The Official AMANDA Documentation

Stefan G. Weichinger and Amanda Core Team

July 11, 2007

ABSTRACT

NOTE

This is the Docbook/XML-version of the official *Amanda* documentation. Please refer to <http://www.amanda.org/docs/AMANDA-docs.html> for the latest versions of these documents. They are available as html, ps and pdf.

This is the official documentation for *Amanda* 2.5.0. The release 2.5.0 brings various new features:

- Communication security/authentication
- Data security
- Enhanced possibilities for Compression
- Dump images spanning multiple media volumes
- Auto tape labelling
- Improved code quality

Feel free to contact me with corrections, additions or updates. Thank you.

Stefan G. Weichinger, for the AMANDA Core Team, March 2006.

sgw@amanda.org <mailto:sgw@amanda.org>

PREFACE

Conversion to Docbook/XML by Stefan G. Weichinger, member of the *Amanda* Core Team.
XML-Buildtree by Jelmer R. Vernooij, member of the Samba-Team. Thanks, Jelmer ... !

COPYRIGHT INFORMATION

CONTENTS

Contents

Part I

Installation

INITIAL INSTALLATION

This section of the *Amanda*-docs contains general info on how to install *Amanda* and how to configure its basic parts. Please read this!

Chapter 1

AMANDA 2.5.0 - SYSTEM-SPECIFIC INSTALLATION NOTES

Please read the notes that correspond to the architectures you are installing for. If you find additional gotchas, or anything incorrect in these notes, please send your updates to <mailto://amanda-hackers@amanda.org> after checking that they are not known/fixed problems in the *Amanda* patches page: <http://www.amanda.org/patches/>.

1.1 Solaris 2.6

You may have compilation errors on statfs.c if you're running, on a Solaris 2.6 host, a **gcc** that was not build on a Solaris 2.6 host. This happens because **gcc** stores fixed copies of some Solaris header files on an internal directory. You must rebuild **gcc** if you get this kind of trouble. Note, however, that **gcc** 2.7.2.3 does not support Solaris 2.6, you should upgrade to 2.8.0 or higher, or egcs.

1.2 Solaris

You may get errors running **make**: Assure that you use the BSD-version of **make**, usually `/usr/ccs/bin/make`. Add `/usr/ccs/bin` to the path before running **configure**.

1.3 Trusted Solaris

According to Julian Stevens julian.stevens@baedsl.co.uk <mailto:julian.stevens@baedsl.co.uk>, the format of **inetd** on Trusted Solaris 1.2 is a bit different. Just before the user name, you should insert the word 'untrusted':

```
amanda dgram udp wait untrusted amuser /usr/local/libexec/amandad amandad
```

The patch-system script is *NOT* aware of this detail; you must fix it yourself.

1.4 SunOS 4.x

A bug in **GNU-tar** 1.12 causes it to miscalculate (in fact, to misreport) the size of filesystems. A patch for **GNU-tar** is available in the patches directory.

1.5 Ultrix

The Ultrix dump program contains an explicit check that it is being run by root. This defeats the usual practice of a non-root "operator" userid for dumps. For this reason, the rundump program (a setuid-root wrapper for dump) is enabled by default. If you find rundump is not necessary for you, just run

```
configure --without-rundump
```

The Ultrix restore program will fail if it is not connected to a tty. Since the restore program is invoked in the clients in order to create index files, and it the client is not connected to a tty, index creation will fail. Using **GNU-tar** instead of DUMP is an option. Thanks to Edwin Wiles ewiles@mclean.sterling.com <mailto:ewiles@mclean.sterling.com> for the investigation. Another alternative proposed by Martyn Johnson Martyn.Johnson@cl.cam.ac.uk <mailto:Martyn.Johnson@cl.cam.ac.uk> is to use a modified restore program: use a binary program editor and replace '/dev/tty' with '/dev/nul', for instance, and link /dev/nul to /dev/null. Note that the chosen file name must be exactly 8 bytes long, otherwise you'll break the restore program. A nice one-liner perl script by Martyn Johnson will do the trick (make sure you preserve a copy of the original restore program, it will be rewritten by running this script!):

```
perl -pi -e 'BEGIN { $/ = "/dev/tty" } s-$/-/dev/nul-' restore
```

The Ultrix C compiler seems to be too broken to compile *Amanda*. You are likely to need **gcc** or egcs.

1.6 HP/UX

You may run into an internal /bin/sh limit when running the configure script. The error message is:

```
./configure: sh internal 2K buffer overflow
```

Using /bin/posix/sh usually works around the problem. Another solution is to install GNU bash and use it instead of /bin/sh.

If 'configure' complains about not finding 'lex', you'll have to get 'flex' installed. Look for its URL in Amanda Installation Notes.

If you use logical volumes, you may refer to mountpoints or full device pathnames instead of device names in the disk list file.

According to Stan Brown stanb@awod.com <mailto:stanb@awod.com>, **amverify** won't work with HP/UX's stock **mt**. The work-around is to install GNU **cpio**, that contains an

implementation of **mt**, and edit **amverify** so that MT points to GNU **mt** and MTOPT is '-f', or reconfigure and rebuild amanda from scratch, making sure it finds GNU **mt** before the stock **mt** in the PATH.

If you have vxfs filesystems to back up, *Amanda* will pick vxdump automatically.

GNU-tar 1.12 will incorrectly report the size of backups. There's a patch in the patches directory that fixes this problem.

The use of 'amhpfixdevs' is deprecated, since you can list mount-points or full device names in the `disklist`. The script may be removed in future releases of *Amanda*.

Sometimes you may get the error message 'Link severed' from an HP/UX client. This is just a cute name the developers of HP/UX found for 'Network error'. Reported by Michael Brehl mbr@condor.de <`mailto:mbr@condor.de`>

1.7 Linux

Linux hosts intended to back up efs partitions with **dump** should install the dump package, as it is not installed by default on most Linux distributions. It is possible to find compiled versions of **dump** on most Linux sites and CD-ROMs.

Note, however, that DUMP for Linux has been quite unreliable, sometimes being unable to perform backups, producing weird error messages such as 'master/slave protocol botched', and sometimes creating unrestorable dump images, especially of active filesystems. The first problem can sometimes be fixed by cleaning up outdated entries in `/etc/dumpdates`, but your best bet is probably **GNU-tar**.

Make sure the user that runs configure has permission to run the **dump** program, otherwise configure may misdetect an ability of **dump** to accept a -E (for estimates) switch.

GNU-tar 1.11.8, distributed with some Linux versions, will cause index failures (Index returned -1). Upgrading to **GNU-tar** 1.12 fixes this problem. This is not a Linux-specific problem, but it is quite common in this platform.

Amanda now supports the ftape driver version 3.04d. It adjusts the blocksize automatically to 32k and supports QIC volume tables. More details can be found in the file ZFTAPE in this directory.

Some releases of dump for Linux, such as the one distributed with Debian 2.0, have modified dump so that it stores dumpdates in /var/lib. If this is your case, you should create a link to it in /etc. Suggested by David Wolfskill dhw@whistle.com <`mailto:dhw@whistle.com`>

1.8 Digital Unix 4

According to Michael Galloway mgx@spruce.lsd.ornl.gov <`mailto:mgx@spruce.lsd.ornl.gov`>, the stock DUX4 dump is broken. There is a patch available at <`ftp://ftp.service.digital.com/public/dunix/v4.0b/duv40bas00005-19970926.README`>

When both dump and vdump are available, *Amanda* will use vdump for backing up advfs filesystems only, and **dump** will be used for the rest. If you'd rather back up all filesystems with vdump, #undef DUMP in config/config.h after running configure.

Unfortunately, the output of 'dump -E' incorrectly matches a line of output from SAMBA, which gets *Amanda*'s estimate process confused. `client-src/sendsize.c` will refuse to compile if both HAVE_DUMP_ESTIMATE and SAMBA_CLIENT are defined in `config/config.h`. *Amanda* will work correctly if you undefine HAVE_DUMP_ESTIMATE in `config/config.h`; if you prefer to have incorrect estimates for SAMBA backups, follow the instructions in `client-src/sendsize.c` on removing the compile-time error.

According to Oren Laadan orenl@cs.huji.ac.il <`mailto:orenl@cs.huji.ac.il`>, DEC compiler version "DEC C V5.2-033 on Digital UNIX V4.0 (Rev. 564)" (obtained with "cc -V") does not build *Amanda* properly, in particular, taper.c. Using **gcc** is OK.

1.9 Sinix 5.43 (Reliant Unix)

The use of 'amsinixfixdevs' is deprecated, since you can list mount-points or full device names in the disklist. The script may be removed in future releases of *Amanda*.

Sinix port originally by Michael Schmitz mschmitz@iname.com <`mailto:mschmitz@iname.com`>.

1.10 IRIX (all)

When setting the tape device name in either `amanda.conf` or one of the changer configuration files, make sure you specify the "variable" device name, which has a 'v' on the end. If not, IRIX will write 4KByte blocks instead of the 32KByte blocks *Amanda* tells it to. This apparantly works OK unless you take the tape to a non-IRIX system, where **amrestore** will complain about a short (4096) read.

If you do end up in this situation, the **dd** command to use to pipe into your system **restore** program is:

```
dd if=/dev/whatever bs=4k skip=8 | ...
```

Jean-Francois Malouin Jean-Francois.Malouin@bic.mni.mcgill.ca <`mailto:Jean-Francois.Malouin@bic.mni.mcgill.ca`> reports that, if you are going to use an IRIX host as the tape server, you *must* patch your system with *all* kernel and scsi rollup patches, otherwise you may end up with useless tapes, due to tape rewinding after short periods of inactivity. See <`http://www-viz.tamu.edu/~sgi-faq/`> for more details.

Some unpatched version of **xfsdump** are reported as not printing estimates. This causes estimates to take much longer than they had to, because backups are always performed twice. According to Mike Acar mike@kentinfoworks.com <`mailto:mike@kentinfoworks.com`>, patch 2333 for IRIX 5.3 and 6.1 will fix this problem.

1.11 IRIX 6

Seems like SGI **make** program is a bit broken, in a way that causes it to rebuild some files if doesn't have to if you happen to run **make** again. Using GNU **make** fixes this problem.

If you have xfs filesystems to back up, *Amanda* will pick xfsdump automatically.

1.12 IRIX 6.5.x

Luc Lalonde Luc.Lalonde@polymtl.ca <mailto:Luc.Lalonde@polymtl.ca> contributed the following notes:

If you use a jukebox, you must set the ownership of the robot-changer device to the *Amanda* operator:group in `/etc/ioperms`. Here's my configuration:

```
/etc/ioperms: /dev/scsi/sc8d6l0 0600 amanda backup
```

Otherwise the ownership:group is changed to "root:sys" after each reboot. When you do upgrades, check the file `/var/sysgen/master.d/scsi` to see if it has changed. Otherwise your jukebox could be rendered unuseable. In particular, check if it has been replaced by a new version and renamed to "./scsi.O/.".

If you use the *Amanda* package provided by freeware.sgi.com, you are not affected by the first question since at compile time the *Amanda* operator is "root:sys".

1.13 SCO

Jens Krause jens@transcom.de <mailto:jens@transcom.de> has reported some problems with **GNU-tar** 1.12 on SCO Release 5. Although the 'sparse' files patch was applied, **GNU-tar** would consistently crash. GNUtar had to be built linked with malloc-libraries, and the '–sparse' switches had to be removed from `client-src/sendbackup-gnutar.c` and `client-src/sendsize.c`.

1.14 FreeBSD 3.0

chg-scsi was not updated to support the new camlib.h-dependent chio.h, so chg-scsi will be automatically disabled if camlib.h is found. You may use chg-chio instead.

1.15 FreeBSD 4.9

Sep. 28th, 2004: Jason Miller jwm@interlinc.net <mailto:jwm@interlinc.net> reported problems with setting up the *Amanda*-client on FreeBSD 4.9. He wrote:

Due to the need for read permissions for *Amanda*-client the default user and group for this on FreeBSD 4.9 is "operator:operator" which I found a write up on that as well. Just a note the port wanted to install it with these user permissions by default and I initially

changed them to match my Redhat 9.0 install. So just doing a **make distclean uninstall install *Amanda*_SERVER=servername** fixed that for me. Then I just followed the below instructions and everything was good to go.

Refer to this link for more details: <`http://www.freebsd.org/cgi/query-pr.cgi?pr=59302`>.

1.16 FreeBSD 5.1

Nicolas Ecarnot nicolas.ecarnot@accim.com <`mailto:nicolas.ecarnot@accim.com`> discovered that for FreeBSD 5.1 (maybe earlier, and surely further), you have to set the net.inet.udp.maxdgram TCP/IP variable to 65535. The default is 9216, and this is a problem when trying to backup a large number of clients as indicated by errors in during **amcheck** or the estimate phase.

You can just run the command: **sysctl net.inet.udp.maxdgram=63535** but this won't last until the next reboot.

To make it permanent, just add this line:

```
net.inet.udp.maxdgram=65535
```

in the file `/etc/sysctl.conf`.

1.17 AIX

sendsize is reported to coredump on AIX 4.3.3, this is a linking problem, try configuring *Amanda* with the option "–disable-shared".

1.18 Microsoft Windows

Although *Amanda* won't run standalone on MS-Windows hosts, it is possible to use it to back up their disks, by using SAMBA. Please read Backup PC hosts using Samba for more information.

SAMBA may be unable to back up some files due to file locking restrictions. Particularly, paging and registry files usually present problems. Backing up page files is pointless, but registry files are quite important to back up. It is possible to create regular files that contain registry information by using the Regback utility, from the Windows NT Resource Kit. Unfortunately, it is not part of the Windows NT standard distribution, you have to purchase it separately. Thanks to Ernie Oporto ernie_oporto@mentorg.com <`mailto:ernie_oporto@mentorg.com`> for the tip.

1.19 Mac OS X

For notes on how to setup *Amanda* under Apple's OS X, please refer to <`http://web.brandeis.edu/pages/view/Bio/AmandaMacOSXCompileNotes`>, written by Steven Karelka-

rel@brandeis.edu <mailto:karel@brandeis.edu>. Thanks to Jose L.Hales-Garcia jose@stat.ucla.edu <mailto:jose@stat.ucla.edu> for the tip.

NOTE

Refer to <http://www.amanda.org/docs/systemnotes.html> for the current version of this document.

Chapter 2

AMANDA INSTALLATION NOTES

This document covers the compilation, installation, and runtime setup of *Amanda* 2.4.2 and higher.

2.1 Before doing anything

- Read this document all the way through.
- Consult *Amanda* 2.4.x - System-Specific Installation Notes for installation notes specific to particular operating systems. There is often important information there, so don't forget this step.
- Read Upgrade Issues if you are upgrading from a previous *Amanda* version. There are some issues that you will need to be aware of.
- If you are using KERBEROS authentication, read Kerberos for details on installing and running the kerberized version of *Amanda*.
- Check the *Amanda* Patches Page, <`http://www.amanda.org/patches/`>.

2.2 Compiling the *Amanda* sources

If you have multiple architectures, you only need to install the whole *Amanda* package on the tape server host (the one with tape drive). On the backup client hosts (the ones you are going to dump), you only need to compile some of the *Amanda* programs (see section Set up the Backup Client Hosts below).

2.2.1 Source configuration

- *Amanda* can optionally make use of the following packages to back up different types of clients or clients with different filesystem dumping programs.
  - **GNU-tar**:

    If you wish to use **GNU-tar** to back up filesystems, it is recommended to use **GNU-tar** 1.13.25. Plain **GNU-tar** 1.12 needs to be patched to handle large

files (> 2GB). Plain **GNU-tar** 1.13 creates bad index-lists which **amrecover** cannot handle, as does the rarely used **GNU-tar** 1.13.9x, which changed the index-format again in an incompatible way.

Refer to the *Amanda* FAQ for more information about issues with the various releases of **GNU-tar**.

If you need to use **GNU-tar** 1.12, get it at

<ftp://ftp.gnu.org/pub/gnu/tar/tar-1.12.tar.gz>

and apply the patch from `patches/tar-1.12.patch`. The first hunk may be enough, unless it's a SunOS4 host. Read more about the patches in the patch file itself.

GNU-tar 1.13.25 can be found at:

<ftp://alpha.gnu.org/pub/gnu/tar/tar-1.13.25.tar.gz>

– Samba:

Samba allows Unix systems to talk to PC clients. *Amanda* can back up Microsoft Windows clients using Samba:

<http://www.samba.org>

Read Backup PC hosts using Samba for configuration tips and known limitations.

Look at <http://www.amanda.org/patches/> for up to date information on patches.

– Perl:

If you wish to make use of some of the scripts that come with *Amanda*, you will need to install Perl. You can get Perl from any CPAN site.

<ftp://ftp.cpan.org/pub/CPAN/src/perl-5.6.1.tar.gz>

– Awk:

One of the programs included in this package is amplot, which reads a data file that *Amanda* generates for each dump and translates that information in it into a nice picture that can be used to determine how your installation is doing and if any parameters need to be changed. To use amplot, you need a version of awk that understands command line variable substitutions, such as nawk or gawk, which is available from

<ftp://ftp.gnu.org/pub/gnu/gawk/gawk-3.1.1.tar.gz>

– GNUplot:

Amplot also required that gnuplot be installed on your system. Gnuplot is available at

<http://www.gnuplot.org/> <ftp://ftp.gnuplot.org/pub/gnuplot>

– Other packages:

The process of building *Amanda* requires that some other packages be installed on your system. The following packages are used:

<ftp://ftp.gnu.org/pub/gnu/readline/readline-5.0.tar.gz>

amrecover optionally uses the readline library for its command-line edition mechanisms. (If you use a package-based distribution, check for the package readline-devel-X.Y.rpm.) This library itself requires either termcap, curses or ncurses. termcap is preferred, and it may be obtained from:

<ftp://ftp.gnu.org/pub/gnu/termcap/termcap-1.3.1.tar.gz>.

If you wish to edit and enhance *Amanda*, you may need to install the following tools. Autoconf and automake are required if you are going to rebuild the Makefiles and auto configuration scripts. Bison is only needed if you are going to work on the index server and client code.

<ftp://ftp.gnu.org/pub/gnu/autoconf/autoconf-2.53.tar.gz> <ftp://ftp.gnu.org/pub/gnu/automake/automake-1.6.3.tar.gz> <ftp://ftp.gnu.org/pub/gnu/bison/bison-1.27.tar.gz> <ftp://ftp.gnu.org/pub/gnu/flex/flex-2.5.4a.tar.gz>

- Read about the different configuration options available for building and running *Amanda*. To see the options, do both:
  - Run `./configure --help` to see the available options that **configure** takes.
  - Read the file `example/config.site` which gives longer descriptions to the same options.
- Choose which user and group you will run the dumps under. Common choices for user are 'bin' or another user specifically created for *Amanda*, such as 'amanda'; common choices for group are 'operator' or 'disk'. If you do not specify –with-user=<username> and –with-group=<groupname>, **configure** will abort. Also choose the default name for your configuration, such as 'csd' or 'DailySet1'). This name is used by the *Amanda* commands to choose one of multiple possible configurations. You may specify it using the –with-config=<confgname>.
- Decide where *Amanda* will live. You need to choose a root directory for *Amanda*. Let this root directory be called $prefix. Unless you change the default behavior with the appropriate command line options, *Amanda* will install itself as. Listed below you find the appropriate **configure**-option for each directory to change the location of this part of *Amanda*.

```
--sbindir=$prefix/sbin              Amanda server side programs
--libexecdir=$prefix/libexec           Amanda backup client programs
--libdir=$prefix/lib             Amanda dynamic libraries
--with-configdir=$prefix/etc/amanda       Runtime configuration files
--with-gnutar-listdir=$prefix/var/amanda/gnutar-lists Directory for GNU-tar lists
--mandir=$prefix/man             Directory for manual pages
```

Note that the **GNU-tar** listdir should be a local filesystem on each client that is going to be backed up with **GNU-tar**. If it really must be NFS-mounted, make sure the

filesystem is exported so that the client has root access to it.

- Decide if you are compiling *Amanda* on a server only or a client only platform. If you have a particular operating system that will only be a *Amanda* client and will never run as the master tape host, then add the –without-server option to **configure**. In the unlikely case that you have a particular operating system that will serve as the tape host and you do not wish to back up any machines that run this operating system, add the –without-client option to the configure options. There are many other configuration switches for *Amanda*. You may learn more about them by running `configure --help` and by reading `examples/config.site`.

- Now configure *Amanda*. There are two ways of doing this. If you are running *Amanda* on a single OS, then probably the first method works better for you. If you need to support multiple platforms, then the second method will work better.

  – Run **configure** as non-root-user with the appropriate command line options. You will probably want to remember the command line options for future builds of *Amanda*.

  – Edit `examples/config.site` and install it in the directory $prefix/etc or $prefix/share. When **configure** runs the next time it will look for this file and use it to configure *Amanda*.

2.2.2 Building and installing the binaries

- Back at the top-level source directory, build the sources:

```
make
su root; make install
```

  Make sure that you don't build the software as root, you may run the first make-command as the *Amanda*-user, for example. On the other hand you have to run **make install** as root to get the binaries installed with the proper permissions. If you want to change the compiler flags, you can do so like this:

```
make CFLAGS="-O3 -Wall"
```

- If you have built with USE_VERSION_SUFFIXES, you will want to create symlinks to the version you wish to use, eg: `ln -s amdump-x.y.z amdump` This is not done automatically by the install process, so that you can have multiple *Amanda* versions co-existing, and choose yourself which to make the default version. The script `contrib/set_prod_link.pl` may save you some keystrokes.

- Run `ldconfig` as root to update the paths to the recently installed shared libraries.

2.3 Setting up your *Amanda* Configuration

2.3.1 Setting up the Tape Server Host

- Create the config directory (eg. `/usr/local/etc/amanda/confname`) and copy the example/ files into that directory. Edit these files to be correct for your site, consulting the amanda(8) man page if necessary. You can also send mail to <`mailto://amanda-users@amanda.org`> if you are having trouble deciding how to set things up. You will also need to create the directory for the log and database files for the configuration to use (eg `/usr/local/var/amanda/confname`), and the work directory on the holding disk. These directories need to agree with the parameters in `amanda.conf`. Don't forget to make all these directories writable by the dump user!

  Make sure that you specify the *no-rewind* version of the tape device in your `amanda.conf` file. This is a frequently encountered problem for new sites.

  Note that you might want to temporarily set the option "no-record" in all your dumptypes when first installing *Amanda* if you'd like to run tests of *Amanda* in parallel with your existing dump scheme. *Amanda* will then run but will not interfere with your current dumpdates. However, you don't want to run with "no-record" under normal operations.

- Put *Amanda* into your crontab. Here's a sample:

Example 2.3.1 /etc/crontab

```
0 16 * * 1-5 /usr/local/sbin/amcheck -m confname
45 0 * * 2-6 /usr/local/sbin/amdump confname
```

  This is for SunOS 4.x, which has a per-user crontab; most other systems also require a userid on each cron line. See your cron(8) for details. With these cron lines, *Amanda* will check that the correct tape is in the drive every weekday afternoon at 4pm (if it isn't, all the operators will get mail). At 12:45am that night the dumps will be run.

- Put the *Amanda* services into your `/etc/services` file. Add entries like:

Example 2.3.2 /etc/services

```
amanda      10080/udp
amandaidx   10082/tcp
amidxtape   10083/tcp
```

  You may choose a different port number if you like, but it must match that in the services file on the client hosts too.

  If you are running NIS (aka YP), you have to enter the *Amanda* service into your NIS services database. Consult your NIS documentation for details.

  You may use the 'patch-system' script, from client-src, in order to modify this file. Run it with a '-h' argument for usage.

- If you are going to use the indexing capabilities of *Amanda*, follow one of the following steps:
  - If your server uses **inetd**, then add these lines to your `inetd.conf` on the tape server host:

Example 2.3.3 /etc/inetd.conf

```
amandaidx stream tcp nowait $USER $AMINDEXD_PATH amindexd
amidxtape stream tcp nowait $USER $AMIDXTAPED_PATH amidxtaped
```

    where `$AMINDEXD_PATH` and `$AMIDXTAPED_PATH` are the complete paths to where the amindexd and amidxtaped executables (usually libexec_dir/amindexd and libexec_dir/amidxtaped), and USER is the *Amanda* user.

    You may use the 'patch-system' script, from client-src, in order to modify this file. Run it with a '-h' argument for usage.

  - If your tape server uses **xinetd** instead of **inetd**, then you have to add the following two files to your xinetd-configuration (usually `/etc/xinetd.d`) and edit the paths:

Example 2.3.4 /etc/xinetd.d/amandaidx

```
   service amandaidx
{
   socket_type     = stream
      protocol     = tcp
      wait         = no
      user         = $USER
      group        = $GROUP
      groups          = yes
      server          = $AMINDEXD_PATH/amindexd }
```

Example 2.3.5 /etc/xinetd.d/amidxtape

```
service amidxtape
{
   socket_type     = stream
      protocol     = tcp
      wait         = no
      user         = $USER
      group        = $GROUP
      groups          = yes
      server          = $AMIDXTAPED_PATH/amidxtaped }
```

  - If your tape server uses Dan Bernstein's daemontools <`http://cr.yp.to/daemontools.html`>) instead of (x)inetd, you have to create **amandaidx** and **amidxtape** services by hand.
    - Create service directories:

```
mkdir -p $prefix/etc/amanda/supervise/amandaidx
mkdir -p $prefix/etc/amanda/supervise/amidxtape
```

* Create service startup files and make them executable:

Example 2.3.6 /etc/amanda/supervise/amandaidx/run

```
#!/bin/sh
exec /usr/local/bin/setuidgid amanda \
/usr/local/bin/tcpserver -DHRl0 0 10082 \
/usr/local/libexec/amindexd >/dev/null 2>/dev/null
```

Example 2.3.7 /etc/amanda/supervise/amidxtape/run

```
#!/bin/sh
exec /usr/local/bin/setuidgid amanda \
/usr/local/bin/tcpserver -DHRl0 0 10083 \
/usr/local/libexec/amidxtaped >/dev/null 2>/dev/null
```

* Link service directories into your svscan directory:

```
cd /service
ln -s $prefix/etc/amanda/supervise/amandaidx .
ln -s $prefix/etc/amanda/supervise/amidxtape .
```

- If the tape server host is itself going to be backed up (as is usually the case), you must also follow the client-side install instructions below on the server host, INCLUDING setting up the file `.amandahosts` so that the server host lets itself in. This is a frequently encountered problem for new sites.

2.3.2 Set up the Backup Client Hosts

- When using BSD-style security (enabled by default), set up your `~dumpuser/.amandahosts` (or `~dumpuser/.rhosts` and/or `/etc/hosts.equiv`, if you have configured –without-amandahosts) so that the dumpuser is allowed in from the server host. Only canonical host names will be accepted in `.amandahosts`, and usernames must be present in every line, because this is safer.
- Set up your raw disk devices so that the dumpuser can read them, and `/etc/dumpdates` so that the dumpuser can write to it. Normally this is done by making the disk devices readable by (and dumpdates read/writable by) group 'operator', and putting the dumpuser into that group.
- Put the *Amanda* service into your `/etc/services` file. Add entry like:

  You may choose a different port number if you like, but it must match that in the services file on the tape server host too.

Example 2.3.8 /etc/services

```
amanda      10080/udp
amandaidx   10082/tcp
amidxtape   10083/tcp
```

If you are running NIS (aka YP), you have to enter the *Amanda* service into your NIS services database. Consult your NIS documentation for details.

You may use the 'patch-system' script, from client-src, in order to modify this file. Run it with a '-h' argument for usage.

- Follow one of the following steps to set up the *Amanda* client service:
  - If your *Amanda* client uses **inetd**, put the *Amanda* client service into inetd's config file. This file is usually found in `/etc/inetd.conf`, but on older systems it is `/etc/servers`. The format is different on different OSes, so you must consult the **inetd** man page for your site. Here is an example from our site, again from SunOS 4.x:

Example 2.3.9 /etc/inetd.conf

```
amanda dgram udp wait USER AMANDAD_PATH amandad
```

You may use the 'patch-system' script, from client-src, in order to modify this file. Run it with a '-h' argument for usage.

  - If your *Amanda* client uses **xinetd**, you have to add the following file to your **xinetd**-configuration (usually `/etc/xinetd.d`) and edit it to reflect your settings and paths:

Example 2.3.10 /etc/xinetd.d/amanda

```
service amanda
{
socket_type     = dgram
protocol    = udp
wait        = yes
user        = $USER
group       = $GROUP
groups          = yes
server          = $AMANDAD_PATH/amandad
}
```

  - If your *Amanda* client uses Dan Bernstein's daemontools (<`http://cr.yp.to/daemontools.html`>) instead of (x)inetd, you have to create the amanda service by hand. You will need also an UDP super-server (netcat in this example).
    * Create service directory:

```
mkdir -p /etc/amanda/supervise/amanda
```

* Create service startup file and make it executable:

Example 2.3.11 /etc/amanda/supervise/amanda/run

```
#!/bin/sh
exec /usr/local/bin/setuidgid amanda \
  /usr/bin/netcat -l -u -p 10080 -q 0 \
  -e /usr/local/libexec/amandad >/dev/null 2>/dev/null
```

NOTE

The netcat-binary used in this run-file might also be called `/usr/bin/nc` on your system, depending on the OS-distribution you use. Refer to <`http://netcat.sourceforge.net`> for details of **netcat**.

* Link service directory into your svscan directory:

```
cd /service
ln -s /etc/amanda/supervise/amanda .
```

- If you are using (x)inetd, kick **inetd**/**xinetd** to make it read its config file. On most systems you can just execute `kill -HUP inetd` (or xinetd). On older systems you may have to kill it completely and restart it. Note that killing/restarting (x)inetd is not safe to do unless you are sure that no (x)inetd services (like rlogin) are currently in use, otherwise (x)inetd will not be able to bind that port and that service will be unavailable.

  If you are using the daemontools, svscan should detect and start your new services automatically.

- If you intend to back up xfs filesystems on hosts running IRIX, you must create the directory `/var/xfsdump/inventory`, otherwise xfsdump will not work.

THAT'S IT! YOU ARE READY TO RUN, UNLESS WE FORGOT SOMETHING. Please send mail to <`mailto://amanda-users@amanda.org`> if you have any comments or questions. We're not afraid of negative reviews, so let us have it!

Before writing questions, you may prefer to take a look at the *Amanda* FAQ and at the *Amanda* home page, at <`http://www.amanda.org`>. Browsable archives of *Amanda* mailing-lists are available at <`http://marc.theaimsgroup.com/?l=amanda-users`> and <`http://marc.theaimsgroup.com/?l=amanda-hackers`>.

NOTE

Refer to <http://www.amanda.org/docs/install.html> for the current version of this document.

Chapter 3

EXCLUDING

3.1 Introduction

There are times when data needs to be excluded from a backup. When these times arise be confident that *Amanda* has this capability. (Actually it's not *Amanda*, it's tar.) There are three ways of excluding data in an *Amanda* backup:

- Exclude an individual item explicitly in the dumptype
- Utilize an "Exclude List"
- Do not include the data in the disklist

This document is based on *Amanda* 2.4.2 and some of this might not work with older versions. This was compiled from my personal experience and with help from the members of the amanda-users mailing list (<mailto://amanda-users@amanda.org>) when I was originally setting this up, to whom I wish to thank for all of their support.

3.2 Please Read

As far as I am able to tell the only way to exclude files or directories with *Amanda* is to use **GNU-tar** as the dump program (others?). The file system dump programs provided with unix systems (e.g. dump, ufsdump) get data at a raw drive level and generally do not allow exclusion of specific files or directories.

The GNU version of tar, (**GNU-tar** or gtar), reads its data at a file system, (or higher), level and does include the option to exclude specific files and/or directories. It should be mentioned here that tar will change the access times on files. Tar has the ability to preserve the access times however, doing so effectively disables incremental backups since resetting the access time alters the inode change time, which in turn causes the file to look like it needs to be archived again.

The only exception that I am aware of is to just not include the data in question in the disklist. This option may not be suitable for everyone's needs and can confuse the issue some, so I have elected to include this mechanism in its own section named Do not include the data in the disklist.

For the purpose of this document an *Amanda* backup configuration named "exclude-test" will be used. The machine that contains the tape drive which receives data to be archived will be referred to as "SERVER". The machine that data is being archived from will be referred to as "CLIENT". These two systems are usually different machines but are not required to be, and may be the same machine. Parts of this setup are on the server and some are on the client.

NOTE

When *Amanda* attempts to exclude a file or directory it does so relative to the area being archived. For example if `/var` is in your disklist and you want to exclude `/var/log/somefile`, then your exclude file would contain `./log/somefile`. You may use one exclude file in multiple dumptypes without any restriction.

3.3 Before We Begin

The first step that should be taken is to verify that backups are currently working. Connect to SERVER and run **amcheck** as your *Amanda* user, to verify that there are no errors in the current setup.

```
$ amcheck -cl CLIENT
```

Output should look something like below for success:

```
Amanda Tape Server Host Check
-----------------------------

/path/to/holding-disk: 4771300 KB disk space available, that's plenty.
Amanda Backup Client Hosts Check
--------------------------------
Client check: 1 host checked in 0.084 seconds, 0 problems found.
```

Next make sure that **GNU-tar** is the dump program currently in use. The easiest way to tell if your dumptype is using gnutar is to run the following:

```
$ amadmin exclude-test disklist CLIENT
```

Among all the output is the "program" value currently in use. This value is also specified with the "program" option in the dumptype. If the dumptype has the line "program GNUTAR" your setup should be ready to exclude data.

If **GNU-tar** is not in use add the line "program GNUTAR" to the dumptype, and then run **amcheck** again to verify that backups should work. The capitalization of GNUTAR is required.

The dumptype should look something like:

```
define dumptype exclude-test {
comment "test dumptype for documentation"
priority high
program "GNUTAR"
}
```

3.4 Choosing an exclude mechanism

If the need is to exclude only one file or directory then the easiest way to accomplish this is to exclude an individual item explicitly in the dumptype. If the need is to exclude multiple files or directories then use an Exclude List.

3.4.1 Exclude Mechanisms

3.4.1.1 Exclude an individual item explicitly in the dumptype

The easiest way to exclude a file or directory is to specify it with the "exclude" option in the dumptype. This option accepts an argument of the file or directory to be excluded. *Amanda* allows only one exclude option in any dumptype at a time.

NOTE

UPDATE: Recent *Amanda*-releases bring the option "exclude append" which enables the administrator to define more than one exclusion-pattern within one dumptype without using a exclude-list. Please look at the amanda.conf.5-manpage for details.

Any path specified to be excluded must be encapsulated with quotes. Continuing with our example from above `/var/log/somefile` and using the same dumptype as above, the dumptype would now look like:

```
define dumptype exclude-test {
comment "test dumptype for documentation"
priority high
program "GNUTAR"
exclude "./log/somefile"
}
```

Next run **amcheck** again to verify that there are no problems with the revised *Amanda* configuration. If the data is not being excluded as expected please see the Troubleshooting section below. This completes the setup of excluding an individual item in the dumptype.

3.4.1.2 Utilize an Exclude List

An exclude list is a file that resides on the CLIENT machine and contains paths to be excluded, one per line. This file can be in any location on the CLIENT so long as the same path is specified in the dumptype. Some find `/usr/local/etc/amanda` an appropriate location, but it is up to you. I personally like to have a subdirectory for exclude files but it is up to you where you place this file.

The exclude file may also be placed in the area being archived. This is an easy way to have a different exclusion file for each disklist entry without needing separate dumptype definitions. To use this technique, enter a path relative to the area being archived as the exclude file below instead of an absolute path.

Connect to CLIENT and create the exclude directory as root. For example:

```
$ mkdir -p /usr/local/etc/amanda/exclude
$ cd /usr/local/etc/amanda/exclude
```

Next create the exclude list for *Amanda* to use. You can name the exclude file anything you wish it to be. Create a file, and in this file place all paths to files and directories that are to be excluded. Keeping with the `/var` example, assume that `/var/log/XFree86.0.log`, and `/var/log/maillog` need to be excluded. Remember that all paths are relative. The exclude list would look like:

```
./log/XFree86.0.log
./log/maillog
```

Make sure that permissions are restricted on this file. Run the following as root, where exclude-filename is the name of the file you just created. For example:

```
$ chmod 644 /usr/local/etc/amanda/exclude/exclude-filename
```

This concludes the necessary configuration on the client.

Connect to SERVER and cd to the exclude-test *Amanda* configuration directory. Edit the *Amanda* configuration file e.g. `amanda.conf`. Add an entry similar to the following line, to the dumptype for the client in question, where the exclude-filename is the file that was created on CLIENT in the step above including the quotes. For example:

```
exclude list "/usr/local/etc/amanda/exclude/exclude-filename"
```

The new dumptype should look something like:

```
define dumptype exclude-test{
```

```
comment "test dumptype for documentation"
priority high
program "GNUTAR"
exclude list "/usr/local/etc/amanda/exclude/exclude-filename"
}
```

Save the file. Run **amcheck** again to verify that there are no problems with the revised *Amanda* configuration. If **amcheck** succeeds then run **amdump** to verify the data is being excluded correctly. If the data is not being excluded as expected please see the Troubleshooting section below. This completes the setup of an exclude list.

3.4.1.3 Do not include the data in the disklist

Amanda uses disklist entries to define which directories or partitions should be archived. This allows us to exclude data by just not placing the data in question in the disklist. Assume that there is a disk mounted on `/example`. The directory `/example` has five subdirectories "a", "b", "c", "d", and "e". The directories "a", "b", and "c" need to be archived, while "d" and "e" should not. This can be accomplished by not specifying "d" and "e" in the disklist. Using the same dumptype and host in the above examples the disklist would contain:

```
CLIENT /examples/a   exclude-test
CLIENT /examples/b   exclude-test
CLIENT /examples/c   exclude-test
```

Run **amcheck** to verify that *Amanda* is working correctly. If the data is not being excluded as expected please see the Troubleshooting section below. This completes the setup of using a disklist to exclude data.

3.5 Expression

Quiz: what is the difference between the following entries in an exclude list?

```
./foo
./foo/
./foo/*
```

case 1 : directory ./foo won't be in the backup image (that's what you want) case 2 : matches nothing (don't use it) case 3 : directory ./foo will be in the backup image but nothing below it.

3.6 Wildcard Expansion

Amanda has the ability to use wildcard expansion while excluding data as implemented by tar(1). The only places that wildcard expansion is allowed is in the "exclude" option in the dumptype, or in the exclude list. Some simple examples:

Exclude any file or directory that ends in ".log" e.g. ppp.log, XFree86.0.log

`./*.log`

Exclude any file or directory with the string "log" e.g. logfile, maillog, syslog, ppp.log, XFree86.0.log

`*/*log*`

Exclude any file or directory that starts with string "cron" and ends in ".gz" e.g. cron.1.gz, cron.2.gz, log/cron.1.gz

`./*cron*.gz`

The question mark can be used to specify a single character. e.g. log.1, log.2, etc

`./log.?`

3.7 Troubleshooting

3.8 Section

If you find that you are having trouble getting the exclude patterns to match correctly, check out this really cool script written by John R. Jackson.

<`ftp://gandalf.cc.purdue.edu/pub/amanda/gtartest-exclude`>

This script allows you to test your patterns before placing them in an exclude list or in the dumptype. Instructions on how to run the script are included in the script.

3.8.1 Broken gnutar?

There are versions of **GNU-tar** that do not correctly exclude data. Version 1.12 (plus the *Amanda* patches from <`http://www.amanda.org`>) are known to work correctly, as does version 1.13.19 (and later). Anything else is questionable.

NOTE

*UPDATE:*Using **GNU-tar** 1.13.25 is recommended.

3.8.2 The ps command is your friend.

Connect to CLIENT and run a

```
ps ax | grep tar
```

(or

```
ps ef | grep tar
```

on Solaris) to see exactly how the tar command is running. Look in the output for the –exclude or –exclude-from options in the running tar process. For example:

```
$ ps ax | grep tar

?        R      0:37 /bin/tar --create --directory /var
--listed-incremental /var/lib/amanda/gnutar-lists/CLIENTvar_0.new
--sparse --one-file-system --ignore-failed-read --totals --file
/dev/null --exclude-from=/usr/local/etc/amanda/exclude-test/exclude.var
.
```

In the above output notice the string "–exclude-from=". The string following the "=" is the exclude file currently in use. If the string was "–exclude" then the string following the "=" is the file or directory that is currently set to be excluded.

Contact the amanda-users mailing list: <mailto://amanda-users@amanda.org>. Subscription information is available at <http://www.amanda.org>.

NOTE

Refer to <http://www.amanda.org/docs/exclude.html> for the current version of this document.

Chapter 4

INDEXING WITH *AMANDA*

This file describes how the index files are generated and how **amrecover** is used.

4.1 Database Format

The database consists of a directory tree of the format: `$host/$disk/$date_$level.gz`

The host and disk are those listed in the disklist file, the "$host/$disk/" is like the curinfo database, '/' are changed for '_'. There is an index file for each dump, the name of the file is made of the date and the level, they will have the .gz suffix if they are compressed with gzip.

ex. The file `foo/_usr/19991231_0.gz` is the index of the level 0 made on 19991231 of the disk /usr of the host foo.

The files are ASCII text files containing a list of the directory and files of the dump, one per line. Each entry is the filename relative to the mount point, starting with a /, e.g., /home/user1/data from the disk mounted on /home would generate the entry /user1/data. The index files are stored in compressed format (eg gzip or compress).

4.2 Database Browsing

The client is called **amrecover** and is loosely based on the functionality of the program **recover** from Backup Copilot. A user starts up **amrecover**. This requires specifying the index server and the *Amanda* config name (defaults for both are compiled in as part of the installation). Then the user has to specify the name of the host information is wanted about, the disk name, and (optionally) the disk mount point. Finally a date needs to be specified. Given all this, the user can then roam around a virtual file system using **ls** and **cd** much like in a FTP client. The file system contains all files backed up on the specified date, or before that date, back to the last level 0 backup. Only the most recent version of any file is shown.

As the file system is traversed, the user can add and delete files to a "shopping list", and print the list out.

4.3 File Extraction

When a user has built up a list of files to extract, they can be extracted by issuing the command **extract** within **amrecover**.

Files are extracted by the following, for each different tape needed.

As part of the installation, a "tape server" daemon **amidxtaped** is installed on one or more designated hosts, which have an attached tape drive. This is used to read the tapes. See the config files for the options for specifying a default.

amrecover contacts **amidxtaped** on the tape server host specifying which tape device to use, which host and disk files are needed for. On the tape server host, **amidxtaped** executes **amrestore** to get the dump image file off the tape, and returns the data to **amrecover**.

If dumps are stored compressed for the client, then **amrecover** pipes the data through the appropriate uncompression routine to uncompress it before piping it into **restore**, which then extracts the required files from the dump image.

Note that a user can only extract files from a host running the same operating system as he/she is executing **amrecover** on, since the native dump/restore tools are used - unless **GNU-tar** is used.

4.4 Protocol Between amindexd and amrecover

The protocol talked between **amindexd** and **amrecover** is a simple ASCII chat protocol based on that used in FTP. **amrecover** sends a 1 line command, and **amindexd** replies with a 1 line or multi-line reply. Each line of the reply starts with a three digit code, starting with a '5' if an error occurred. For 1 line replies, and the last line of a multi-line reply, the 4th character is a space. For all but the last line of a multi-line reply, the 4th character is a '-'.

The commands and replies other than acknowledgments are:

Table 4.1 Protocol between amindexd and amrecover

QUIT	finish up and
HOST <host>	set ho
DISK <disk>	set dis
LISTDISK [<device>]	list the disks fo
SCNF <config>	set *Amanda* con
DATE <date>	set da
DHST	return dump his
OISD <dir>	Opaque is directory? query. Is the directory `dir` present in the ba
OLSD <dir>	Opaque list directory. Give all filenames present in `dir` in the bac
ORLD <dir>	Opaque recursive list directory. Give all filenames present in `dir` and subdir
TAPE	return value of tapedev
DCMP	returns "YES" if dumps for disk a

4.5 Installation Notes

1 Whether or not an index is created for a disk is controlled by a disk configuration option *index*. So, in `amanda.conf` you need to define a disktype with this option, e.g.,

```
define dumptype comp-user-index {
comment "Non-root partitions on reasonably fast machines"
compress client fast
index yes
priority medium
}
```

2 You need to define disks that you want to generate an index for to

be of one of the disktypes you defined which contain the index option. This cause sendbackup-dump on the client machine to generate an index file which is stored local to the client, for later recovery by **amgetidx** (which is called by **amdump**).

3 *Amanda* saves all the index files under a directory specified by

"indexdir" in `amanda.conf`. You need to create this directory by hand. It needs to have read/write permissions set for the user you defined to run *Amanda*.

If you are using the "text database" option you may set indexdir and infofile to be the same directory.

4 The index browser, **amrecover**, currently gets installed as part of the client software. Its location may not be appropriate for your system and you may need to move it to a more accessible place such as `/usr/local/bin`. See its man page for how to use it.

Note that **amindexd**, **amgetidx**, **amidxtaped**, and **amtrmidx** all write debug files on the server in `/tmp` (unless this feature is disabled in the source code), which are useful for diagnosing problems. **amrecover** writes a debug file in `/tmp` on the machine it is invoked.

4.6 Permissions

The userid chosen to run the *Amanda* client code must have permission to run **restore** since this is used by createindex-dump to generate the index files.

For a user to be able to restore files from within **amrecover**, that user must have permission to run **restore**.

4.7 Changes from amindex-1.0

Get index directory from `amanda.conf`.

Integration into *Amanda*-2.3.0.4.

Rewriting of **amgetidx** to use **amandad** instead of using rsh/rcp.

4.8 Changes from amindex-0.3

Support for index generation using **GNU-tar**.

Support for restoring files from within **amrecover**.

Bug fixes:

- index/client/amrecover.c (guess_disk): Removed inclusion of mntent.h and use of MAXMNTSTR since this was non-portable, as pointed out by Izzy Ergas erga00@nbhd.org <mailto:erga00@nbhd.org>.
- index/client/display_commands.c (list_directory): Removed point where list_directory() could sleep for ever waiting for input that wasn't going to come.
- index/server/amindexd.c index/client/uscan.l Installed patches from Les Gondor les@trigraph.on.ca <mailto:les@trigraph.on.ca> to make **amrecover** handle spaces in file names.
- server-src/amcontrol.sh: As pointed out by Neal Becker neal@ctd.comsat.com <mailto:neal@ctd.comsat.com> there were still a few sh-style comments that needed conversion to c-style.

4.9 Changes from amindex-0.2

- index/client/Makefile.in
- index/client/help.c
- index/client/amrecover.h
- index/client/uparse.y
- index/client/uscan.l Added a help command.
- index/client/set_commands.c: set_disk() and set_host() now check for empty extract list.
- index/client/extract_list.c:
- index/client/amrecover.h:
- index/client/uparse.y:
- index/client/uscan.l: Added clear extract list command.
- index/client/set_commands.c (set_disk): Added code so working directory set to mount point.
- index/client/extract_list.c: If the last item on a tape list is deleted, the tape list itself is now deleted from the extract list.
- index/client/amrecover.c:
- index/server/amindex.c: If the server started up and found that the index dir doesn't exist, then it exited immediately and the client got informative message. Corrected

this so it is obvious what is wrong to the user, since this is most likely to occur when somebody is setting up for the first time and needs all the help they can get.

- server-src/amgetidx.c Added patch from Pete Geenhuizen pete@gasbuggy.rockledge.fl.us <mailto:pete@gasbuggy.rockledge.fl.us> so that it works even when remote shell is csh.
- server-src/amcontrol.sh
- server-src/Makefile.in Amcontrol is now parameterized like other scripts and run through munge to generate installable version.
- index/server/amindexd.c (main): Added code to set userid if FORCE_USERID set.
- index/server/amindexd.c Removed #define for full path of grep. Assumed now to be on path.
- client-src/createindex-dump.c
- client-src/sendbackup-dump.c
- man/Makefile.in Added patch from Philippe Charnier charnier@lirmm.fr <mailto:charnier@lirmm.fr> so they work when things are installed with version numbers. This was also reported by Neal Becker neal@ctd.comsat.com <mailto:neal@ctd.comsat.com>. Also patch to set installed man page modes and create directory if needed.
- config/options.h-sunos4 Corrected definition for flex library.
- server-src/amtrmidx.c Added some pclose() commands, used remove() instead of system("rm .."). Problems reported by Pete Geenhuizen (pete@gasbuggy.rockledge.fl.us <mailto:pete@gasbuggy.rockledge.fl.us>) on a system with small ulimits set.
- index/server/amindexd.[ch]
- index/server/list_dir.c
- index/client/amrecover.c
- index/client/set_commands.c
- index/client/uparse.y Changes developed with the help of Pete Geenhuizen pete@gasbuggy.rockledge.fl.us <mailto:pete@gasbuggy.rockledge.fl.us> to support disks specified by logical names. Also, now debug files generated by amrecover include PID so multiple users can use amrecover simultaneously and without file deletion permission problems.
- config/config.h-hpux:
- config/config-common.h:
- server-src/amgetidx.c: Changes from Neal Becker re remote shell, making it a configuration parameter.
- config/options.h-sunos4 Had -Lfl instead of -lfl

4.10 Changes from amindex-0.1

- index/client/uscan.l: added support for abbreviated date specs
- index/client/amrecover.c (guess_disk): guess_disk got disk_path wrong if mount point other than / (as subsequently pointed out by Eir Doutreleau ed@cti.ecp.fr <mailto:ed@cti.ecp.fr>)
- server-src/amtrmidx: Added amtrmidx which removes old index files.
- index/client: Added a pwd command
- server-src/amgetidx.c (main): Added use of CLIENT_LOGIN username on r commands. (as pointed out by Eric Payan Eric.Payan@ufrima.imag.fr <mailto:Eric.Payan@ufrima.imag.fr>)
- server-src/amgetidx.c: Bug: It was copying from all clients irrespective of whether the client was configured for indices. A '}' in the wrong place.
- server-src/amgetidx.c: Removed user configuration section. Instead include amindexd.h to get information.

4.11 Changes/additions to 2.3.0

common-src/conffile.[ch]

- added "index" as a valid option

server-src/driverio.c

- added code to optionstr() to write "index" into option string

client-src/sendback-dump.c

- added code to generate index if requested.

client-src/indexfilename.[ch] client-src/createindex-dump.c

- code to generate index.

client-src/Makefile.in

- a new target. Another file for sendbackup-dump

config/config-common.h

- added def of restore.

4.12 Known Bugs

- Empty directories don't get into the listing for a dump (at all dump levels).
- When **amrecover** starts up, it tries to guess the disk and mount point from the current directory of the working system. This doesn't work for disks specified by logical names, nor when an automounter is being used, or a link is in the path.

NOTE

Refer to <http://www.amanda.org/docs/indexing.html> for the current version of this document.

Chapter 5

BACKUP PC HOSTS USING SAMBA

5.1 Installation

Amanda is able to back up Microsoft Windows shared disks by using Samba, a package that implements a SMB client and server for Unix:

<http://www.samba.org>

NOTE

This is *old* stuff and will be (re)moved soon:

Releases from 1.9.18p5 up to 1.9.18p10 logged information in the tar files produced, making them unusable! If you really must use a release prior to Samba 2.0.6, a patch that fixes the problem is available in the *Amanda* patches page:

<http://www.amanda.org/patches/>

Amanda no longer supports Samba releases prior to 1.9.18. If you're using Samba from 1.9.18 through 1.9.18p3, make sure you don't set a low logging/debugging level in `smb.conf`. This flag may prevent estimate sizes from printing correctly and *Amanda* will report an estimate failure.

This problem may also occur if you have large (>2GB) shares with Samba prior to 2.0.4. In this case, apply samba2-largefs.patch from the *Amanda* patches page (<http://www.amanda.org/patches/>).

After building and installing Samba, *Amanda* must be configured with support for smbclient. *Amanda* will automatically find smbclient if it is in your PATH when you run **configure**, or you may add the following argument:

```
--with-smbclient=/full/path/to/smbclient
```

5.2 Setup

Once *Amanda* and Samba are installed, the only difference between a Unix client/disk and PC client/share is in how the backup disks are specified in the file `disklist`. For each PC share, the entry lists the 'samba server' host (where the patched Samba software is installed) and the disk field is the share name. The remaining fields are like any other DLE.

A user must be created on the PC with full access rights (read/write) to the share. *Amanda*, via the Samba server, will connect to the PC via this user. If the user does not have full access, incremental backups will not work and the whole share will be backed up every time (the archive bits are never reset).

The file `/etc/amandapass` must be created by hand. It contains share name to user name, password and workgroup mapping. Each line consists of two or three fields, separated by whitespace:

- Share name followed by optional directory.

  You have to use forward slashes (/), not backslashes (\). This must match the disklist entry exactly (case sensitive). This may be asterisk (*) to match all remaining shares for this Samba server. The first match in the file is used, so specific entries must be listed first. The directory is appended to the share name as full MS network path. Like `//thepc/c$/mydir`. No blanks are allowed in directory!

- User name and password.

  Separated by a percent sign (%). See the description of the -U option in the manpage of smbclient. No whitespace is allowed in either the user name or password.

- Workgroup (optional).

This file must be owned by the *Amanda*-user, and disallow world access privileges. Blank lines are ignored. A "#" on a line at the start of a field (including start of line) causes the rest of the line to be ignored.

5.3 Example

The *Amanda* client software and (patched) Samba is installed on host "pcserver". A share to be backed up called "backupc" is on PC "thepc". The share will be accessed via PC user "bozo" and password "f00bar" and does not require a workgroup.

The entry in the file `disklist` is:

```
pcserver //thepc/backupc      nocomp-user-gnutar

^ samba installed unix host
                ^ pc host and share name
               ^ dumptype must include the tar option
```

In `/etc/amandapass` on the machine 'pcserver':

```
//thepc/backupc       bozo%f00bar
```

If smbclient requires a workgroup specification (-W), you may add it as a third argument in the line in the file `/etc/amandapass` :

```
//thepc/backupc       bozo%f00bar NTGROUP
```

This will cause smbclient to be invoked with -W NTGROUP.

An example dumptype in `amanda.conf` would be:

```
define dumptype nocomp-user-gnutar {
    program "GNUTAR"
    comment "user partitions dumped with tar and no compression"
    options no-compress
    priority medium
}
```

Essentially, the entry in `disklist` is a 'pseudo-disk' which contains all the relevant information needed by smbclient to backup the disk, but in a way that is compatible to *Amanda*.

amcheck does a quick check to see if smbclient exists and tries to connect to the PC clients. It also checks for the existence and permissions of `/etc/amandapass`.

5.4 Bugs and notes

Samba will not back up open files (such as PAGEFILE.SYS and registry files) nor Access Control List data. Furthermore, at restore time, smbclient is unable to overwrite read-only files. Hence, *Amanda*+Samba is not a perfect solution for backing up (restoring, actually) system disks.

Samba does not use the Windows Backup API, so configuring the *Amanda* backup user as a member of group backup on the Windows host is useless. You will probably have to configure it as an Administrator, and make sure it can read and change permission of *all* files in the share.

It seems impossible to detect when a per-user based login fails, e.g. the username doesn't have sufficient access. It connects but cannot see any files (e.g. backups do nothing). The selfcheck code isn't particularly robust in this area either, so you may get no warnings when a disk isn't being backed up. Just check to see that level 0 dumps are bigger than 64K, otherwise it means the backup was empty.

The estimate and totals are probably a bit off since tar pads to the nearest 512 bytes after each file (I think). Not sure how much of a problem this is.

smbclient only supports excluding a single file from the command line, not a file of patterns like GNU tar. So "exclude" is supported from a dumptype but not "exclude list".

NOTE

Also the new option "exclude append" is not yet supported with smbclient.

NOTE

Since Samba-3.0.2a smbclient supports multiple exclusion-patterns. It is one of the "Ongoing Projects" to make use of this in *Amanda*. Refer to <`http://www.amanda.org/ongoing.php`> for details.

The size estimate calculation does not use the same method as the dump, so it may be inaccurate. It also does not support any type of exclusion ("exclude" is ignored). Things are done this way because doing a simulated dump to `/dev/null`, like other dump programs, would take forever with current implementations of Samba.

If you compile with support for smbclient, **GNU-tar** support is automatically enabled. If you aren't using the **GNU-tar** part, you may get warnings about the availability of `/usr/local/bin/gtar` (or whatever it was compiled with). These may safely be ignored, unless you enable index generation for those filesystems.

NOTE

Refer to <`http://www.amanda.org/docs/samba.html`> for the current version of this document.

Chapter 6

RESTORE

This document describes how to restore files backed up with *Amanda* either with or without *Amanda* tools.

All these cases assume you're trying to restore a complete disk, that is, you've replaced the lost disk with a new one, or created a new filesystem on it. Tweaking with the arguments to restore (not **amrestore**), you will be able to restore individual files.

Also, this text does not cover **amrecover**, a program that provides a text user interface similar to interactive **restore** (**restore** -i), but it allows you to select individual files to recover and automatically determines the tapes where they were stored. The backups must be performed with the 'index' option enabled for this to work.

I considered the following cases.

The server machine (machine Aaron) runs solaris, the client machine (machine Barney) runs sunos.

1 Client machine fails, non-system critical.

Example: /home fails on Barney.

First, use **amadmin** to find the tapes most recently used to backup the partition.

```
amadmin <config> info Barney '/home$'

Current info for Barney /home:
  Stats: dump rates (kps), Full:   41.1,  33.1,  38.8
                     Incremental:    9.5,   2.1,  24.7
           compressed size, Full:  63.1%, 54.0%, 52.9%
                     Incremental:  43.7%, 15.5%, 47.8%
  Dumps: lev datestmp  tape             file   origK   compK secs
           0  19971223  Barney01           16  329947  208032 5060
           1  19980108  Barney16            8    1977     864   91
           2  19971222  Barney06            7    1874     672   83
           3  19970926  Barney03           11   12273    3040  211
```

This tells us that we will need two tapes to do a full restore (Barney01, Barney16). Note that, even if Barney06 and Barney03 are listed, they are actually older than the full backup, so they should not be used to restore any data.

Log into Barney. Cd to the /home directory. Insert the tape with the level 0 dump on it into the tape drive of Aaron.

Become super-user in the client host and run (replace <amanda> with the username under which amanda runs):

```
rsh -n -l <amanda> Aaron amrestore -p /dev/rmt/0cn Barney '/home\$' |
restore -ivbf 2 -
```

This requires client root to have login access to <amanda>@Aaron, with a `.rhosts` entry (`.amandahosts` won't do). If you use **ssh**, you may be able to type a password in order to be authenticated. Another alternative is to start the operation in the server, and **rsh** to the client. You should be the amanda user or root in the tape server and run:

```
amrestore -p /dev/rmt/0cn Barney '/home$' |
rsh Barney -l root /usr/etc/restore -ivbf 2 -
```

If you don't want to use **rsh** at all, you may run:

```
amrestore /dev/rmt/0cn Barney '/home$'
```

This should create a file whose name contains the hostname, directory name, dump level and dump date of the backup. Now you have to move this file to the client somehow: you may use NFS, rcp, ftp, floppy disks :-), whatever. Suppose you rename that file to 'home.0'. Then, on the client, you should become root and run:

```
restore -ivbf 2 home.0
```

Repeat one of these steps, incrementing the level of the dump, until there are no more available backups.

2 Client machine fails, system critical disk.

Example: / fails on Barney.

First of all, boot off the CD, and reinstall the system critical partition, restoring it to vendor supplied state. Then, go through all of Scenario 1.

3 Server machine fails, non-system critical, non-*Amanda* disk.

Proceed just as described in Scenario 1. However, you won't have to go through the **rsh** process, because you can just use **amrestore** to replace the lost data directly.

4 Server machine fails, system critical, non-*Amanda* disk.

Example: / on Aaron

First of all, boot off the CD, and reinstall the system critical partition, restoring it to vendor supplied state.

Then, follow steps in Scenario 3.

5 Server machine fails, non-system critical, *Amanda* disk, with db.

Example: /opt on Aaron

If the disk that contains the *Amanda* database is toast, then you need to rebuild the database. The easiest way to do it is to take the text file that you had mailed to you via the 'amadmin export' command, and import via the 'amadmin import' command. Then you should be able to follow the steps outlined in Scenario 4.

Note that *Amanda* does not mail the exported database automatically; you may add this to the crontab entry that runs amanda.

Maybe it's a good idea to print out the text files as well and store the last 2 dumpcycles worth of paper (the disc text files might have got toasted as well). From the paper you still are able to reconstruct where your discs are.

6 Server machine fails, non-system critical, *Amanda* disk, with binaries.

Example: /usr/local on Aaron

This is where things get hairy. If the disk with the amanda binaries on it is toast, you have three options.

i reinstall the *Amanda* binaries from another tape, on which you have conveniently backed up the binaries within the last couple of weeks (not using *Amanda*).

ii recompile *Amanda*, making sure not to overwrite your db files.

iii use **dd** to read *Amanda* formatted tapes. This is the option I am going to explore most fully, because this seems the most likely to occur.

a Find out the device name used by *Amanda*, by looking in `amanda.conf`. Turns out to be `/dev/rmt/0cn` for this system.

If `amanda.conf` isn't at hand: this must be a non-rewinding tape device specifier (which I believe the trailing 'n' stands for).

b Look over the copy of the output of 'amadmin <config> export', and find out which tapes /usr/local was backed up on.

c Grab the tapes that /opt was backed up on, and stick the level 0 into the drive. cd to /usr/local.

d Skip the first record, which is just the tape header, by using the appropriate tape command.

```
mt -f /dev/rmt/0cn fsf 1
```

e Now you want to start looking for /usr/local on this tape.

```
dd if=/dev/rmt/0cn bs=32k skip=1 | gzip -d | /usr/sbin/ufsrestore -ivf -
```

This command gives us an interactive restore of this record, including telling us what partition, what host, and what level the backup was. The gzip -d portion of the pipe can be omitted if there was no compression.

f If you don't find /usr/local on the first try, quit ufsrestore, and move forward one record.

```
mt -f /dev/rmt/0cn fsf 1
```

and try the **dd**/**restore** command shown above. Do this until you find /opt on the disk.

Another possibility: quick and dirty tape index in case you don't know which partition /usr/local was on: (from ralf@atg.venture.de <mailto:ralf@atg.venture.de>)

```
#!/bin/sh
TAPEDEV=/dev/nrtape
while mt -f $TAPEDEV fsf 1 ; do
  dd if=$TAPEDEV bs=32k count=1 | head -1
  sleep 1
done
```

Example output:

```
Amanda: FILE 19971220 uri /root-sun4 lev 1 comp .gz program DUMP
Amanda: FILE 19971220 uri /misc lev 1 comp .gz program DUMP
Amanda: FILE 19971220 uri / lev 1 comp .gz program DUMP
```

g Restore the *Amanda* binaries (what else do you need??), and then bail out of ufsrestore. You can use **amrestore**, as in Scenario 3.

NOTE

Refer to <http://www.amanda.org/docs/restore.html> for the current version of this document.

Part II

About Tapes and Changers

TAPE-DRIVES AND TAPE-CHANGERS

Amanda is able to use a wide range of Tape-Drives and Tape-Changers. This section contains information on the concept of the tapetypes as well as information on how to make use of your tape-changer by using the appropriate changer-scripts.

Chapter 7

TAPETYPES

You may find tapetype-definitions in the example `amanda.conf`, in the mailinglist-archives of the amanda-users-mailinglist at <`http://marc.theaimsgroup.com/?l=amanda-users`> or in the *Amanda*-FAQ-O-Matic at <`http://www.amanda.org/fom-serve/cache/1.html`>. They inform *Amanda* how much it is supposed to be able to store in a tape (length), how much space is wasted at the end of a dump image with the EOF mark (filemark) and how fast the tape unit is (speed).

The most inportant parameter is length, since *Amanda* may decide to delay a backup if length is too small, but, if it is too large, *Amanda* may end up leaving dumps in the holding disk or having to abort some dump.

Filemark is important if you have many disks, particularly with small incremental backups. The space wasted by so many filemarks may add up and considerably modify the available tape space.

The speed is currently unused.

If none of the sample tapetype entries match your needs, you may search the mailing list archives or look up the on-line list of tapetype entries. Just follow the links from <`http://www.amanda.org`>.

Amanda provides the **amtapetype** utility to calculate the size of a tape, to generate a "tapetype" entry for your `amanda.conf`.

Specifying the appropriate tape device, but beware that it may take many hours to run (it fills the tape twice ...). Make sure you do not use hardware compression, even if you plan to use hardware compression in the future. **amtapetype** writes random data to tape, and random data will expand instead of compressing, therefore you'll get an estimate that's smaller than expected.

NOTE

Please note that **amtapetype** in releases up to snapshot 20040915 expects to be given a tape that can be overwritten without causing harm.

It does NOT warn you if the tape contains a label or data. **amtapetype** will OVERWRITE the tape so be cautious. You have been warned.

Snapshot 20040915 introduces a warning and the flag -o to overwrite a labelled tape.

NOTE

Refer to <`http://www.amanda.org/docs/tapetypes.html`> for the current version of this document.

Chapter 8

AMANDA TAPE CHANGER SUPPORT

8.1 Introduction

This document outlines the tape changer support design of *Amanda* 2.2 and beyond. It defines a small interface for changer software to follow so that *Amanda* can remain device-independent but still support the widest range of tape software and hardware possible.

The interface is a bit simplistic and has only had complications added when there is a body of field experience.

8.2 Specifying a tape changer in amanda.conf

All device-specifics are hidden by a glue program that the rest of *Amanda* calls to interact with the tape changer.

The name of this changer program is given by the "tpchanger" variable in the file `amanda.conf`. Example entry:

```
tpchanger "chg-multi"      # use multi-unit tape changer emulator
```

The tapedev parameter is ignored if a tpchanger program is specified, unless the changer program itself reads tapedev from `amanda.conf`. The chg-multi changer doesn't, as it reads all its configuration arguments from its own configuration file, specified as changerfile.

If the tpchanger program does not begin with a '/', *Amanda* expects it to reside in libexecdir, and possibly have the version suffix appended depending on how *Amanda* was configured.

Two other `amanda.conf` parameters are available for changer use, however their definition is left entirely up to the changer script itself. They are changerfile and changerdev. Typically changerfile will point to the configuration file specific to the changer, or be a base name of several related changer files. The changerdev variable may point to the /dev entry used to access the changer device.

See the documentation with the specific changer you're interested in to see exaclty how these variables are used (if at all).

8.3 Device-specific tapechanger script

The tape changer script/program is always from the directory with `amanda.conf`. It is never passed the configuration name it is running on behalf of, but since **amgetconf** works without a name from the current directory, that should be sufficient.

The script/program must support the following commands:

- <tpchanger> -slot <slot-specifier>

If changer is loaded, unloads the current slot (if different than "slot-specifier") and puts that tape away, then loads the requested slot. See the next section for the list of valid slot requests.

Outputs to stdout the slot name and name of the device file to access the tape on success, or a slot name and error text.

Returns 0 on success.

Returns 1 on positioning error (eg at bottom of gravity stacker or slot empty).

Returns 2 any other fatal error. The slot name may be invalid, but must be present. Error message goes to stdout in place of device name.

Examples:

```
% chg-multi -slot 0
0 /dev/nrst8         # exitcode returned is 0

% chg-multi -slot 1
1 slot 1 is empty    # exitcode returned is 1

% chg-multi -slot bogus-slot
<none> no slot 'bogus-slot'   # exitcode returned is 2
```

- <tpchanger> -info

Outputs to stdout three or more fields:

The current slot string (required)

The number of slots (required)

Flag indicating whether the changer can go backwards

(0 if it can't, 1 if it can). (required)

Flag indicating whether the changer is searchable

(optional). Shows whether the changer supports the -search and -label commands and is able to load a tape given only the *Amanda* label string (0 or omitted if it can't, 1 if it can). (optional)

Examples:

```
% chg-multi -info
0 10 1            # exitcode returned is 0

% chg-zd-mtx -info
0 10 1 1
```

- <tpchanger> -reset

Resets the changer to known state and loads the first slot.

Output and error handling are the same as

"<tpchanger> -slot first"

In the case of a gravity stacker that must be reset by hand, this could be run (via " **amtape** <conf> reset") to inform the software the stacker is positioned back at the top.

Examples:

```
% chg-multi -reset
0 /dev/nrst8         # exitcode returned is 0

% chg-multi -reset
0 slot 0 is empty    # exitcode returned is 1

% chg-multi -reset
0 tape-changer not responding # exitcode returned is 2
```

- <tpchanger> -eject

Unloads the current slot (if loaded) and puts that tape away.

Output and error handling are the same as the -slot command.

Note that a tape may or may not be loaded when this command completes, depending on the hardware.

Examples:

```
% chg-multi -eject
0 /dev/nrst8         # exitcode returned is 0

% chg-multi -eject
0 drive was not loaded     # exitcode returned is 1
```

The tape changer program MAY support the following commands:

- <tpchanger> -search <labelstr>

Loads an *Amanda* tape by name (labelstr).

Output and error handling are the same as the -slot command.

taper, **amcheck** and **amtape** will use this command if the changer reports it is searchable.

Example:

```
% chg-zd-mtx -search DailySet005
5 /dev/nrst8         # exitcode returned is 0
```

-<tpchanger> -label <labelstr> Associates the *Amanda* label <labelstr> with the barcode of the currently loaded (in the tape drive) tape.

Outputs to stdout the current slot and tape device. **amlabel** will use this command if your changer is searchable to build up the barcode database.

Example:

```
% chg-zd-mtx -label DailySet006
6 /dev/nrst8         # exitcode returned is 0
```

For all the commands:

An exit code of 0 implies that the operation was completely successful, and the output may be parsed by the *Amanda* code as described above.

For non-zero exit codes, the first field is still the slot name, but the actual error messages are not fixed. They are just displayed and/or logged as-is by the calling *Amanda* program.

An exit code of 1 implies the operation failed in a benign way, for example an empty slot or an attempt to go backwards in a gravity stacker. The calling *Amanda* program will print the error message if appropriate and continue, perhaps requesting a different slot be loaded.

Any other exit code is considered fatal and will cause *Amanda* to stop attempting to talk to the tape changer.

8.4 Slot names and the "current" slot

Some tape changers, such as carousels and gravity stackers, have a hardware notion of current position. Others have no current position when no tape is loaded: all tapes are in their slots and the changer arm is docked away from the slots.

Nevertheless, *Amanda* requires tape-changer scripts to maintain the notion of a "current" position. This is for performance reasons: as tapes tend to be loaded into the rack in order, and *Amanda* uses them in order, the next tape to use can be found much quicker if the position of the current one is remembered. As an example, the chg-multi script maintains the current position in a `chg-multi.state` file (or any other file specified in a 'statefile' line in the changer configuration file).

Amanda does not care how slots are available or how they are named. They could be numbered 0 to N-1, numbered 1 to N, or even designated by letter, A .. Z. The only requirement is that the names do not contain whitespace and that the names "current", "next", "prev", "first", "last" and "advance" retain their meaning as follows:

current The position of the last loaded tape, as described above

next The position after current, wrapping from the last slot to the first.

prev The position before current, wrapping from the first slot to the last.

first The first slot in the tape rack.

last The last slot in the tape rack.

advance The same as "next" except the next tape may not be loaded if the changer supports advancing to the next slot without putting that tape in the drive.

The current position must be updated even if there is a positioning error (such as "empty slot"). This allows amanda to step through the entire tape rack by issuing successive "slot next" positioning commands.

8.5 Operator interface

The **amtape** program is the main operator interface to *Amanda*'s tape changer support. The commands supported include:

amtape <conf> slot <slot-specifier> Load the tape from the specified slot into the drive

amtape <conf> eject Send an eject command to the tape-changer. Effect is changer specific.

amtape <conf> reset Send a reset command to the tape-changer. Effect is changer specific.

amtape <conf> show Go through the entire tape rack, showing the labels of all amanda tapes encountered.

amtape <conf> label <label> Find and load the tape with the specified label

amtape <conf> taper Perform taper's scan algorithm (see below), loading the tape which would be picked for the next amdump run.

amtape <conf> clean If a cleaning tape is defined by the changer, load it in the drive and put it away when done.

amtape <conf> device Output the current tape device name.

amtape <conf> current Display the contents of the current slot.

amtape <conf> update Scan the entire tape rack and update the barcode database.

See the **amtape**(8) man page for more details.

In addition to **amtape**, **amlabel** has been modified to allow optionally specifying a slot:

amlabel <conf> <label> [slot <slot-specifier>]

amcheck looks for the next tape in the rack the same way the **taper** does. If multiple tapes are used in one night, **amcheck** attempts to find all the needed tapes in turn if the tape-changer is random access. On a one-way gravity stacker, **amcheck** only finds the first tape, since finding the subsequent ones would put the first one out of reach of that night's **amdump** run.

amrestore and **amrecover** do not yet include any tape changer support directly, as **amrestore** knows nothing about the `amanda.conf` files or server-side databases. This is a

deliberate decision to keep **amrestore** independent, so it can be run from any host with a tape drive, even if the original tape server host is down. To use **amrestore** in a tape-changer environment, use **amtape** to find the right tape, then run **amrestore** giving the resulting tape device name.

8.6 How amdump interacts with the tape changer

Amanda does not require a particular tape label for a run. Any label that matches the labelstr regex and is determined to be "inactive" according to the tapelist database, may be written to. However, there is a preferred 'next' tape, the one that is next in the cycle implied by the current tapelist.

amdump uses two algorithms, depending on whether the tape changer can go backwards in the rack or not. If multiple tapes are needed in a single run, this algorithm is repeated in turn whenever a new tape is required.

Normal tape changers (those that can go backwards):

With a full-access tape changer, **amdump** searches the entire rack for the preferred tape label. This tape will usually be found at the current or next position, but might be located anywhere. If the tape is found, it is used. If it is not found, the first tape encountered that matches the labelstr and is not active is picked.

Gravity stackers (anything that can not go backwards):

To avoid going all the way to the bottom of the stacker only to find that the preferred tape isn't present and nothing can be done, *Amanda* picks the first tape (starting at the current position) that matches the labelstr and is not active, regardless of whether it is the preferred tape.

8.7 Builtin tape-changers

8.7.1 chg-multi (formerly chg-generic)

This tape changer script supports several common configurations:

- Using multiple tape drives in a single host to emulate a tape changer. This can also be used with a single physical drive to write several tapes in an *Amanda* run.
- Using a gravity stacker or a real changer configured to sequentially load the next tape when the current one is ejected. Also supports a changer which cycles to the first tape after loading the last one.
- Using a changer accessed through several "virtual" tape devices which determine which slot actually gets loaded in the tape drive.

The advantage of this changer script is that you do not need to get into the complexity of dealing with a real changer interface. All the action goes through the tape device interface with standard **mt** commands, which eases many portability issues. Many common tape jukeboxes can be configured in a sequential or cycle mode.

chg-multi ignores 'tapedev' and 'changerdev' because 'changerfile' may specify several tape devices to be used. A sample configuration file may be found in `example/chg-multi.conf`.

8.7.2 chg-manual (formerly no-changer)

This is a poor man's tape changer that requires the backup operator to change tapes manually. It expects 'tapedev' in `amanda.conf` to point to a valid tape device, and stores some status data in files whose names start with the 'changerfile'. 'changerdev' is ignored.

8.7.3 chg-mtx (formerly hp-changer)

An mtx-based tape changer script. 'changerdev' must specify the tape device controlled by the **mtx** program, and 'tapedev' must point to the no-rewind tape device to be used. More than likely, both 'changerdev' and 'tapedev' will reference the same device file. 'changerfile' must specify a prefix for status files maintained by this script. It will maintain files named 'changerfile'/changer-clean and 'changerfile'/changer-access. You may have to edit the script to specify which slot contains a cleaning tape (cleanslot).

The **mtx** program must support commands such as '-s', '-l' and '-u'. If the one you've got requires 'status', 'load' and 'unload', you should use chg-zd-mtx instead (see below).

8.7.4 chg-zd-mtx

Based on chg-mtx, but modified in order to support the Zubkoff/Dandelion version of **mtx**. Eric DOUTRELEAU <Eric.Doutreleau@int-evry.fr>, who contributed this script, reported that it works on a Solaris/sparc box with a HP 1557A stacker.

In addition to the 'changerfile'-clean and the 'changerfile'-access files, it maintains a 'changerfile'-slot file that indicates the currently loaded slot.

There are lots of comments at the start of the script describing how to set it up.

8.7.5 chg-scsi-chio (formerly seagate-changer, then chg-chio)

A C program that relies on scsi tape-changer interfaces. It may either use the tape changer interface specified in chio.h (Gerd Knor's SCSI media changer driver, a Linux kernel loadable module), or it may use built-in tape changer interfaces available on HPUX, Solaris 2.5, IRIX and possibly others, but only the chio and HPUX interfaces are currently implemented . 'tapedev' specifies the tape device to be used; 'changer_dev' is the device used to talk to the kernel module (for chio, usually /dev/ch0), and 'changerfile' specifies a filename in which the current slot number will be stored.

Now there is another way, to get the chg-scsi a little bit more flexible. If you use only one digit in the 'tapedev' parameter, the chg-scsi expects that changerfile points to a real configuration file, not only a counting file. In this configuration file you may specify that the tapedrive needs an eject command and an optional waittime, necessary after inserting the tape into the drive. You are also able to configure a range of slots which should be used by your configuration. The idea behind this is, that you don't want *Amanda* to cycle all the

tapes if *Amanda* searches exactly one tape. If you have a library which supports more than one drive you can also specify which drive to use. For each configuration (there should be at least one) you have to specify a file, where *Amanda* remembers which tape is actually in the drive. For future use there is also some stuff for cleaning the drives.

In `amanda.conf`:

```
tapedev "x"       with x between 0 and 9, selects the configuration to use
changerfile "filename"            specifies the changer configuration file
```

In the changer-config-file the following could be set:

```
number_configs x
#  x between 0 and 9    number of configurations defined. This should be the first par
eject x
#  x 0 or 1    1 means that the drives need an eject command, before the robot can han
sleep x
#  x between 0 and MAX_INT specifies the seconds to wait before the drive could be use
cleanmax x
#  x some positive int  How many cleanings does a cleaning tape survive
changerdev  <device>
#           The device for the robot
```

And then there come some configuration sections, separated by the word ‘config‘ followed by the ordinal of that configuration (0 to 9). In each configuration section you should specify:

```
drivenum x     x between 0 and the number of drives in the library
This one specifies the drive to
use with this configuration
dev   <device>          The device for the tapedrive
startuse    x     x between 0 and maximum slotnumber of your library
Starting here we may use the tapes
enduse    x    x between start and maximum slotnumber
This is the last tape we may use
in this configuration. If we reach
this one the next will be start..
statfile <filename>           Here we remember the last used
slot for this configuration
cleancart   x  x between 0 and maximum slotnumber
In this slot we find the
cleaning tape
cleanfile <filename>          Here we will remember how
often we used the cleaning tape
usagecount <filename>            This points to a file which is
deleted after cleaning the drive
e.g. the usagetime of the drive
```

Comments begin with an '#' until end of line. Allowed separators are TAB and SPACE.

8.7.6 chg-scsi (new interface, try to drive a robot with direct scsi commands)

The config and the syntax is the same as for chg-scsi-chio. New is the config type

emubarcode 1

With this option and the option labelfile chg-scsi will try to create an inventory. With this inventory it should be possible to use the search feature for loading tapes.

debuglevel x:y

This option will set the debug level and select for which part debug messages should be sent to the debug file. In case of problems you should set it to 9:0

havebarcode 1

This will force the program to read the barcodes, and don not try to figure out if there is an barcode reader available.

scsitapedev <devicename>

This device is used to control the tape, read status infos etc.

tapestatus <filename>

If this option is given on every eject/move the log pages of the tape device will be dumped in this file. There are 2 log pages were you can see how many read/write errors (corrected) are processed by the tape

labelfile <filename>

This file is used for the mapping from barcode labels to *Amanda* volume labels. It is used if the changer has a barcode reader. To initialize run **amtape** show, this will add the mapping for the tapes in the magazine.

eject > 1

Use the mtio ioctl to eject the tape, use only if the standard (1) does not work, and send the debug output (`/tmp/amanda/chg-scsi.debug`) to th@ant.han.de

changerident <ident>

With this it is possible to specify which internal driver to use for controlling/error handling of the robot

tapeident <ident>

Some as above but for the tape.

New command line option: -status [all|types|robot|sense|ModeSenseRobot|ModeSenseTape|fd]

<all> will show the result form all options. <types> will list the known driver types. <robot> will show the status of all elements (tape/robot/slots..) <sense> will show the result from a request sense <ModeSenseRobot> will show the sense page from the robot <ModeSenseTape> will show the sense page from the tape <fd> will show the devices which are open, and some info about it.

At the moment changer with tape and robot on the same SCSI id (but on different luns) will run on the following platforms:

- HP_UX 10.20 - IRIX 6.x - Solaris - Linux - AIX - FreeBSD 3.0/4.0

Tape and robot on different IDs run native on - Linux - HP-UX 10.20 - Irix 6.x - FreeBSD

Tape and robot on different IDs with special modules run on: Solaris with sst kernel module, which is not any longer needed in solaris 2.8. See in the contrib/sst directory The configuration on solaris 2.8 with the sgen driver is done by creating the file /kernel/drv/sgen.conf

This file should contain at the beginning the following device-type-config-list="changer","sequential"

This will force the driver to attach only to the devices with type either changer (the robot) and sequential (the tape). Next you must tell the driver on which id it should check for devices (tape on id 5, robot on id 6 in this example),

name="sgen" class="scsi" target=5 lun=0; name="sgen" class="scsi" target=6 lun=0;

This will create the 2 device files /dev/scsi/sequential/c0t5d0 (scsitapedev option in chg-scsi.conf) /dev/scsi/changer/c0t6d0 (changer option in chg-scsi.conf)

So the complete sgen.conf looks like: device-type-config-list="changer","sequential name="sgen" class="scsi" target=5 lun=0; name="sgen" class="scsi" target=6 lun=0;

For HP you have to create the special device files for the pass throu interface. Check if the ctl driver is installed. Example:

```
# lsdev -C ctl
    Character     Block       Driver          Class
      203          -1         sctl            ctl
```

Next check on which bus your drives are connected. (ioscan) with the Character device num form the lsdev and the card instance from ioscan create the special file. Example:

```
mknod /dev/scsi/1 c 203 0x001000
                          ||||
                          ||| LUN of device
                          ||SCSI ID of the device
                          2 digit instance number from ioscan
```

On FreeBSD 4.0 the syntax for the device files has changed. Now you have to tell chg-scsi the bus:target:lun combination. If you for example on your scsi bus 0 target 3 an robot the syntax is changerdev 0:3:0 To get this info you can use the camcontrol command, <camcontrl devlist> will give you a list of known devices. Don't specify dev and scsitapedev in your

chg-scsi.conf !!, this will not work. On Linux you need either sg (generic scsi) as module or it must be compiled into the kernel. If the sg driver doses not work try to use the ioctl interface. For that you have to undef the LINUX_CHG define in changer-src/scsi-linux.c Also you have to change the NORMAL_TIMEOUT in /usr/src/linux/drivers/scsi/scsi_ioctl.c from (10 * HZ) to (5 * 60 * HZ). On linux it does not run if you are using an aha1542 SCSI controller. The driver can not handle the extended request sense.

On IRIX you find the SCSI pass through interfaces for every device in /dev/scsi.

chg-scsi has been tested/run with the following devices: Exabyte 10h and eliant tape HP-Surestore 1200e and C1553A tape BreeceHill Q2.15 (EXB-120) and DLT7000 tape Powerstor L200 and DLT7000 ARCHIVE Python 28849-XXX TANDBERG TDS 1420 ADIC VLS DLT Library

It is now possible with a changer that has barcode reader to load tapes faster. Also **amdump** will find tapes faster. Every time a tape is labeled the information in the labelfile will be updated. To initialize the label mapping you can also do an **amtape** config show. This will put the mapping for all tapes in the magazine into the mapping file.

For all problems please contact th@ant.han.de. Please include in your mail the debug file. (`/tmp/amanda/chg-scsi.debug`)

8.7.7 chg-chio

(new perl script that replaces the original chg-chio written in C) This script is based on the FreeBSD version of chio, a program to communicate with the jukebox. This script has for the moment only been test with FreeBSD and is likely not to work on any other system. Let me know if this is the case and send me the output of the chio program for your version of chio. It does not restrict the number of tapes, except that if there is only one tape in the juke, it is supposed to be in max_slot and not in slot 1. [This is the first version of the changer script and I would appreciate all comments on it, at nick.hibma@jrc.it. It has been tested only with FreeBSD 2.2.5 and the accompanying chio program.]

8.7.8 chg-chs

(formerly chs-changer) A tape changer script very similar to chg-multi, that uses the 'chs' program to change tapes. As in chg-multi, 'tapedev' is ignored. 'changerfile' names its configuration file, similar to chg-multi.conf. 'changerdev' will be passed to CHS in a -f command-line switch, unless it is set to an empty string or "/dev/null" (watch out for default values!)

8.7.9 chg-rth

(formerly rth-changer) A perl5 script that controls an HPc1553 tape drive via a Peripheral Vision Inc. SCSI control subsystem that interprets commands sent on the SCSI bus. It expects 'tapedev' to specify the tape device to be used. 'changerfile' and 'changerdev' are ignored.

8.7.10 chg-juke

A shell script that uses the Fermilab "juke" software (see <`http://www.fnal.gov/fermitools`>, the "juke" link) to control tape chagners. It supports mounting multiple tapes for RAIT tapedrive sets, both multiple jukeboxes, or one jukebox with multiple tape drives, or combinations. 'juke' must be configured to know tape drives by the same name *Amanda* calls them. It uses 'changerfile' to track *Amanda*'s current tape state, 'tapedev' must be the tape drive (or RAIT set) name, and 'changerdev' is the juke software's name for the changer, or a csh-glob that expands to several jukebox names (i.e. "changer{a,b,c}").

8.7.11 chg-rait

A shell script that runs other changers in tandem, and returns a rait:{dev1,dev2,...} tape device based on the results of each other changer. So if you wanted to have 2 stackers striped with no parity, and you have chg-mtx support for your stackers, you would use the follwing changerfile:

```
nchangers=3
tpchanger_1="chg-mtx"
changerdev_1="/dev/mtx1"
changerfile_1="/some/file1"
tapedev_1="/some/dev"
tpchanger_2="chg-mtx"
changerdev_2="/dev/mtx2"
changerfile_2="/some/file2"
tapedev_2="/some/dev"
tpchanger_3="chg-null"
changerdev_3="/dev/null"
changerfile_3="/some/file3"
tapedev_3="/some/dev"
```

The third uses the null changer. The tapedev_n entries are only needed if the changerfile in question uses them.

8.7.12 chg-disk

Clone of the chg-zd-mtx, but modified to be applied on local directories instead of tapes. This changer emulates a robotic that uses virtual tapes instead of real ones, where the virtual tapes are real directories on a hard disk.

The directory tree should be:

```
slot_root_dir -|
               |- info
               |- data -> slot1/
               |- slot1/
```

```
|- slot2/
|- ...
|- slotn/
```

Where "slot_root_dir" is the tapedev "file:xxx" parameter and "n" the tapecycle parameter.

Please refer to How to use the *Amanda* file-driver for details of usage.

8.7.13 chg-iomega

This changer script is designed for IOMEGA or JAZZ disks of various sizes as well as any other removable disk media. This is a PURELY MANUAL changer. It requests insertion of disk media via messages on /dev/tty. So it cannot be used via crontab. Make sure you comply with any of the following. - Add statements

```
tpchanger "chg-iomega"
tapedev "file:<mount_point_of_removable_disk>"
(e.g. tapedev "file:/mnt/iomega" )
tapetype IOMEGA


define tapetype IOMEGA {
    comment "IOMega 250 MB floppys"
    length 250 mbytes
    filemark 100 kbytes
    speed 1 mbytes
}
```

to your `amanda.conf`. - Add entry to `/etc/fstab` to specify mount point of removable disk and make this disk mountable by any user. - Format all disks, add a "data" sub directory and label all disks by using **amlabel**. - Be aware that as of version 2.4.4p1, *Amanda* can't handle backups that are larger than the size of the removable disk media. So make sure `/etc/amanda/<backup_set>/disklist` specifies chunks smaller than the disk size.

8.7.14 chg-null

A trivial changer which loads/unloads on a null: device. Useful with chg-rait to throw away a parity stripe by puttin on a null jukebox, or for testing.

NOTE

Refer to <`http://www.amanda.org/docs/tapechangers.html`> for the current version of this document.

Chapter 9

CHG-SCSI

Jason's new and improved chg-scsi documentation.

9.1 Section

This documentation will also include an occasional reference to the **mtx** suite as I have attempted to use chg-zd-mtx. I use **mtx** often as a fast query tool. Please also refer to *Amanda* Tape Changer Support for additional details.

My equipment list is as follows:

- Redhat 7.0 machine
- – Dual Processor P-III
  - Sony DDS3 tape drive
  - lots of hard disk space
- Quantum/ATL L500 SCSI changer
- DLT tape drive (three possible)
- barcode reader fourteen tape slots

I base this documentation on the following:

- **mtx** version 1.2.16rel
- *Amanda* version 2.4.3b3
- SCSI2 specification: X3T9.2/375R revision 10L
- Quantum/ATL Field Service Manual 6321101-03 Ver.3, rel.0
- Quantum DLT800 Tape system product manual 02 April, 2001 81-60118-04
- the SCSI bus and IDE interface 2nd ed by Friedhelm Schmidt pub: Addison-Wesley 1998

Note that Quantum/ATL's L-series changers follow the SCSI command set, and do not use any proprietary commands. Thus, it was fairly simple to make this work.

I had to install *Amanda* –with-suffixes and setup my server's client side of things using *Amanda*-2.4.2p2 –with-suffixes.

Please note that my usage of "barcode" and "barcode reader" throughout this document really refers to "physical tape identification system". for example: the EEPROM in the AIT cartridge.

9.2 Command line options

chg-scsi's command line options:

```
-slot <param>
-info
-reset
-eject
-clean
-label <param>
-search <param>
-status <param>
-trace <param>
-inventory
-dumpdb
-scan
-genconf
```

Note that chg-scsi is called by *Amanda* in the context of whatever *Amanda* configuration *Amanda* is currently using. In short, to call chg-scsi by hand, change to your *Amanda* configuration directory, then run chg-scsi.

-slot <param> command:

this command takes either a slot number, or any one of the following: current, next, prev, first, last, advance

It then loads the appropriate tape into the drive.

Note that if the tape is already loaded, no changer motion will happen. No reading of the tape is done here.

-info command:

four numbers are printed: <slot# loaded> <number of slots> <reversable> <searchable>

with chg-scsi, the reversable number is always 1. Searchable is set based on if a barcode reader is available and working correctly.

-reset command:

Tape is unloaded, and slot 0 is loaded. No actual reset command is sent to the changer.

-eject command:

Tape is unloaded, and put back into its original slot.

-clean command:

Cleaning tape (if configured) is loaded into the tape. It is probably unloaded when done. I haven't looked closely at this yet.

-label <param> command:

This appears to take the currently loaded tape's barcode and write it to the labelfile with the given parameter as it's tape header. I have not tested this.

-search <param> command:

this only should be used if a barcode reader is present, or emulate barcode is turned on.

the required parameter is an *Amanda* tape label. The label searched in the labelfile. If a barcode is found, then that tape is loaded directly.

I believe the fallback is to search the entire magazine.

-status command:

The tape changer is queried, and the results are printed out. Values printed are what slots exist, are they full or empty, and what volume labels (barcodes) they are.

Q. what about changers that don't retain current slot? A. this is what the "statfile" is for.

-trace command:

present only for a particular type of changer.

-inventory: (this takes a LONG time to do)

unloads tape back to its slot issues command to changer to do an inventory of itself (read all barcodes...)

loads each tape, retrieves the barcode, and reads the *Amanda*

label off of the tape itself stores/updates the label database file

-dumpdb:

prints out in human readable form the label database contents from the labelfile.

-scan: (aka scanbus)

scans the SCSI generic interfaces (eg: linux sg), and prints out the device name and the device types. I found that linux didn't classify either of my tape devices as generic, but this facility did.

USE THIS FOR FINDING VALUE OF SCSItapedev. Be certain though you have the correct tape drive: I came close to wreaking havoc with my DDS3 drive while it was flushing *Amanda* data...and my changer has a DLT drive! Please refer to my configuration notes below.

-genconf:

prints out a SAMPLE `changer.conf` file. Note that I said sample. except for that it also does a scanbus. if you have more than one tape drive, please be certain it is correct as chg-scsi uses the first one it finds.

Please refer to my configuration notes below.

9.3 Notes about changer.conf

Here, I try to be a bit more clear on what these config values mean.

"number_configs" MUST be the first entry (not counting comments)

"emubarcode" will set this value in chg-scsi regardless of the value you try to assign it. If you don't want "emubarcode" set, don't list it!

"havebarcode" have barcode is tempered by what devices chg-scsi is aware of. if chg-scsi doesn't know about your changer explicitly, it will default to a default changer. the default changer has no barcode reader.

In a patch I plan to post, I have added a generic changer that does use a barcode. set "changerident" to "genericbarcode" to use this.

"sleep" number of seconds for chg-scsi to wait for a "tape drive ready" command after loading a new tape. Mine is 140 as I have a DLT tape drive, and my tape drive manual reports a max of 133 seconds to be ready.

"config" this is a header telling chg-scsi that all values following, up to the next "config" line apply to this drive number. It is this number that is refered to in your `amanda.conf` file as line tapedev

"drivenum" this is the tape drive number in your changer. For a single tape drive, this is zero. Mine can handle up to three, so I expect I could make this 0, 1, or 2.

"dev" the SCSI device of your tape drive. under linux, in my case it is `/dev/nst1`. This MUST be defined.

"SCSItapedev" The generic SCSI device of your tape drive. this is simply the generic interface to the device "dev" above. This entry is optional. In my case, it is `/dev/sg2`. If this entry exists and is legitimate, then SCSI commands are formed directly instead of using ioctl commands.

"startuse" "enduse" The start and end slots of your changer. Note that these also start with zero.

"eject" chg-scsi tells the tape changer to eject. You might need to explicitly tell the tape drive to unload first. That's what this setting is for. Have you ever removed a loaded VCR tape by hand?

"changerident" chg-scsi will first read the changer's identification from the return of the SCSI "inquiry" command. If changerident is set in the configuration file, the SCSI inquiry's value is overridden. chg-scsi will attempt to match the value with its built-in changertypes.

"generic" is what the chg-scsi will default to "genericbarcode" is a hack of mine that forces the changer's barcode reader to work. Note that if your changer uses a superset of the SCSI

command set, this probably won't work. "L500" is another addition of mine that will enable a quantum/atl L500 to work correctly. It might even work with an L200.

other values are (taken from the code)

- C1553A (HP Auto Loader)
- EXB-10e, EXB-120 (Exabyte Robot)
- EXB-85058HE-0000 (Exabyte Tape)
- TDS 1420 (Tandberg Robot)
- VLS DLT, VLS SDX (ADIC VLS DLT Library)
- FastStor DLT (ADIC FastStor DLT Library)
- Scalar DLT 448 (ADIC DLT 448)
- 215 (Spectra Logic TreeFrog)
- Quad 7 (Breece Hill)
- DLT7000, DLT4000 (DLT Tape)

"debuglevel" setting this to "9:0" is very informative.

"statfile" stores which "slot" that the tape in the drive came from

"labelfile" binary file that stores the tape header and barcode information

cleanmax cleancart cleanfile I have my changer set to autoclean, and the slot the cleaning cartridge is in is not available for any other use.

9.4 *Amanda*'s actual usage of chg-scsi

this should be brief: *Amanda* really only issues "slot next" type commands. Currently *Amanda* doesn't ask chg-scsi to load "tape x with label Daily_set023". the chg-scsi mechanism is there for use, and functions quite well for the user to load a particular tape. I understand they (the *Amanda* team) are working on this.

9.5 Configuration notes

(assuming one changer with one tape drive!)

in `amanda.conf`:

set tpchanger to chg-scsi set changerfile to <pathname>/changer.conf set tapedev to 0 set changerdev to /dev/<changer generic device>

- this value is usually listed in the system boot messages or will be printed via syslog when the appropriate kernel module is loaded. eg: linux **modprobe sg**

in changer.conf: set number_configs to 1 set dev to <non-rewinding tape device eg: /dev/nst1> set debug to 9:0

run "chg-scsi -scan" from your *Amanda* configuration directory I get: name /dev/sg0 Tape Count 1 name /dev/sg1 Changer Count 2 name /dev/sg2 Tape Count 3

I set SCSItapedev to `/dev/sg0` to test with, then ran **chg-scsi -info**. Check the chg-scsi debug file for tapeidentification details. This is where I discoverd that "sg0" was not the correct tape!

set SCSItapedev to "/dev/<generic device>"

if you have no barcode, try "changerident generic" if you have a barcode reader try "changerident genericbarcode"

9.6 Hacking notes

My hacks are:

- adding printout of "emubarcode" values in the debug file
- added dlt8000 tape drive to ChangerIO[], and SenseType[]
- added genericbarcode to ChangerIO[], and SenseType[]
- added L500 to ChangerIO[], and SenseType[]
- added a couple of basic sensecodes listed in the SCSI specs for the generic and genericbarcode tape changers.

My desires are:

- modify the sensecode code such that all SCSI devices inherit the standard SCSI codes and may override or append to them as needed.
- modify the configuration reading code and the **inquiry** command to allow spaces. eg: my changer displays the following ident data: "L500 6320000" but I had to create a type called "L500" or else the spaces would throw off the comparisons.

NOTE

Refer to <`http://www.amanda.org/docs/chgscsi.html`> for the current version of this document.

Chapter 10

RAIT (REDUNDANT ARRAY OF INEXPENSIVE TAPE) SUPPORT

Currently it is only integrated with the chg-manual changer script; collaboration on integrating it with the other tape changers is needed.

10.1 What is a RAIT?

RAIT is an acronym for "Redundant Array of Inexpensive Tapes", where data is striped over several tape drives, with one drive writing an exclusive-or-sum of the others which can be used for error recovery. Any one of the data streams can be lost, and the data can still be recovered.

This means that a 3-drive RAIT set will write 2 "data" streams and one "parity" stream, and give you twice the capacity, twice the throughput, and the square of the failure rate (i.e. a 1/100 failure rate becomes 1/10,000, since a double-tape failure is required to lose data).

Similarly, a 5-drive RAIT set will give you 4 times the capacity, 4 times the throughput (with sufficient bus bandwidth), and the square of the failure rate.

This means you can back up partitions as large as four times your tape size with *Amanda*, with higher reliability and speed.

10.2 Using a RAIT

If you have several tape devices on your system [currently either 3 or 5 drive sets are supported] you tell *Amanda* to use them as a RAIT by listing them as a single tape device using **/bin/csh** curly-brace-and-comma notation, as in:

```
tapedev = "rait:/dev/rmt/tps0d{4,5,6}n"
```

which means that `/dev/rmt/tps0d4n`, `/dev/rmt/tps0d5n`, and `/dev/rmt/tps0d6n` are to be treated as a RAIT set. You can now mount three tapes, and label them with **amlabel**, etc.

Also, you want to create a new tape-type entry, which lists an n-drive RAIT set, for this RAIT-set. So if you were using an entry like:

```
define tapetype EXB-8500 {
    comment "Exabyte EXB-8500 drive on decent machine"
    length 4200 mbytes
    filemark 48 kbytes
    speed 474 kbytes
}
```

You would want to make a new one like:

```
define tapetype EXB-8500x3 {
    comment "Exabyte EXB-8500 3 drive stripe on decent machine"
    length 8400 mbytes
    filemark 200 kbytes
    speed 948 kbytes
}
```

and change your tapetype entry to:

```
tapetype EXB-8500x3
```

to tell *Amanda* about the multiple drive set.

10.3 Disaster Recovery

To assist in disaster recovery (as well as changer scripts) the *Amanda* package now also includes **amdd**, which is a simple **dd**(1) replacement which supports (only) the "if=xxx", "of=xxx", "bs=nnn[kMb]" "skip=nnn" and "count=nnn" options, but which can read and write RAIT tapesets.

Using **amdd** and your usual *Amanda* unpack instructions will suffice for disaster recovery from RAIT tape-sets.

NOTE

Refer to <http://www.amanda.org/docs/rait.html> for the current version of this document.

Chapter 11

PRINTING OF LABELS

11.1 The New Feature

Amanda now has the ability to print postscript paper tape labels. The labels have what machines, partitions, and the level of the dump the tape has on it. This is achieved by adding the lbl-templ field to the tapetype definition. Since the labels are specific to the type of tape you have, that seemed to most logical place to add it.

You can also specify an alternate "printer" definition to print the label to other than the system default printer.

If you don't add this line to your tapetype definition, *Amanda* works as it always has.

11.2 Labels provided

The author has provided label templates for the following tape types. These are pretty generic labels, and should be easy to customize for other tape types. Others are encouraged to do so.

- Exabyte 8mm tapes
- DDS 4mm tapes
- DLT tapes (in progress).

11.3 History

At the University of Colorado at Boulder, we used to use some dump scripts that printed out paper tape labels that went with the tape. When we started using *Amanda* for our dumps, my boss insisted we still generate them, in case we weren't able to access the *Amanda* database. The thought was that as long as we had an **amrestore** binary on a machine, we could just look at the label, grab the tapes, and do the restore.

As a result of this we have had to hack this feature into every version of *Amanda* from 2.1.1 through 2.4.0-prerelease.

Our hope in adding this feature is that others find it as useful as we have.

11.4 How it works

The majority of the changes are in `reporter.c`. Just as you might run the reporter by itself to see what the report will (or did) look like with a logfile. When the reporter prints out the report, the postscript label template is copied, and the successful machines, partitions, and dump levels are appended to this. The output either goes to `/tmp/reporter.out.ps` (when running in testing mode) or through a pipe to the printer (default printer, if an alternate "printer" is not specified).

NOTE

Refer to <`http://www.amanda.org/docs/labelprinting.html`> for the current version of this document.

Part III

HOWTOs

HOW TO DO THIS?

This section contains some HOWTO-style-documents which should help you to get *Amanda* up and going with Cygwin, AFS or chg-disk ...

Chapter 12

AMANDA ON CYGWIN HOWTO

by Doug Kingston, 30 January 2003. Based on Cygwin 1.3.18, and *Amanda* 2.4.3-20021027 and some fixes which will be in the official release by the time you see this.

With thanks to Enrico Bernardini from whom I have borrowed some material from an earlier attempt at documenting the installation of *Amanda* on Cygwin in 2001. Please send annotations and corrections to <mailto://amanda-hackers@amanda.org>. I can be reached as dpk (at) randomnotes.org (do the obvious).

12.1 Install Cygwin

The following Cygwin packages are required for binary installation (may be incomplete):

- Category BASE: standard
- Category MISC: gzip
- Category MISC: tar
- Category NET: inetutils

You need also these packages to build from source (may be incomplete):

- Category DEVELOP: ALL
- Category INTERPRETERS: m4, gawk ?
- Category LIBS:default selection? (libc, libiconv, others?)

I have most or the basic utilities and libraries installed so I cannot give you a more specific list of what is required. If someone has a more definitive list, I would appreciate and email to <mailto://amanda-hackers@amanda.org>.

One user reported some problems with access rights when running under Cygwin, which he solved by setting the CYGWIN environment variable to nontsec. I do not believe this is necessary if you run the *Amanda* daemon as System (see below).

12.2 Other Preparation

When doing backups on a NT, Windows 2000 or Windows XP system, the choice of user and group will be important if you are to properly interact with the security mechanisms of these more modern Microsoft product. For Windows 95/98/ ME this is probably a non-issue. The most privileged account on the Windows systems is 'System', and I have chosen to use this account for *Amanda* backups to ensure that I can access the widest set of files. On Unix we would run as root, with equivalent access permissions. I have also chose to run under the 'Administrators' group, another standard Windows group. Ensure these exist before you continue - or identify another account to use. The Cygwin installation postinstall script should have already populated `/etc/passwd` and `etc/group` with these entries.

- Make sure that System (or SYSTEM) has a home directory specified in `/etc/passwd`.

I used _/home/root_. You'll need to put the file `.amandahosts` here later. The relevant lines from my file `/etc/passwd` are:

```
SYSTEM:*:18:18:,S-1-5-18:/home/root:
root:*:18:18:,S-1-5-18:/home/root:
```

12.3 Compile *Amanda*

After installing Cygwin, unpack the *Amanda* sources, typically in `/usr/src/Amanda` or something similar. In the *Amanda* directory, you will need to execute:

```
automake  # this may not be necessary in the official release
autoconf  # this may not be necessary in the official release

./configure --without-server \
  --without-force-uid \
  --with-user=yourlogin \
  --with-group=Administrators
make
make       # yes, I needed to run it a second time
make install
```

The use of your own login instead of SYSTEM requires some explanation. If you were to call runconfigure with SYSTEM instead of your own login id as part of the -with-user parameter, the installation process will fail due to the way Cygwin and the NT/W2K/XP security system interact. Once you chown a file to another user (like SYSTEM) you are no longer able to chgrp or chmod the file. The installation process will abort at this point. By installing the files owned by yourself, you will be able to chgrp and chmod them as expected. Note that you still RUN as SYSTEM from `/etc/inetd.conf` (see below).

12.4 Configure Cygwin files

You have to modify some config files:

- `/etc/inetd.conf`: cleanup un-needed entries: Comment out any entries you do not need by placing a '#' at the start of the lines. This is just good practice, and if any of the entries reference non- existent users (e.g. uucp) inetd may not start up.
- `/etc/inetd.conf`: add

```
amanda dgram udp wait System /usr/local/libexec/amandad amandad
```

  ATTENTION: Use tabs, don't use spaces.
- create _/home/root/.amandahosts_ (or whereever System's home directory is): <amanda server> <amanda user>

Then create the following *Amanda* directories and the file `amandates`:

```
mkdir -p /usr/local/var/amanda/gnutar-lists

mkdir /tmp/amanda

touch /etc/amandates
```

12.5 Configure Windows System Files

Update the Windows services list

- WINDIR\Services: add

```
amanda    10080/udp   # Amanda backup services
amandaidx    10082/tcp   # Amanda backup services
amidxtape    10083/tcp   # Amanda backup services
```

where WINDIR is `C:\WINNT\system32\drivers\etc` or something similar. The last two lines are needed if you want to use **amrecover**.

Ensure that the default Windows PATH environment variable include your Cygwin / bin directory. This is necessary since **inetd** and hence the **amandad** that it spawns will not have the advantage of being started by the standard bash shell startup script and won't find the needed dynamic libraries (e.g. `cygwin1.dll`). My PATH is:

```
_%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;C:\cygwin\bin_
```

This is on XP; **My Computer**, right click **Properties**, click on **Environment Variables** (at the bottom). Yours may vary, but make sure the Cygwin bin directory is represented somewhere in the PATH.

12.6 Configure inetd to run automatically as a service

If you want to test your installation, you can call **inetd** from bash prompt:

```
/usr/sbin/inetd -d
```

12.7 Windows 98/ME

- To start after the user logs in: Create a shortcut to `c:\cygwin\usr\sbin\inetd.exe` in `WINDIR\start menu\programs\startup`
- To start before the user logs in: Add the string key

  `CygwinInetd=C:\cygwin\usr\sbin\inetd.exe`

  under

  `HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices`

  in the registry. You'll see a dos-like window on the startup: I did not find a solution to iconize or to make invisible (suggestions are welcome).

12.8 Windows NT/2000/XP

From bash prompt, type:

```
/usr/sbin/inetd --install-as-service
```

Then, to start/stop the **inetd** service use the Services control panel or the following Windows command: **net start/stop inetd**

12.9 Notes on *Amanda* backup options

12.9.1 Compression

Currently, client side compression does not work, probably due to problems in pipe emulation in Cygwin. I have not tried to debug this yet. This may be addressed in a subsequent release, or it could be fixed in later releases of Cygwin. Due to this issue, we recommend that if you want compressed dumps from Windows clients, you configure *Amanda* for server compression in `amanda.conf` on your *Amanda* server:

```
define dumptype srv-comp-tar {
    global
    comment "partitions dumped via tar with server compression"
    program "GNUTAR"
    compress server fast
    exclude list ".Amanda.exclude"
}
```

12.9.2 Exclude Lists

A note on exclude lists is also in order. If you specify a relative path, it will be expected that the file is in or relative to the root of the directory you are planning to dump. Typically this will not be '/' but '/cygdrive/c' or something similar if you want to get the Windows files and the Cygwin files. '/' is taken to be the root of the Cygwin tree, normally something like C: \cygwin or possibly C:\Program Files\cygwin.

12.9.3 Debugging Files

Amanda will leave debugging files in `/tmp/amanda` if it exists. I have recommended to create this directory above.

NOTE

Refer to <http://www.amanda.org/docs/howto-cygwin.html> for the current version of this document.

Chapter 13

HOW TO USE THE *AMANDA* FILE-DRIVER

This document covers the use of the file-driver in *Amanda* 2.4.3 and higher.

Examples given here have been taken from a SuSE-Linux-8.2-Pro-environment, using *Amanda* 2.4.4p1 and the snapshot 2.4.4p1-20031202. Please adjust paths, configuration names and other parameters to your system.

Stefan G. Weichinger, November - December, 2003 ; minor updates in April, 2005.

13.1 Introduction

Since release 2.4.3 *Amanda* supports the usage of a output driver called "file". See **man amanda**, section OUTPUT DRIVERS, for more information on its implementation. As the name suggests, this driver uses files as virtual (or file) tapes. Once created and labeled, these file tapes can be selected and changed with the standard tape-changer-interface of the *Amanda* server.

13.2 Possible Uses

- test installations

  You can easily explore the rich features of *Amanda* on systems without tape drives.

- cheap installations

  Without buying a tape drive you can enjoy the benefits of *Amanda* and backup to a bunch of harddisks. You can create CD/DVD-sized backups which you can burn onto optical disks later.

- disk-based installations

  You can use the file-driver to backup onto a set of file tapes hosted on a bunch of hard-disks or a RAID-system. Combined with another *Amanda*-configuration that dumps the file tapes to real tapes, you can provide reliable backup with faster tapeless recovery. This is called "disk-to-disk-to-tape"-backup by some people today.

13.3 Setup

13.3.1 Basics

This guide assumes you have setup the basic *Amanda*-services as described in Amanda Installation Notes

The configuration in this HOWTO is called "daily". The file tapes are also called *vtapes* in this document, which stands for "virtual tapes".

Please be sure to understand the differences between holding disks and file tapes. The two serve different purposes; holding disks allow for parallelism of multiple DLE's being backed up while file tapes are a replacement for physical tapes.

Before beginning you will need to decide on (a) dedicated part(s) of your hard disk(s) for your file tape storage. While this space could be spread among several file systems and hard disks, I recommend to dedicate at least a specific partition, better a specific physical harddisk to the task of keeping your vtapes. The use of a dedicated disk will speed things up definitely.

The disk space you dedicate for your vtapes should NOT be backed up by *Amanda*. Also, for performance reasons there should be NO holding disks on the same partition as the vtapes, preferably not even on the same physical drive.

If you only have one harddisk, it will work out, too, but you will suffer low performance due to massive head-moving in your harddisk, resulting from copying data between the filesystems.

Steps

1 Prepare the filesystem(s) used for the tapes. Decide on where to put your files, create the appropriate partition(s) and filesystem(s) and mount them.

  In our example we have the dedicated partition `hdc1`, mounted on `/amandatapes` for vtape storage.

```
$ mount
[...]
/dev/hdc1 on /amandatapes type reiserfs (rw)
[...]
```

  Make sure there is space left. Determine the amount of space you will use.

```
$ df -h /amandatapes
Filesystem      Size  Used  Avail  Use%   Mounted on
/dev/hdc1        20G    0G    20G    0%   /amandatapes
```

  In our example we have 20GB diskspace left on `/amandatapes`.

2 Determine length and number of tapes After deciding on the number of vtapes you want to create, evenly allocate the available space among them.

Look at the following rule of thumb:

As many filesystems exhibit dramatically reduced performance when they are nearly full I have chosen to allocate only 90% of the available space. So we have:

(Available Space * 0.9) >= tapelength * tapecycle

This is a very conservative approach to make sure you don´t suffer any performance drop due to a nearly-full-filesystem.

As it is uncommon for *Amanda* to fill, or almost fill an entire tape you may also wish to use more space than that.

So you could determine possible combinations of tapelength/tapecycle with the more general formula:

Available Space >= tapelength * tapecycle

In our example we take the conservative approach:

- 20 GB * 0.9 = 18 GB to use

  and so we could create the following combinations:

- 18 GB = 18 GB * 1
- 18 GB = 9 GB * 2
- 18 GB = 6 GB * 3
- 18 GB = 3 GB * 6
- 18 GB = ... you get the picture.

Using only one tape is generally considered a bad idea when it comes to backup, so we should use at least 3 tapes (for testing purposes), better 6 or more tapes.

- 18 GB = 3 GB * 6

  so we get the value 3 GB for the tapelength if we want to use 6 tapes.

3 Create a tapetype definition. Add a new tapetype definition similar to the following to your `amanda.conf`. I named my definition "HARD-DISK". Choose whatever name you consider appropriate.

```
define tapetype HARD-DISK {
   comment "Dump onto hard disk"
   length 3072 mbytes   # specified in mbytes to get the exact size of 3GB
}
```

You don´t have to specify the parameter speed (as it is commonly listed in tapetype definitions and reported by the program **amtapetype**). *Amanda* does not use this parameter right now.

There is also an optional parameter filemark, which indicates the amount of space "wasted" after each tape-listitem. Leave it blank and *Amanda* uses the default of 1KB.

4 Think about tapechangers. As you will use a set of vtapes, you have to also use a kind of vtape-changer. There are several tape-changer-scripts included in the *Amanda*-tarball. Read more about tape-changer-scripts in *Amanda* Tape Changer Support.

Right now there are two scripts that can be used with vtapes. These scripts take different approaches to the handling of tapes.

The script chg-multi handles many drives with a tape in each drive. The script chg-disk handles a library with one drive and multiple tapes.

So with vtapes you could look at it this way:

chg-multi simulates multiple tape drives with one tape in each drive. chg-disk simulates one tape-library with multiple tapes in.

As chg-multi exists for a much longer time than chg-disk, it is still used in many *Amanda*-vtape-installations.

chg-disk was introduced with the snapshot 20031202. Contrary to chg-multi, which is a generic changer-script that must be somewhat adjusted to the use of the file-driver, chg-disk offers exactly the behavior needed for handling vtapes

IMHO the approach is much more logical, so I recommend to use chg-disk in new *Amanda*-vtape-installations.

NOTE

To use chg-disk you need to have at least amanda-2.4.4p1-20031202.

Choose the one that fits your way of vtape-handling and -maintenance.

In this HOWTO I only cover the use of chg-disk. Usage of chg-multi is pretty similar and will maybe covered in a later version of this document.

5 Set up your tape-config. In the general section you have to set the parameters tapecycle , tapetype , tpchanger , changerfile , tapedev , rawtapedev and changerdev.

Example:

```
$ vi /usr/local/etc/amanda/daily/amanda.conf
...

tapecycle 6
tapetype HARD-DISK
tpchanger "chg-disk"
```

```
     changerfile "/usr/local/etc/amanda/daily/changer"
tapedev  "file:/amandatapes/daily"
```

This reflects the use of your defined tapetype.

The parameter tapecycle tells *Amanda* how much tapes can be used, Set this value according to the number of tapes you want to use.

The parameter tapetype , points to the tapetype definition you have created before.

The parameter tpchanger tells *Amanda* to use the generic tape-changer-script to handle the vtapes. You can think of it as a virtual tape-changer-device.

The parameter changerfile is used to give chg-disk the "prefix" for the "%s-changer, %s-clean, %s-slot" files it needs. Use something like "changer" in your config-dir. Please note that this file does NOT have to exist, but it won't hurt anyway.

The parameter tapedev tells the chg-disk-script where the root-dir for your vtapes is.

In our example the vtape-files go to `/amandatapes`.

To separate multiple configurations, we decided to use subdirectories according to the configuration name "daily".

NOTE

The parameter changerdev is NOT needed with chg-disk as it is not parsed by chg-disk.

6 Create the virtual tapes.

NOTE

Gene Heskett has committed a shell-script which creates and labels the vtapes in one step. Stefan G. Weichinger will generalize this script and contribute it, this script will just read your settings in `amanda.conf` and create the appropriate vtape-directories.

Now you have to create the tape-directories. chg-disk needs a directory structure like:

```
slot_root_dir -|
               |- info
               |- data -> slot1/
               |- slot1/
               |- slot2/
```

```
|- ...
|- slotn/
```

where 'slot_root_dir' is the tapedev 'file:xxx' parameter and 'n' is the tapecycle parameter.

So in our example we do:

```
$ mkdir /amandatapes/daily
```

for the 'slot_root_dir' and

```
$ mkdir /amandatapes/daily/slot1
$ mkdir /amandatapes/daily/slot2
....
```

for the virtual slots that will later contain the vtapes.

If you have many vtapes to create and their names follow a pattern you may be able to do them all with a single loop such as:

```
$ for n in 1 2 3 4 5 6 7 8 9 10 11 12
> do
>    mkdir /amandatapes/daily/slot${n}
> done
```

Create the info-file:

```
$ touch /amandatapes/daily/info
```

and link the first slot to the data-file (to "load" the vtape into the first slot):

```
$ ln -s /amandatapes/daily/slot1 /amandatapes/daily/data
```

Make sure the *Amanda*-user has write-permissions on these directories:

```
$ chown -R amanda_user /amandatapes
$ chgrp -R amanda_group /amandatapes
$ chmod -R 750 /amandatapes
```

7 Label the virtual tapes. As the virtual tapes are handled just like physical tapes by the *Amanda*-Server they have to be labeled before use.

```
Usage: amlabel [-f] <conf> <label> [slot <slot-number>]
```

Example:

```
$ amlabel daily daily1 slot 1
....
$ amlabel daily daily2 slot 2
....
```

If you have many vtapes to label and their names follow a pattern you may be able to do them all with a single loop such as:

```
     $ for n in 1 2 3 4 5 6 7 8 9 10 11 12
> do
>    amlabel daily daily${n} slot ${n}
> done
```

Label all your created tapes according to the "labelstr"-parameter in your `amanda.conf`. Consult the **amlabel**-man-page for details.

8 Test your setup with **amcheck**. Run **amcheck daily** (or, more general, **amcheck** <config>) and look for anything *Amanda* complains about.

A proper output looks like:

```
$ amcheck daily
Amanda Tape Server Host Check
--
Holding disk /amhold: 6924940 KB disk space available,
that's plenty
amcheck-server: slot 2: date 20031115 label daily02
(exact label match)
NOTE: skipping tape-writable test
Tape daily02 label ok
Server check took 0.377 seconds
```

Recheck your files if errors occur.

13.4 Recovery

Recovering files from vtapes is very similar to recovering files from a "real" tapechanger.

Make sure you read the chapter Restore.

I will simply paste a **amrecover**-session here (provided by JC Simonetti, author of chg-disk):

```
# /usr/local/amanda/sbin/amrecover woo
AMRECOVER Version 2.4.4p3. Contacting server on backupserver.local ...
220 backupserver Amanda index server (2.4.4p3) ready.
200 Access OK
Setting restore date to today (2004-10-08)
200 Working date set to 2004-10-08.
Scanning /BACKUP2/holding...
Scanning /BACKUP/holding...
200 Config set to woo.
200 Dump host set to backupserver.local.
Trying disk /tmp ...
$CWD '/tmp/RECOVER' is on disk '/tmp' mounted at '/tmp'.
200 Disk set to /tmp.
Invalid directory - /tmp/RECOVER
amrecover> sethost backupserver.local
200 Dump host set to backupserver.local.
amrecover> setdisk /
200 Disk set to /.
amrecover> cd /etc
/etc
amrecover> add passwd
Added /etc/passwd
amrecover> list
TAPE B3_14 LEVEL 0 DATE 2004-09-26
        /etc/passwd
amrecover> extract

Extracting files using tape drive file:/BACKUP2/slots/ on host
backupserver.local. The following tapes are needed: B3_14

Restoring files into directory /tmp/RECOVER
Continue [?/Y/n]? Y

Extracting files using tape drive file:/BACKUP2/slots/ on host
backupserver.local. Load tape B3_14 now
Continue [?/Y/n/s/t]? Y
./etc/passwd
amrecover> quit
200 Good bye.
```

Nothing spectacular? The trick is this:

When *Amanda* asks you

```
Load tape B3_14 now Continue [?/Y/n/s/t]?
```

you have to run the following in a second terminal:

```
$ amtape woo slot 14
amtape: changed to slot 14 on file:/BACKUP2/slots/
```

This step is necessary to load the proper tape into your virtual changer.

Let me express this in a more general way:

When **amrecover** prompts for the tape it needs to restore the files you requested, you have to "load" the tape it requests.

The recommended way to do this is to use **amtape**. The options that make sense in this context are:

```
# amtape
Usage: amtape <conf> <command>
        Valid commands are:
      [...]
                slot <slot #>        load tape from slot <slot #>
      [...]
                label <label>        find and load labeled tape
      [...]
```

If you know which slot contains the requested tape (for example, if you have tape daily01 in slot 1, tape daily02 in slot 2, and so on) you may use the first option. If you just know the label of the tape you need, use the second option.

To continue the upper example:

```
amtape woo slot 14   # option 1 OR
amtape woo label B3_14  # option 2
```

amtape will return something like:

```
amtape: label B3_14 is now loaded.
```

After this you can return to your **amrecover**-session and continue restoring your files.

Please be aware of the fact reported by JC Simonetti: " I have never never used the "settape" command of **amrecover** [with chg-disk] since there's some problems with it (tape not loaded correctly, or impossible to change from tape to tape when restoring data shared accross multiple tapes...) "

NOTE

Refer to <http://www.amanda.org/docs/howto-filedriver.html> for the current version of this document.

Chapter 14

AFS HOWTO

You need to download the following package if you want to backup AFS volume with amanda:
<ftp://ftp.ccmr.cornell.edu/pub/amanda-afs/amanda-afs.tar.gz>
or anonymous cvs from :pserver:anonymous@cvs.ccmr.cornell.edu:/usr/common/cvs
and checkout project 'amanda-afs'
The patch to *Amanda* is already included in this distribution.

NOTE

Refer to <http://www.amanda.org/docs/howto-afs.html> for the current version of this document.

Chapter 15

HOW TO USE A WRAPPER

NOTE

The script used in this document is not part of the official *Amanda* release. The *Amanda* core team does not take any responsibility for this script.

15.1 Bert de Ridder's suggestions

This is a mini-howto explaining how to control other running tasks on a server where the *Amanda* software is used to backup data.

Problem : Lots of software is picky about their datafiles being backed up while the files are in use. It sometimes is even necessary to know the state of the datafiles at the moment of backup so that when restoring you know exactly *what* you are restoring. And most of the time there are dependencies between the datafiles as well (for instance, the pure datafiles and the controlfiles of an Oracle database.)

The solution is actually quite simple; you just use a custom made backupscript instead of the standard tar command. Inside this tar command, you do some necessary processing before executing the tar command and - if necessary - do some more processing. This way, you can easily stop an Oracle database, tar the files, send them to the tape server and restart the Oracle database. This of course is just an example, anything you can do in a shell script can be done.

1 Create the script

This is the most important step, this script is the work horse of the solution. I've called it `/bin/amandatar`. You can call it whatever you want though. It's a Perl script, it may not be very pretty code, but it does the job. In the script, an example is given for the backup of a Lotus Notes Domino server.

```
#!/usr/bin/perl

# Tar wrapper for Amanda's tar.
#

use Getopt::Long qw(:config pass_through);

# Obtain directory and file information from the command line.

$result = GetOptions (
 'directory=s' => \$dir,
 'file=s' => \$file
);

# Check whether Amanda wants to do some administrative task (eg. indexinfo
# or obtain the number of bytes to be backed up)
# if file = /dev/null it's an administrative task and most of the time, no extra
#   processing is necessary.

# What you see here is just a log of the backup start time, and more important
#   the stopping of the domino server

if ( $file ne '/dev/null' )
{
  if ( $dir eq '/local/notesdata' )
  {
    system "echo 'Start backup notes at ' >> /var/lib/amanda/runtime" ;
    system "date >> /var/lib/amanda/runtime";
    system ( "/etc/init.d/domino stop >> /var/lib/amanda/runtime" );
  }
}

# The command line is being 'reconstructed'. Necessary because the GetOptions
#   call above has stripped the file and directory information.
# This is what I meant with 'ugly'  code ;-)

while ( $ARGV[0] ne '' )
{
  $val = $ARGV[0] ;
  unshift ( @NEWARGV, $val, ) ;
  shift @ARGV;
}

while ( $NEWARGV[0] ne '' )
{
  $val = $NEWARGV[0] ;
  unshift ( @ARGV, $val ) ;
```

```
  shift @NEWARGV;
}

if ( $dir ne '' )
{
  unshift ( @ARGV, '--directory', $dir );
}
if ( $file ne '' )
{
  unshift ( @ARGV, '--file', $file );
}

if ( $file ne '/dev/null' )
{
  system "echo 'Backing up directory ' $dir >> /var/lib/amanda/runtime" ;
}

# And finally make sure tar is called :-)
#   (path may differ on your installation)
unshift ( @ARGV , "/bin/tar" ) ;

system ( @ARGV ) ;

# Postprocessing
#
# If Notes backup was requested, restart the server.
# Log the backup end time.
#

if ( $file ne '/dev/null' )
{
  if ( $dir eq '/local/notesdata' )
  {
    system ( "/etc/init.d/domino start >> /var/lib/amanda/runtime" );
    system "echo 'End backup notes at ' >> /var/lib/amanda/runtime" ;
    system "date >> /var/lib/amanda/runtime";
  }
}

exit 0;

# End script
```

On some systems it may be necessary to setuid root the script.

2 Rebuild *Amanda* so that it uses your newly created script.

Download the sources, untar them to a directory. I'm sure there are lots of documents

already available on how to do this, so I won't go into too much detail. (Refer to Amanda Installation Notes).

fast path :

```
/usr/local/src # tar -xvzf amanda-source.tar.gz
/usr/local/src # cd amanda-version
/usr/local/src/amanda-version # ./configure \
  --with-user=amanda \
  --prefix=/usr/local \
  --exec-prefix=/usr \
  --bindir=/usr/bin \
  --sbindir=/usr/sbin \
  --libexecdir=/usr/lib/amanda \
  --with-configdir=/etc/amanda \
  --with-group=disk \
  --with-gnutar=/bin/amandatar \
  --with-gnutar-listdir=/var/lib/amanda/gnutar-lists \
  --with-tmpdir=/tmp/amanda \
  --with-smbclient=/usr/bin/smbclient \
  --mandir=/usr/local/man
```

Here, it may be necessary to adjust some paths to match your installation. This setup works on SuSE Linux (also SLES) and MacOSX although you may have to use another binary tar.

As you see, you may also "replace" the smbclient if necessary. I haven't yet tested it though. I'll leave it as an exercise for the reader <g>.

```
/usr/local/src/amanda-version # make
/usr/local/src/amanda-version # make install
```

Now proceed as with a "normal" installation.

15.2 Paul Bijnens's suggestions

How do I run pre- and post dump programs, e.g. database stop/start?

Currently (*Amanda* 2.4.5) there is no direct support to run a program before or after a backup on a client. But there is an easy workaround by using a wrapper for **GNU-tar** that does the additional tasks.

Let's suppose you want to stop a database before the backup, and start it up again when the backup is finished. You have already two scripts "shutdb" and "startdb" to shutdown and startup the database.

First you have to configure *Amanda* on the client to use the gnutar-wrapper instead of the real **GNU-tar**:

```
./configure ... --with-gnutar=/usr/local/bin/amgtar ...
```

and re-compile *Amanda*. The program "amgtar" can be a simple link to the real **GNU-tar**-binary on clients that don't need special handling, or it can be a script.

Amanda expects that the bytestream on `stdout` is the backup image, and the bytestream on `stderr` are messages. The `stderr` messages are filtered against a known set of strings, and anything unexpected is flagged as "STRANGE" in the *Amanda* report. The return-codes of the program should be the same as the return-codes of **GNU-tar**:

- 0 = ok (backup image will be put on tape)
- 1 = not ok (backup image will not be put on tape, same level will be tried next time).

The arguments passed to the program are pretty static (see in the sources `client-src/sendbackup-gnutar.c`, line 483). To decide if you need to stop/start the database you have to check if:

- this run makes a backup and not a restore: look for "–create"
- this it is not an estimate run: look for "–file /dev/null" (estimate) or "–file -" (real run)
- this run is for the database directory: look for "–directory /my/data/base"

In all other cases, we just pass the args and run the real **GNU-tar**.

Here is an example script in Bourne shell:

Here is an example script in perl:

NOTE

Refer to <http://www.amanda.org/docs/howto-wrapper.html> for the current version of this document.

Example 15.2.1

```
#!/bin/sh

# # uncomment next block to follow the flow
# LOG=/tmp/amanda/mytar.debug
# date >> $LOG
# echo "$@" >> $LOG
# if [ "$3" = "/dev/null" ]
# then echo "Estimate only" >> $LOG
# else echo "Real backup" >> $LOG
# fi

# - Avoid output to stdout! (the backup stream by tar)
# - Any output to stderr is flagged as "strange" by amanda
#   and may be used to pass error messages into the report

if [ "$1" = "--create"  -a  "$3" = "-"  -a  "$5" = "/my/dir" ]
then
    # echo "/my/dir: want to execute some progs first" >>$LOG
    /usr/local/bin/shutdb thedb >&2
    /usr/local/bin/gtar "$@"
    rc=$?
    # echo "Finished the real backup; some postprocessing" >>$LOG
    /usr/local/bin/startdb thedb >&2
    exit $rc
else
    /usr/local/bin/gtar "$@"
fi
```

Example 15.2.2

```
#!/usr/bin/perl -w

use Getopt::Long qw(:config pass_through);

my @saveopts = @ARGV;
GetOptions (
        'create' => \$create,
        'directory=s' => \$dir,
        'file=s' => \$file,
);
@ARGV = @saveopts;

my $postproc = 0;
if ($create  &&  $dir eq '/my/data/base' &&  $file ne '/dev/null') {
    system '/usr/local/bin/dbshut thedb >/tmp/amanda/dbshut.debug 2>&1';
    $postproc = 1;
}

unshift(@ARGV, "/usr/local/bin/gtar");
system @ARGV;

my $rc = $? >> 8;

if ($postproc) {
    system '/usr/local/bin/dbstart thedb >/tmp/amanda/dbstart.debug 2>&1';
}

exit $rc;
```

Chapter 16

HOW TO DO *AMANDA*-SERVER-SIDE GPG-ENCRYPTED BACKUPS.

NOTE

THIS IS *NOT* YET INTENDED FOR PRODUCTION SERVERS !!!

Bruce Fletcher asked for a "simple" encryption method to be used with *Amanda*-server. gpg-amanda <`http://security.uchicago.edu/tools/gpg-amanda/`> seems to create problems at restore-time, as it uses a wrapper for gzip.

My solution uses a wrapper for **GNU-tar** instead, so there are several disadvantages avoided.

NOTE

This is based on a *Amanda*-vtape-setup with the *Amanda*-release 2.4.5. As this is still in the testing-stage, I have coded the home-dir of the *Amanda*-user into my scripts (`/var/lib/amanda`). This should be done with variables later, I agree ...

What you need:

- aespipe <`http://loop-aes.sourceforge.net/aespipe/aespipe-v2.3b.tar.bz2`> and the bz2aespipe-wrapper that comes with it. It gets patched as described later.

- the wrapper-script `/usr/local/libexec/amgtar`, as listed down below,
- GNU-PG <`http://www.gnupg.org/(en)/download/index.html`>. This should be part of most current operating systems already.
- *Amanda* ;)

16.1 Setup

- Configure and compile aespipe:

```
tar -xjf aespipe-v2.3b.tar.bz2
cd aespipe-v2.3b
./configure
make
make install
```

- Generate and store the gpg-key for the *Amanda*-user:

```
# taken from the aespipe-README
head -c 2925 /dev/random | uuencode -m - | head -n 66 | tail -n 65 | \
gpg --symmetric -a > /var/lib/amanda/.gnupg/am_key.gpg
```

This will ask for a passphrase. Remember this passphrase as you will need it in the next step.

Store the passphrase inside the home-directory of the *Amanda*-user and protect it with proper permissions:

```
echo my_secret_passphrase > ~amanda/.am_passphrase
chown amanda:disk ~amanda/.am_passphrase
chmod 700 ~amanda/.am_passphrase
```

We need this file because we don't want to have to enter the passphrase manually everytime we run **amdump**. We have to patch bz2aespipe to read the passphrase from a file. I have called that file `~amanda/.am_passphrase`.

It should NOT ;) look like this:

```
# cat ~amanda/.am_passphrase
my_secret_passphrase
```

NOTE

Store the key and the passphrase in some other place as well, without these information you can't access any tapes that have been encrypted with it (this is exactly why we are doing all this, isn't it? ;)).

- Create the wrapper for **GNU-tar**:

Example 16.1.1 /usr/local/libexec/amgtar

```
#!/bin/sh
#
# Original wrapper by Paul Bijnens
#
# crippled by Stefan G. Weichinger
# to enable gpg-encrypted dumps via aespipe

GTAR=/bin/tar
AM_AESPIPE=/usr/local/bin/amaespipe
AM_PASSPHRASE=/var/lib/amanda/.am_passphrase
LOG=/dev/null
LOG_ENABLED=1

if [ "$LOG_ENABLED" = "1" ]
then
LOG=/var/log/amanda/amgtar.debug
date >> $LOG
echo "$@" >> $LOG
fi

if [ "$3" = "/dev/null" ]
then
   echo "Estimate only" >> $LOG
   $GTAR "$@"
else
   echo "Real backup" >> $LOG
   $GTAR --use-compress-program="$AM_AESPIPE" "$@" 3< $AM_PASSPHRASE
fi

rc=$?
exit $rc
```

- Copy the wrapper-script bz2aespipe, which comes with the aespipe-tarball, to `/usr/local/bin/amaespipe` and edit it this way:

  or apply this small patch

Things I have changed:

- Decreased WAITSECONDS: No need to wait for 10 seconds to read the passphrase.
- Removed bzip2 from the pipes: *Amanda* triggers **GNU-zip**-compression by itself, no need to do this twice (slows down things, blows up size).
- Added options -K and -p: This enables aespipe to use the generated gpg-key and tells it the number of the file-descriptor to read the passphrase from.

NOTE

You may set various parameters inside bz2aespipe. You may also call bz2aespipe with various command-line-parameters to choose the encryption-algorithm, hash-function etc. . For a start I have chosen to call bz2aespipe without command-line-options.

- Reconfigure and recompile *Amanda* (yes, I'm sorry ...):

  As described in How to use a wrapper you have to run **configure** again with the option –**with-gnutar=/usr/local/libexec/amgtar**, after that recompile and reinstall *Amanda*. These steps are described in the mentioned document.

16.2 Test

Still to come ...

16.3 Plans

There are several wishes:

- Ability to switch encryption inside a dumptype. This HOWTO describes a method that enables/disables encryption for the whole installation. You might remove the amgtar-wrapper and simply link to plain **GNU-tar** again to disable encryption, but be aware that you also disable decryption with this step. You will hit problems when you then try to restore encrypted tapes.
- Ability to switch encryption-parameters inside a dumptype. Choice of algorithm, hash-functions etc. I don't know if it makes sense to put it into a dumptype or if it would be enough to configure it once inside amaespipe (I assume the latter).
- All this leads to the need to code this into *Amanda* itself: new dumptype-options and corresponding calls to **GNU-tar** etc. inside `client-src/sendbackup-gnutar.c`.

This is it so far. Release early, release often. Feel free to contact me with your thoughts on this paper.

NOTE

Refer to <http://www.amanda.org/docs/howto-gpg.html> for the current version of this document.

Example 16.1.2 /usr/local/bin/amaespipe

```
#! /bin/sh

# FILE FORMAT
# 10 bytes: constant string 'bz2aespipe'
# 10 bytes: itercountk digits
# 1 byte: '0' = AES128, '1' = AES192, '2' = AES256
# 1 byte: '0' = SHA256, '1' = SHA384, '2' = SHA512, '3' = RMD160
# 24 bytes: random seed string
# remaining bytes are bzip2 compressed and aespipe encrypted

# These definitions are only used when encrypting.
# Decryption will autodetect these definitions from archive.
ENCRYPTION=AES256
HASHFUNC=SHA256
ITERCOUNTK=100
WAITSECONDS=1
GPGKEY="/var/lib/amanda/.gnupg/am_key.gpg"
FDNUMBER=3

if test x$1 = x-d ; then
    # decrypt
    n=`head -c 10 - | tr -d -c 0-9a-zA-Z`
    if test x${n} != xbz2aespipe ; then
        echo "bz2aespipe: wrong magic - aborted" >/dev/tty
        exit 1
    fi
    itercountk=`head -c 10 - | tr -d -c 0-9`
    if test x${itercountk} = x ; then itercountk=0; fi
    n=`head -c 1 - | tr -d -c 0-9`
    encryption=AES128
    if test x${n} = x1 ; then encryption=AES192; fi
    if test x${n} = x2 ; then encryption=AES256; fi
    n=`head -c 1 - | tr -d -c 0-9`
    hashfunc=SHA256
    if test x${n} = x1 ; then hashfunc=SHA384; fi
    if test x${n} = x2 ; then hashfunc=SHA512; fi
    if test x${n} = x3 ; then hashfunc=RMD160; fi
    seedstr=`head -c 24 - | tr -d -c 0-9a-zA-Z+/`
    #aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${encryption} -H ${hashfunc} -S "${seedstr
    aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${encryption} -H ${hashfunc} -S "${seedstr}
else
    # encrypt
    echo -n bz2aespipe
    echo ${ITERCOUNTK} | awk '{printf "%10u", $1;}'
    n=`echo ${ENCRYPTION} | tr -d -c 0-9`
    aesstr=0
    if test x${n} = x192 ; then aesstr=1; fi
    if test x${n} = x256 ; then aesstr=2; fi
    n=`echo ${HASHFUNC} | tr -d -c 0-9`
    hashstr=0
```

Example 16.1.3 bz2aespipe.patch

```
@@ -15,3 +15,5 @@
 ITERCOUNTK=100
-WAITSECONDS=10
+WAITSECONDS=1
+GPGKEY="/var/lib/amanda/.gnupg/am_key.gpg"
+FDNUMBER=3

@@ -36,3 +38,4 @@
     seedstr=`head -c 24 - | tr -d -c 0-9a-zA-Z+/`
-    aespipe -e ${encryption} -H ${hashfunc} -S "${seedstr}" -C ${itercountk} -d | bzi
+    #aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${encryption} -H ${hashfunc} -S "${seedst
+    aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${encryption} -H ${hashfunc} -S "${seedstr
 else
@@ -52,3 +55,4 @@
     echo -n ${aesstr}${hashstr}${seedstr}
-    bzip2 | aespipe -e ${ENCRYPTION} -H ${HASHFUNC} -S ${seedstr} -C ${ITERCOUNTK} -T
+    #bzip2 | aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${ENCRYPTION} -H ${HASHFUNC} -S $
+    aespipe -K ${GPGKEY} -p ${FDNUMBER} -e ${ENCRYPTION} -H ${HASHFUNC} -S ${seedstr}
 fi
```

Chapter 17

HOW TO USE DIFFERENT AUTH WITH *AMANDA*

This document covers the use of the auth in *Amanda* 2.5.1 and higher.

17.1 Introduction

17.2 BSD

You must configure amanda with –with-bsd-security and –with-amandahosts.

The xinetd.d/amanda file on the client:

```
service amanda
{
      only_from               = 127.0.0.1
      socket_type             = dgram
      protocol                = udp
      wait                    = yes
      user                    = amanda
      group                   = amanda
      groups                  = yes
      server                  = /path/to/amandad
      server_args             = -auth=bsd amdump
      disable                 = no
}
```

The only_from line should list your tape server ip address.

The ~amanda/.amandahosts file on the client:

```
tapeserver.fqdn amanda amdump
```

If you want to also enable amindexd and amidxtaped, you must change the server_args line in the xinetd.d/amanda file on the tape server:

```
server_args             = -auth=bsd amdump amindexd amidxtaped
```

The only_from line should list all machine that can use amdump/amrecover. It's the .amandahosts that will limit which client can use amdump/amindexd/amidxtaped.

The ~amanda/.amandahosts file on the tape server must have a line for each machi ne:

```
clientmachine1 amanda amindexd amidxtaped
clientmachine2 amanda amindexd amidxtaped
```

17.3 BSDTCP

Like bsd but you must configure amanda with –with-bsdtcp-security and –with-amandahosts and do 4 changes in the xinetd.d/amanda file:

```
socket_type             = stream
protocol                = tcp
wait                    = no
server_args             = -auth=bsdtcp amdump
```

17.4 BSDUDP

Like bsd but you must configure amanda with –with-bsdudp-security and –with-amandahosts and do 1 change in the xinetd.d/amanda file:

```
server_args             = -auth=bsdudp amdump
```

17.5 KRB4

You must configure amanda with –with-krb4-security.

17.6 KRB5

You must configure amanda with –with-krb5-security.

17.7 RSH

You must configure amanda with –with-rsh-security.

It's your system that should allow your server user to rsh to your client user.

If your server username and client username are different, you must add the client_username option in all DLE for that host.

```
client_username "client_username"
```

If your server amandad path and client amandad path are different, you must set the amandad_path option in all DLE for that hosts.

```
amandad_path "client/amandad/path"
```

17.8 SSH

You must configure amanda with –with-ssh-security.

17.8.1 For amdump:

You must create an ssh key for your server. In this example, the key is put in the id_rsa_amdump file:

```
ssh-keygen -t rsa
Enter file in which to save the key (/home/amanda/.ssh/id_rsa)? /home/amanda/.ssh/id_r
```

You must set the ssh_keys option in all DLE for that host:

```
ssh_keys "/home/amanda/.ssh/id_rsa_amdump"
```

You mush append the /home/amanda/.ssh/id_rsa_amdump.pub file to the .ssh/authorized_keys file of all client host.

For security reason, you must prepend the line with the following:

```
from="tape_server_fqdn_name",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,
```

That will limit that key to connect only from your server and only be able to execute amandad.

Like rsh if your server username and client username are different, you must add the client_username option in all DLE for that host:

```
client_username "client_username"
```

Like rsh, if your server amandad path and client amandad path are different, you must set the amandad_path option in all DLE for that hosts:

```
amandad_path "client/amandad/path"
```

17.8.2 For amrecover:

You must create an ssh key for root on all clients that can use amrecover. In this example, the key is put in the /root/.ssh/id_ rsa_amrecover file:

Log in as root:

```
ssh-keygen -t rsa
Enter file in which to save the key (/root/.ssh/id_rsa)? /root/.ssh/id_rsa_amrecover
```

You must set the ssh_keys option in the amanda_client.conf file

```
ssh_keys "/root/.ssh/id_rsa_amrecover"
```

You mush append all client /home/root/.ssh/id_rsa_amrecover.pub file to the /home/amanda/.ssh/autho
of the server.

For security reason, you must prefix all lines with the following:

```
from="aclient_fqdn_name",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,comm
```

That will limit every client key to connect from the client and only be able to execute amandad.

NOTE

Refer to <http://www.amanda.org/docs/howto-auth.html> for the current version of this document.

Part IV

Various Information

ADDITIONAL INFORMATION ABOUT *AMANDA*

Here you find various other documents like the *Amanda* FAQ, the Top Ten Questions and the *Amanda*-Wishlist. You can also find the last *Amanda*-Survey in here (although it still needs formatting ... sorry, Jon ...). The latest addition is the "famous" *Amanda*-chapter by John R. Jackson.

Chapter 18

USING *AMANDA*

18.1 An Introduction

NOTE

This chapter was written by John R. Jackson with input from Alexandre Oliva. It is part of the O'Reilly book "Unix Backup & Recovery" by W. Curtis Preston and has been provided online at <http://www.backupcentral.com/amanda.html> since the first edition of this book.

During the Docbook-conversion of the *Amanda*-docs we asked for permission to include this chapter in the Official *Amanda* documentation and W. Curtis Preston allowed to us to include the now converted version. There will be some updates to this chapter in the next few months to reflect various changes and enhancements.

You can find online versions of this chapter at <http://www.amanda.org/docs/using.html> and at <http://www.backupcentral.com/amanda.html>.

Amanda, the Advanced Maryland Automated Network Disk Archiver, is a public domain utility developed at the University of Maryland. It is as advanced as a free backup utility gets, and has quite a large user community. *Amanda* allows you to set up a single master backup server to back up multiple hosts to a single backup drive. (It also works with a number of stackers.) *Amanda* uses native dump and/or **GNU-tar**, and can back up a large number of workstations running multiple versions of Unix. Recent versions can also use SAMBA to back up Microsoft Windows (95/98/NT/2000)-based hosts. More information about *Amanda* can be found at <http://www.amanda.org>

Amanda was written primarily by James da Silva at the Department of Computer Science of the University of Maryland around 1992. The goal was to be able to back up large numbers of client workstations to a single backup server machine.

Amanda was driven by the introduction of large capacity tape drives, such as ExaByte 8mm and DAT 4mm. With these drives, and the increased number of personal workstations, it no longer made sense to back up individual machines to separate media. Coordinating access and providing tape hardware was prohibitive in effort and cost. A typical solution to this problem reaches out to each client from the tape host and dumps areas one by one across the network. But this usually cannot feed the tape drive fast enough to keep it in streaming mode, causing a severe performance penalty.

NOTE

Since *Amanda* is optimized to take advantage of tape drives, we will use the word *tape* throughout this section. However, that doesn't mean that you couldn8217;t use it with an optical or CD-R drive.

The *Amanda* approach is to use a "holding disk" on the tape server machine, do several dumps in parallel into files in the holding disk, and have an independent process take data out of the holding disk. Because most dumps are small partials, even a modest amount of holding disk space can provide an almost optimal flow of dump images onto tape.

Amanda also has a unique approach to scheduling dumps. A "dump cycle" is defined for each area to control the maximum time between full dumps. *Amanda* takes that information, statistics about past dump performance, and estimates on the size of dumps for this run to decide which backup level to do. This gets away from the traditional static "it's Friday so do a full dump of /usr on client A" approach and frees *Amanda* to balance the dumps so the total run time is roughly constant from day to day.

Amanda is freely-available software maintained by the *Amanda* Users Group. Based on membership of *Amanda*-related mailing lists, there are probably well over 1500 sites using it. This chapter is based on *Amanda* version 2.4.2. Updated versions of this section will be available with the *Amanda* source code.

18.2 *Amanda* Features

Amanda is designed to handle large numbers of clients and data, yet is reasonably simple to install and maintain. It scales well, so small configurations, even a single host, are possible. The code is portable to a large number of Unix platforms. It calls standard backup software, such as vendor provided **dump** or **GNU-tar**, to perform actual client dumping. There is also support for backing up Windows-based hosts via SAMBA. There is no Macintosh support yet.

Amanda provides its own network protocols on top of TCP and UDP. It does not, for instance, use rsh or rdump/rmt. Each client backup program is instructed to write to standard output, which *Amanda* collects and transmits to the tape server host. This allows *Amanda* to insert compression and encryption and also gather a catalogue of the image for recovery. Multiple clients are typically backed up in parallel to files in one or more holding

disk areas. A separate tape writing process strives to keep the tape device streaming at maximum throughput. *Amanda* can run direct to tape without holding disks, but with reduced performance.

Amanda supports using more than one tape in a single run, but does not yet split a dump image across tapes. This also means it does not support dump images larger than a single tape. *Amanda* currently starts a new tape for each run and does not provide a mechanism to append a new run to the same tape as a previous run, which might be an issue for small configurations.

Amanda supports a wide range of tape storage devices. It uses basic operations through the normal operating system I/O subsystem and a simple definition of characteristics. New devices are usually trivial to add. Several tape changers, stackers, and robots are supported to provide truly hands-off operation. The changer interface is external to *Amanda* and well-documented, so unsupported changers can be added without a lot of effort.

Either the client or tape server may do software compression, or hardware compression may be used. On the client side, software compression reduces network traffic. On the server side, it reduces client CPU load. Software compression may be selected on an image-by-image basis. If Kerberos is available, clients may use it for authentication and dump images may be encrypted. Without Kerberos, .amandahosts authentication (similar to .rhosts) is used, or *Amanda* may be configured to use .rhosts (although rsh/rlogin/rexec are not themselves used). *Amanda* works well with security tools like TCP Wrappers (ftp://info.cert.org/pub/network_tools) and firewalls.

Since standard software is used for generating dump images and software compression, only normal Unix tools such as mt, dd, and gunzip/uncompress are needed to recover a dump image from tape if *Amanda* is not available. When *Amanda* software is available, it locates which tapes are needed and finds images on the tapes.

Amanda is meant to run unattended, such as from a nightly cron job. Client hosts that are down or hung are noted and bypassed. Tape errors cause *Amanda* to fall back to ?degraded? mode where backups are still performed but only to the holding disks. They may be flushed to tape by hand after the problem is resolved.

Amanda has configuration options for controlling almost all aspects of the backup operation and provides several scheduling methods. A typical configuration does periodic full dumps with partial dumps in between. There is also support for:

- Periodic archival backup, such as taking full dumps to a vault away from the primary site.
- Incremental-only backups where full dumps are done outside of *Amanda*, such as very active areas that must be taken offline, or no full dumps at all for areas that can easily be recovered from vendor media.
- Always doing full dumps, such as database areas that change completely between each run or critical areas that are easier to deal with during an emergency if they are a single-restore operation.

It's easy to support multiple configurations on the same tape server machine, such as a periodic archival configuration along side a normal daily configuration. Multiple configurations can run simultaneously on the same tape server if there are multiple tape drives.

Scheduling of full dumps is typically left up to *Amanda*. They are scattered throughout the dump cycle to balance the amount of data backed up each run. It's important to keep logs of where backup images are for each area (which *Amanda* does for you), since they are not on a specific, predictable, tape (e.g., the Friday tape will not always have a full dump of /usr for client A). The partial backup level is also left to *Amanda*. History information about previous levels is kept and the backup level automatically increases when sufficient dump size savings will be realized.

Amanda uses a simple tape management system and protects itself from overwriting tapes that still have valid dump images and from tapes not allocated to the configuration. Images may be overwritten when a client is down for an extended period or if not enough tapes are allocated, but only after *Amanda* has issued several warnings. *Amanda* can also be told to not reuse specific tapes.

A validation program may be used before each run to note potential problems during normal working hours when they are easier to correct. An activity report is sent via e-mail after each run. *Amanda* can also send a report to a printer and even generate sticky tape labels.

There is no graphical interface. For administration, there is usually only a single simple text file to edit, so this is not much of an issue. For security reasons, *Amanda* does not support user controlled file recovery. There is an ftp-like restore utility for administrators to make searching online dump catalogues easier when recovering individual files.

18.3 Future Capabilities of *Amanda*

In addition to the usual enhancements and fixes constantly being added by the *Amanda* Core Development Team, three main changes are in various stages of development.

- A new internal security framework will make it easier for developers to add other security methods, such as SSH (ftp://ftp.cs.hut.fi/pub/ssh/) and SSL (Secure Socket Layer).
- Another major project is a redesign of how *Amanda* runs the client dump program. This is currently hardcoded for a vendor dump program, **GNU-tar** or SAMBA tar. The new mechanism will allow arbitrary programs such as cpio, star, and possibly other backup systems. It will also add optional pre-dump and post-dump steps that can be used for locking and unlocking, and snapshots of rapidly changing data such as databases or the Windows registry.
- The third major project is a redesign of the output subsystem to support non-tape media such as CD-ROM, local files, remote files via tools like rcp and ftp, remote tapes, etc. It will also be able to split dump images across media, handle multiple simultaneous media of different types such as writing to multiple tapes or a tape and a CD-ROM, and handle writing copies of images to multiple media such as a tape to keep on site and a CD-ROM or duplicate tape for archiving.
- In addition, the output format will be enhanced to include a file-1 and a file-n. The idea is to put site-defined emergency recovery tools in file-1 (the first file on the output) that can be retrieved easily with standard non-*Amanda* programs like tar, then use those tools to retrieve the rest of the data. The file-n area is the last file on the output

and can contain items such as the *Amanda* database, which would be complete and up to date by the time file-n is written.

18.4 *Amanda* Resources

Amanda may be obtained via the web page <`http://www.amanda.org`> or with anonymous ftp at <`ftp://ftp.amanda.org/pub/amanda`>.A typical release is a gzip compressed tar file with a name like amanda-2.4.1.tar.gz, which means it is major version 2.4 and minor version 1. There are occasional patch releases that have a name like amanda-2.4.1p1.tar.gz (release 2.4.1 plus patch set 1). Beta test pre-releases have a names like amanda-2.5.0b3.tar.gz (third beta test pre-release of 2.5.0).

Some operating system distributions provide pre-compiled versions of *Amanda*, but because *Amanda* hardcodes some values into the programs, they may not match the configuration. Work is being done to move these values to run-time configuration files, but for now *Amanda* should be built from source.

The *Amanda* web page contains useful information about patches not yet part of a release, how to subscribe to related mailing lists, and pointers to mailing list archives. Subscribe to at least amanda-announce to get new release announcements or amanda-users to get announcements plus see problems and resolutions from other *Amanda* users. The amanda-users mailing list is a particularly good resource for help with initial setup as well as problems. When posting to it, be sure to include the following information:

- *Amanda* version
- OS version on the server and client(s)
- Exact symptoms seen, such as error messages, relevant sections of e-mail reports, debugging and log files
- Anything unusual or recent changes to the environment
- A valid return e-mail address

Finally, the docs directory in the release contains several files with helpful information, such as a FAQ.

18.5 Installing *Amanda*

After downloading and unpacking the *Amanda* release, read the README, docs/INSTALL, and docs/SYSTEM.NOTES files. They contain important and up-to-date information about how to set up *Amanda*.

18.5.1 Install Related Packages

Several other packages may be required to complete an *Amanda* install. Before continuing, you should locate and install packages your environment will need. In particular, consider the following:

GNU-tar 1.12 or later 8212; www.gnu.org The GNU version of the standard tar program with enhancements to do partial backups and omit selected files. It is one of the client backup programs *Amanda* knows how to use.

Samba 1.9.18p10 or later 8212; www.samba.org SAMBA is an implementation of the System Message Block (SMB) protocol used by Windows-based systems for file access. It contains a tool, smbclient, that *Amanda* can use to back them up.

Perl 5.004 or later 8212; www.perl.org Perl is a scripting programming language oriented toward systems programming and text manipulation. It is used for a few optional *Amanda* reporting tools and by some tape changers.

GNU readline 2.2.1 or later 8212; www.gnu.org The GNU readline library may be incorporated into interactive programs to provide command-line history and editing. It is built into the *Amanda* amrecover restoration tool, if available.

GNU awk 3.0.3 or later 8212; www.gnu.org The GNU version of the awk programming language contains a common version across platforms and some additional features. It is used for the optional *Amanda* amplot statistics tool.

Gnuplot 3.5 or later 8212; ftp://ftp.dartmouth.edu/pub/gnuplot/ This gnuplot library (which has nothing to do with the GNU tools, see the accompanying README) is a graph plotting package. It is used for the optional *Amanda* amplot statistics tool.

Be sure to look in the *Amanda* patches directory and the patches section on the web page for updates to these packages. SAMBA versions before 2.0.3, in particular, must have patches applied to make them work properly with Amanda. Without the patches, backups appear to work but the resulting images are corrupt.

When *Amanda* is configured, locations of additional software used on the clients, such as **GNU-tar** and SAMBA, get built into the *Amanda* programs, so additional software must be installed in the same place on the *Amanda* build machine and all the clients.

18.5.2 Perform Preliminary Setup

A typical *Amanda* configuration runs as a user other than root, such as backup or amanda, given just enough permissions to do backups. Often, direct login as the user is disallowed. To use the vendor dump program instead of **GNU-tar**, the *Amanda* user must be in a group with read access to the raw disk devices. Membership in this group should be tightly controlled since it opens up every file on the client for viewing.

There are two ways to link *Amanda* and the raw device group membership. Either put the *Amanda* user in the group that currently owns the raw devices, as the primary group or as a secondary, or pick a new group for *Amanda* and change the group ownership of the

devices. *Amanda* (actually, the vendor dump program) needs only read access, so turn off group write permission. Turn off all "world" access.

To use **GNU-tar**, *Amanda* runs it under a setuid-root program that grants the needed permissions. The GNU version of tar must be used with *Amanda*. Vendor supplied versions (unless they originated from GNU and are at least version 1.12) do not work because *Amanda* depends on additional features.

18.5.3 Configure the *Amanda* Build

Use the *Amanda* user and group for the –**with-user** and –**with-group** options to **./configure**. For instance, to use amanda for the user and backup as the group: **./configure –with-user=amanda –with-group=backup ...**

No other options are required for **./configure**, but all the possibilities may be seen with **./configure –help**. Don't get carried away changing options. The defaults are usually suitable and some require experience with *Amanda* to fully understand. Leave –**with-debugging** enabled so debug log files are created on the clients. They take very little space but are often necessary for tracking down problems.

The normal build creates both tape server and client software. The tape server host is often backed up by *Amanda* and needs the client parts. However, the clients usually do not need the tape server parts. A little disk space and build time may be saved by adding –**without-server** to the **./configure** arguments when building for them.

The default security mechanism uses a file formatted just like `.rhosts` but called `.amandahosts`. This keeps *Amanda* operations separate from normal **rsh/rcp** work that might use the same user. It is not recommended, but `.rhosts` and `hosts.equiv` may be used by adding –**without-amandahosts** to the **./configure** arguments.

The TCP ports used for data transfer may be restricted with –**with-portrange** to use *Amanda* between hosts separated by a firewall. A typical entry would be: **./configure –with-portrange=50000,50100 ...** This does not affect the initial UDP requests made from the tape server to the clients. The amanda UDP port (typically 10080) must be allowed through the firewall.

If more than just a few **./configure** options are used, they may be put in `/usr/local/share/config.site` or `/usr/local/etc/config.site` to keep them the same from build to build. An example is in `example/config.site`.

18.5.4 Build and Install *Amanda*

After `./configure` is done, run make to build *Amanda*, then make install to install it. The make install step must be done as root because some *Amanda* programs require system privileges. Unless the base location is changed, *Amanda* installs into these areas:

`/usr/local/sbin` Programs administrators run.

`/usr/local/lib` Libraries.

`/usr/local/libexec` Private programs only *Amanda* uses.

`/usr/local/man` Documentation.

Now is a good time to read the main *Amanda* man page. It provides an overview of *Amanda*, a description of each program, and detailed configuration information.

The following programs must be *setuid-root* (which **make install** as root does). The first group (*amcheck*,*dumper*, and *planner*) run on the tape server machine and need a privileged network port for secure communication with the clients. The others are utility routines optionally used on the clients, depending on the *dump* program used and operating system type.

`sbin/amcheck` *Amanda* sanity checker program

`libexec/dumper` Client communication program

`libexec/planner` Estimate gathering program

`libexec/killpgrp` Used to kill vendor dump programs that run as root

`libexec/rundump` Setuid wrapper for systems that need to run the vendor dump program as root

`libexec/runtar` Setuid wrapper to run **GNU-tar** as root

All these programs are installed with world access disabled and group access set to the *Amanda* group from –**with-group**. Be sure all members of that group are trustworthy since rundump and runtar in particular give access to every file on the system. If *Amanda* software is made available via NFS, be sure the mount options allow setuid programs. Also, if **GNU-tar** is used, root needs write access to `/usr/local/var/amanda/gnutar-lists` (or the –**with-gnutar-list** value to **./configure**) to store information about each partial level.

If the build has trouble or *Amanda* needs to be rebuilt, especially with different **./configure** options, the following sequence makes sure everything is cleaned up from the previous build: `make distclean ./configure ... make make install (as root)` Problems with the **./configure** step can sometimes be diagnosed by looking at the `config.log` file. It contains detailed output of tests **./configure** runs. Note that it is normal for many of the tests to "fail" as part of **./configure** determining how to access various features on the system.

A common problem when using the GNU C compiler is not re-installing it after the underlying operating system version changes. Gcc is particularly sensitive to system header files and must be re-installed or have its fixincludes step rerun (see the gcc release installation

notes) if the operating system is upgraded. Running gcc –**verbose** shows where gcc gets its information, and contains an indication of the operating system version expected.

Amanda needs changes to the network services and inetd configuration files. The `client-src/patch-system` script should be able to set up systems in most cases. It does not currently handle systems that deliver service entries via YP/NIS. If the script does not work, add the following entries to the services file (e.g., `/etc/services`) or YP/NIS map: `Amanda 10080/udp Amandaidx 10082/tcp Amidxtape 10083/tcp`

Each client needs an entry in the inetd configuration file (e.g., /etc/inetd.conf) like this, substituting the *Amanda* user for *Amanda* and the full path to the *Amanda* `libexec` directory for `PATH`: `amanda dgram udp wait Amanda /PATH/libexec/amandad amandad`

The *amanda* service is used by all *Amanda* controlling programs to perform functions on the clients.

The tape server host needs entries like these if the amrecover tool is to be used: `amandaidx stream tcp nowait Amanda /PATH/libexec/amindexd amindexd amidxtape stream tcp nowait Amanda /PATH/libexec/amidxtaped amidxtaped`

The amandaidx service provides access to the catalogues, while amidxtape provides remote access to a tape device. After every *inetd configuration* file change, send a HUP signal to the inetd process and check the system logs for errors.

18.5.5 Configuring *Amanda*

Once installed, *Amanda* must be configured to your environment.

18.5.6 Decide on a Tape Server

The first thing to decide is what machine will be the *Amanda* tape server. *Amanda* can be CPU-intensive if configured to do server compression, and almost certainly network and I/O-intensive. It does not typically use much real memory. It needs direct access to a tape device that supports media with enough capacity to handle the expected load.

To get a rough idea of the backup sizes, take total disk usage (not capacity), *Usage*, and divide it by how often full dumps will be done, *Runs*. Pick an estimated run-to-run change rate, *Change*. Each *Amanda* run, on average, does a full dump of *Usage/Runs*. Another Usage/Runs*Change is done of areas that got a full dump the previous run, *Usage/Runs*Change** is done of areas that got a full dump two runs ago, and so on.

For example, with 100 GB of space in use, a full dump every seven runs (e.g., days) and estimated run-to-run changes (new or altered files) of 5 percent:

```
100 GBytes / 7              = 14.3 GB
100 GBytes / 7 * 5%         =  0.7 GB
100 GBytes / 7 * 5% * 2     =  1.4 GB
100 GBytes / 7 * 5% * 3     =  2.1 GB
100 GBytes / 7 * 5% * 4     =  2.9 GB
100 GBytes / 7 * 5% * 5     =  3.6 GB
```

```
100 GBytes / 7 * 5% * 6      =  4.3 GB
                             = 29.3 GB
```

If 50 percent compression is expected, the actual amount of tape capacity needed for each run, which might be on more than one tape, would be 14.7 GB. This is very simplistic, and could be improved with greater knowledge of actual usage, but should be close enough to start with. It also gives an estimate of how long each run will take by dividing expected capacity by drive speed.

18.5.7 Decide Which Tape Devices to Use

Unix operating systems typically incorporate device characteristics into the file name used to access a tape device. The two to be concerned with are "rewind" and "compression." *Amanda* must be configured with the non-rewinding tape device, so called because when the device is opened and closed it stays at the same position and does not automatically rewind. This is typically a name with an *n* in it, such as `/dev/rmt/0n` or `/dev/nst0`. On AIX, it is a name with a .1 or .5 suffix.

Put the *Amanda* user in the group that currently owns the tape device, either as the primary group or as a secondary, or pick a new group for *Amanda* and change the group ownership of the device. *Amanda* needs both read and write access. Turn off all "world" access.

18.5.8 Decide Whether to Use Compression

Dump images may optionally be compressed on the client, the tape server, or the tape device hardware. Software compression allows *Amanda* to track usage and make better estimates of image sizes, but hardware compression is more efficient of CPU resources. Turn off hardware compression when using software compression on the client or server. See the operating system documentation for how hardware compression is controlled; on many systems it is done via the device file name just like the non-rewinding flag. AIX uses the **chdev** command.

18.5.9 Decide Where the Holding Space Will Be

If at all possible, allocate some holding disk space for *Amanda* on the tape server. Holding disk space can significantly reduce backup time by allowing several dumps to be done at once while the tape is being written. Also, for streaming tape devices, *Amanda* keeps the device going at speed, and that may increase capacity. *Amanda* may be configured to limit disk use to a specific value so it can share with other applications, but a better approach is to allocate one or more inexpensive disks entirely to *Amanda*.

Ideally, there should be enough holding disk space for the two largest backup images simultaneously, so one image can be coming into the holding disk while the other is being written to tape. If that is not practical, any amount that holds at least a few of the smaller images helps. The *Amanda* report for each run shows the size of the dump image after software compression (if enabled). That, in addition to the amplot and amstatus tools, may be used to tune the space allocated.

18.5.10 Compute Your Dump Cycle

Decide how often *Amanda* should do full dumps. This is the "dump cycle." Short periods make restores easier because there are fewer partials, but use more tape and time. Longer periods let *Amanda* spread the load better but may require more steps during a restore.

Large amounts of data to back up or small capacity tape devices also affect the dump cycle. Choose a period long enough that *Amanda* can do a full dump of every area during the dump cycle and still have room in each run for the partials. Typical dump cycles are one or two weeks. Remember that the dump cycle is an upper limit on how often full dumps are done, not a strict value. *Amanda* runs them more often and at various times during the cycle as it balances the backup load. It violates the limit only if a dump fails repeatedly, and issues warnings in the e-mail report if that is about to happen.

By default, *Amanda* assumes it is run every day. If that is not the case, set "runs per cycle" (described below) to a different value. For instance, a dump cycle of seven days and runs per cycle of five would be used if runs are done only on weekdays.

Normally, *Amanda* uses one tape per run. With a tape changer (even the `chg-manual` one), the number of tapes per run may be set higher for extra capacity. This is an upper limit on the number of tapes. *Amanda* uses only as much tape as it needs. *Amanda* does not yet do overflow from one tape to another. If it hits end of tape (or any other error) while writing an image, that tape is unmounted, the next one is loaded, and the image starts over from the beginning. This sequence continues if the image cannot fit on a tape.

Runs per cycle and tapes per run determine the minimum number of tapes needed, called the "tape cycle." To ensure the current run is not overwriting the last full dump, one more run should be included. For instance, a dump cycle of two weeks, with default runs per cycle of 14 (every day) and default tapes per run of one, needs at least 15 tapes (14+1 runs * one tape/run). Using two tapes per run needs 30 tapes (14+1 runs * two tapes/run). Doing backups just on weekdays with a dump cycle of two weeks, runs per cycle of 10, and two tapes per run needs 22 tapes (10+1 runs * two tapes/run).

More tapes than the minimum should be allocated to handle error situations. Allocating at least two times the minimum allows the previous full dump to be used if the most recent full dump cannot be read. Allocating more tapes than needed also goes back further in time to recover lost files. *Amanda* does not have a limit on the number of tapes in the tape cycle.

18.5.11 Copy and Edit the Default Configuration File

Pick a name for the configuration (the name `Daily` will be used for the rest of this section). Create a directory on the tape server machine to hold the configuration files, typically `/usr/local/etc/amanda/Daily`. Access to this directory (or perhaps its parent) should be restricted to the *Amanda* group or even just the *Amanda* user.

Each tape assigned to a configuration needs a unique label. For this example, we'll use the configuration name, a dash, and a three-digit suffix, `Daily-000` through `Daily-999`. Do not use blanks, tabs, slashes (/), shell wildcards, or non-printable characters.

Amanda limits network usage so backups do not take all the capacity. This limit is imposed when *Amanda* is deciding whether to perform a dump by estimating the throughput and

adding that to dumps that are already running. If the value exceeds the bandwidth allocated to *Amanda*, the dump is deferred until enough others complete. Once a dump starts, *Amanda* lets underlying network components do any throttling.

Copy the template `example/amanda.conf` file to the configuration directory and edit it. Full documentation is in the `amanda` man page. There are many parameters, but probably only a few need to be changed. Start with the following (some of which are described later):

org This string will be in the Subject line of *Amanda* e-mail reports.

mailto Target address for *Amanda* e-mail reports.

dumpuser Same as `--with-user` from `./configure`.

dumpcycle The dump cycle.

runspercycle The runs per cycle.

tapecycle The tape cycle.

runtapes Number of tapes to use per run.

tapedev The no-rewind tape device if a changer is not being used, or if the manual changer is being used.

tapetype Type of tape media.

netusage Network bandwidth allocated to *Amanda*.

labelstr A regular expression (**grep** pattern) used to make sure each tape is allocated to this *Amanda* configuration. Our example might use `Daily-[0-9][0-9][0-9].`

The following parameters probably do not need to be changed, but look at their values to know where *Amanda* expects to find things:

infofile Location of *Amanda* history database. Older versions of *Amanda* used this as the base name of a database file. Newer versions use this as a directory name.

logdir Directory where *Amanda* logs are stored.

indexdir Location of optional *Amanda* catalogue database.

18.5.12 Configure the Holding Disk

Define each holding disk in an `amanda.conf` `holdingdisk` section. If partitions are dedicated to *Amanda*, set the `use` value to a small negative number, such as -10 MB. This tells *Amanda* to use all but that amount of space. If space is shared with other applications, set the value to the amount *Amanda* may use, create the directory and set the permissions so only the *Amanda* user can access it.

Set a `chunksize` value for each holding disk. Positive numbers split dumps in the holding disk into chunks no larger than the chunksize value. Negative numbers are no longer supported. Even though the images are split in the holding disk, they are written to tape as a single image. At the moment, all chunks for a given image go to the same holding disk.

Older operating systems that do not support individual files larger than 2GB need a chunk size slightly smaller, such as 2000 MB, so the holding disk can still be used for very large dump images. Systems that support individual files larger than 2 GB should have a very large value, such as 2000 GBytes.

18.5.13 Configure Tape Devices and Label Tapes

Amanda needs to know some characteristics of the tape media. This is set in a `tapetype` section. The example `amanda.conf`, web page, and amanda-users mailing list archives have entries for most common media. Currently, all tapes should have the same characteristics. For instance, do not use both 60-meter and 90-meter DAT tapes since *Amanda* must be told the smaller value, and larger tapes may be underutilized.

If the media type is not listed and there are no references to it in the mailing list archives, go to the `tape-src` directory, **make tapetype**, mount a scratch tape in the drive and run **./tapetype NAME DEV** where *NAME* is a text name for the media and *DEV* is the no-rewind tape device with hardware compression disabled. This program rewinds the tape, writes random data until it fills the tape, rewinds, and then writes random data and tape marks until it fills the tape again. This can take a very long time (hours or days). When finished, it generates a new tapetype section to standard output suitable for adding to the `amanda.conf` file. Post the results to the *amanda-users* mailing list so others may benefit from your effort.

When using hardware compression, change the `length` value based on the estimated compression rate. This typically means multiplying by something between 1.5 and 2.0.

The `length` and `filemark` values are used by *Amanda* only to plan the backup schedule. Once dumps start, *Amanda* ignores the values and writes until it gets an error. It does not stop writing just because it reaches the tapetype length. *Amanda* does not currently use the tapetype speed parameter.

Once the `tapetype` definition is in `amanda.conf`, set the `tapetype` parameter to reference it.

Without special hardware to mount tapes, such as a robot or stacker, either set the `tapedev` parameter to the no-rewind device name or set up the *Amanda* `chg-manual` changer. The manual changer script prompts for tape mounts as needed. The prompts normally go to

the terminal of the person running *Amanda*, but the changer may be configured to send requests via e-mail or to some other system logging mechanism.

To configure the manual changer, set `tapedev` to the no-rewind tape device and set `tpchanger` to `chg-manual`. To send tape mount prompts someplace other than the terminal, which is necessary if *Amanda* is run from a **cron** job, see the request shell function comments in `changer-src/chg-manual.sh.in`.

Another common tape changer is `chg-multi`. This script can drive stackers that advance to the next tape when the drive is unloaded or it can use multiple tape drives on the tape sever machine to emulate a changer. The `chg-multi` script has a configuration file and a state file. Put the path to the configuration file in the `amanda.conf changerfile` parameter. There is a sample in `example/chg-multi.conf`. It has the following keyword/value pairs separated by whitespace:

firstslot Number of the first slot in the device.

lastslot Number of the last slot in the device.

gravity Set to 1 if the device is gravity fed and cannot go backwards, otherwise set to 0.

needeject Set to 1 if the tape needs to be ejected to advance to a new tape, otherwise set to 0.

multieject Set to 1 if sending multiple ejects causes the changer to advance through the tapes, otherwise set to 0. If set to 1, gravity must also be set to 1 because the script currently does not handle carousels that wrap back around to the first tape after the last one. Also, `needeject` must be set to 0.

ejectdelay Set to a number of seconds of extra delay after ejecting a tape if it takes a while before the next tape is ready.

statefile Set to the path to a file `chg-multi` builds and maintains with the current state of the changer.

slot Repeat as needed to define all the slots and corresponding tape devices. The first field after slot is the slot number. The next field is the no-rewind tape device name. For changers that have a single tape device, repeat the device name for each slot. To emulate a changer by using multiple tape devices, list a different no-rewind tape device for each slot.

`chg-multi` may also be used as a framework to write a new changer. Look for `XXX` comments in the script and insert calls to commands appropriate for the device. Make any source changes to the `changer-src/chg-multi.sh.in` file. That file is processed by **./configure**

to generate `chg-multi.sh`, which turns into `chg-multi` with **make**. If `chg-multi.sh` or `chg-multi` is altered, the changes will be lost the next time *Amanda* is rebuilt.

A third popular changer is `chg-scsi`. It can drive devices that have their own SCSI interface. An operating system kernel module may need to be installed to control such devices, like **sst** for Solaris, which is released with *Amanda*, or **chio**, available for various systems. As with `chg-multi`, set the `amanda.conf changerfile` parameter to the changer configuration file path. There is a sample in `example/chg-scsi.conf`. The initial section has parameters common to the entire changer:

number_configs Set to the number of tape drives connected to this changer. The default is 1.

eject Set to 1 if tape drives need an explicit eject command before advancing to the next tape, otherwise set to 0.

sleep Set to the number of seconds to wait for a tape drive to become ready.

changerdev Set to the device path of the changer. This may be set in the `amanda.conf` file instead of here if preferred. Following the common parameters is a section for each tape device:

config Set to the configuration number, starting with 0.

drivenum Set to the tape drive number, usually the same as the configuration number.

dev Set to the no-rewind device name of the tape drive.

startuse Set to the number of the first slot served by this drive.

enduse Set to the number of the last slot served by this drive.

statfile Set to the path to a file `chg-scsi` will build and maintain with the current state of this drive.

Test any changer setup with the **amtape** command. Make sure it can load a specific tape with the `slot NNN` suboption, eject the current tape with **eject** and advance to the next slot with **slot next**.

Tapes must be pre-labeled with **amlabel** so *Amanda* can verify the tape is one it should use. Run **amlabel** as the *Amanda* user, not root. For instance:

```
# su amanda -c "amlabel Daily Daily-123 slot 123"
```

18.5.14 Configure Backup Clients

After tapes are labeled, pick the first client, often the tape server host itself, and the filesystems or directories to back up. For each area to back up, choose either the vendor **dump** program or **GNU-tar**. Vendor **dump** programs tend to be more efficient and do not disturb files being dumped, but are usually not portable between different operating systems. **GNU-tar** is portable and has some additional features, like the ability to exclude patterns of files, but alters the last access time for every file backed up and may not be as efficient. **GNU-tar** may also deal with active filesystems better than vendor **dump** programs, and is able to handle very large filesystems by breaking them up by subdirectories.

Choose the type of compression for each area, if any. Consider turning off compression of critical areas needed to bring a machine back from the dead in case the decompression program is not available. Client compression spreads the load to multiple machines and reduces network traffic, but may not be appropriate for slow or busy clients. Server compression increases the load on the tape server machine, possibly by several times since multiple dumps are done at once. For either, if GNU **GNU-zip** is used, compression may be set to `fast` for faster but less aggressive compression or *best* for slower but more aggressive compression. Set `compression` to *none* to disable software compression or use hardware compression.

Pick or alter an existing `dumptype` that matches the desired options, or create a new one. Each `dumptype` should reference the *global* `dumptype`. It is used to set options for all other `dumptypes`. For instance, to use the indexing facility, enable it in the *global* `dumptype` and all other dumptypes will inherit that value.

The indexing facility generates a compressed catalogue of each dump image. These are useful for finding lost files and are the basis of the **amrecover** program. Long dump cycles or areas with many or very active files can cause the catalogues to use a lot of disk space. *Amanda* automatically removes catalogues for images that are no longer on tape.

Create a file named `disklist` in the same directory as `amanda.conf` and either copy the file from `example/disklist` or start a new one. Make sure it is readable by the *Amanda* user. Each line in `disklist` defines an area to be backed up. The first field is the client host name (fully qualified names are recommended), the second is the area to be backed up on the client and the third is the dumptype. The area may be entered as a disk name, `( sd0a)`, a device name, `(/dev/rsd0)`a, or a logical name, `(/usr)`. Logical names make it easier to remember what is being backed up and to deal with disk reconfiguration.

To set up a Windows client, set the host name to the name of the Unix machine running SAMBA and the area to the Windows share name, such as *//some-pc/C$*. Note that Unix-style forward slashes are used instead of Windows-style backward slashes.

Enable *Amanda* access to the client from the tape server host (even if the client is the tape server host itself) by editing `.amandahosts` (or `.rhosts`, depending on what was set with **./configure**) in the *Amanda* user home directory on the client. Enter the fully qualified tape server host name and *Amanda* user, separated by a blank or tab. Make sure the file is owned by the *Amanda* user and does not allow access to anyone other than the owner (e.g. mode 0600 or 0400).

For Windows clients, put the share password in `/etc/amandapass` on the SAMBA host. The first field is the Windows share name, the second is the clear text password and the optional third field is the domain.

NOTE

This info isn't correct anymore. Please refer to Backup PC hosts using Samba for details on this file.

Because this file contains clear text passwords, it should be carefully protected, owned by the *Amanda* user and only allow user access. By default, *Amanda* uses SAMBA user *backup*. This can be changed with `--with-samba-user` to **./configure**.

18.5.15 Test and Debug Setup

Test the setup with **amcheck**. As with all *Amanda* commands, run it as the *Amanda* user, not root:

```
# su amanda -c "amcheck Daily"
```

Many errors reported by **amcheck** are described in `docs/FAQ` or the **amcheck** man page. The most common error reported to the *Amanda* mailing lists is selfcheck request timed out, meaning **amcheck** was not able to talk to **amandad** on the client. In addition to the ideas in `docs/FAQ`, here are some other things to try:

- Are the *Amanda* services listed properly in `/etc/services` or a YP/NIS map? The C program in Figure 4-1 uses the same system call as *Amanda* to look up entries:

Run it on both the tape server and client and make sure the port numbers match:

```
$ cc check-service.c -lnsl -lsocket (Solaris)
$ a.out amanda udp
  amanda/udp: 10080
$ a.out amandaidx
  amandaidx/tcp: 10082
$ a.out amidxtape
  amidxtape/tcp: 10083
```

- Is there a line in the inetd configuration file on the client to start **amandad**?
- Was inetd sent a `HUP` signal after the configuration file was changed?

Example 18.5.1 A C Program to Check the *Amanda* Service Numbers

```
 #include <stdio.h>
 #include <string.h>
 #include <netdb.h>

 main (
     int argc,
char **argv)
 {
     char *pn;
char *service;
char *protocol = "tcp";
struct servent *s;

if ((pn = strrchr (*argv, '/')) == NULL) {
    pn = *argv;
} else {
    pn++;
     } if (argc < 2) {
      fprintf (stderr, "usage: %s service [protocol]\n", pn);
      return 1;
}
service = *++argv;
if (argc > 2) {
protocol = *++argv;
}
if ((s = getservbyname (service, protocol)) == NULL) {
    fprintf (stderr, "%s: %s/%s lookup failed\n", pn,
      service, protocol);
    return 1;
}
printf ("%s/%s: %d\n", service, protocol,
  (int) ntohs (s->s_port));
return 0;
 }
```

- Are there system log messages from inetd about amanda or **amandad**? For instance, inetd complains if it cannot look up the *Amanda* services.
- Is `/tmp/amanda/amandad/debug` being updated?
- Is the access time on the **amandad** executable **(ls -lu)** being updated? If not, inetd is probably not able to run it, possibly because of an error in the inetd configuration file or a permission problem.

- Run the **amandad** program by hand as the *Amanda* user on the client. It should sit for about 30 seconds, then terminate. Enter the full path exactly as it was given to inetd, perhaps by using copy/paste.

Do not proceed until **amcheck** is happy with the configuration.

For initial testing, set the `record` option to *no* in the *global* dumptype, but remember to set it back to `yes` when *Amanda* goes into normal production. This parameter controls whether the **dump** program on the client updates its own database, such as `/etc/dumpdates` for vendor **dump**.

To forget about an individual test run, use **amrmtape** to remove references to the tapes used, then use **amlabel** to relabel them. To completely start over, remove the files or directories named in the `infofile` and `indexdir` parameters, the `tapelist` file named in the `tapelist` parameter, all `amdump.*` files in the configuration directory and all `log.*` files in the directory named by the `logdir` parameter. These files contain history information *Amanda* needs between runs and also what is needed to find particular *dump* images for restores and should be protected when *Amanda* goes into production.

18.6 Operating Amanda

Once configured, you will need to setup the automated use of *Amanda*.

18.6.1 Run amdump

The **amdump** script controls a normal *Amanda* backup run. However, it's common to do site-specific things as well with a wrapper shell script around **amdump**. **amdump** is meant to run unattended from **cron**. See the operating system documentation for how to set up a **cron** task. Be sure it runs *as the Amanda user*, not root or the installer.

The **amdump** script does the following:

- If a file named `hold` is in the configuration directory, **amdump** pauses until it goes away. This may be created and removed by hand to temporarily delay *Amanda* runs without having to change the **cron** task.
- If it looks like another copy of **amdump** is running, or a previous run aborted, **amdump** logs an error and terminates. If an earlier run aborted, **amcleanup** must be run. An **amcleanup** step should be added to the tape server system boot sequence to handle crashes. No backups can be performed after an abort or crash until **amcleanup** is run.
- The *Amanda* **planner** program decides what areas to back up and at what level. It does this by connecting to each client and getting estimated sizes of a full dump, the same partial level that was done on the previous run and possibly the next partial level. All clients are done in parallel, but it can take a while to gather all this information.
- The schedule is then passed to the **driver** program that controls actual dumping. It, in turn, starts up several **dumper** processes (based on the `inparallel` `amanda.conf` parameter) and a single **taper** process. The **taper** process splits into two parts, a reader and a writer, to keep streaming tape drives busy.

- **driver** commands **dumpers** to start backups, telling each its client, area, options such as compression and whether the result should go to the holding disk or direct to tape. Each **dumper** connects to **amandad** on the client and sends a request describing the dump program to run and options such as whether to do compression or indexing. The image comes back to the **dumper** who writes it, possibly via the server compression program, into the holding disk or directly to a **taper** connection. If enabled, **dumper** also collects catalogue information generated on the client and compresses it into the `indexdir` area. The **driver** also commands **taper** to write files from the holding disk to tape or to prepare to receive an image directly from a **dumper**.
- After backups are done, **amreport** is run to generate the e-mail report. It also renames the log file for the run to a unique `log.YYYYMMDD.N` name.
- Old `amdump.NN` debug log files are rolled so only enough to match the tape cycle are retained.
- The **amtrmidx** program is run to remove old catalogues if indexing has been used.

There are several ways to determine which tapes *Amanda* will need for a run. One is to look at the *Amanda* e-mail report from the previous run. The tapes used during that run and those expected for the next run are listed. Another is to run **amcheck** during normal working hours. In addition to showing which tapes are needed, it makes sure things are set up properly so problems can be fixed before the real *Amanda* run. A third is to use the `tape` suboption of **amadmin**. Without a tape changer, *Amanda* expects the first tape to be mounted in the drive when it starts. Automated tape changers should be able to locate the tapes. The **chg-manual** changer prompts for the tapes.

18.6.2 Read *Amanda*'s Reports

An *Amanda* report has several sections:

```
These dumps were to tape Daily-009, Daily-010
Tonight's dumps should go onto 2 tapes: Daily-011, Daily-012.
```

This shows which tapes were used during the run and which tapes are needed next.

```
FAILURE AND STRANGE DUMP SUMMARY:
  gurgi.cc.p /var lev 0 FAILED [Request to gurgi.cc.purdue.edu timed out.]
  gurgi.cc.p / lev 0 FAILED [Request to gurgi.cc.purdue.edu timed out.]
  pete.cc.pu /var/mail lev 0 FAILED ["data write: Broken pipe"]
  samba.cc.p //nt-test.cc.purdue.edu/F$ lev 1 STRANGE
  mace.cc.pu /master lev 0 FAILED [dumps too big, but cannot incremental dump n
  disk]
```

Problems found during the run are summarized in this section. In this example:

- *gurgi.cc.purdue.edu* was down, so all its backups failed.
- The `/var/mail` problem on *pete.cc.purdue.edu* and *F$* problem on *nt-test.cc.purdue.edu* are detailed later.
- The `/master` area on *mace.cc.purdue.edu* is new to *Amanda* so a full dump is required, but it would not fit in the available tape space for this run.

```
STATISTICS:
                          Total           Full         Daily
                        --------        --------     --------
  Dump Time (hrs:min)       5:03            3:23         0:33   (0:14 start,
  Output Size (meg)        20434.4         17960.0      2474.4
  Original Size (meg)      20434.4         17960.0      2474.4
  Avg Compressed Size (%)     --              --           --
  Tape Used (%)             137.4           120.0         17.4    (level:#dis
  Filesystems Dumped           90              21           69    (1:64 2:2 3
  Avg Dump Rate (k/s)       1036.5          1304.3        416.2
  Avg Tp Write Rate (k/s)   1477.6          1511.2       1271.9
```

This summarizes the entire run. It took just over five hours, almost 3.5 hours writing full dumps and about half an hour for partials. It took 14 minutes to get started, mostly in the **planner** step getting the estimates, and **taper** was idle almost one hour waiting on dumps to come into the holding disk.

In this example, hardware compression was used so Avg Compressed Size is not applicable and Output Size written to tape matches Original Size from the clients. About 137% of the length of the tape as defined in the `tapetype` was used (remember that two tapes were written), 120% for full dumps and 17% for partials. The Rate lines give the dump speed from client to tape server and tape writing speed, all in KBytes per second. The Filesystems Dumped line says 90 areas were processed, 21 full dumps and 69 partials. Of the partials, 64 were level 1, two were level 2 and three were level 3.

```
FAILED AND STRANGE DUMP DETAILS:

      /-- pete.cc.pu /var/mail lev 0 FAILED ["data write: Broken pipe"]
  sendbackup: start [pete.cc.purdue.edu:/var/mail level 0]
  sendbackup: info BACKUP=/usr/sbin/ufsdump
  sendbackup: info RECOVER_CMD=/usr/sbin/ufsrestore -f... -
  sendbackup: info end
  | DUMP: Writing 32 Kilobyte records
  | DUMP: Date of this level 0 dump: Sat Jan 02 02:03:22 1999
  | DUMP: Date of last level 0 dump: the epoch
```

```
  | DUMP: Dumping /dev/md/rdsk/d5 (pete.cc.purdue.edu:/var/mail) to standard ou
  | DUMP: Mapping (Pass I) [regular files]
  | DUMP: Mapping (Pass II) [directories]
  | DUMP: Estimated 13057170 blocks (6375.57MB) on 0.09 tapes.
  | DUMP: Dumping (Pass III) [directories]
  | DUMP: Dumping (Pass IV) [regular files]
  | DUMP: 13.99% done, finished in 1:02
  | DUMP: 27.82% done, finished in 0:52
  | DUMP: 41.22% done, finished in 0:42

  /-- samba.cc.p //nt-test.cc.purdue.edu/F$ lev 1 STRANGE
  sendbackup: start [samba.cc.purdue.edu://nt-test/F$ level 1]
  sendbackup: info BACKUP=/usr/local/bin/smbclient
  sendbackup: info RECOVER_CMD=/usr/local/bin/smbclient -f... -
  sendbackup: info end
  ? Can't load /usr/local/samba-2.0.2/lib/smb.conf - run testparm to debug it
  | session request to NT-TEST.CC.PURD failed
  |                 directory \top\
  |                 directory \top\Division\
  |        238 (    2.7 kb/s) \top\Division\contract.txt
  |      19456 ( 169.6 kb/s)  \top\Division\stuff.doc
...
```

Failures and unexpected results are detailed here. The dump of `/var/mail` would not fit on the first tape so was aborted and rerun on the next tape, as described further in the next section.

The dump of F$ on *nt-test.cc.purdue.edu* failed due to a problem with the SAMBA configuration file. It's marked STRANGE because the line with a question mark does not match any of the regular expressions built into *Amanda*. When dumping Windows clients via SAMBA, it's normal to get errors about busy files, such as `PAGEFILE.SYS` and the registry. Other arrangements should be made to get these safely backed up, such as a periodic task on the PC that creates a copy that will not be busy at the time *Amanda* runs.

```
NOTES:
  planner: Adding new disk j.cc.purdue.edu:/var.
  planner: Adding new disk mace.cc.purdue.edu:/master.
  planner: Last full dump of mace.cc.purdue.edu:/src on tape Daily-012 overwrit
           in 2 runs.
  planner: Full dump of loader.cc.purdue.edu:/var promoted from 2 days ahead.
  planner: Incremental of sage.cc.purdue.edu:/var bumped to level 2.
  taper: tape Daily-009 kb 19567680 fm 90 writing file: short write
  taper: retrying pete.cc.purdue.edu:/var/mail.0 on new tape: [writing file: sh
         write]
  driver: pete.cc.purdue.edu /var/mail 0 [dump to tape failed, will try again]
```

```
taper: tape Daily-010 kb 6201216 fm 1 [OK]
```

Informational notes about the run are listed here. The messages from planner say:

- There are new `disklist` entries for *j.cc.purdue.edu* and *mace.cc.purdue.edu.*
- Tape `Daily-012` is due to be overwritten in two more runs and contains the most recent full dump of `/src` from *mace.cc.purdue.edu*, so the tape cycle may not be large enough.
- The next scheduled full dump of /var on loader.cc.purdue.edu was moved up two days to improve the load balance.
- The partial dump of /var on sage.cc.purdue.edu was bumped from level 1 to level 2 because the higher level was estimated to save enough space to make it worthwhile.

The rest of the notes say **taper** was not able to write as much data as it wanted, probably because of hitting end of tape. Up to that point, it had written 19567680 KBytes in 90 files on tape `Daily-009`. Another attempt at the full dump of `/var/mail` from *pete.cc.purdue.edu* was made on the next tape (`Daily-010`) and it succeeded, writing 6201216 KBytes in one file.

```
DUMP SUMMARY:

                                 DUMPER STATS                    TAPER STATS
HOSTNAME  DISK           L   ORIG-KB   OUT-KB COMP%   MMM:SS   KB/s  MMM:SS   KB
--------------------------- ---------------------------------------- ------------
boiler.cc /              1      2624     2624   --      0:13  200.1    0:02 1076
boiler.cc /home/boiler/a 1       192      192   --      0:07   26.7    0:02  118
boiler.cc /usr           1       992      992   --      0:41   24.2    0:02  514
boiler.cc /usr/local     1       288      288   --      0:09   31.2    0:04   86
boiler.cc /var           1       425     4256   --      0:21  205.9    0:04 1104
egbert.cc /              1     41952    41952   --      1:26  487.3    0:37 1149
egbert.cc /opt           1       224      224   --      0:06   37.5    0:02  136
egbert.cc -laris/install 1        64       64   --      0:11    5.8    0:02   49
gurgi.cc. /              0    FAILED ----------------------------------------
gurgi.cc. /var           0    FAILED ----------------------------------------
pete.cc.p /              1     13408    13408   --      0:41  328.2    0:08 1600
pete.cc.p /opt           1      3936     3936   --      1:04   61.2    0:03 1382
pete.cc.p /usr           1      1952     1952   --      0:29   67.0    0:03  584
pete.cc.p /var           1    300768   300768   --      2:33 1963.8    2:50 1768
pete.cc.p /var/mail      0   6201184  6201184   --     73:45 1401.3   73:47 1400

...
(brought to you by Amanda version 2.4.1p1)
```

This section (which has been abbreviated) reports each area dumped showing client, area, backup level, sizes, time to dump and time to write to tape. Entries are in alphabetic order by client and then by area. This is not the same as the tape order. Tape order can be determined with the **find** or **info** suboption of the **amadmin** command, **amtoc** can generate a tape table of contents after a run, or **amreport** can generate a printed listing. By default, client names are truncated on the right, area names on the left, to keep the report width under 80 character. This typically leaves the unique portions of both.

Two log files are created during an *Amanda* run. One is named `amdump.NN`, where `NN` is a sequence number (1 is most recent, 2 is next most recent, etc), and is in the same directory as amanda.conf. The file contains detailed step by step information about the run and is used for statistics by **amplot** and **amstatus**, and for debugging. The other file is named `log.YYYYMMDD.N` where `YYYYMMDD` is the date of the *Amanda* run and `N` is a sequence number in case more than one run is made on the same day (0 for the first run, 1 for the second, etc). This file is in the directory specified by the `logdir amanda.conf` parameter. It contains a summary of the run and is the basis for the e-mail report. In fact, **amreport** may be run by hand and given an old file to regenerate a report.

Old `amdump.NN` files are removed by the **amdump** script. Old `log.YYYYMMDD.N` files are not automatically removed and should be cleared out periodically by hand. Keeping a full tape cycle is a good idea. If the tape cycle is 40 and *Amanda* is run once a day, the following command would do the job:

```
#find log.????????.* -mtime +40 -print | xargs rm
```

If `--with-pid-debug-files` was used on **./configure**, clients accumulate debug files in `/tmp/amanda` (or whatever `--with-debug` was set to) and should be cleaned out periodically. Without this option, client debug files have fixed names and are reused from run to run.

18.6.3 Monitor Tape and Holding Disk Status

While **amdump** is running, **amstatus** can track how far along it is. **amstatus** may also be used afterward to generate statistics on how many *dumpers* were used, what held things up and so on.

When a tape error happens on the last tape allowed in a run (as set by **runtapes**), *Amanda* continues to do backups into the holding disks. This is called *degraded* mode. By default, full dumps are not done and any that were scheduled have a partial done instead. A portion of the holding disk area may be allocated to do full dumps during degraded mode by reducing the value of the parameter *reserve* in `amanda.conf` below 100%.

A tape server crash may also leave images in the holding disks. Run **amflush**, as the *Amanda* user, to flush images in the holding disk to the next tape after correcting any problems. It goes through the same tape request mechanism as **amdump**. If more than one set of dumps are in the holding disk area, **amflush** prompts to choose one to write or to write them all. **amflush** generates an e-mail report just like **amdump**.

Operating systems vary in how they report end of tape to programs. A no space or short write error probably means end of tape. For I/O error, look at the report to see how much was written. If it is close to the expected tape capacity, it probably means end of tape, otherwise it means a real tape error happened and the tape may need to be replaced the next time through the tape cycle.

To swap out a partially bad tape, wait until it is about to be used again so any valid images can still be retrieved. Then swap the tapes, run **amrmtape** on the old tape and run **amlabel** on the replacement so it has a proper *Amanda* label.

If a tape is marked to not be reused with the `no-reuse` suboption of **amadmin**, such as one that has been removed or is failing, *Amanda* may want a freshly labeled tape on the next run to get the number of tapes back up to the full tape cycle.

If a tape goes completely bad, use **amrmtape** to make *Amanda* forget about it. As with marking a tape `no-reuse`, this may reduce the number of tapes *Amanda* has in use below the tape cycle and it may request a newly labeled tape on the next run.

18.6.4 Adding Tapes at a Particular Position in the Cycle

- Run **amlabel** on the new tapes.
- Edit the `tapelist` file by hand and move the new tapes *before* the tape to be used just ahead of them. For instance, move *Daily-100* before *Daily-099*.
- Set the date stamp on the new tapes to the same as the previous tape, e.g. make them the same for *Daily-099* and *Daily-100*.
- Update the `tapecycle` `amanda.conf` parameter if new tapes are being added.

These steps let *Amanda* know about all tapes, including those that do not have data yet. When the cycle gets to the last old tape (*Daily-099*), the next tape used will be the first new one (*Daily-100*). A new option is planned for **amlabel** to do these steps automatically.

18.6.5 Miscellanous Operational Notes

Multiple **amdump** runs may be made in the same day, although catalogues are currently stored without a timestamp so **amrecover** may not show all restore possibilities. To redo a few areas that failed during the normal run, edit the `disklist` file by hand to comment out all the other entries, run **amdump**, then restore the `disklist` file.

Use the `force` suboption of **amadmin** to schedule a full dump of an area on the next run. Run this *as the Amanda user*, not root. *Amanda* automatically detects new `disklist` entries and schedules an initial full dump. But for areas that go through a major change, such as an operating system upgrade or full restore, force *Amanda* to do a full dump to get things back into sync.

Amanda does not automatically notice new client areas, so keep the `disklist` in sync by hand. *Amanda* usually notices areas that are removed and reports an error as a reminder to remove the entry from the `disklist`. Use the `delete` suboption of **amadmin** (as the

Amanda user) to make *Amanda* completely forget about an area, but wait until the information is not needed for restores. This does not remove the entry from the `disklist` file 8212; that must be done by hand.

Non8212;*Amanda* backups may still be done with *Amanda* installed, but do not let the client **dump** program update its database. For vendor **dump** programs, this usually means not using the `u` flag, or saving and restoring `/etc/dumpdates`. For **GNU-tar** it means the `--listed-incremental` flag (if used) should not point to the same file *Amanda* uses.

As with all backup systems, verify the resulting tapes, if not each one then at least periodically or by random sample. The **amverify** script does a reasonably good job of making sure tapes are readable and images are valid. For **GNU-tar** images, the test is very good. For vendor **dump** images of the same operating system type as the tape server machine, the test is OK but does not really check the whole image due to the limited way the catalogue option works. For vendor dump images from other operating systems, **amverify** can tell if the image is readable from tape but not whether it is valid.

Tape drives are notorious for being able to read only what they wrote, so run **amverify** on another machine with a different drive, if possible, so an alternate is available if the primary drive fails. Make a copy of the *Amanda* configuration directory on the other machine to be able to run **amverify**. This copy is also a good way to have a backup of the *Amanda* configuration and database in case the tape server machine needs to be recovered.

18.7 Advanced *Amanda* Configuration

Once you have *Amanda* running for a while, you may choose to do some additional advanced configuration.

18.7.1 Adjust the Backup Cycle

Several `dumptype` parameters control the backup level *Amanda* picks for a run:

dumpcycle Maximum days between full dumps.

strategy nofull Never schedule (or run) a full dump.

strategy incronly Only schedule non-full dumps.

Note that **dumpcycle** is both a general `amanda.conf` parameter and a specific **dumptype** parameter. The value in a specific **dumptype** takes precedence. To handle areas that change significantly between each run and should get a full dump each time (such as the mail spool on a busy e-mail server or a database area), create a **dumptype** based on another **dumptype** with attributes changed as desired (client **dump** program, compression, etc) and set **dumpcycle** in the new **dumptype** to 0:

```
define mail-spool {
```

```
    comp-user-tar
dumpcycle 0
 }
```

To run full dumps by hand outside of *Amanda* (perhaps they are too large for the normal tape capacity, or need special processing), create a new **dumptype** and set `strategy` to `incronly`:

```
define full-too-big {
    comp-user-tar
    strategy incronly
}
```

Tell *Amanda* when a full dump of the area has been done with the `force` suboption of **amadmin**. Take care to do full dumps often enough that the tape cycle does not wrap around and overwrite the last good non-full backups.

To never do full dumps (such as an area easily regenerated from vendor media), create a new **dumptype** and set `strategy` to `nofull`:

```
define man-pages {
  comp-user-tar
strategy nofull
 }
```

Only level 1 backups of such areas are done, so wrapping around the tape cycle is not a problem.

To do periodic archival full dumps, create a new *Amanda* configuration with its own set of tapes but the same `disklist` as the normal configuration (e.g. symlink them together). Copy `amanda.conf`, setting all **dumpcycle** values to `0` and `record` to `no`, e.g. in the global **dumptype**. If a changer is used, set `runtapes` very high so tape capacity is not a planning restriction. Disable the normal *Amanda* run, or set the *hold* file as described in "Operating *Amanda*", so *Amanda* does not try to process the same client from two configurations at the same time.

18.7.2 Adjust Parallelism

Amanda starts several **dumper** processes and keeps as many as possible running at once. The following options control their activity:

inparallel Total number of dumpers.

maxdumps Maximum dumpers for a single client.

The default `maxdumps` is one, meaning only one **dumper** is assigned to a client at a time. If a client can support the load, increase `maxdumps` so more than one dump on that client is running at once. Note that `maxdumps` is both a general `amanda.conf` parameter and a specific **dumptype** parameter. The value in a specific **dumptype** takes precedence.

Field four of the `disklist` file is a "spindle number". Areas with the same non-negative spindle number are not backed up at the same time if `maxdumps` is greater than one. This prevents thrashing on an individual physical disk. Set spindle number to -1 (which is the default) for independent areas that can be done in conjunction with any other area, such as a whole physical disk. If the tape server has multiple network connections, an `amanda.conf` `interface` section may be set up for each one and clients allocated to a particular interface with field five of the `disklist`. Individual interfaces take precedence over the general `netusage` bandwidth limit and follow the same guidelines described above in "Configuring *Amanda*": the limit is imposed when deciding whether to start a dump, but once a dump starts, *Amanda* lets underlying network components do any throttling.

Individual *Amanda* `interface` definitions do not control which physical connection is used. That is left up to the operating system network software. While it's common to give an *Amanda* interface definition the same name as a physical connection, e.g. `le0`, it might be better to use logical names such as `back-door-atm` to avoid confusion.

The `starttime` **dumptype** parameter delays a backup some amount of time after *Amanda* is started. The value is entered as *HHMM*, so `230`, for instance, would wait 2.5 hours. This may be used to delay backups of some areas until they are known to be idle.

18.7.3 Monitor for Possible Improvements

amstatus may be used to get a summary of dumper activity:

```
# su amanda -c "amstatus Daily --file amdump.1 --summary"
...
 dumper0  busy  :  5:52:01  ( 98.03%)
 dumper1  busy  :  0:23:09  (  6.45%)
 dumper2  busy  :  0:13:27  (  3.75%)
 dumper3  busy  :  0:16:13  (  4.52%)
 dumper4  busy  :  0:06:40  (  1.86%)
 dumper5  busy  :  0:03:39  (  1.02%)
   taper  busy  :  3:54:20  ( 65.26%)
     0 dumpers busy  :  0:03:21  (  0.93%)       file-too-large:  0:03:21  (100.00%
1 dumper  busy  :  4:03:22  ( 67.78%)         no-diskspace:  3:40:55  ( 90.77%)
                                             file-too-large:  0:21:13  (  8.72%)
                    no-bandwidth:  0:01:13  (  0.50%)
```

```
  2 dumpers busy  :  0:17:33  (  4.89%)         no-bandwidth:  0:17:33  (100.00%)
  3 dumpers busy :  0:07:42  (  2.14%)        no-bandwidth:  0:07:42  (100.00%)
  4 dumpers busy :  0:02:05  (  0.58%)        no-bandwidth:  0:02:05  (100.00%)
  5 dumpers busy :  0:00:40  (  0.19%)        no-bandwidth:  0:00:40  (100.00%)
  6 dumpers busy :  0:03:33  (  0.99%)            not-idle:  0:01:53  (  53.10%)
                                                 no-dumpers:  0:01:40  ( 46.90%)
```

This says:

- **dumper** 0 was busy almost all the time.
- **dumper** 1 (and above) were not used very much.
- **taper** was busy about 2/3 of the total run time.
- All **dumpers** were idle less than 1% of the total runtime.
- One **dumper** was busy 67.78% of the total run time and the reason two **dumpers** were not started when one was busy was not enough holding disk space (`no-diskspace`) 90.77% of that time, the next image to dump was too large to fit in the holding disk at all (`file-too-large`) 8.72% of that time and network bandwidth was exhausted (`no-bandwidth`) 0.50% of that time

This configuration would benefit from additional holding disk space, which would allow more **dumpers** to run at once and probably keep **taper** busy more of the time.

Other common status indicators are:

not-idle Everything is running that can be.

no-dumpers All **dumpers** are busy and there are other dumps that could be started.

client-constrained The maximum number of **dumpers** for remaining clients are already running, or all spindles are already in use.

start-wait All remaining dumps are delayed until a specific time of day.

If the tape server machine has multiple tape drives, more than one *Amanda* configuration may run at the same time. Clients and holding disks should be assigned to only one configuration, however.

Amanda waits a fixed amount of time for a client to respond with dump size estimates. The default is five minutes per area on the client. For instance, if a client has four areas to back up (entries in `disklist`), *Amanda* waits at most 20 minutes for the estimates. During dumping, *Amanda* aborts a dump if the client stops sending data for 30 minutes. Various conditions, such as slow clients, which **dump** program is used and characteristics of the area, may cause timeouts. The values may be changed with the `amanda.conf etimeout` parameter for estimates and `dtimeout` for data. Positive `etimeout` values are multiplied by

the number of areas. The absolute value of a negative number is used for the whole client regardless of the number of areas.

18.7.4 Excluding Files

GNU-tar can exclude items from the dump image based on file name patterns controlled by the `dumptype exclude` parameter. A single pattern may be put on the `exclude` line itself or multiple patterns may be put in a file *on the client*. The `dumptype exclude` line in that case includes a `list` keyword and the path to the file.

Exclusion entries are shell-style wildcard expressions except * matches through any number of / characters. If a matched item is a directory, it and all its contents are omitted. For instance:

`./usr` Omit the `usr` directory at the top level of the area and everything under it.

`core` Omit all items named `core`.

`*/core*` Omit all items starting with *core*, e.g. `core`, `core19970114`, `corespondent`, or `corexx/somefile` (probably not a good idea).

`*/test*.c` Omit all items starting with `test` and ending with `.c`, e.g. `test.c`, `testing.c` or `testdir/pgm/main.c` (probably not a good idea).

`*.o` Omit all items ending with `.o`.

`*/OLD/*` Omit all items within directories named `OLD`, including subdirectories and their contents, but dump the `OLD` directory entry itself.

18.8 Restoring with *Amanda*

Remember that no one cares if you can back up ?only if you can restore.

18.8.1 Configuring and Using amrecover

One way to restore items with *Amanda* is with **amrecover** on the client. Before **amrecover** can work, *Amanda* must run with the `dumptype index` parameter set to `yes` and the **amindexd** and **amidxtaped** services must be installed and enabled to inetd, usually on the tape server machine (the default build sequence installs them). Also, add the client to `.amandahosts` (or `.rhosts`) for the *Amanda* user on the server machine. Since **amrecover** must run as root on the client, the entry must list root as the remote user, not the *Amanda* user. **amrecover** should not be made setuid-root because it would open up catalogues of the entire system to everyone.

For this example, user *jj* has requested two files, both named `molecule.dat`, in subdirectories named `work/sample-21` and `work/sample-22` and said they want the versions last modified on the 13th of January. Become root on the client, `cd` to the area and start **amrecover**:

```
$ su
Password:
# cd ~jj
# amrecover Daily
AMRECOVER Version 2.4.1p1.
Contacting server on amanda.cc.purdue.edu ...
220 amanda Amanda index server (2.4.1p1) ready.
200 Access OK
Setting restore date to  today (1999-01-18)
200 Working date set to 1999-01-18.
200 Config set to Daily.
200 Dump host set to pete.cc.purdue.edu.
$CWD '/home/pete/u66/jj' is on disk '/home/pete/u66' mounted at '/home/pete/u66'.
200 Disk set to /home/pete/u66.
amrecover>
```

At this point, a command line interface allows browsing the image catalogues. Move around with the **cd** command, see what is available with **ls**, change date with **setdate**, add files and directories to the extraction list with **add** and so on. The **extract** command starts actual recovery:

```
amrecover> setdate ---14
200 Working date set to 1999-01-14.
amrecover> cd work/sample-21
/home/pete/u66/jj/work/sample-21
amrecover> add molecule.dat
Added /jj/work/sample-21/molecule.dat
amrecover> cd ../sample-22
/home/pete/u66/jj/work/sample-22
amrecover> add molecule.dat
Added /jj/work/sample-22/molecule.dat
amrecover> extract
Extracting files using tape drive /dev/rmt/0mn on host amanda.cc.purdue.edu.
The following tapes are needed: Daily-034

Restoring files into directory /home/pete/u66
Continue? [Y/n]: y
```

```
Load tape Daily-034 now
Continue? [Y/n]: y
Warning: ./jj: File exists
Warning: ./work: File exists
Warning: ./work/sample-21: File exists
Warning: ./work/sample-22: File exists
set owner/mode for '.'? [yn] n
amrecover> quit
```

amrecover finds which tapes contain the images, prompts through mounting them in the proper order, searches the tape for the image, optionally decompresses it, brings it across the network to the client and pipes it into the appropriate restore program with the arguments needed to extract the requested items. **amrecover** does not know how to run every client restore program. See the **amrecover** manpage for current information. **amrecover** should not be used to do full filesystem recovery with vendor restore tools, but does work with **GNU-tar**. Vendor tools should be run with the `r` flag for a full recovery and **amrecover** is oriented toward extracting individual items with the `x` flag. Full filesystem recovery with vendor restore should be done with **amrestore**. **amrecover** (actually the **amidxtaped** server) does not know about tape changers, so mount the tapes by hand or use **amtape** if a changer is available.

18.8.2 Using amrestore

The **amrestore** command retrieves whole images from tape. First, find which tapes have the desired images. The `find` suboption of **amadmin** generates output like this (abbreviated):

```
# su amanda -c "amadmin Daily find pete u66"
Scanning /amanda...

date       host                 disk           lv tape or file   file status
...
1999-01-12 pete.cc.purdue.edu   /home/pete/u66    1  Daily-032        14 OK
1999-01-13 pete.cc.purdue.edu   /home/pete/u66    1   Daily-033        26 OK
1999-01-14 pete.cc.purdue.edu   /home/pete/u66    1  Daily-034        40 OK
1999-01-15 pete.cc.purdue.edu   /home/pete/u66    1  Daily-000        34 OK
1999-01-16 pete.cc.purdue.edu   /home/pete/u66    1  Daily-001        31 OK
1999-01-17 pete.cc.purdue.edu   /home/pete/u66    0  Daily-002        50 OK
1999-01-18 pete.cc.purdue.edu   /home/pete/u66    1  Daily-003        20 OK
```

The `Scanning /amanda...` message says **amadmin** looked in the holding disk (`/amanda`) for any images left there. It then lists all tapes or files in the holding disk that contain the requested area.

The `info` suboption to **amadmin** shows tapes with the most recent images:

```
# su amanda -c "amadmin Daily info pete u66"
Current info for pete.cc.purdue.edu /home/pete/u66:
Stats: dump rates (kps), Full:  652.0, 648.0, 631.0
                   Incremental:  106.0, 258.0, 235.0
         compressed size, Full: -100.0%,-100.0%,-100.0%
              Incremental: -100.0%,-100.0%,-100.0%
Dumps: lev datestmp  tape             file   origK   compK secs
         0  19990117  Daily-002          50  582239  582272 892
    1  19990118  Daily-003          20    3263    3296    31
    2  19981214  Daily-032          21    7039    7072   37
```

Old information may appear, such as `19981214` (14-Dec-1998) in this example. While it's true this was the last level 2 dump of this area, it is of little interest because at least one full and level 1 dump have been done since then. The compressed size values here may be ignored because this particular configuration uses hardware compression so no software compression data are available.

A third way to know what tape has an image is to generate a tape table of contents with **amtoc** after each *Amanda* run:

```
 #  partition                        lvl  size[Kb]  method
 0  Daily-002                          -         -  19990117
 1  boiler.cc.purdue.edu:/usr/local    1        31  normal
 2  egbert.cc.purdue.edu:/opt          1       127  normal
 3  boiler.cc.purdue.edu:/usr          1        95  normal
...
50 pete.cc.purdue.edu:/home/pete/u66   0    582239  normal
...
```

A printed report similar to the **amtoc** output may be automatically generated by **amreport** for each run with the `lbl-templ tapetype` parameter in `amanda.conf` using the `example/3hole.ps` template.

The `find` and `info` suboptions to **amadmin** need the *Amanda* log files and database. These are not usually large amounts of information and a copy should be pushed after each **amdump** run to an alternate machine that also has the *Amanda* tape server software installed so they are available if the primary tape server machine dies. Tools like rdist (<`ftp://usc.edu/pub/rdist/`>) or rsync (<`ftp://samba.anu.edu.au/pub/rsync/`>) are useful.

If *Amanda* was built using `--with-db=text` (the default), the database is stored in a set of text files under the directory listed in the `infofile amanda.conf` parameter. Here is the file that matches the above `info` **amadmin** output:

```
# cd /usr/local/etc/amanda/Daily/curinfo
# cat pete.cc.purdue.edu/_home_pete_u66/info
version: 0
command: 0
full-rate: 652.000000 648.000000 631.000000
full-comp:
incr-rate: 106.000000 258.000000 235.000000
incr-comp:
stats: 0 582239 582272 892 916549924 50 Daily-002
stats: 1 3263 3296 31 916637269 20 Daily-003
stats: 2 7039 7072 37 913614357 21 Daily-032
//
```

The first field of each `stats` line is the dump level. The last field is the VSN and the field just before it is the tape file number. The field with the large number just before that is a Unix epoch time value, which may be converted to text with this Perl script:

```
$ cat epoch.pl
#!/usr/local/bin/perl
use warnings;
use strict;
require 'ctime.pl';
foreach (@ARGV) {
  s/,//;
  if (m/[a-fA-FxX]/) {
     unless (m/^0[xX]/) {
 $_ = '0x' . $_;
}
$_ = oct;
   }
  print &ctime ($_);
}
exit (0);
$ epoch.pl 916549924
Sun Jan 17 0:12:04 US/East-Indiana 1999
```

Prepositioning the tape to the image with **mt fsf** may significantly reduce the time needed to do a restore. Some media contain an index for very fast file searching compared to the one file at a time scanning done by **amrestore**. Each tape location method listed above also shows the tape file. Use that number with **mt fsf** after a rewind to position to a particular image.

amrestore takes client, area and date stamp patterns as optional arguments to search for matching images. Each argument is a *grep*-style regular expression, so multiple images may match. This also means an image may need a specific pattern. For instance:

```
# amrestore $TAPE pete /
```

finds not just the root area for the *pete* client, but images for any client with *pete* someplace in the hostname and a slash anywhere in the area name. Assuming only one client matches pete, the following gets just the root area:

```
# amrestore $TAPE pete '^/$'
```

The up arrow (caret) at the beginning says the pattern must start with this string. The dollar sign at the end says it must end there. The quote marks around the pattern protect the special characters from shell expansion.

Without flags, **amrestore** finds every matching image, uncompresses it if needed and creates a disk file in the current working directory with a name made up of the client, area and dump level. These images may be used directly by the client restore program.

amrestore may be used to generate a tape table of contents by giving it a host pattern that cannot match:

```
# mt rewind
# amrestore $TAPE no.such.host
```

As it searches in vain for *no.such.host* it reports images that are skipped:

```
amrestore: 0: skipping start of tape: date 19990117 label Daily-002
amrestore: 1: skipping boiler.cc.purdue.edu._.19990117.1
amrestore: 2: skipping egbert.cc.purdue.edu._opt.19990117.1
amrestore: 3: skipping boiler.cc.purdue.edu._.19990117.1
...
```

For large images, the `p` flag writes the first match to standard output, which may then be piped into the client restore program. This flag is also useful for moving an image across

the network. For instance, here is one way to restore a file directly from the tape server (*amanda.cc.purdue.edu*) while logged in to the client:

```
    # rsh -n amanda.cc.purdue.edu amrestore -p $TAPE pete
?'^/$? ' \ | gtar xf - ./the-file
```

Tell vendor restore programs to use a small blocking factor to handle the arbitrary size chunks of data available through a pipeline:

```
# rsh -n amanda.cc.purdue.edu amrestore -p $TAPE pete u66 \ |
ufsrestore -ivbf 2 -
```

18.8.3 Restoring Without *Amanda*

The *Amanda* tape format is deliberately simple and restoring data can be done without any *Amanda* tools if necessary. The first tape file is a volume label with the tape VSN and date it was written. It is not in *ANSI VOL1* format, but is plain text. Each file after that contains one image using 32 KByte blocks. The first block is an *Amanda* header with client, area and options used to create the image. As with the volume label, the header is not in ANSI format, but is plain text. The image follows, starting at the next tape block, until end of file.

To retrieve an image with standard Unix utilities if **amrestore** is not available, position the tape to the image, then use **dd** to read it:

```
# mt rewind
# mt fsf NN
# dd if=$TAPE bs=32k skip=1 of=dump_image
```

The `skip=1` option tells **dd** to skip over the *Amanda* file header. Without the `of=` option, **dd** writes the image to standard output, which can be piped to the decompression program, if needed, and then to the client **restore** program.

Since the image header is text, it may be viewed with:

```
# mt rewind
# mt fsf NN
```

```
# dd if=$TAPE bs=32k count=1
```

In addition to describing the image, it contains text showing the commands needed to do a restore. Here's a typical entry for the root filesystem on *pete.cc.purdue.edu*. It is a level 1 dump done without compression using the vendor **ufsdump** program:

```
Amanda: FILE 19981206 pete.cc.purdue.edu / lev 1
comp N program /usr/sbin/ufsdump
```

To restore, position the tape at start of file and run:

```
# dd if=$TAPE bs=32k skip=1 | /usr/sbin/ufsrestore -f... -
```

As with any backup system, test these procedures while in normal production so the principles and techniques are familiar when disaster strikes.

NOTE

Refer to <`http://www.amanda.org/docs/using.html`> for the current version of this document.

Chapter 19

AMANDA **FAQ**

This file contains answers to some questions that are frequently asked in the *Amanda* mailing lists, specially by new users. Please take a look at this file before posting, this can save us time that could be spent improving *Amanda* and its documentation.

New entries and modifications are welcome; send them to <mailto://amanda-users@amanda.org> or <mailto://amanda-hackers@amanda.org>.

You may also want to take a look at the *Amanda* FAQ-O-Matic <http://www.amanda.org/fom-serve/cache/1.html>.

F.A.Q.

1. **Q:** *Why does* Amanda *fail to build on my system?*
A: One of the most common reasons for compile-time errors is stale information in `config.cache`, after a build on a different platform using the same build tree. In order to avoid this problem, make sure you don't ever reuse build trees across platforms, or at least run **make distclean** before running **configure** on another platform.

Another common reason for failure, that causes link-time errors, is a problem in libtool that causes it to search for symbols in already-installed amanda libraries, instead of in the just-built ones. This problem is known to affect SunOS 4.1.3 and FreeBSD. You can usually work around it by specifying a different prefix when you configure the new version of *Amanda*. However, it may not work if the previous version of *Amanda* was installed in `/usr/local` and **gcc** searches this directory by default; in this case, you must either remove the old libraries (which you don't want to do, right? :-) or call **configure** with the flag **–disable-libtool**. In this case, *Amanda* won't create shared libraries, so binaries will be larger, but you may worry about that later.

You may also want to take a look at *Amanda* 2.5.0 - System-Specific Installation Notes, as well as to the *Amanda* Patches Page (<`http://www.amanda.org/patches/`>) for other known problems. If everything fails, you should read the manual, but since we don't have one yet, just post a help request to the amanda-users mailing list (<`mailto://amanda-users@amanda.org`>), showing the last few lines of the failed build.

2. **Q:** *Why does* ***amdump*** *report that all disks failed?*
A: Probably because the *Amanda* clients are not properly configured. Before you ever run **amdump**, make sure **amcheck** succeeds. When it does, so should **amdump**.

Make sure you run **amcheck** as the same user that is supposed to start **amdump**, otherwise you may get incorrect results.

3. **Q:** *Why does* ***amcheck*** *say "port NNN is not secure"?*
A: Because **amcheck**, as some other *Amanda* programs, must be installed as setuid-root. Run **make install** as "root", or **chown** all *Amanda* setuid programs to "root", then **chown u+s** them again, if **chown** drops the setuid bit.

4. **Q:** *Why does* ***amcheck*** *claim that the tape is "not an* Amanda *tape"?*
A: Because *Amanda* requires you to label tapes before it uses them. Run **amlabel** in order to label a tape.

If, even after labeling a tape, **amcheck** still complains about it, make sure the regular

expression specified in `amanda.conf` matches the label you have specified, and check whether you have configured non-rewinding tape devices for *Amanda* to use. For example, use `/dev/nrst0` instead of `/dev/rst0`, `/dev/rmt/0bn` instead of `/dev/rmt/0b`, or some other system-dependent device name that contains an "n", instead of one that does not. The "n" stands for non-rewinding.

If you have labeled any tapes using the rewiding device configuration, you'll have to label them again.

5. **Q:** *Why does* ***amcheck*** *report "selfcheck request timed out"?*
A: This can occur under several different situations. First, make sure this problem is repeatable; if *Amanda* programs are NFS-auto-mounted, some clients may fail to mount the *Amanda* binaries in time.

If the error is repeatable, log into the client, and check whether the directory `/tmp/amanda` exists, and a file named `amandad.debug` exists in there: **amandad** will create this file whenever it starts. If this file does not exist, **amandad** is not starting properly, or it lacks permission to create `/tmp/amanda/amandad.debug`.

In the latter case, wipe out `/tmp/amanda`, and **amandad** should create it next time it runs. In the former case, check your **inetd** configuration. Make sure you have added the *Amanda* services to `/etc/services` (or the NIS services map), that `/etc/inetd.conf` was properly configured, and that you have signalled **inetd** to reread this file (some systems may need rebooting). Check section 2.2 from in the Amanda Installation Notes for details. Check the **inetd** man-page for possible differences between the standard format of `/etc/inetd.conf` and the one in your system.

Pay special attention to typos in `/etc/inetd.conf`; error messages will probably appear in `/var/adm/messages` or `/var/log/messages` if you have typed the **amandad** program name incorrectly. Make sure the same user that you have specified at configure-time (**configure –with-user=<USERNAME>**) is listed in `/etc/inetd.conf`. Check whether this user has permission to run **amandad**, as well as any shared libraries **amandad** depends upon, by running the specified **amandad** command by hand, as the *Amanda* user. It should just time-out after 30 seconds waiting for a UDP packet. If you type anything, it will abort immediately, because it can't read a UDP packet from the keyboard.

As soon as you have properly configured `/etc/inetd.conf` so as to run **amandad**, you should no longer get the "selfcheck request timed out" message. A nice tool to help make sure **inetd** is really listening on the **amandad** port is **lsof**, available at <`ftp://vic.cc.purdue.edu/pub/tools/unix/lsof.`>

6. **Q:** *Why does* `amandad.debug` *contain "error receiving message"?*
A: One possibility is that you have run **amandad** from the command line prompt and typed anything instead of waiting for it to time-out: in this case, it will try to read a UDP packet from the keyboard, and this was reported not to work on most keyboards :-). However, if you have run **amandad** as any user other than the one listed in `/etc/inetd.conf`, it may have created a `/tmp/amanda` directory that the *Amanda* user cannot write to, so you should wipe it out.

Another possibility is that the *Amanda* service was not properly configured as a UDP service; check `/etc/services` and `/etc/inetd.conf`.

7. **Q:** *Why does* ***amcheck*** *say "access as <username> not allowed..."?*
A: There must be something wrong with `.amandahosts` configuration (or `.rhosts`, if you have configured –**without-amandahosts**).

First, if the <username> is not what you expect (i.e., not what you have specified in the –**with-user flag**, at configure time), check the **inetd** configuration file: you must have specified the wrong username there.

Make sure you specify the names exactly as they appear in the error message after the '@' sign in `.amandahosts/.rhosts`. You'll need a fully-qualified domain name or not, depending on how your client resolves IP addresses to host names.

8. **Q:** *Why does* ***amcheck*** *report "ip address #.#.#.#" is not in the ip list list for <hostname>'?*
A: Check your DNS configuration tables. In order to avoid DNS-spoofing, *Amanda* double-checks hostname<->IP address mapping. If the IP address the request comes from maps to a hostname, but this hostname does not map back to the incoming IP address, the request is denied.

9. **Q:** *Why does* ***amcheck*** *say "cannot overwrite active tape"?*
A: Because, if you configure *Amanda* to use N tapes, by setting tapecycle to N in `amanda.conf`, before *Amanda* overwrites a tape, it must write to at least other N-1 tapes. Of course, *Amanda* will always refuse to overwrite a tape marked for 'noreuse' with **amadmin**. Furthermore, such tapes are not counted when *Amanda* computes 'N-1' tapes.

If, for some reason, you want to tell *Amanda* to overwrite a particular tape, regardless of its position in the cycle, use **amrmtape**. This command will remove this tape from the `tapelist` file, that is used to manage the tape cycle, and will delete information about backups stored in that tape from the *Amanda* database.

10. **Q:** *Why does* ***amcheck*** *tell me "DUMP program not available"?*
A: Because **configure** could not find **dump** when it was first run. This is a common problem on Linux hosts, because most Linux distributions do not install **dump** by default.

If you don't have a DUMP program installed, install it, remove `config.cache`, run **configure** again and rebuild *Amanda*. While **configure** is running, make sure it can find the installed DUMP program. If it cannot, you may have to set the environment variables DUMP and RESTORE by hand, before running **configure**.

If you can't or don't want to install DUMP, you may use GNU tar, but make sure it as release 1.12 or newer; release 1.11.8 may work, but estimates will be slow as hell.

11. **Q:** *Which tape changer configuration should I use in* ***amanda.conf****?*
A: If you only have one tape unit, you have two choices:

i Don't use a tape changer at all, i.e., set runtapes to 1, set tapedev to the non-rewinding device corresponding to the tape unit, and comment out tpchanger, changerfile and changerdev

ii Set up chg-manual, so that you can change tapes manually. If you select chg-manual, you will not be able to start **amdump** as a cron job, and you should always run **amflush** -f, because chg-manual will ask you to press return in the terminal where you started the controlling program.

If you have several tape units, which you want to use to emulate a tape changer, you want chg-multi. Even if you do own a real tape changer, that operates based on ejecting a tape or such, chg-multi may be useful.

Actual tape changers usually require specialized changer programs, such as **mtx**, `chio` or specific system calls. The availability of these programs is much more dependent on the operating system you're running than on the particular tape changer hardware you have.

mtx, for example, is available for several platforms. However, even if you find it for your platform, beware that there exist several different programs named **mtx**, that require different command line arguments, and print different output, and *Amanda*'s chg-mtx does not support them all. You may have to edit the script, which shouldn't be hard to do.

In section BUILT-IN TAPE CHANGERS of *Amanda* Tape Changer Support you will find details about the tape changer interfacing programs provided with *Amanda*, that can interact with common tape changer programs and with tape changer-related system calls provided by some operating system. If none of them matches your needs, you may have to develop your own tape changer interface script.

Before posting a question to the *Amanda* mailing lists, *please* search the archives, and try to obtain as much information about driving your tape changer hardware from the vendor of the changer hardware and of the operating system, rather than from the *Amanda* mailing lists. We usually don't have much to say about tape changer units, and several questions about them remain unanswered. :-(

Anyway, if you decide to post a question, make sure you specify both the tape changer hardware *and* the OS/platform that is going to interface with it. Good luck! :-)

12. **Q:** *Where do I get my tapetype-definition from? Do I have to run* ***amtapetype****?*
A: It is not mandatory to run **amtapetype** at installation-time. It is very likely that your tapedrive or -changer is one of the devices that are already covered by one of the existing tapetype-definitions.

You may find tapetype-definitions in the example `amanda.conf`, in the mailinglist-archives of the amanda-users-mailinglist at <`http://marc.theaimsgroup.com/?l=amanda-users`> or in the *Amanda*-FAQ-O-Matic at <`http://www.amanda.org/fom-serve/cache/1.html`>.

Reasons to run **amtapetype** for your device:

- You want to generate your own tapetype-definition because you can't find any suitable tapetype-definition for your device.
- You want to determine the performance of your device.
- You want to determine if your device has hardware-compression enabled.

If you decide to run **amtapetype**, please refer to the chapter Tapetypes and the manpage amtapetype(8).

13. **Q:** *Should I use software or hardware compression?*
A: When you enable software compression, you drastically reduce the compression that might be achieved by hardware. In fact, tape drives will usually use *more* tape if you tell them to try to further compress already compressed data.

Thus, you must choose whether you're going to use software or hardware compression; don't ever enable both unless you want to waste tape space.

Since *Amanda* prefers to have complete information about tape sizes and compression rates, it can do a better job if you use software compression. However, if you can't afford the extra CPU usage, *Amanda* can live with the unpredictability of hardware compression, but you'll have to be very conservative about the specified tape size, specially if there are filesystems that contain mostly uncompressible data.

You might want to run **amtapetype** to determine if you have hardware-compression enabled for your tape-drive.

14. **Q:** *How can I configure* Amanda *so that it performs full backups on the week-end and incrementals on weekdays?*
A: You can't. *Amanda* doesn't work this way. You just have to tell *Amanda* how many tapes you have (tapecycle), and how often you want it to perform full backups of each filesystem (dumpcycle). If you don't run it once a daily (including Saturdays and Sundays :-), you'll also want to tell *Amanda* how many times you'll run it per dumpcycle (runspercycle). It will spread full backups along the dumpcycle, so you won't have any full-only or incremental-only runs.

Please also refer to "the friday-tape-question" in Collection of the top ten *Amanda* questions. And answers..

15. **Q:** *What if my tape unit uses expensive tapes, and I don't want to use one tape per day? Can't* Amanda *append to tapes?*
A: It can't, and this is good. Tape drives and OS drivers are (in)famous for rewinding tapes at unexpected times, without telling the program that's writing to them. If you have a month's worth of backups in that tape, you really don't want them to be overwritten, so *Amanda* has taken the safe approach of requiring tapes to be written from the beginning on every run.

This can be wasteful, specially if you have a small amount of data to back up, but expensive

large-capacity tapes. One possible approach is to run **amdump** with tapes only, say once a week, to perform full backups, and run it without tape on the other days, so that it performs incremental backups and stores them in the holding disk. Once or twice a week, you flush all backups in the holding disk to a single tape.

If you don't trust your holding disk, and you'd rather have all your data on tapes daily, you can create an alternate configuration, with two tapes, that backs up the holding disk only, always as a full backup. You'd run this configuration always after your regular backup, so you always have a complete image of the holding disk on tape, just in case it fails.

16. **Q:** *How can I configure* Amanda *for long-term archiving?*
A: The best approach is to create a separate configuration for your archive backups. It should use a separate set of tapes, and have all dumptypes configured with 'record no', so it doesn't interfere with regular backups.

17. **Q:** *Can I backup separate disks of the same host in different configurations?*
A: Yes, but you have to be careful. *Amanda* uses UDP to issue estimate and backup requests and, although replies to backup requests are immediate (so that TCP connections for the actual backup can be established), replies to estimate requests are not and, while one request is being processed, any other request is ignored. The effect is two-fold:

i If another configuration requests for estimates, the request will be ignored, and the requester will end up timing out;

ii If another configuration has already finished the estimates, and is now requesting for backups, the backup requests will time-out.

So, there are two easy ways out:

i Ensure that the configurations never run concurrently, or

ii set up two different installations of the *Amanda* server, using different services names to contact the clients, i.e., different port numbers. This can be attained with the configure flag –**with-testing=<service-suffix>**. Yes, the flag name is not appropriate, but so what?

If you don't want to set up two installations of *Amanda* (I agree, it's overkill), but you still want to back up disks of the same host in separate configurations, you can set up *Amanda* so that one configuration only starts after the first one has already finished its One possible way to work-around this limitation is to start one configuration only after you know the estimates for the first one have already finished (modifying the crontab entries, according to history data). You'll also have to delay the starttime (a dumptype option) of the disks in the first configuration, so that they don't start backing up before the estimates of the second configuration finish.

18. **Q:** *Can* Amanda *span large filesystems across multiple tapes?*
A: Not yet :-(

This is an open project, looking for developers. If you'd like to help, please take a look at the *Amanda* Ongoing Projects Page (<`http://www.amanda.org/ongoing.php`>), where more up-to-date information is likely to be found about this project.

Update September 2004: Refer to the archive of the amanda-hackers mailinglist (<`http://marc.theaimsgroup.com/?l=amanda-hackers`>). A patch by John Stange is being discussed there, which allows splitting and spanning.

The current work-around is to use GNU tar to back up subdirectories of the huge filesystem separately. But be aware of the problems listed in the question about "results missing".

19. **Q:** *What's the difference between option "skip-full" and "strategy nofull"?*
A: "strategy nofull" is supposed to handle the following situation: you run a full dump off-line once a millenium :-), because that disk isn't supposed to change at all and, if it does, changes are minimal. *Amanda* will run only level 1 backups of that filesystem, to avoid the risk of overwriting a level 1 backup needed to do a restore. Remember, you run full dumps once a millenium, and your tape cycle probably won't last that long :-)

"skip-full", OTOH, is supposed to let the user run full dumps off-line regularly (i.e., as often as specified in the dumpcycle), while *Amanda* takes care of the incrementals. Currently, *Amanda* will tell you when you're supposed to run the level 0 backups but, if you fail to do so, *Amanda* will not only skip a full day's worth of valuable backups of the filesystem, on the day it told you to the full backup manually, but it will also run a level 1 backup on the next day, even if you have not performed the full backup yet. Worse yet: it might perform a level 2 on the next day, just after you have run the level 0, so, if the disk should crash, you'd have to restore a level 0 then a level 2, but not the level 1! Not a real problem, but definitely strange, eh?

20. **Q:** *Why does* ***amdump*** *report "results missing"?*
A: One of the possible reasons is that you have requested too many backups of the host. In this case, the estimate request or the reply may not fit in a UDP packet. This will cause *Amanda* not to perform some of the backups. Fixing this problem involves modifying the way estimate requests are issued, so that no packet exceeds the maximum packet size, and issuing additional requests that did not fit in a UDP packet after a reply for the previous set is obtained. The probability of getting this problem has been considerably reduced since we increased the maximum UDP packet size from 1Kb to 64Kb, but some operating systems may not support such large packets.

One possible work-around is to try to shorten the pathnames of the directories and the exclude file names, so that more requests fit in the UDP packet. You may create short-named links in some directory closer to the root (/) so as to reduce the length of names. I.e., instead of backing up `/usr/home/foo` and `/usr/home/bar`, create the following links:

```
/.foo -> /usr/home/foo
/.bar -> /usr/home/bar
```

then list `/.foo` and `/.bar` in the `disklist`.

Another approach is to group sub-directories in backup sets, instead of backing up them all separately. For example, create `/usr/home/.bkp1` and move 'foo' and 'bar' into it, then create links so that the original pathnames remain functional. Then, list `/usr/home/.bkp1` in the `disklist`. You may create as many '.bkp<N>' directories as you need.

A simpler approach, that may work for you, is to backup only a subset of the subdirectories of a filesystem separately. The others can be backed up together with the root of the filesystem, using an exclude list that prevents duplicate backups.

21. **Q:** *Why does **amdump** report "disk offline"?*
A: Well, assuming the disk is not really off line :-), it may be a permission problem, but then, **amcheck** would have reported it.

Another possible reason for this failure is a filesystem error, that causes DUMP to crash before it estimates the backup size; a **fsck** may help.

Yet another possibility is that the filesystem is so large that the backup program is incorrectly reporting the estimated size, for example, by printing a negative value that *Amanda* will not accept as a valid estimate. If you are using **dump**, contact your vendor and request a patch for dump that fixes this bug. If you are using **GNU-tar**, make sure it is release 1.12 or newer; 1.11.8 won't do! Even release 1.12 may require a patch to correctly report estimates and dump sizes, as well as to handle sparse files correctly and quickly instead of printing error messages like 'Read error at byte 0, reading 512 bytes, in file ./var/log/lastlog: Bad file number' in sendsize.debug and being very slow. Check the patches directory of the *Amanda* distribution.

22. **Q:** *What if **amdump** reports "dumps way too big, must skip incremental dumps"?*
A: It means *Amanda* couldn't back up some disk because it wouldn't fit in the tape(s) you have configured *Amanda* to use. It considered performing some incrementals instead of full dumps, so that all disks would fit, but this wouldn't be enough, so the disk really had to be dropped in this run.

In general, you can just ignore this message if it happens only once in a while. Low-priority disks are discarded first, so you'll hardly miss really important data.

One real work-around is to configure *Amanda* to use more tapes: increase 'runtapes' in `amanda.conf`. Even if you don't have a real tape changer, you can act yourself as a changer ('chg-manual'; more details in the question about tape changer configuration), or use 'chg-multi' with a single tape unit, and lie to *Amanda* that it will have two tapes to use. If you have a holding disk as large as a tape, and configure *Amanda* (2.4.1b1 or newer) not to reserve any space for degraded dumps, dumps that would be stored in the second tape of a run will be performed to the holding disk, so you can flush them to tape in the morning.

23. **Q:** ***amdump** reported "infofile update failed". What should I do?*
A: Make sure all directories and files are readable and writable by the *Amanda* user, within the directory you specified as 'infofile' in `amanda.conf`. From then on, only run amanda

server commands (**amadmin**, **amdump**, **amflush**, **amcleanup**) as the *Amanda* user, not as root.

24. **Q:** *Why does* Amanda *sometimes promote full dumps?*
A: To spread the full dumps along the dumpcycle, so that daily runs take roughly the same amount of tape and time. As soon as you start using *Amanda*, it will run full dumps of all filesystems. Then, on the following runs, it will promote some backups, so as to adjust the balance. After one or two dumpcycles, it should stop promoting dumps. You can see how well it is doing with **amadmin <conf> balance**. If you find the results surprising, you may want to adjust dumpcycle or runspercycle.

25. **Q:** *Why does* ***amrecover*** *report "no index records" or "disk not found"?*
A: The most common cause of this problem is not having enabled index generation in `amanda.conf`. The 'index yes' option must be present in every dumptype for whose disks indexes should be generated.

Another possibility is that **amrecover** is not selecting the configuration name that contains the backups for the selected disk. You may specify a configuration name with the '-C' switch, when you invoke **amrecover**. The default configuration name can only be specified at *Amanda* configure time (**configure –with-config=<name>**).

Indexes are currently generated at backup-time only, so, if a backup was performed without creating an index, you won't be able to use **amrecover** to restore it, you'll have to use **amrestore**.

26. **Q:** *Ok, I'm done with testing* Amanda*, now I want to put it in production. How can I reset its databases so as to start from scratch?*
A: First, remove the 'curinfo' database. By default, it is a directory, but, if you have selected any other database format (don't, they're deprecated), they may be files with extensions such as .dir and .pag.

Then, remove any log files from the log directory: `log.<TIMESTAMP>.<count>` and `amdump.<count>`. Finally, remove the `tapelist` file, stored in the directory that contains `amanda.conf`, unless `amanda.conf` specifies otherwise. Depending on the tape changer you have selected, you may also want to reset its state file.

27. **Q:** *The man-page of* ***dump*** *says that active filesystems may be backed up inconsistently. What does* Amanda *do to prevent inconsistent backups?*
A: Nothing. When you back up an active filesystem, there are two possibilities:

dump may print strange error messages about invalid blocks, then fail; in this case, *Amanda* will retry the backup on the next run.

Files that are modified while **dump** runs may be backed up inconsistently. But then, they will be included in the next incremental backup, which should usually be enough.

Large, critical files such as databases should be locked somehow, to avoid inconsistent backups, but there's no direct support for that in *Amanda*. The best bet is to configure *Amanda* to use a wrapper to **dump**, that locks and unlocks the database when appropriate.

28. **Q:** *Which version of **GNU-tar** should I use?*
A: (This answer was slightly adapted from a posting by Paul Bijnens paul.bijnens@xplanation.com <mailto:paul.bijnens@xplanation.com>, Mon, 11 Apr 2005):

- 1.13.19 is good.

  However it still sets return code 2 for some infrequent conditions even with –ignore-failed-read option. This results in *Amanda* thinking the total archive is bad, and drops the complete archive. Those conditions are very rare on a quiet filesystem.

- 1.13.25 is good: no problems found (yet).

- 1.13.9x is not good.

  It has changed the format of "tar -t", resulting in **amrecover** not able to use the indexes.

- 1.14.x is not good.

  It writes good archives, but when restoring, it has trouble with sparse files; the sparse file itself, and *all* files after it cannot be read anymore. But you can read the archive with a good tar version (i.e. the tar images produced are fine).

- 1.15.1 is good: no problems found (yet).

  Paul Bijnens: "I'm using this version on most of my clients since january this year (2005), and have already done successful restore too."

29. **Q:** *What does "bumping" mean?*
A: The term "bumping" is used to describe the change from one backup-level to the next higher level. If *Amanda* changes from Level 0 to Level 1 for a specific DLE, it "bumps".

The basic goal of "bumping" is to save precious space on the backup media as higher level incremental backups are smaller in size than lower level incremental backups.

The disadvantage of increasing backup levels is the fact that restoring from higher level incremental backups needs more tapes. This increases the amount of work time that are needed to fully restore a DLE as well as the possibility of tape-errors and similar problems during the process of restore. So in general it is recommended to keep the levels as low as possible with the given hardware and data.

There are various `amanda.conf` parameters to control and fine-tune *Amanda*'s behavior when it comes to "bumping":

Please refer to the amanda-manpage and the example `amanda.conf` for details on the parameters bumppercent, bumpsize, bumpdays and bumpmult.

30. **Q:** *How do I backup a Windows server?*
A: *Amanda* is able to use smbclient to dump SMB/CIFS-shares. Refer to the Backup PC hosts using Samba for details.

31. **Q:** *How do I tell my iptables-based firewall to allow* Amanda *through?*
A: posted by Matt Hyclak hyclak@math.ohiou.edu <mailto:hyclak@math.ohiou.edu>:

Use something like

```
iptables -A INPUT -p udp -s $AMANDA_SERVER -d $AMANDA_CLIENT --dport 10080 -j ACC
```

and load the ip_conntrack_amanda kernel module. I use the following in `/etc/modprobe.conf`:

```
options ip_conntrack_amanda master_timeout=2400
install ip_tables /sbin/modprobe --ignore-install ip_tables && /sbin/modprobe ip_
```

This sets the UDP timeout for *Amanda* packets to 2400 seconds, up from the default 300 (don't hold me to that, it might be 600). I was getting estimate timeouts since they were taking longer than 300/600 seconds and the firewall would close the port.

Makes things a little more secure than opening up everything > 1024 ;-)

32. **Q:** *How do I get rid of pressing "q" to get rid of a pager prompt when using* ***amrecover****?*
A: compiled from postings by Paul Bijnens paul.bijnens@xplanation.com <mailto:paul.bijnens@xplanation.com> and Jon LaBadie jon@jgcomp.com <mailto:jon@jgcomp.com>

Paul Bijnens wrote:

If you have to press "q" all the time in **amrecover** this is related to the pager-binary you use. If you use Linux this will be most likely **less**. To teach **less** to quit when hitting EOF, you need to set something like `LESS=--QUIT-AT-EOF; export LESS`, for example in your `.profile`. Refer to the manpage of **less** for details.

Jon LaBadie wrote:

If you don't like the quit at EOF behavior "except" when in **amrecover** create an alias or a wrapper; something like:

```
alias amrecov='LESS="$LESS -E" _pathto_your_amrecover'
```

33. **Q:** *Is there a way to tell the pager that my terminal has "y" lines?*
A: Jon LaBadie jon@jgcomp.com <mailto:jon@jgcomp.com> wrote:

The pager normally does it's best to find out how many lines your terminal has, given the right TERM-variable. Even terminals with elastic boundaries (e.g. xterms) work. But I have to admit that on Solaris the settings are not always correct. You can fix it quickly by setting an environment variable to e.g. LINES=24 (and export it).

NOTE

Refer to <http://www.amanda.org/docs/faq.html> for the current version of this document.

Chapter 20

COLLECTION OF THE TOP TEN *AMANDA* QUESTIONS. AND ANSWERS.

20.1 Reason for starting this list.

Jon LaBadie once wrote to me:

" I think a good "what is *Amanda*", "how is it different", "can I use it in my setup", "why is it so different" kinda document is needed to stop the constant "how do I put 10 dumps on one tape", or "how do I make *Amanda* do full backups on saturday and incrementals ..." queries off the list :)) "

Stefan G. Weichinger

20.2 the DLE-question

A posting from the amanda-users mailing-list (<mailto://amanda-users@amanda.org>) asked:

"What, please, is a "DLE"? May it mean: Down Loadable Entity ??? Stupid. Do Less Errors ??? Stupid again. Hmmmm ..."

People consulting the amanda-users-mailinglist for the first time often get confused by the use of the abbreviation DLE.

It has become very common for regular mailinglist-participants to use the abbreviation DLE, which means in its long form

DiskList Entry

A DLE refers to one entry in the disklist of an *Amanda*-configuration. General usage was to describe them as partitions, or file systems. But in fact they do not have to be either. They can be directory trees, or multiple trees, or trees with some branches cut off. So the more generic term DLE was coined.

20.3 the localhost-question

People get something like:

```
>Amanda Backup Client Hosts Check
>--------------------------------
>ERROR: localhost: [access as amanda not allowed from
>amanda@localhost.localdomain] amandahostsauth failed
```

and ask "Why?"

SHORT ANSWER:

DO NOT USE "localhost" as host entry in your `disklist` entries (aka DLEs). Use the FQDN (Fully Qualified Domain Name) instead.

In *Amanda*-releases newer than 2004-03-22 there is a WARNING issued when you use something like "localhost" or **localhost.localdomain.net** in your `disklist`.

Example (applies to Linux, syntax may be different on other systems):

```
$ hostname --fqdn
oops1.oops.co.at

$ cat disklist
oops1.oops.co.at /root root-tar # do it like this
localhost        /root root-tar # DON'T DO IT LIKE THIS
```

GOOD ANSWER (provided by John R. Jackson):

There are (at least) two things going on here and they should have their own question/answer.

Completely independent of the "localhost" vs. FQDN issue are the people who get this message because of any number of problems. Let me reword the error and then give some typical goofs:

```
ERROR: some.amanda.client: access as amanda not allowed from amanda@some.amanda.server
amandahostsauth failed
```

(error message reformatted here ...)

The first thing to understand is how to read this message. When it says "access as amanda ..." it is telling you the client side (**amandad**) is running as user "amanda". The "... from amanda@some.amanda.server" part tells you the server trying to connect is "some.amanda.server" and the *Amanda* command (e.g. **amcheck** or **amdump**) is running as user "amanda".

The user names are typically the same on both client and server, but some situations use different names and it is important to understand which is which. For instance, **amrecover** connects as root ("... from root@some.amanda.server") regardless of what the usual *Amanda* user is.

Potential problems:

- "some.server" is not spelled *exactly* that way in ~amanda/`.amandahosts`. A typical error is to not use a fully qualified name (although simple typos happen as well). For instance, this line:

```
some  amanda
```

does not match "some.amanda.server" even though both names may be equivalent. When *Amanda* looks up the host name in `.amandahosts`, it uses the *exact* name it lists in the message. It does not try to look up abbreviations.

The only exception to this is that the lookup is case insensitive.

- The user name listed in ~amanda/`.amandahosts` is not the one trying to connect from the server. In particular, watch out for the "root" case listed above for **amrecover**. The *Amanda* server typically needs lines like this in its `.amandahosts` file:

```
some.amanda.client   root
```

- There are permission problems on the client preventing user "amanda" from reading its own `.amandahosts` file. Make sure the file itself is readable to the user "amanda" and all the parent directories down to it can be traversed. A simple test is:

```
su - amanda -c "cat ~amanda/.amandahosts"
```

Now, back to the localhost issue. This:

Do NOT USE "localhost" as host entry in your `disklist` entries (aka DLEs). Use the FQDN (Fully Qualified Domain Name) instead.

is not really an answer, more of a command :-).

There are a couple of reasons to NOT use "localhost". First is **amrecover** will not work as expected. When it connects to the server (even though they are the same machine), the server will look for the matching DLE's using the real host name, not "localhost". The **sethost** command inside **amrecover** can "fix" this, but why not just set it up right in the first place?

Another reason to not use "localhost" is because it helps with future changes. As the *Amanda* configuration grows, it's not at all unusual to take a server and make it a client of a new, larger, server. But now "localhost" does not point to the same machine it used to. If the FQDN of the machine had been used all along, this upgrade would have been much easier.

There is also no performance reason (any more) to use "localhost" instead of the FQDN. Modern OS network stacks know to shortstop packets destined for the local machine and never let them hit the wire. Yes, I'm old enough to remember when they didn't :-).

20.4 the friday-tape-question

"How do I make *Amanda* do full backups on Saturday and incrementals ... ?"

"My backup screwed up on tuesday and now it keeps asking for the tuesday tape even though it is wednesday!"

ANSWER:

The short answer is: You can't.

The longer answer is: You can. But you should not.

The reason: *Amanda* is designed to schedule your backups. Let "her" do it.

When you want to make the best use of *Amanda*, you have to let go the classic schedule where one used to have one tape dedicated to each day of the week, and one for the friday.

The main difference in concept is this:

In the classic backup scheme you said: "I want to have incremental backups from Mo-Th and a full backup on Fr." Using *Amanda* you say: "I want to have at least one full backup in 5 days." So you don't have to specify exactly WHEN the full backup should happen. You just tell *Amanda* some goals it should reach and let it work out the details.

There are several advantages in this:

Imagine that you have your classic backup-schedule running fine. Everything is calculated and designed well, so your tape gets filled well each night.

Now one user generates an unforeseen huge amount of data. For example, he duplicates one big data-directory by mistake.

So the size of the directory raises within one day, maybe for multiple GBs.

Would your classic backup-scheme catch that? Or would it run out of tape, simply because it was not calculated to have that filesystem with that size?

Amanda would try to catch it (and most of the time succeed ...).

As there is the estimate-phase before actually dumping something, *Amanda* can look at the DLEs and determine the actual size at the time. It also determines the size of an incremental backup so it can test for the Plan B to just run a level-1 if it does not work out to do a level-0 for that DLE.

If the size of the DLE is much bigger than it has been the run before, *Amanda* still tries to meet your goals. It just reschedules stuff, combining full and incremental backups to meet the goals as good as possible.

So you can think of it as some algorithm which lets *Amanda* adapt to your data. If you set the goals in a reasonable way, *Amanda* will just do the rest.

20.5 the multiple-dumps-question

"How do I put 10 dumps on one tape?"

ANSWER (provided by Jon LaBadie):

Use another backup scheduler.

This question is most often asked by individual computer users as a cost consideration.

Amanda was developed at the University of Maryland Computing Center for use in moderately sized computer centers. That it can be used by users of small computers is a testament to its designers and maintainers.

While it may seem cost effective to put as many dumps as possible on a single tape, in a computing center that would be considered a very risky decision. The loss of, or damage to, a single tape would be the loss of many days worth of dumps. That is too much to chance.

Thus, *Amanda* was designed to never overwrite a non-*Amanda* tape, nor an *Amanda* tape from a different configuration, nor an *Amanda* tape from the current configuration that is still "active", i.e. has backups on the tape more recent than the dumpcycle length.

If you still feel you want *Amanda* to put multiple dumps on a single tape, there is a crude way to accomplish your goal.

But first ask yourself, "If my data is worth so little that I can not afford a few more tapes, why am I backing it up?"

NOTE

Most of the time it won't be YOU paying for the tapes as you may be working for some company. If your boss tries to force you into doing this multiple-dumps-on-one-tape thing, be sure to point him at this risk. Business people tend to understand the price-difference between some tapes and a major data-loss. Stefan G. Weichinger

A common way to put multiple dumps on a single tape is to let them accumulate on the holding disk and use the **amflush** command when you want to put them on tape. I.e. if you want a weeks' worth of backups on a single tape, leave the tape out for a week. Then stick it in and run **amflush**.

(Better make sure you have sufficient disk space on your holding disk.)

Note, a slight variant of this is to have the parameter autoflush in `amanda.conf` set to "yes". (Users of older *Amanda*-releases should check out if their version already supports that parameter.)

Then after several dumps have collected in the holding disk, put the tape in before that day's **amdump** is scheduled. **amdump** will both flush the holding disk to tape and add the regularly scheduled dump.

20.6 the mailing-list-question

"How do i get off this damn mailing list?"

ANSWER:

Frequent users of the *Amanda*-users-mailing-list get mails like containing "unsubscribe" as people are trying desperately to get off the list.

Everyone that subscribes to *Amanda*-users gets a mail in which the following is contained:

>Welcome to the amanda-users mailing list!

>Please save this message for future reference. Thank you.

>If you ever want to remove yourself from this mailing list, >you can send mail to <Majordomo@amanda.org> with the following >command in the body of your email message:

> unsubscribe amanda-users

Did you see that? You have to send your mail to <Majordomo@amanda.org>, and NOT to <amanda-users@amanda.org> !

20.7 the distro-question

"Where can i get binary distributions of *Amanda*?"

ANSWER:

It is well known that various distributions of Linux contain precompiled packages of *Amanda*-servers and -clients.

Due to the design of the *Amanda* source code, in which MANY features can be configured at compile-time, it is heavily and heartily recommended to take the effort and roll your own special flavour.

Thinking about these things before actually doing backups with *Amanda* will help you in many ways. And you get the benefits of compiling your own paths/devices/configurations right into your *Amanda*-binaries. You also get the benefit of a much more improved understanding of the way *Amanda* does backups.

20.8 the index-question

"Why does **amrecover** say there are no index files?"

ANSWER:

It is very likely that *Amanda* is right about that. Check your dumptypes and make sure they include the line:

```
index yes
```

If this is the case and you still get that message, recheck the installation of your **amindexd**-binary.

Is the line in your (x)inetd-configuration pointing to the proper binary? Is this line active (= uncommented)? Did (x)inetd reread that configuration since that line was edited?

20.9 the tapetype-questions

" **amtapetype** has been running for 9 days, is this normal?"

"Will *Amanda* work with my frozboz tape drive/library?"

"Which device is my changer?"

" **amtapetype** is broken, it says my 200GB tape only holds 65GB."

"My file marks are HUGE, 1.3MB (on a 200GB tape, i.e. about 0.05% of the total capacity, or expressed another way, maybe 2 mm of a 125000 mm tape ...)"

ANSWER:

It is crucial to tell *Amanda* the truth about the tape-device(s) you want to use. Given the wrong values, *Amanda* can't calculate proper dumpsizes, free tape-space or make valuable use of compression.

Before you consider running **amtapetype**, think twice. Twice.

As tapedrives tend to be produced by not-so-small companies and as those not-so-small companies tend to produce more than one unit to maximize profits, it is very likely that someone else has the same device you have or at least one that uses the same technology.

Many people have already run **amtapetype** to determine the proper values to fill in their `amanda.conf`-files. Browse the example `amanda.conf` in your *Amanda*-tarball for various tapetypes. Browse the *Amanda*-FAQ on <`http://www.amanda.org`>. Chances are high that you find just your device described.

As in every other topic discussed in internet mailing lists, please try finding an answer there before asking on the *Amanda*-users list.

If your device is so exotic that even the *Amanda*-users can't help you, you still have your copy of **amtapetype**.

Before you start running it, note this:

- *DISABLE* hardware compression on your drive.

A common mistake is to have hw-compression enabled. **amtapetype** uses random data to test for the size and speed of your drive. Random data is pretty bad at getting compressed. In fact it gets even bigger so the results given back are useless. Disable it even if you are planning to use your drive with enabled hw-compression.

- Expect it running long.

As you can read in the man page, **amtapetype** writes the full tape twice, which can be a lot of data for modern drives (approaching a TByte). It also writes tape marks every 10 MBytes (by default) which forces the drive to flush its internal buffers and slows the process down. You can shorten this by giving **amtapetype** a better estimate of the expected capacity: **$ amtapetype -e 100g -f /dev/nst0** This "prepares" **amtapetype** to expect a tape with 100 GB capacity.

If **amtapetype** really runs for 9 days, you can be pretty sure there is something wrong with your approach.

And for the filemark-size: Just read the question again.

20.10 the size-question

"How do I back up a partition that won't fit on a tape?"

aka

"Can *Amanda* span one file over multiple tapes?"

ANSWER:

There are two basic rules when it comes to these things:

- *Amanda* supports using more than one tape in a single run of **amdump**
- *Amanda* does not support splitting a dump image across tapes

The first rule lets you make use of two or more tapes for a single **amdump** when using a tapechanger-robot or a tape-library. You could even use multiple tapes with the chg-manual-script, waiting patiently for one tape to be filled, then change tapes manually.

No matter how many tapes you can put in your robot or how long you can stay awake to change tapes you can NOT split the backup image of one of your `disklist` entries (aka DLEs) across multiple tapes. No way.

So you may ask the first question listed above. As the size of harddisk- drives grows steadily it is not uncommon to have multiple hundreds of gigabytes of harddrive capacity in one system. Compared to the size of your maybe not-so-shiny-anymore tapedrive this seems (and maybe is) huge.

What to do?

Don't split your dump image (it can't be done), split your DLEs.

You have to use **GNU-tar** in your dumptypes for this.

Try to redefine your `disklist` as in the following example:

```
fatboy  /bigmama_BIGDIR  /bigmama {     # a big subdirectory
comp-user-tar
include "./bigdir"
}
fatboy  /bigmama_FILES01 /bigmama {     # all files beginning with...
nocomp-user-tar
include "./file[01]*"
}
fatboy  /bigmama_FILES23 /bigmama {
nocomp-user-tar
include "./file[23]*"
}
...
fatboy  /bigmama_REST /bigmama {        # Catch-all
nocomp-user-tar
exclude "./file[0-9]*"
```

```
exclude append "./bigdir"
}
```

(example taken from a mail by Paul Bijnens on the *Amanda*-users-list)

The trick is to form several chunks of data of which each fits on tape. In the example above the chunks are formed by regular expressions matching files named like file00, file123 and file9999. You have to look at your DLEs to find the patterns describing your chunks.

As this technique forms data-chunks that fit on your tape it also helps *Amanda* to schedule your backups more flexible. Having more and smaller DLEs, the **planner** has more variations to possibly schedule your backups, so this will help getting nice output from **amadmin <conf> balance**, too.

NOTE

DLE-spanning might be supported by *Amanda* in a future release.

20.11 the GUI-question

"Is anyone working on a GUI for *Amanda*?"

ANSWER:

Actually there are people working on GUIs for *Amanda*. Aside from that the question really is: "Does anyone need a GUI for *Amanda*?"

Given the fact that backups tend to be run at night while people tend to sleep, who would need a fancy GUI showing 3D-backup-diagrams via X11? The only part of backups where GUIs maybe could add some comfort is recovery for unexperienced users.

20.12 the holding-disk question

"Why does it say "Some dumps may have been left in the holding disk." and there is nothing in the holding disk?"

ANSWER:

The third word in the message. Some dumps *MAY* have been left.

20.13 ...

Please feel free to suggest additions and corrections. Write to the amanda-users-mailinglist at <mailto://amanda-users@amanda.org>.

NOTE

Refer to <http://www.amanda.org/docs/topten.html> for the current version of this document.

Chapter 21

AMANDA WISHLIST

This is a major update.

These are items that we are planning to address, OR which we would like to see happen sometime in the future.

They appear in random order.

Of course, we aren't promising to deliver anything.

This document can be found at <`http://www.amanda.org/docs/wishlist.html`>.

You may find more up-to-date information at <`http://www.amanda.org/ongoing.php`>.

If you have any ideas about any of the following, please send an e-mail note to <`mailto://amanda-users@amanda.org`> or <`mailto://amanda-hackers@amanda.org`>.

Jean-Louis Martineau, Stefan G. Weichinger,

Amanda Core Team

Oct. 2004.

- Samba: Ports to non-Unix platforms, specifically Macs and PCs. The hooks are in the *Amanda* protocol to support non-dump backup programs, but no-one has volunteered to implement the client side. sgw: Mac OS X is able to run the client, so only the PC is left, and I suggest that we should go the SAMBA-way, I think. Samba is working well, is documented and developed further, so I think this will be a stable path to go and support. And people don't need to compile/install anything on their Win-boxes, just add a user and shares ... opinions welcome.
- Samba: Samba should be treated as a different backup program, not as GNUTAR, because it cannot handle dump-style incrementals.
- Samba: multiple exclusion-patterns. Since Samba 3.0.3 smbclient supports the usage of multiple exclude-patterns. This would enable *Amanda* to exclude more than one pattern per SMB-share, allowing to exclude pagefile.sys AND the registry-files, for example.
- Instead of hard-coding the interface with tape devices in *Amanda*, there should be a higher level interface that allowed different storage devices to be used. *Amanda* should also be able to retain backups in disk, even after they are taped, for faster restore of

recently backed up data. It should also be possible to store a single backup in multiple tapes, for redundancy.

- We need a better protocol between the driver and dumpers. setup terminated (to not start to dump on the same host at the same time). driver should ask periodicaly if the dumper is still alive (in case the dumper hang).
- retry failed backups in a single run. If backup fails because of active filesystems or lack of memory, *Amanda* could throw the failed backup away and run it again, instead of trying it again in the next run only.
- Support for client-initiated backups might be interesting, but the server would have to keep listening for clients backup requests for a configurable period of time. This could be used to back up secure hosts, for instance.
- Backups to remote tape devices (i.e., not in the main *Amanda* server), as well as to remote filesystems should be supported.
- multi-tape : *Amanda* should be able to write to many tape at the same time. Need some criteria to select which dump should go on which tape? By level, host name, ???
- A way to tell if some dump must be done before/after some other. (eg. DLE X must be started after DLE Y is started/dumped/taped).
- Write to tape and holding disk in parallel (For dump to tape), the dump to tape could be started first, while doing some dump to holding disk.
- Keep files on holding disk after taped, that will permit faster recovery because they will be from holding disk, these dump will be erase when the same is needed for newer dump.
- Append to tape
- chg-disk

  This script writes to disks which can be accessed in a parallel way (contrary to the serial access to tapes). This could enable *Amanda* to do writes and reads to several vtapes in parallel (e.g. doing an **amrestore** while the regular **amdump** is running).

  It would be helpful to have a script which generates the needed directory-structure for a given chg-disk configuration. This script should test for valid settings (using **amgetconf** to get the values out of `amanda.conf`), create the necessary slot-directories and label the new vtapes by using **amlabel**. (there are drafts available already)
- **amrecover** should be able to set and "rewind" the correct vtape. Currently it is necessary to do this manually in another tty.
- It should be possible to re-generate databases and indexes from tapes.
- *Amanda* could append meta-data like databases and indexes to tape, so that each tape contains its own current indexes and everything to rebuild the server-config.
- *Amanda* should install man-pages for installed programs only.
- It should be possible to configure whether **amidxtaped** should decompress the dump stream or not (so **amrecover** could decompress it locally).

- **amstatus**:

  It should read degraded schedule and write which are delayed.

  It should print number of byte written to tape for the current flush.

  The **taper** should write a file with a byte count for the current files (every GB) and **amstatus** could read it.

  It could report the expected time before the dump finishes.

- **amverify/ amverifyrun**:

  It should look at the log file and compare the result.

- **amrecover**:

  should cd, add, remove, ... with a path with glob or regex (cd o*/linux)

  find [file] # where is that file in the current DLE? (I don't know the path)

  when [file] # when was this file dumped?

  parsing accept '\': cd HP890\ Color

  our own completion

- amkill:

  A new script to kill all process on client and server

- Enhance the protocol between client-server to allow white-space and any character in DLE/exclude/include.

- More tools in **amadmin**. The administrator should be able to look at the database in various ways. Adding / deleting / moving disks and hosts should be done through **amadmin** instead of editing the disklist directly. This will allow *Amanda* to do some sanity checks for the operators, to make sure permissions are set up right, etc.

  Suggested by Chris Jones cjones@honors.montana.edu <`mailto:cjones@honors.montana.edu`>.

- You should be able to force full dumps for nights other than tonight. Rather than one command at a time on the command line, **amadmin** could be a little shell with a help facility (ala ckermit or gnuplot, if you've seen those).

- A tape-verify pass after the *Amanda* run (we already have one, but it doesn't work with **dump** as well as it does with GNU **tar**). Perhaps taper could calculate a CRC for each file and store that in the database, to be checked by the verifier.

- More sophisticated tape management. Should *Amanda* track tapes globally, counting the number of times tapes were used, and recommending retirement for tapes at the appropriate time?

- Automatically notice that external dumps have been done. Sendsize could also notice if a filesystem was dumped externally from *Amanda*. Right now the **planner** assumes that the incrementals it is doing are relative to the full dumps it is doing. If someone

does a full dump of one of its filesystems (and writes `/etc/dumpdates`) outside of *Amanda*, data could be lost. Sun's Backup-Copilot handles this well. We should too.

- What if we made the "length" in a tapetype definition always be the "no compression" value? Then change the dumptype "compress" option to accept "hardware" as another type (ala "client" and "server") and let **planner** do its normal thing with that information (including "comprate", which at the current default of 50% is the usual first guess for hardware compression). This would make setting the tape length value less confusing, and make the **amtapetype** program easier to run. You could even get more accurate planning than what is currently available by setting the comprate to what you know the data is like on a dumptype by dumptype basis. Suggested by John R. Jackson jrj@gandalf.cc.purdue.edu <`mailto:jrj@gandalf.cc.purdue.edu`>.

- The way to specify the schedule should be redesigned, all those strategy (standard, nofull, noinc, incronly, force-full) and options (skip-full, skip-incr) are confusing.

  We should have two options, one for full dump and one for incrementals.

  full [AUTOMATIC | SKIP | NOTIFY | FORCE | FIXED]

  incr [NONE | BUMP | NOBUMP]

  with the following values:

  AUTOMATIC: follow *Amanda* scheduling (allow promoted and delayed)

  SKIP : full dump are done externally on an fixed schedule, dump nothing when a full is due (like skip-full).

  NOTIFY : full dumps are done externally, but are notified with the NOTIFY command (**amadmin** <conf> notify <host> <disk>).

  FORCE : full dumps are done only with the FORCE_FULL command.

  FIXED : do full dumps on a fixed schedule (like skip-incr).

  NONE : don't do incremental dumps.

  BUMP : allow incremental dumps to bump.

  NOBUMP : do not allow incremental dumps to bump.

- Remove all compiled options that can be moved to a (the?) configuration file. (eg. GNU **tar** path, if **configure** can't find it, *Amanda* should be able to use GNU **tar** if the path is specified on a client config file) Many people would like this, it would maybe also bring us closer to the possibility of working and usable rpms?

- Documentation:

  There is pretty much going on with the *Amanda*-docs. The docs have been converted to Docbook/XML and form the new module xml-docs in the *Amanda*-CVS-repository.

  The FAQ-O-Matic could be replaced by a Wiki. Suggested by Harlan Stenn Harlan. Stenn@pfcs.com <`mailto:Harlan.Stenn@pfcs.com`>.

  The xml-docs need more formatting and reviewing.

  The tapetypes from the FOM should go into the XML-docs.

The docs would benefit from adding some illustrations.

The WISHLIST should get shortened.

NOTE

Refer to <http://www.amanda.org/docs/wishlist.html> for the current version of this document.

Part V

Technical Background

HOW *AMANDA REALLY* WORKS ...

This section contains some papers which describe the technical concepts behind *Amanda*. You find descriptions of the various APIs as well as a basic draft of the internals of *Amanda*.

Chapter 22

HOW *AMANDA* USES UDP AND TCP PORTS

Amanda uses both UDP and TCP ports during its operation. The **amandad** service is listening (via **inetd/xinetd**) at a well known (fixed) port on each client for UDP connections. The **amindexd** and **amidxtaped** services are listening (also via **inetd/xinetd**) at well known ports on the tape server for TCP connections.

When a process on the tape server wants to talk to a client, it creates a UDP socket and binds it to a port on its side, then sends the packet to the well known **amandad** service port on the client. Because security information is passed, the port bound on the connecting (tape server) side must be privileged (less than 1024). This "proves" to **amandad** whoever is connecting is running as root, and therefore is trustworthy (there are all sorts of issues with the validity of this "trust" that are beyond the scope of this document).

A similar sequence of events happens when **amrecover** on a client wants to contact **amindexd** or **amidxtaped** on the tape server. The port that **amrecover** binds to its TCP socket must be privileged, which is one of the reasons it must be run as root.

Amanda also uses TCP connections for transmitting the backup image, messages and (optionally) the index list from a client back to the **dumper** process on the tape server. A process called **sendbackup** is started by **amandad** on the client. It creates two (or three, if indexing is enabled) TCP sockets and sends their port numbers back to **dumper** in a UDP message. Then **dumper** creates and binds TCP sockets on its side and connects to the waiting **sendbackup**.

Because **sendbackup** does not run as root on the client, it cannot allocate privileged TCP ports to listen on. The **dumper** process is setuid root and could bind privileged ports on its side (it currently does not), but because **sendbackup** does not care what port connects back to it (it assumes the only process that could have knowledge of the port numbers to use is **dumper**), it does not check the peer (connecting) port number.

22.1 TCP port allocation

When *Amanda* creates a TCP server socket to listen for incoming connections (**sendbackup**), it goes through the following bind steps:

- try for the user TCP port range (–with-tcpportrange), if defined. If that fails ...
- try for a privileged port (512 .. 1023). If that fails ...
- get any available port.

In all cases, it will not use a port that has been assigned to other well-known services. This sequence is implemented in stream_server().

When *Amanda* (**dumper**) creates an unprivileged TCP client socket to connect to a server, it goes through the following bind steps:

- try for the user TCP port range (–with-tcpportrange), if defined. If that fails ...
- get any available port.

In all cases, it will not use a port that has been assigned to other well-known services. This sequence is implemented in stream_client().

When *Amanda* (**amrecover**) creates a privileged TCP client socket to connect to a server, it goes through the following bind step:

- try for a privileged port (512 .. 1023). If that fails, the whole request is aborted.

This sequence is implemented in stream_client_privileged().

The stream_server() routine is used in two ways:

- **taper** to set up direct to tape communication with **dumper** on localhost.

If a user TCP port range is defined, it needs to be unprivileged because **taper** is not running as root.

- **sendbackup** to set up a transfer with its **dumper**.

If a user TCP port range (–with-tcpportrange) is defined, it needs to be unprivileged because **sendbackup** is not running as root.

A user TCP port range needs to be large enough for three ports (data, message and index) times the number of simultaneous backups the client may be asked to perform ("maxdumps" in `amanda.conf`).

The stream_client() routine is used in two ways:

- **dumper** to connect to **taper** for a direct to tape operation. Except for making sure what is connecting is not (ftp) port 20 (a common attack entry point), **taper** does not pay any attention to the source (**dumper**) port number.
- **dumper** to connect to **sendbackup** on a client. Again, except for port 20, **sendbackup** does not care what port the request comes from.

If a user TCP port range (–with-tcpportrange) is defined, it needs to be unprivileged because **dumper** is not running as root (at this point).

A user TCP port range needs to be large enough for two ports (one to **sendbackup** on the client, and possibly one to **taper**) times the number of **dumper**s ("inparallel" in `amanda.conf`).

The stream_client_privileged() routine is used in one way:

- **amrecover** to connect to amindexd and amidxtaped.

Because security information is passed, **amindexd**/ **amidxtaped** (via security_ok() in security.c) insist the other end (**amrecover**) be bound to a privileged port.

22.2 User TCP port range (–with-tcpportrange) summary

Pick the max of (2 * inparallel) and (3 * largest maxdumps). Allocate at least that many ports in the unprivileged (1024 or larger) range. Stay away from other well known ports (e.g. in your `/etc/services` file) or account for their potential use by making the portrange larger.

22.3 UDP port allocation

When *Amanda* creates a UDP socket, the same order of assignment as above is used by dgram_bind():

- try for the user UDP port range (–with-udpportrange), if defined. If that fails ...
- try for a privileged port (512 .. 1023). If that fails ...
- get any available port.

In all cases, it will not use a port that has been assigned to other well-known services. The dgram_bind() routine is called from three places, **amcheck**, **planner** and **dumper**. In each case, a connection to **amandad** on a client is being set up. **amandad**, in turn, calls security_ok(), which insists the other end of the connection be a privileged port, so a user UDP port range (–with-udpportrange) must specify privileged port numbers.

A user UDP port range must allow for one port for each client that might be contacted at a time. **planner** and **amcheck** use a single socket to contact all their clients, but there may be multiple **dumper**s (based on "inparallel" in `amanda.conf`) and each needs its own port.

22.4 User UDP port range (–with-udpportrange) summary

Allocate at least "inparallel" many ports in the privileged (1023 or smaller) range. Stay away from other well known ports (e.g. in your `/etc/services` file) or account for their potential use by making the portrange larger.

22.5 Firewalls and NAT

I'm not familiar with firewalls or NAT – one of the benefits of working in a University environment :-). So the following is likely to be completely wrong, but I have tried to get the advice of folks who do really understand this stuff.

Firewalls and *Amanda* should be pretty easy to set up. Just pick user UDP and TCP port ranges, build *Amanda* with them (–with-udpportrange and –with-tcpportrange) and

let them through the firewall. You also need to let the well known *Amanda* ports through, just as you would ftp or telnet.

NAT has other issues. If the *Amanda* client is "outside" NAT, there should not be a problem for backups. Sendbackup will set up the ports and tell **dumper** what they are. Then **dumper** will connect to them from "inside" and NAT should leave that alone, although it doesn't really matter since **sendbackup** does not care who connects to it (other than it not be ftp port 20).

If the *Amanda* tape server is outside, NAT will have to be told how to translate the incoming connections from **dumper** to the client. To do that, the UDP and TCP port ranges will have to be known and only one client can be inside.

The reverse is true for **amrecover**. If **amrecover** is run from inside NAT, there should not be a problem – it's just like running ftp or telnet. But from the outside, NAT will have to know where the amindexd/amidxtaped services are and allow them through (much like ftp or telnet daemons). Since they are on known port numbers, the user TCP port range is not an issue.

A user TCP port range is probably not important in the case of **dumper** and **taper** talking to each other since only the one machine (localhost) is involved and so it does not go through a firewall. But I could be wrong, especially if NAT is involved.

The details of how you configure a specific firewall or NAT are beyond the scope of this document (although examples would be welcome). You need to read up on the documentation that comes with them.

NOTE

Refer to <`http://www.amanda.org/docs/portusage.html`> for the current version of this document.

Chapter 23

AMANDA DUMPER API

23.1 Introduction

This is a proposal of a mechanism for *Amanda* to support arbitrary backup programs, that relies on a generic backup driver and scripts or programs that interface with backup programs such as **dump**, **tar**, smbclient, and others. It can also be used to introduce pre- and post-backup commands.

The interface is simple, but supports everything that is currently supported by *Amanda*, and it can be consistently extended to support new abstractions that may be introduced in the backup driver in the future.

This proposal does not imply any modification in the *Amanda* protocol or in *Amanda* servers; only *Amanda* clients have to be modified. By *Amanda* clients, we refer to hosts whose disks are to be backed up; an *Amanda* server is a host connected to a tape unit.

Currently (as of release 2.4.1 of *Amanda*), *Amanda* clients support three operations: self-check, estimate and backup.

Selfcheck is used by the server program **amcheck**, to check whether a client is responding or if there are configuration or permission problems in the client that might prevent the backup from taking place.

Estimates are requested by the *Amanda* **planner**, that runs on the server and collects information about the expected sizes of backups of each disk at several levels. Given this information and the amount of available tape space, the **planner** can select which disks and which levels it should tell dumper to run.

dumper is yet another server-side program; it requests clients to perform dumps, as determined by **planner**, and stores these dumps in holding disks or sends them directly to the **taper** program. The interaction between **dumper** and **taper** is beyond the scope of this text.

We are going to focus on the interaction between the *Amanda* client program and wrappers of dump programs. These wrappers must implement the DUMPER API. The dumptype option 'program' should name the wrapper that will be used to back up filesystems of that dumptype. One wrapper may call another, so as to extend its functionality.

23.2 The Problem

Different backup programs present distinct requirements; some must be run as super-user, whereas others can be run under other user-ids. Some require a directory name, the root of the tree to be backed up; others prefer a raw device name; some don't even refer to local disks (SAMBA). Some wrappers may need to know a filesystem type in order to decide which particular backup program to use (dump, vdump, vxdump, xfsdump, backup).

Some provide special options for estimates, whereas others must be started as if a complete dump were to be performed, and must be killed as soon as they print an estimate.

Furthermore, the output formats of these backup programs vary wildly. Some will print estimates and total sizes in bytes, in 512-byte tape blocks units, in Kbytes, Mbytes, Gbytes, and possibly Tbytes in the near future. Some will print a timestamp for the backup; some won't.

There are also restrictions related with possible scheduling policies. For example, some backup programs only support full backups or incrementals based on the last full backup (0-1). Some support full backups or incrementals based on the last backup, be it a full or an incremental backup (0-inf++). Some support incrementals based on a timestamp (incr/date); whereas others are based on a limited number of incremental levels, but incrementals of the same level can be repeated, such as dump (0-9).

Amanda was originally built upon DUMP incremental levels, so this is the only model it currently supports. Backup programs that use other incremental management mechanisms had to be adapted to this policy. Wrapper scripts are responsible for this adaptation.

Another important issue has to do with index generation. Some backup programs can generate indexes, but each one lists files in its own particular format, but they must be stored in a common format, so that the *Amanda* server can manipulate them.

The DUMPER API must accomodate for all these variations.

23.3 Overview of the API

We are going to define a standard format of argument lists that the backup driver will provide to wrapper programs, and the expected result of the execution of these wrappers.

The first argument to a wrapper should always be a command name. If no arguments are given, or an unsupported command is requested, an error message should be printed to stderr, and the program should terminate with exit status 1.

23.3.1 The 'support' command

As a general mechanism for *Amanda* to probe for features provided by a backup program, a wrapper script must support at least the 'support' command. Some features must be supported, and *Amanda* won't ever ask about them. Others will be considered as extensions, and *Amanda* will ask the wrapper whether they are supported before issuing the corresponding commands.

23.3.1.1 The 'level-incrementals' subcommand

For example, before requesting for an incremental backup of a given level, *Amanda* should ask the wrapper whether the backup program supports level-based incrementals. We don't currently support backup programs that don't, but we may in the future, so it would be nice if wrappers already implemented the command 'support level-incrementals', by returning a 0 exit status, printing, say, the maximum incremental level it supports, i.e., 9. A sample session would be:

```
% /usr/local/amanda/libexec/wrappers/dump support level-incrementals hda0 9
```

Note that the result of this support command may depend on filesystem information, so the disklist filesystem entry should be specified as a command line argument. In the next examples, we are not going to use full pathnames to wrapper scripts any more.

We could have defined a 'support' command for full backups, but I can't think of a backup program that does not support full backups...

23.3.1.2 The 'index' subcommand

The ability to produce index files is also subject to an invocation of 'support' command. When the support sub-command is 'index', like in the invocation below, the wrapper must print a list of valid indexing mechanisms, one per line, most preferred first. If indexing is not supported, nothing should be printed, and the exit status should be 1.

DUMP support index hda0

The currently known indexing mechanisms are:

output: implies that the command 'index-from-output' generates an index file from the output produced by the backup program (for example, from **tar -cv**).

image: implies that the command 'index-from-image' generates an index file from a backup image (for example, **tar -t**).

direct: implies that the 'backup' command can produce an index file as it generates the backup image.

parse: implies that the 'backup-parse' command can produce an index file as it generates the backup formatted output .

The indexing mechanisms will be explicitly requested with the additionnal option 'index-<mode>' in the 'backup' and 'backup-parse' command invocation.

'index-from-image' should be supported, if possible, even if other index commands are not, since it can be used in the future to create index files from previously backed up filesystems.

23.3.1.3 The 'parse-estimate' subcommand

The 'parse-estimate' support subcommand print a list of valid mechanisms to parse the estimate output and write the estimate size to its output, the two mechanisms are:

direct: implies that the 'estimate' command can produce the estimate output.

parse: implies that the 'estimate-parse' command can produce the estimate output when fed with the 'estimate' output.

The estimate parsing mechanisms will be explicitly requested with the additionnal option 'estimate-<mode>' in the 'estimate' and 'estimate-parse' command invocation.

23.3.1.4 The 'parse-backup' subcommand

The 'parse-backup' support subcommand print a list of valid mechanisms to parse the backup stderr, the two mechanisms are:

direct: implies that the 'backup' command can produce the backup-formatted-ouput.

parse: implies that the 'backup-parse' command can produce the backup-formatted-ouput when fed with the 'backup' stderr.

The backup parsing mechanisms will be explicitly requested with the additional option 'backup-<mode>' in the 'backup' and 'backup-parse' command invocation.

23.3.1.5 Other subcommands

Some other standard 'support' sub-commands are 'exclude' and 'exclude-list'.

One may think (and several people did :-)) that there should be only one support command, that would print information about all supported commands. The main arguments against this proposal have to do with extensibility:

The availability of commands might vary from filesystem to filesystem. No, I don't have an example, I just want to keep it as open as possible :-) one support subcommand may require command line arguments that others don't, and we can't know in advance what these command line arguments are going to be

The output format and exit status conventions of a support command may vary from command to command; the only pre-defined convention is that, if a wrapper does not know about a support subcommand, it should return exit status 1, implying that the inquired feature is not supported.

23.4 The 'selfcheck' command

We should support commands to perform self-checks, run estimates, backups and restores (for future extensions of the *Amanda* protocol so as to support restores)

A selfcheck request would go like this:

DUMP selfcheck hda0 option option=value ...

The options specified as command-line arguments are dumptype options enabled for that disk, such as 'index', 'norecord', etc. Unknown options should be ignored. For each successful check, a message such as:

OK [/dev/hda0 is readable] OK [/usr/sbin/dump is executable]

Errors should be printed as:

ERROR [/etc/dumpdates is not writable]

A wrapper script will certainly have to figure out either the disk device name or its mount point, given a filesystem name such as 'hda0', as specified in the disklist. In order to help these scripts, *Amanda* provides a helper program that can guess device names, mount points and filesystem types, when given disklist entries.

The filesystem type can be useful on some operation systems, in which more than one dump program is available; this information can help automatically selecting the appropriate dump program.

The exit status of **selfcheck** and of this alternate script are probably going to be disregarded. Anyway, for consistency, **selfcheck** should return exit status 0 for complete success, 1 if any failures have occurred.

23.5 The 'estimate' and 'estimate-parse' commands

Estimate requests can be on several different forms. An estimate of a full backup may be requested, or estimates for level- or timestamp-based incrementals:

DUMP estimate full hda0 option ... DUMP estimate level 1 hda0 option ... DUMP estimate diff 1998:09:24:01:02:03 hda0 option ...

If requested estimate type is not supported, exit status 3 should be returned.

If the option 'estimate-direct' is set, then the 'estimate' command should write to stdout the estimated size, in bytes, a pair of numbers that, multiplied by one another, yield the estimated size in bytes.

If the option 'estimate-parse' is set, then the 'estimate' command should write to stdout the informations needed by the 'estimate-parse' command, that should extract from its input the estimated size.

The syntax of 'estimate-parse' is identical to that of 'estimate'.

Both 'estimate' and 'estimate-parse' can output the word 'KILL', after printing the estimate. In this case, *Amanda* will send a SIGTERM signal to the process group of the 'estimate' process. If it does not die within a few seconds, a SIGKILL will be issued.

If 'estimate' or 'estimate-parse' succeed, they should exit 0, otherwise exit 1, except for the already listed cases of exit status 3.

23.6 The 'backup' and 'backup-parse' commands

The syntax of 'backup' is the same as that of 'estimate'. The backup image should be written to standard output, whereas stderr should be used for the user-oriented output of the backup program and other messages.

If the option 'backup-direct' is set, then the 'backup' command should write to stderr a formatted-output-backup.

If the option 'backup-parse' is set, then the 'backup' command should write to stderr the informations needed by the 'backup-parse' command, that should edit its input so that it prints to standard output a formatted-output-backup.

If the option 'no-record' is set, then the 'backup' command should not modify its state file (ex. **dump** should not modify `/etc/dumpdates`).

The syntax of 'backup-parse' is identical to that of 'backup'.

The syntax of the formatted-output-backup is as follow: All lines should start with either '| ' for normal output, '? ' for strange output or '& ' for error output. If the wrapper can determine the total backup size from the output of the backup program, it should print a line starting with '# ', followed by the total backup size in bytes or by a pair of numbers that, multiplied, yield the total backup size; this number will be used for consistency check.

The option 'index-direct' should cause commands 'backup' to output the index directly to file descriptor 3. The option 'index-parse' should cause commands 'backup-parse' to output the index directly to file descriptor 3. The syntax of the index file is described in the next section.

23.7 The 'index-from-output' and 'index-from-image' commands

The syntax of the 'index-from-output' and 'index-from-image' commands is identical to the one of 'backup'. They are fed the backup output or image, and they must produce a list of files and directories, one per line, to the standard output. Directories must be identified by the '/' termination.

After the file name and a blank space, any additional information about the file or directory, such as permission data, size, etc, can be added. For this reason, blanks and backslashes within filenames should be quoted with backslashes. Linefeeds should be represented as '\n', although it is not always possible to distinguish linefeeds in the middle of filenames from ones that separate one file from another, in the output of, say 'restore -t'. It is not clear whether we should also support quoting mechanisms such as '\xHH', '\OOO' or '\uXXXX'.

23.8 The 'restore' command

Yet to be specified.

23.9 The 'print-command' command

This command must be followed by a valid backup or restore command, and it should print a shell-command that would produce an equivalent result, i.e., that would perform the backup to standard output, or that would restore the whole filesystem reading from standard input. This command is to be included in the header of backup images, to ease crash-recovery.

23.10 Conclusion

Well, that's all. Drop us a note at the amanda-hackers mailing list (<mailto://amanda-hackers@amanda.org>) if you have suggestions to improve this document and/or the API. Some help on its implementation would be welcome, too.

NOTE

Refer to <http://www.amanda.org/docs/dumperapi.html> for the current version of this document.

Chapter 24

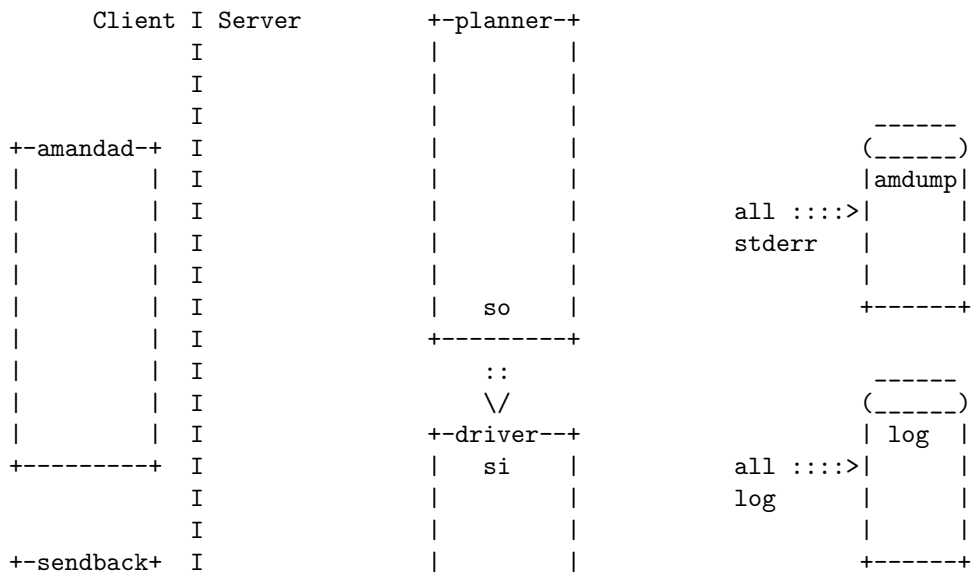
AMANDA INTERNALS

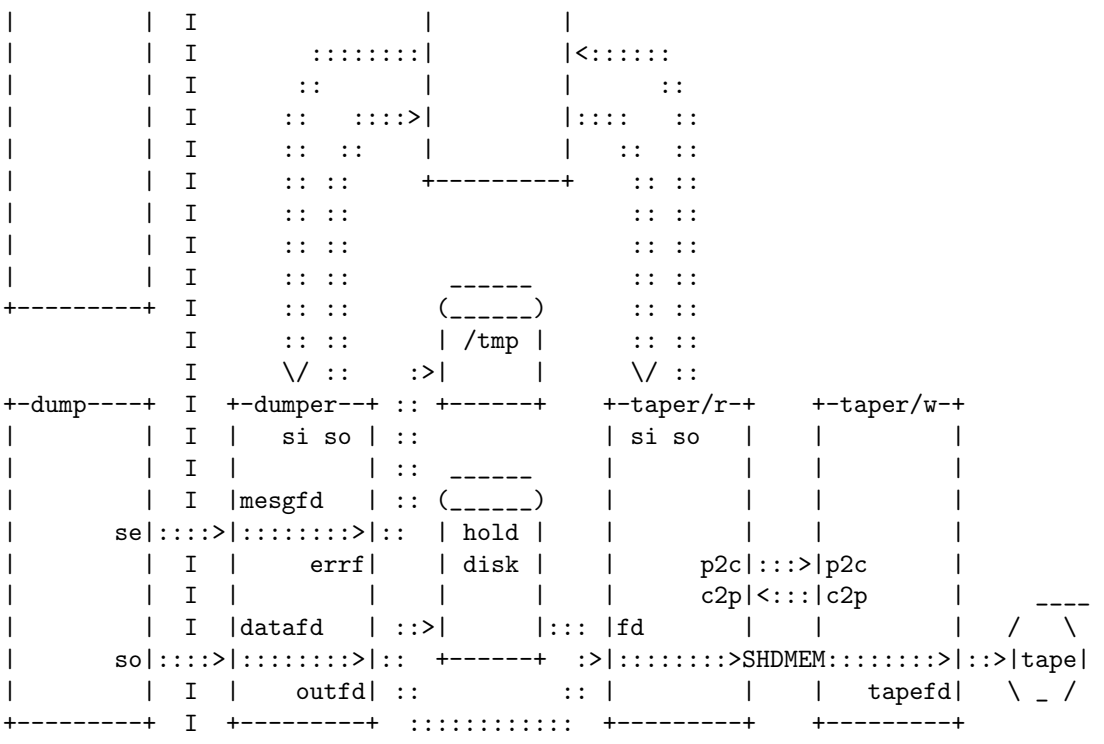
This is an attempt to document *Amanda*'s internals. Please feel free to make comments and suggest changes. Text for new sections gratefully accepted!

24.1 Protocols

NOTE

The following was an ASCII-illustration in the original docs, I managed to transfer it at last. Maybe someone will convert this to the first image in the *Amanda*-docs ;-) . sgw.





24.2 server and amandad on client

XXX - still to be done

24.3 planner and driver

planner interrogates all clients and generates a plan of which disks to backup and what dump level to do them at. The plan is plain text with one line per disk to be dumped. It is piped from planners stdout to drivers stdin. Plan lines come in two flavours:

For total dumps: <host> <disk> <pri> <lev> <size> <time> <deg lev> <deg size> <deg time>

For incremental dumps: <host> <disk> <pri> <lev> <size> <time>

Where: <host> Host name of client (from disklist file) <disk> Name of disk (from disklist file) <pri> Priority of backup (pri from disklist and amanda.conf + days overdue for total) <lev> Dump level for dump (0 for total, 1-9 for incremental) <size> Estimated size (in Kb after compression if requested) <time> Estimated time for backup (in seconds) <deg lev> <lev> to use if in degraded mode <deg size> <size> to use if in degraded mode <deg time> <time> to use if in degraded mode

24.4 driver and dumper

dumper talks via two pipes connected to each dumper's stdin and stdout. The commands and responses are plain text.

driver can ask **dumper** to do a dump to a file on the holding disk: FILE-DUMP <handle> <filename> <host> <disk> <level> <dumpdate> <chunksize> <prog> <options> or directly to taper: PORT-DUMP <handle> <port> <host> <disk> <level> <dumpdate> <prog> <options> or exit at the end of the run: QUIT

If the dump finishes correctly **dumper** replies with: DONE <handle> [<message>]

If something goes wrong with the dump, **dumper** can request that the dump be retried at a later time with: TRY-AGAIN <handle> [<message>] or, for fatal errors, be abandoned with: FAILED <handle> [<message>]

If the holding disk runs out of space, **dumper** will give: NO-ROOM <handle> and wait for **driver** to either fix the problem and say: CONTINUE or just say: ABORT in which case **dumper** kills the dump and replies with: ABORT-FINISHED <handle>

If **driver** says something that **dumper** doesn't recognise it responds with: BAD-COMMAND <message>

Where: <handle> Request ID <filename> Name of file (on holding disk) to write dump <port> Port (of **taper**) to send dump directly <host> Hostname of client <disk> Disk to backup <level> Dump level to do backup at <prog> Dump program to use <options> Options to pass to **sendbackup** <message> Error or status message

*driver and taper

driver talks via two pipes connected to taper's stdin and stdout. The commands and responses are plain text.

driver initialises **taper** with: START-TAPER <datestamp> to which **taper** replies with: TAPER-OK or, for fatal errors, with: TAPER-ERROR [<message>]

driver can ask **taper** to to copy a file from the holding disk to tape: FILE-WRITE <handle> <filename> <host> <disk> <level> or directly from a **dumper**: PORT-WRITE <handle> <host> <disk> <level> or exit at the end of the run: QUIT

taper responds to the PORT-WRITE command with: PORT <port> which **driver** should then hand on to **dumper** in a PORT-DUMP command.

taper responds to the QUIT command with: QUITING

If the copy to tape finishes correctly **taper** replies with: DONE <handle> [<message>]

If something goes wrong with the tape, **taper** can request that the dump be retried at a later time with: TRY-AGAIN <handle> [<message>] or, for fatal errors, be abandoned with: TAPE-ERROR <handle> [<message>]

If **driver** says something that **taper** doesn't recognise it responds with: BAD-COMMAND <message>

Where: <datestamp> Todays date as "yymmdd" <handle> Request ID <filename> Name of file (on holding disk) to write dump <port> Port (of **taper**) to send dump directly

<host> Hostname of client <disk> Disk to backup <level> Dump level to do backup at <message> Error or status message

24.5 taper(read) and taper(write)

There are two parts to **taper**: the file reader and the tape writer. Communication between the two sides is via a bit of shared memory for data transfer and two pipes (one in each direction) for synchronisation.

The shared memory area is made up of NBUFS (=20) buffers each of which contains a status word and a BUFFER_SIZE (=32*1024) byte data buffer.

The sync pipes are used to transfer a simplistic command sequence:

reader writer —— ——

Startup S<datestamp> —> <— S Start OK <— E<messge> Error

Open tape O<datestamp><hostname><diskname><level> —> <— O Opening

Write buffer W<bufnum> —> <— R<bufnum> Buffer empty <— E<message> Error <— T<message> Error, try again E ack e —> Protocol error X —> <— x X ack

Close tape C —> <— C<label><filenum><stats> Closing

Quit Q —>

NOTE

Refer to <`http://www.amanda.org/docs/internals.html`> for the current version of this document.

Chapter 25

AMANDA EVENT API

25.1 Introduction

This is a document of the API for the event handler. The purpose of the event handler is to allow scheduling and serialization of multiple different types of events.

25.2 The API

25.2.1 event_register

event_handle_t *event_register(event_id_t data, event_type_t type, event_fn_t callback, void *arg);

Sets up an event of the given type to call the given function with the given argument.

The 'data' argument is type specific.

EV_READFD, EV_WRITEFD - the file descriptor to monitor EV_SIG - the signal number to monitor EV_TIME - the number of seconds between each pulse EV_WAIT - the wait identifier used with event_wakeup() EV_DEAD - internal use only

25.2.2 event_release

void event_release(event_handle_t *handle);

Remove an event from the queue. This can happen at any time, even while the event is firing.

25.2.3 event_loop

void event_loop(int dontblock);

Process all pending events. If the argument is zero, this will keep running until all events have been released. If the argument is nonzero, this will do one pass over all pending events, and fire the ones that are immediately ready, and then return.

25.2.4 event_wait

void event_wait(event_id_t id);

Like event_loop(0), except that it will stop as soon as the event id is serviced.

25.2.5 event_wakeup

int event_wakeup(event_id_t id);

Fire all EV_WAIT events registered with an argument value of 'id' immediately. Returns the number of events that were fired.

25.3 Data types

25.3.1 event_handle_t

This is an opaque structure that describes a registered event. It is only useful to keep if you need to unregister the event later.

25.3.2 event_id_t

This is type-specific data. The contents and format depend on on the type of the event. This is an unsigned integral type.

25.3.3 event_type_t

This is an enumerated type describing the different events we handle.

25.3.4 event_fn_t

typedef void (*event_fn_t)(void *arg);

This is a function signature for the type of function that needs to get passed to event_register. The argument to the function is a pointer of the caller's choosing.

25.4 Event Types

25.4.1 EV_READFD

This type of event will fire when the file descriptor passed to event_register has data waiting to be read.

25.4.2 EV_WRITEFD

This type of event will fire when the file descriptor passed to event_register can have data written to it without blocking.

25.4.3 EV_SIG

This type of event will fire when the signal number passed to event_register has been caught. Note that if a signal is caught while processing is not in event_loop(), the event will be delayed until processing returns to event_loop().

25.4.4 EV_TIME

This type of event will fire repeatedly with a delay of the number of seconds passed to event_register between each interval.

25.4.5 EV_WAIT

This type of event will fire when someone calls event_wakeup() with the numeric argument equal to the argument this event was registered with.

NOTE

Refer to <http://www.amanda.org/docs/eventapi.html> for the current version of this document.

Chapter 26

AMANDA SECURITY API

26.1 Introduction

This is a document of the API for defining and utilizing multiple security and transport mechanisms for the *Amanda* network protocol.

The goal of this API is to allow several different forms of communication and authentication to exist between the *Amanda* server and its clients.

26.2 The Problem

There exist many potential ways that a user might wish to grant access to the *Amanda* daemon. The two currently supported are BSD (reserved port) and Kerberos IV security. The current implementation of these two methods is not very general, and adding additional methods requires a large amount of code to be modified.

Additionally, the current methods require the protocol and dump transport to be transmitted across the network using a pre-defined method. The *Amanda* protocol currently must be sent using udp datagrams to a well-known port, and the dumps are transported using tcp connections between ports negotiated via the protocol.

26.3 The API

The security API was designed to be a layer in between the core logic of *Amanda* and the transport and authentication of the protocol and dumps.

The component server and client programs now deal with abstract concepts instead of concrete udp and tcp handles.

The prefix "security_" is reserved for use as the namespace of this API.

26.3.1 protocol packet transmission functions

These functions exist for transmitting pkt_t's between the client and server.

These functions operate on security_handle_t objects. These objects are described later.

26.3.1.1 security_getdriver

const security_driver_t *security_getdriver(const char *drivername);

Given a security type ("KRB4", "BSD", "SSH", etc), returns a pointer to that type's security_driver_t (section 4.1), or NULL if no driver exists.

26.3.1.2 security_connect

void security_connect(const security_driver_t *h, const char *hostname, char *(*conf_fn)(char *arg, void *arg), void (*fn)(void *arg, security_handle_t *h, security_status_t s), void *arg);

Given a security driver, and a hostname, calls back with a security_handle_t (section 4.2) that can be used to communicate with that host. The status arg to the callback is reflects the success of the request. Error messages can be had via security_geterror().

This is expected to be the *Amanda* server's interface for setting up connections to clients.

conf_fn is used to determine configuration information. If NULL, no configuration information is available.

26.3.1.3 security_accept

void security_accept(const security_driver_t *h, int in, int out, void (*callback)(security_handle_t *, pkt_t *));

Given a security driver, an input file descriptor, and an output file descriptor, and a callback, when new connections are detected on the given file descriptors, the function is called with a newly created security handle and the initial packet received.

This is expected to be the *Amanda* daemon's interface for setting up incoming connections from the *Amanda* server. The file descriptors are typically 0 and 1 (stdin/stdout).

This function uses the event interface, and only works properly when event_loop() is called later in the program.

26.3.1.4 security_close

void security_close(security_handle_t *h);

Closes a connection created by a security_connect() or security_accept().

26.3.1.5 security_sendpkt

int security_sendpkt(security_handle_t *h, const pkt_t *pkt);

Transmits a pkt_t over a security handle. Returns 0 on success, or negative on error. A descriptive error message can be obtained via security_geterror().

26.3.1.6 security_recvpkt

int security_recvpkt(security_handle_t *h, void (*callback)(void *arg, pkt_t *pkt, security_status_t), void *arg, int timeout);

Requests that when incoming packets arrive for this handle, the given function is called with the given argument, the received packet, and the status of the reception.

If a packet does not arrive within the number of seconds specified in the 'timeout' argument, RECV_TIMEOUT is passed in the status argument of the timeout.

On receive error, the callback's status argument will be set to RECV_ERROR. An error message can be retrieved via security_geterror().

On successful reception, RECV_OK will be passed in the status argument, and the pkt argument will point to a valid packet.

This function uses the event interface. Callbacks will only be generated when event_loop() is called.

26.3.1.7 security_recvpkt_cancel

int security_recvpkt_cancel(security_handle_t *h);

Cancels a previous recvpkt request for this handle.

26.3.1.8 security_geterror

const char *security_geterror(security_handle_t *h);

Returns a descriptive error message for the last error condition on this handle.

26.3.1.9 security_seterror

void security_seterror(security_handle_t *h, const char *msg, ...);

Sets the string that security_geterror() returns.

26.3.1.10 security_handleinit

void security_handleinit(security_handle_t *, const security_driver_t *);

Initializes a security_handle_t. This is meant to be called only by security drivers to initialize the common part of a newly allocated security_handle_t.

26.3.2 stream functions

These functions exist for transmitting random data over a stream-like connection.

These functions operate on security_stream_t objects, which are described later.

26.3.2.1 security_stream_server

security_stream_t *security_stream_server(security_handle_t *h);

Creates the server end of a security stream, and will receive a connection from the host on the other end of the security handle passed.

Returns a security_stream_t on success, and NULL on error. Error messages can be obtained by calling security_geterror() on the security handle associated with this stream.

26.3.2.2 security_stream_accept

int security_stream_accept(security_stream_t *);

Given a security stream created by security_stream_server, blocks until a connection is made from the remote end.

Returns 0 on success, and -1 on error. Error messages can be obtained by calling security_stream_geterror().

26.3.2.3 security_stream_client

security_stream_t *security_stream_client(security_handle_t *h, int id);

Creates the client end of a security stream, and connects it to the machine on the other end of the security handle. The 'id' argument identifies which stream on the other end to connect to.

Returns a security_stream_t on success, and NULL on error. Error messages can be obtained by calling security_geterror() on the security handle associated with this stream.

26.3.2.4 security_stream_close

void security_stream_close(security_stream_t *s);

Closes a security stream and frees up resources associated with it.

26.3.2.5 security_stream_auth

int security_stream_auth(security_stream_t *s);

Authenticate a connected security stream.

Returns 0 on success, and -1 on error. Error messages can be obtained by calling security_stream_geterror().

26.3.2.6 security_stream_id

int security_stream_id(security_stream_t *s);

Returns an identifier which can be used to connect to this security stream with security_stream_client().

Typical usage would be for one end of a connection to create a stream with security_stream_server(), and then transmit the id for that stream to the other side. The other side will then connect to that id with security_stream_client().

26.3.2.7 security_stream_write

int security_stream_write(security_stream_t *s, const void *buf, size_t bufsize);

Writes a chunk of data to the security stream. Returns 0 on success, or negative on error. Error messages can be obtained by calling security_stream_geterror().

26.3.2.8 security_stream_read

void security_stream_read(security_stream_t *s, void (*callback)(void *arg, void *buf, int bufsize), void *arg);

Requests that when data is ready to be read on this stream, the given function is called with the given arg, a buffer full of data, and the size of that buffer.

On error, the bufsize will be negative. An error message can be retrieved by calling security_stream_geterror().

This function uses the event interface. Callbacks will only be generated while in event_loop().

26.3.2.9 security_stream_read_cancel

void security_stream_read_cancel(security_stream_t *s);

Cancels a previous read request.

26.3.2.10 security_stream_geterror

const char *security_stream_geterror(security_stream_t *h);

Returns a descriptive error message for the last error condition on this stream.

26.3.2.11 security_stream_seterror

void security_stream_seterror(security_stream_t *h, const char *msg, ...);

Sets the string that security_stream_geterror() returns.

26.4 Data Types

All visible data types are meant to be opaque to the caller. At no time should a caller have to access a member of any data type directly. The API should always be used instead.

26.4.1 security_driver_t

This is a static object containing function vectors that implement the API for a particular security type.

26.4.2 security_handle_t

This is an object that describes a protocol connection to a remote server. There is one security_handle_t per request, and there can be many to the same remote host.

26.4.3 security_stream_t

This is an object that describes a data connection to a remote host. It is always associated and derived from a security_handle_t. Arbitrary data can be passed over a security stream.

26.4.4 security_status_t

This is an enumerated type that is passed to the callback of security_recvpkt and security_connect. The possible values it can have are:

S_OK - the pkt_t was received fine S_TIMEOUT - no pkt_t was received within the time specified in the timeout argument to security_recvpkt(). S_ERROR - an error occurred during reception. Call security_geterror() for more information.

26.5 SECURITY DRIVERS

Each security type is defined by a struct of function vectors. These methods implement the details of this security type.

This section will document each element of security_driver_t.

26.5.1 name

const char *name;

This is the name of the driver. This is used by security_getdriver() to associate a name with a driver type.

26.5.2 connect

void (*connect)(const char *hostname, void (*fn)(void *, security_handle_t *, security_status_t), void *);

This is the implementation of security_connect(). It actually sets up the connection, and then returns a structure describing the connection. The first element of this structure MUST be a security_handle_t, because it will be cast to that after it is passed up to the caller.

The first argument is the host to connect to. The second argument is a function to call when a connection is made. The third argument is passed to the callback.

The callback takes three arguments. The first is the caller supplied void pointer. The second is a newly allocated security handle. The third is a security_status_t flag indicating the success or failure of the operation.

26.5.3 accept

void (*accept)(int in, int out, void (*callback)(security_handle_t *handle, pkt_t *pkt));

This is the implementation of security_accept(). It is passed the input and output file descriptors and a callback. The callback takes a security handle argument and also an initial packet received for that handle.

26.5.4 close

void (*close)(void *handle);

The implementation of security_close().

26.5.5 sendpkt

int (*sendpkt)(void *handle, pkt_t *pkt);

The implementation of security_sendpkt(). Security information is usually added by the driver before transmission.

26.5.6 recvpkt

void (*recvpkt)(void *handle, void (*callback)(void *arg, pkt_t *pkt, security_status_t), void *arg);

The implementation of security_recvpkt(). It will typically be layered onto the event interface somehow. It can assume that a caller will eventually call event_loop().

26.5.7 recvpkt_cancel

void (*recvpkt_cancel)(void *handle);

The implementation of security_recvpkt_cancel(). Drivers should allow this to be run even if no recvpkt was scheduled, or if one was previously cancelled.

26.5.8 stream_server

void *(*stream_server)(void *handle);

Implementation of security_stream_server(). This function returns a object describing the stream. The first member of this object MUST be a security_stream_t, because it will be cast to that.

26.5.9 stream_accept

int (*stream_accept)(void *stream);

After calling stream_server, stream_accept must be called on the stream before it is fully connected.

26.5.10 stream_client

void *(*stream_client)(void *handle, int id);

Implementation of security_stream_client(). The id argument is something returned by security_stream_id(). Again, the handle is referenced counted.

This function returns a object describing the stream. The first member of this object MUST be a security_stream_t, because it will be cast to that.

26.5.11 stream_close

void (*stream_close)(void *stream);

Close and free up resources for an open stream.

26.5.12 stream_auth

int (*stream_auth)(void *stream);

Authenticate a connected stream.

26.5.13 stream_id

int (*stream_id)(void *stream);

Return a unique id for this stream. This is to be used by stream_client() to connect to this stream.

26.5.14 stream_write

int (*stream_write)(void *stream, const void *buf, size_t bufsize);

Implementation of security_stream_write.

26.5.15 stream_read

void (*stream_read)(void *stream, void (*callback)(void *arg, void *buf, int bufsize), void *arg);

Implementation of security_stream_read.

26.5.16 stream_read_cancel

void (*stream_read_cancel)(void *stream);

Implementation of security_stream_read_cancel.

NOTE

Refer to <http://www.amanda.org/docs/security-api.html> for the current version of this document.

Chapter 27

VIRTUAL TAPE API

The upper level *Amanda* code (including some of the other tape_xxx routines) calls the following routines which implement a virtual tape table:

- int tape_access(filename, mode) Acts like access(2) on possibly virtual devices
- int tape_open(filename, mode) Acts like open(2) on possibly virtual devices
- int tape_stat(filename, buf) Acts like stat(2) on possibly virtual devices
- int tapefd_close(tapefd) Acts like close(2) on possibly virtual devices
- int tapefd_fsf(tapefd, count) Forward skips the (possibly virtual) device.
- int tapefd_read(tapefd, buffer, count) Reads a block from the (possibly virtual) device.
- int tapefd_rewind(tapefd) Reads a block from a (possibly virtual) device.
- void tapefd_resetofs(tapefd) Uses lseek() tricks to reset the write/read offset counter on a virtual tape device.
- int tapefd_unload(tapefd) Unloads the media from a (possibly virtual) device.
- int tapefd_status(tapefd) pr- ints status of the (possibly virtual) device to standard output.
- int tapefd_weof(tapefd, count) writes a filemark/moves to the next file on a device for writing.
- int tapefd_write(tapefd, buffer, count) writes a block of data to a (possibly virtual) device.

For a tape type xxx, the following routines must be provided, and entered - into the table "vtable" in tape-src/tapeio.c:

- int xxx_tape_access(filename, mode)
- int xxx_tape_open(filename, mode)
- int xxx_tape_stat(filename, buf)
- int xxx_tapefd_close(xxx_tapefd)
- int xxx_tapefd_fsf(xxx_tapefd, count)

- int xxx_tapefd_read(xxx_tapefd, buffer, count)
- int xxx_tapefd_rewind(xxx_tapefd)
- void xxx_tapefd_resetofs(xxx_tapefd)
- int xxx_tapefd_unload(xxx_tapefd)
- int xxx_tapefd_status(xxx_tapefd)
- int xxx_tapefd_weof(xxx_tapefd, count)
- int xxx_tapefd_write(xxx_tapefd, buffer, count)

Along with a prefix string which will identify the type. The initial vtape layer has two types "plain" and "rait" (Redundant Array of Inexpensive Tapes) but we hope to add a "rmt" (remote tape client), and possibly "dvd" types soon.

NOTE

Refer to <http://www.amanda.org/docs/vtape-api.html> for the current version of this document.

Chapter 28

USING KERBEROS WITH *AMANDA*

28.1 *Amanda* 2.5.0 - KERBEROS v4 SUPPORT NOTES

28.1.1 Configuration

The configure script defaults to:

```
#  define SERVER_HOST_PRINCIPLE "amanda"
#  define SERVER_HOST_INSTANCE  ""
#  define SERVER_HOST_KEY_FILE  "/.amanda"

#  define CLIENT_HOST_PRINCIPLE "rcmd"
#  define CLIENT_HOST_INSTANCE  HOSTNAME_INSTANCE
#  define CLIENT_HOST_KEY_FILE  KEYFILE

#  define TICKET_LIFETIME       128
```

You can override these with configure options if you so desire, with:

```
--with-server-principal=ARG    server host principal  [amanda]
--with-server-instance=ARG     server host instance   []
--with-server-keyfile=ARG      server host key file   [/.amanda]
--with-client-principal=ARG    client host principal  [rcmd]
--with-client-instance=ARG     client host instance   [HOSTNAME_INSTANCE]
--with-client-keyfile=ARG      client host key file   [KEYFILE]
--with-ticket-lifetime=ARG     ticket lifetime        [128]
```

The configure script will search under `/usr/kerberos/lib`, `/usr/cygnus/lib`, `/usr/lib`, and `/opt/kerberos/lib` for libkrb.a. (in that order) for the kerberos bits. If it finds them, kerberos support will be added in, if it doesn't, it won't. If the kerberos bits are found under some other hierarchy, you can specify this via the –with-krb4-security=DIR, where

DIR is where the kerberos bits live. It'll look under the 'lib' directory under this hierarchy for libkrb.a.

28.1.2 Installation

The kerberized *Amanda* service uses a different port on the client hosts. The `/etc/services` line is:

```
kamanda       10081/udp
```

And the `/etc/inetd.conf` line is:

```
kamanda dgram udp wait root /usr/local/libexec/amanda/amandad amandad -auth=krb4
```

Note that you're running this as root, rather than as your dump user. *Amanda* will set it's uid down to the dump user at times it doesn't need to read the srvtab file, and give up root permissions entirely before it goes off and runs dump. Alternately you can change your srvtab files to be readable by user amanda.

28.1.3 conf file

The following dumptype options apply to krb4:

```
auth "krb4" # use krb4 auth for this host
      # (you can mingle krb hosts & bsd .rhosts in one conf)
kencrypt # encrypt this filesystem over the net using the krb4
      # session key.  About 2x slower.  Good for those root
      # partitions containing your keyfiles.  Don't want to
      # give away the keys to an ethernet sniffer!
      # This is currently always enabled.  There is no
      # way to disable it.  This is a bug.
```

28.2 *Amanda* 2.5.0 - KERBEROS v5 SUPPORT NOTES

28.2.1 Building

You must specify –with-krb5-security to **configure**, otherwise there will be no attempt to look for kerberos binaries. You may specify a path that the system should look for the kerberos libraries, or leave it to the default.

By default, when –with-krb5-security is specified with with no path, the configure script will search under `/usr/kerberos/lib`, `/usr/cygnus/lib`, `/usr/lib`, and `/opt/kerberos/lib` for libkrb.a. (in that order) for the kerberos bits. If it finds them, kerberos support will be

added in, if it doesn't, it won't. If the kerberos bits are found under some other hierarchy, you can specify this via the –with-krb5-security=DIR, where DIR is where the kerberos bits live. It'll look under the 'lib' directory under this hierarchy for libkrb.a.

The krb5 driver script defaults to:

```
/*
 * The lifetime of our tickets in minutes.
 */
#define Amanda_TKT_LIFETIME     (12*60)

/*
 * The name of the service in /etc/services.
 */
#define Amanda_KRB5_SERVICE_NAME        "k5amanda"
```

You can currently only override these by editing the source.

The principal and keytab file that the amanda uses are genearlly set in the `amanda.conf` file (see below). You can hardcode this in the source if you really want to and that's described in common-src/krb5-security.c

28.2.2 Installation

The kerberized *Amanda* service uses a different port on the client hosts. The `/etc/services` line is:

```
k5amanda      10082/tcp
```

And the `/etc/inetd.conf` line is:

```
k5amanda stream tcp nowait root /usr/local/libexec/amanda/amandad amandad -auth=krb5
```

Note that you're running this as root, rather than as your dump user. *Amanda* will set it's uid down to the dump user at times it doesn't need to read the keytab file, and give up root permissions entirely before it goes off and runs **dump**. Alternately you can change your keytab files to be readable by user amanda. You should understand the security implications of this before changing the permissions on the keytab.

28.2.3 conf file

The following dumptype options apply to krb5:

```
auth "krb5" # use krb5 auth for this host
```

```
# (you can mingle krb hosts & bsd .rhosts in one conf)
```

The following two configuration directives are required in the `amanda.conf` file for kerberos 5 dumps to work:

```
krb5keytab
krb5principal
```

For example:

```
krb5keytab  "/etc/krb5.keytab-amanda"
krb5principal  "amanda/saidin.omniscient.com"
```

The principal in the second option must be contained in the first. The keytab should be readable by the amanda user. (and definitely not world readable!) This is (obviously) on the server. In MIT's kadmin, the following:

```
addprinc -randkey amanda/saidin.omniscient.com
ktadd -k /etc/krb5.keytab-amanda amanda/saidin.omniscient.com
```

will do the trick. You will obviously want to change the principal name to reflect something appropriate for the conventions at your site.

You must also configure each client to allow the amanda principal in for dumps. This is described in section 4.

28.2.4 Destination Host Permissions file

There are several ways to go about authorizing a server to connect to a client.

The normal way is via a `.k5amandausers` file or a `.k5login` file in the client user's home directory. The determination of which file to use is based on the way you ran configure on *Amanda*. By default, *Amanda* will use `.k5amandahosts`, but if you configured with –without-amandahosts, *Amanda* will use `.k5login`. (similar to the default for .rhosts/.amandahosts-style security). The `.k5login` file syntax is a superset of the default krb5 .k5login. The routines to check it are implemented in amanda rather than using krb5_kuserok because the connections are actually gssapi based.

This .k5amandahosts/.k5login is a hybrid of the .amandahosts and a .k5login file. You can just list principal names, as in a .k5login file and the principal will be permitted in from any host. If you do NOT specify a realm, then there is no attempt to validate the realm (this is only really a concern if you have cross-realm authentication set up with another realm or something else that allows you multiple realms in your kdc. If you do specify a realm, only that principal@realm will be permitted to connect.

You may prepend this with a hostname and whitespace, and only that principal (with optional realm as above) will be permitted to access from that hostname.

Here are examples of valid entries in the `.k5amandahosts`:

```
service/amanda
service/amanda@TEST.COM
dumpmaster.test.com service/amanda
dumpmaster.test.com service/amanda@TEST.COM
```

Rather than using a `.k5amandahosts` or `.k5login` file, the easiest way is to use a principal named after the destination user, (such as amanda@TEST.COM in our example) and not have either a `.k5amandahosts` or `.k5login` file in the destination user's home directory.

NOTE

There is no attempt to verify the realm in this case (only a concern if you have cross-realm authentication setup).

NOTE

Refer to <`http://www.amanda.org/docs/kerberos.html`> for the current version of this document.

Part VI

Historical files

OLD AND OUTDATED MATERIAL, PROPOSALS AND DRAFTS.

Here you find some chapters which contain outdated informations. These chapters nonetheless can help to get a more complete picture of *Amanda*.

NOTE

Please realize that the following does not necessarily describe current features of *Amanda*. There are also features described which were never added to *Amanda*.

Chapter 29

RESPONSE TO CPIO SECURITY NOTICE ISSUE 11:

The *Amanda* development team confirms the existence of the **amrecover** security hole in recent versions of *Amanda*. We have made a new release, *Amanda* 2.4.0b5, that fixes the **amrecover** problem and other potential security holes, and is the product of a security audit conducted in conjunction with the OpenBSD effort. The new version is available at:

<ftp://ftp.amanda.org/pub/amanda/amanda-2.4.0b5.tar.gz>

Here's some more information about the **amrecover** problem to supplement the information given in the CPIO Security Notice:

29.1 Affected Versions

The *Amanda* 2.3.0.x interim releases that introduced **amrecover**, and the 2.4.0 beta releases by the *Amanda* team are vulnerable.

Amanda 2.3.0 and earlier UMD releases are not affected by this particular bug, as **amrecover** was not part of those releases. However, earlier releases do have potential security problems and other bugs, so the *Amanda* Team recommends upgrading to the new release as soon as practicable.

29.2 Workaround

At an active site running *Amanda* 2.3.0.x or 2.4.0 beta, **amrecover**/ **amindexd** can be disabled by:

- removing **amandaidx** and **amidxtape** from `/etc/inetd.conf`
- restarting `/etc/inetd.conf` (kill -HUP should do)

This will avoid this particular vulnerability while continuing to run backups. However, other vulnerabilities might exist, so the *Amanda* Team recommends upgrading to the new release as soon as practicable.

29.3 Acknowledgements

This release (2.4.0) has addressed a number of security concerns with the assistance of Theo de Raadt, Ejovi Nuwere and David Sacerdote of the OpenBSD project. Thanks guys! Any problems that remain are our own fault, of course.

The *Amanda* Team would also like to thank the many other people who have contributed suggestions, patches, and new subsystems for *Amanda*. We're grateful for any contribution that helps us achieve and sustain critical mass for improving *Amanda*.

NOTE

Refer to <http://www.amanda.org/docs/security.html> for the current version of this document.

Chapter 30

UPGRADE ISSUES

Amanda 2.4.0 has introduced a major incompatibility in the *Amanda* protocol. This means that pre-2.4.0 clients won't interoperate with a 2.4.0 server, nor will 2.4.0 clients interoperate with pre-2.4.0 servers. You have to upgrade them all at the same time.

To ease the upgrade process *Amanda* has, from release 2.4.0 on, a configure flag (–with-testing) that will cause *Amanda* to use alternate service names (*Amanda*-test) instead of the standard ones. This allows you to keep using your old version of *Amanda* while you test the new one.

Depending upon the version of *Amanda* you are upgrading from, *Amanda* may use a different database library to store the backup information, and the new *Amanda* may not be able to read the old *Amanda* database files. In this case, you will want to do something like the following:

Before the upgrade (using the old version of **amadmin**):

```
# cd /var/AMANDA/CONFIG
# amadmin CONFIG export > zzz
# mkdir backup
# mv curinfo* backup
```

and after the upgrade (using the new version of **amadmin**):

```
# cd /var/AMANDA/CONFIG
# amadmin CONFIG import < zzz
```

and a month :-) after you are happy with the new version:

```
# cd /var/AMANDA/CONFIG
# rm -rf backup
```

After 2.4.0, the structure of the directory holding the index files was changed to have three levels instead of being flat. This greatly reduces the number of files in a given directory,

which was a problem for some systems.

The new layout is: `[indexdir]/hostname/filesystem/YYYYMMDD_L.gz` where hostname and filesystem are "sanitized" versions of the names from `disklist`, i.e. '/' characters are converted to '_' and so on. This new naming convention matches the one used for the text formatted database.

A script is available to convert the flat directory structure to the new layout:

<http://www.amanda.org/2.4-conv/msg00428.html>

NOTE

Refer to <http://www.amanda.org/docs/upgrade.html> for the current version of this document.

Chapter 31

WHAT ONCE WAS NEW

31.1 What's new in *Amanda* 2.3

This document contains notes on new features in *Amanda* 2.3 that may not yet be fully documented elsewhere.

31.1.1 Indexing backups for easier restore

Read more about this in the file named Indexing with *Amanda*.

31.1.2 Samba Support

Read more about this in the file named Backup PC hosts using Samba.

31.1.3 GnuTar Support

Amanda now supports dumps made via GnuTAR. To use this, set your dumptypes set the program name to "GNUTAR":

```
dumptype tar-client {
....
program "GNUTAR"
}
```

Since Gnu TAR does not maintain a dumpdates file itself, nor give an estimate of backup size, those need to be done within *Amanda*. *Amanda* maintains an `/etc/amandates` file to track the backup dates analogously to how **dump** does it.

NOTE: if your /etc directory is not writable by your dumpuser, you'll have to create the empty file initially by hand, and make it writable by your dumpuser ala `/etc/dumpdates`.

NOTE: Since **tar** traverses the directory hierarchy and reads files as a regular user would, it must run as root. The two new *Amanda* programs **calcsize** and **runtar** therefore must

be installed setuid root. I've made them as simple as possible to to avoid potential security holes.

31.1.4 Multiple backups in parallel from one client host

A new "maxdumps" parameter for the conf file gives the default value for the amount of parallelism per client:

```
maxdumps 2 # default max num. dumps to do in parallel per client
```

If this default parameter is not specified, the default for the default :-0 is 1. Then, you can override the parameter per client through the dumptype, eg:

```
dumptype fast-client {
....
maxdumps 4
}
```

If the "maxdumps" parameter isn't given in the dumptypes, the default is used. The idea is that maxdumps is set roughly proportional to the speed of the client host. You probably wont get much benefit from setting it very high, but all but the slowest hosts should be able to handle a maxdumps of at least 2.

Amanda doesn't really have any per-host parameters, just per-disk, so the per-client-host maxdumps is taken from the last disk listed for that host.

Just to make things more complicated, I've added the ability to specify a "spindle number" for each filesystem in the `disklist` file. For example:

```
wiggum /        fast-comp-user  0
wiggum /usr     fast-comp-user  0
wiggum /larry   fast-comp-user  1
wiggum /curly   fast-comp-user  1
wiggum /moe     fast-comp-user  1
wiggum /itchy   fast-comp-user  2
wiggum /scratchy fast-comp-user 3
```

The spindle number represents the disk number, eg every filesystem on sd0 can get a spindle number of 0, everything on sd1 gets spindle 1, etc (but there's no enforced requirement that there be a match with the underlying hardware situation). Now, even with a high maxdumps, *Amanda* will refrain from scheduling two disks on the same spindle at the same time, which would just slow them both down by adding a lot of seeks.

The default spindle if you don't specify one is -1, which is defined to be a spindle that doesn't interfere with itself. That is if you don't specify any spindle numbers, any and all filesystems on the host can be scheduled concurrently up to the maxdumps.

Just to be clear, there's no relation between spindle numbers and maxdumps: number the spindles by the disks that you have, even if that's more than maxdumps.

Also, I'm not sure that putting spindle numbers everywhere is of much value: their purpose is to prevent multiple big dumps from being run at the same time on two partitions on the same disk, on the theory that the extra seeking between the partitions would cause the dumps to run slower than they would if they ran sequentially. But, given the client-side compression and network output that must occur between blocks read from the disk, there may be enough slack time at the disk to support the seeks and have a little parallelism left over to do some good.

31.1.5 Multiple tapes in one run

I've rewritten the **taper** - it now supports one run spanning multiple tapes if you have a tape-changer. The necessary changes in support of this have also been made to **driver** and **reporter** - **planner** already had support. There are a couple other places that should probably be updated, like **amcheck**. Dumps are not split across tapes - when **taper** runs into the end of a tape, it loads the next tape and tells **driver** to try sending the dump again.

If you are feeling brave, set "runtapes" to something other than 1.

The new **taper** also keeps the tape open the entire time it is writing the files out - no more having **amcheck**s or other accesses/rewinds in the middle of the run screw you royally if they hit when the tape is closed for writing a filemark.

31.1.6 Bottleneck determination

I've made some experimental changes to **driver** to determine what the bottleneck is at any time. Since *Amanda* tries to do many things at once, it's hard to pinpoint a single bottleneck, but I "think" I've got it down well enough to say something useful. For now it just outputs the current bottleneck as part of its "driver: state" line in the debug output, but once I'm comfortable with its conclusions, I'll output it to the log file and have the **reporter** generate a nice table. The current choices are:

- not-idle - if there were dumps to do, they got done
- no-dumpers - there were dumps to do but no dumpers free
- no-hold - there were dumps to do and dumpers free but the dumps
- couldn't go to the holding disks (no-hold conf flag)
- no-diskspace - no diskspace on holding disks
- no-bandwidth - ran out of bandwidth
- client-constrained - couldn't start any dumps because the clients were busy

31.1.7 2 Gb limit removed

I've fixed the 2-gig limits by representing sizes in Kbytes instead of bytes everywhere. This gives us a new 2 TB dump-file size limit (on 32bit machines), which should last us a couple more years. This seemed preferable to me than going to long-long or some other non-portable type for the size.

31.1.8 amadmin import/export

amadmin now has "import" and "export" commands, to convert the curinfo database to/from text format, for: moving an *Amanda* server to a different arch, compressing the database after deleting lots of hosts, or editing one or all entries in batch form or via a script.

31.2 What's new in *Amanda* 2.2

NOTE

Here's the old 2.2.x stuff from this file. I'm pretty sure most of this is in the mainline documentation already.

This document contains notes on new features in *Amanda* 2.2 that may not yet be fully documented elsewhere.

31.2.1 Client side setup has changed

The new `/etc/services` lines are:

```
amanda        10080/udp     # bsd security Amanda daemon
kamanda       10081/udp     # krb4 security Amanda daemon
```

The new `/etc/inetd.conf` lines are:

```
amanda  dgram udp wait /usr/local/libexec/amanda/amandad amandad
kamanda dgram udp wait /usr/local/libexec/amanda/amandad amandad -krb4
```

(you don't need the vanilla *Amanda* lines if you are using kerberos for everything, and vice-versa)

31.2.2 Version suffixes on executables

The new USE_VERSION_SUFFIXES define in options.h controls whether to install the *Amanda* executables with the version number attached to the name, eg "amdump-2.2.1". I recommend that you leave this defined, since this allows multiple versions to co-exist - particularly important while *Amanda* 2.2 is under development. You can always symlink the names without the version suffix to the version you want to be your "production" version.

31.2.3 Kerberos

Read the comments in Using Kerberos with *Amanda* for how to configure the kerberos version. With KRB4_SECURITY defined, there are two new dumptype options:

- ```
  krb4-auth   # use krb4 auth for this host
        # (you can mingle krb hosts & bsd .rhosts in one conf)
  ```

- ```
  kencrypt # encrypt this filesystem over the net using the krb4
        # session key.  About 2x slower.  Good for those root
        # partitions containing your keyfiles.  Don't want to
        # give away the keys to an ethernet sniffer!
  ```

31.2.4 Multiple holding disks

You can have more than one holding disk for those really big installations. Just add extra diskdir and disksize lines to your `amanda.conf`:

NOTE

sgw: This is OLD syntax now ...

```
diskdir "/Amanda2/Amanda/work"  # where the holding disk is
disksize 880 MB                 # how much space can we use on it

diskdir "/dumps/Amanda/work"    # a second holding disk!
disksize 1500 MB
```

Amanda will load-balance between the two disks as long as there is space. *Amanda* now also actually stats files to get a more accurate view of available and used disk space while running.

31.2.5 Remote self-checks

amcheck will now cause self-checks to run on the client hosts, quickly detecting which hosts are up and communicating, which have permissions problems, etc. This is amazingly fast for what it does: here it checks more than 130 hosts in less than a minute. My favorite gee-whiz new feature! The new -s and -c options control whether server-only or client-only checks are done.

31.2.6 mmap support

System V shared memory primitives are no longer required on the server side, if your system has a version of mmap() that will allocate anonymous memory. BSD 4.4 systems (and OSF/1) have an explicitly anonymous mmap() type, but others (like SunOS) support the trick of mmap'ing /dev/zero for the same effect. *Amanda* should work with both varieties.

Defined HAVE_SYSVSHM or HAVE_MMAP (or both) in config.h. If you have both, SYSVSHM is selected (simply because this code in *Amanda* is more mature, not because the sysv stuff is better).

31.2.7 gzip support

This was most requested feature #1; I've finally slipped it in. Define HAVE_GZIP in options.h. See options.h-vanilla for details. There are two new `amanda.conf` dumptype options "compress-fast" and "compress-best". The default is "compress-fast". With gzip, compress-fast seems to always do better than the old lzw compress (in particular it will never expand the file), and runs faster too. Gzip's compress-best does very good compression, but is about twice as slow as the old lzw compress, so you don't want to use it for filesystems that take a long time to full-dump anyway.

31.2.8 Mount point names in disklist

Most requested feature #2: You can specify mount names in the `disklist` instead of dev names. The rule is, if the filesystem name starts with a slash, it is a mount point name, if it doesn't, it is a dev name, and has DEVDIR prepended. For example:

```
obelix   sd0a     # dev-name: /dev/sd0a
obelix   /obelix     # mount name: /obelix, aka /dev/sd0g
```

31.2.9 Initial tape-changer support included

A new `amanda.conf` parameter, tpchanger, controls whether *Amanda* communicates with a tape changer program to load tapes rather than just opening the tapedev itself. The tpchanger parameter is a string which specifies the name of a program that follows the API specified in *Amanda* Tape Changer Support. Read that for more information.

31.2.10 Generic tape changer wrapper script

An initial tape-changer glue script, chg-generic.sh, implements the *Amanda* changer API using an array of tape devices to simulate a tape changer, with the device names specified via a conf file. This script can be quickly customized by inserting calls tape-changer-specific programs at appropriate places, making support for new changers painless. If you know what command to execute to get your changer to put a particular tape in the drive, you can get *Amanda* to support your changer.

The generic script works as-is for sites that want to cascade between two or more tape drives hooked directly up to the tape server host. It also should work as-is with tape-changer drivers that use separate device names to specify the slot to be loaded, wheres simply opening the slot device causes the tape from that slot to be loaded.

chg-generic has its own small conf file. See example/chg-generic.conf for a documented sample.

31.2.11 New command amtape

amtape is the user front-end to the *Amanda* tape changer support facilities. The operators can use **amtape** to load tapes for restores, position the changer, see what *Amanda* tapes are loaded in the tape rack, and see which tape would be picked by **taper** for the next **amdump** run.

No man page yet, but running **amtape** with no arguments gives a detailed usage statement. See *Amanda* Tape Changer Support for more info.

NOTE

sgw: The manpage exists now.

31.2.12 Changer support added to command amlabel

The **amlabel** command now takes an optional slot argument for labeling particular tapes in the tape rack. See *Amanda* Tape Changer Support for more info.

31.2.13 Tape changer support improved

The specs in *Amanda* Tape Changer Support have been updated, and the code changed to match. The major difference is that *Amanda* no longer assumes slots in the tape rack are numbered from 0 to N-1. They can be numbered or labeled in any manner that suits your tape-changer, *Amanda* doesn't care what the actual slot names are. Also added "first" and "last" slot specifiers, and an -eject command.

The chg-generic.sh tape changer script now has new "firstslot", "lastslot", and "needeject" parameters for the chg-generic.conf file. It now keeps track of whether the current slot is loaded into the drive, so that it can issue an explicit eject command for those tape changers that need one. See example/chg-generic.conf for more info.

31.2.14 A few words about multi-tape runs

I'm still holding back on support for multiple tapes in one run. I'm not yet completely happy with how *Amanda* should handle splitting dumps across tapes (eg when end-of-tape is encountered in the middle of a long dump). For example, this creates issues for **amrestore**, which currently doesn't know about configurations or tape changers — on purpose, so that you can do restores on any machine with a tape drive, not just the server, and so that you can recover with no online databases present.

However, because the current snapshot DOES support tape changers, and multiple runs in one day, some of the benefit of multi-tape runs can be had by simply running *Amanda* several times in a row. Eg, to fill three tapes per night, you can put

```
amdump <conf>; amdump <conf>; amdump <conf>
```

in you crontab. On the down side, this will generate three reports instead of one, will do more incremental dumps than necessary, and will run slower. Not very satisfying, but if you *need* to fill more than one tape per day NOW, it should work.

31.2.15 Big planner changes

The support for writing to multiple tapes in one run is almost finished now. See Multitape support in *Amanda* 2.2 for an outline of the design. The **planner** support for this is included in this snapshot, but the **taper** part is not.

There is a new `amanda.conf` variable "runtapes" which specifies the number of tapes to use on each **amdump** run. For now this should stay at 1, the default. Also, the old "mincycle" and "maxcycle" `amanda.conf` variables are deprecated, but still work for now. "maxcycle" was never used, and "mincycle" is now called "dumpcycle".

There are two visible differences in the new **planner**: First, **planner** now thinks in real-time, rather than by the number of tapes as before. That is, a filesystem is due for a full backup once every <dumpcycle> days, regardless of how many times *Amanda* is run in that interval. As a consequence, you need to make sure the dumpcycle variable marks real time instead of the number of days. For example, previously "mincycle 10" worked for a two week cycle if you ran **amdump** only on weekdays (for 10 runs in a cycle). Now a two week cycle must be specified as "dumpcycle 14" or "dumpcycle 2 weeks". The "2 weeks" specifier works with both the old and new versions of planner, because previously "weeks" multiplied by 5, and now it multiplies by 7.

Second, **planner** now warns about impending overwrites of full backups. If a filesystem's last full backup is on a tape that is due to be overwritten in the next 5 runs, **planner** will give you a heads-up about it, so that you can restore the filesystem somewhere, or switch

that tape out of rotation (substitute a new tape with the same label). This situation often occurs after a hardware failure brings a machine or disk down for some days.

31.2.16 Level-0 dumps allowed with no tape

If there is no tape present (or the tape drive fails during dumping), *Amanda* switches to degraded mode. In degraded mode, level-0 dumps are not allowed. This can be a pain for unattended sites over the weekend (especially when there is a large holding disk that can hold any necessary dumps). *Amanda* now supports a new configuration file directive, "reserve". This tells *Amanda* to reserve that percentage of total holding disk space for degraded mode dumps. Example: your total holding disk space adds up to 8.4GB. If you specify a reserve of 50, 4.2GB (50%) of the holding disk space will be allowed to be used for regular dumps, but if that limit is hit, *Amanda* will switch to degraded dumps. For backward compatibility, if no 'reserve' keyword is present, 100 will be assumed (e.g. never do full dumps if degraded mode is in effect).

NOTE

This percentage applies from run to run, so, as in the previous example, when *Amanda* runs the next day, if there is 3.8GB left on the holding disk, 1.9GB will be reserved for degraded mode dumps (e.g. the percentage keeps sliding).

NOTE

Refer to <http://www.amanda.org/docs/whatwasnew.html> for the current version of this document.

Chapter 32

MULTITAPE SUPPORT IN *AMANDA* 2.2

32.1 Introduction

The goal of this enhancement is to make *Amanda* independent of the number of tapes used per run or even per dump cycle. Specifically, I would like *Amanda* to handle the following:

- output of **amdump** run goes to more than one tape
- a single dump file can straddle two tapes
- more than one **amdump** run can be done in a single day
- **planner** should not care how many runs per cycle occur

And later:

- multiple runs of **amdump** can go onto one tape (eg an append mode)
- any dump files from a previous run that are on the holding disk are written to tape in this run (eg eliminate **amflush**)
- **taper** write to multiple tape drives simultaneously

32.2 New Planner Algorithm

32.2.1 Time

Previously, planner marked time by the number of **amdump** runs, which it equated with number of tapes, and number of days. In *Amanda* 2.2, *Amanda* keeps track of the real passage of time, and doesn't generally care about the number of runs or tapes between any two events.

While *Amanda* 2.2 doesn't care about spacing between runs, dump cycles are still in terms of days, to make things easy to understand for the user. So, time differences are rounded to the nearest 24 hours:

days_diff(A,B) = (<B> - <A> + 86400/2) / 86400

Where the times A and B are in seconds since the Unix epoch, and 86400 is the number of seconds per day. This rounds a 2.49 day difference down to 2 days, and a 2.5 day difference up to 3 days. No, Olafur, Unix time does not handle leap seconds. Give me a break. :-)

32.2.2 Full Backups

The first thing **planner** does is calculate when each filesystem is due for a full backup. This is trivial for normal backups:

full_is_due = days_diff(<time of last full>, <curtime>) >= dumpcycle

There is a complication for "skip-full" filesystems. Under 2.2, these will be skipped on any runs that occur on the day the full is due, but we have to do the right thing if multiple runs are done that day, and if no runs are done that day (in which case we should be doing an incremental). Also, the time of last full dump is a fiction maintained by the **planner** – *Amanda* has no way to tell whether the full backup was actually done or when it was done:

```
if(skip-full) {
   if(full_is_due)
      <time of last full> += dumpcycle;
   if(days_diff(<time of last full>, <curtime>) == 0)
      skip the filesystem on this run;
   else
      do an incremental dump of this filesystem;
}
```

32.2.3 Schedule Balancing

The runtapes parameter tells **planner** how many tapes it should plan to use each run. It multiplies this by the tape length to get the size available for the run. (NOTE: later amend this size if appending to tapes, or if there are dumps on the holding disk waiting to be flushed). Other than the size calculation, **planner** doesn't really care how many tapes will be written to.

The fundamental problem with attempting to balance the schedule is that we no longer know how many **amdump** runs will be done in a full cycle. The number may change from cycle to cycle if there are extenuating circumstances.

So, **planner** must guess at how many runs will be done in one cycle, by looking at the information for the last cycle, or, if this is the first cycle, assuming one run for each day in the dump cycle.

32.2.4 Overwrite Detection

When can a tape be overwritten, considering that it might have old dumps on it? We want to be able to warn when full dumps are going to be overwritten, but given the possibility of old files on the tape, how can we know when the tape is no longer needed? I think we

can get this when going through the info file, considering each full dump and what tape it is on. Make sure we correctly handle stale information.

32.3 Taper Algorithm

32.3.1 Choosing a tape

taper must now handle writing to multiple tapes in one night, but choosing the tapes from the tape rack is done one at a time as needed, re-applying the same algorithm each time (see *Amanda* Tape Changer Support).

32.3.2 End of tape handling

As in earlier versions of *Amanda*, **taper** itself does not try to restrict writing to the tape size given in the config file. It relied on **planner** having correctly estimated backup sizes and limiting itself to what would fit on one tape.

Now, **taper** needs to switch to a new tape when the current tape has filled up. The tape is considered full when **taper** gets a write error. This will most likely occur in the middle of writing a (potentially large) backup file, perhaps even from a direct-to-tape socket, so there is no possibility of starting the backup file over again on the next tape, it must start from where it left off, rewriting the block that got the error on the next tape.

To insure correct operation, the file header of the continued file should contain an indication that it is a continuation, and at what offset. **amrestore** of course needs to be aware of this scheme and handle it correctly, perhaps by double-buffering internally. XXX provide more alg details here, or just leave it with the general idea?

32.3.3 Tape Format Changes

We need to specify the sequence number of the tape in the run, in the tape header file. The file header block specifies whether it is a continuation file or not.

32.3.4 Tapelist File Changes

The lines in the `tapelist` file should contain the sequence number of the tape in its run, as well as the amount of data written on the tape, and perhaps whether or not the end of tape was reached.

NOTE

Refer to <`http://www.amanda.org/docs/multitape.html`> for the current version of this document.

Chapter 33

THOUGHTS ABOUT A STRATEGY API

Subject: STRATEGY API (was: Re: spelunking)
From: Alexandre Oliva oliva@dcc.unicamp.br
Date: 03 Oct 1998 02:44:47 -300

Doug Hughes Doug.Hughes@Eng.Auburn.EDU writes:

> I'm going to (try to) modify the amanda stuff such that there is a new
> parameter called dumplevel available in dumptype. That way I can fix
> the dump level at whatever I want (5 or 9 when I need to), without
> having to worry about whether it's going to try and skip a 0 and miss
> a dump, or try to do a 1, or whatever.

Now that you mention that, it comes to my mind that the current
mechanism to define backup strategies is too limited, and we could try
to improve it just like we are going to do with the DUMPER API.

We could define a STRATEGY API, that planner would use to:

1) define a set levels and/or dates for which estimates should be requested

2) select a subset of the estimate results that planner can choose

and driver would run to inform that a backup has succeeded, after
having updated the database.

I haven't fully analysed the implications of this choice, but it looks
quite feasible and very useful. Opinions? Requests of clarification?
Random flames? :-)

Anyone willing to pursue this issue?

```
--
Alexandre Oliva
```

NOTE

Refer to <http://www.amanda.org/docs/strategy-api.html> for the current version of this document.

Chapter 34

Y2K COMPLIANCY

The *Amanda* developers believe *Amanda* is Y2K-compliant, as long as the underlying operating system and C library are. The only date manipulations performed by *Amanda* use C-language time manipulation functions and/or strings where years are represented with the century-included notation.

NOTE

Refer to <`http://www.amanda.org/docs/y2k.html`> for the current version of this document.

Chapter 35

USAGE OF FLOPPY TAPE DRIVES ON LINUX

Amanda now supports the ftape driver version 3.04d (see <`http://www-math.math.rwth-aachen.de/~LBFM/claus/ftape`>). It adjusts the blocksize automatically to 32k and supports QIC volume tables.

It uses only one **open()** call for writing backups to one tape, so the "busy"-lamp of the drive will be on all the time. (With normal *Amanda* code it would be off in pauses between two files written to tape.)

For volume table support you have to get libvtblc first (available at <`ftp://pc02-stat.sci.uni-klu.ac.at/pub/Linux/libvtblc`>.) This library contains the functions and subroutines from the vtblc utility, distributed with ftape. (Maybe this library will be part of a future ftape package).

You have to set the raw tape device for volume table operations, usually `/dev/rawft0`, either via configure –with-ftape-rawdevice= or with rawtapedev in `amanda.conf` (**configure** checks for `/dev/rawft[0-3]`, to get a guess for this value). Dont forget to make this device read/writeable for your backup user.

For compilation you need the header files from ftape 3.04d in your include tree.

The volumetable of a tape "amlabeled" TEST-VOL2 with 4 backups on it would look like (listed with vtblc utility from ftape):

```
prompt: vtblc

 Nr  Id          Label                    Date          Start      End    Space

  0 VTBL TEST-VOL2              23:08:06 03/15/98        3        4    0.00 %
  1 VTBL gamma //beta/C 0       00:00:00 03/15/98        5      374    0.68 %
  2 VTBL gamma sda2 0           00:00:00 03/15/98      375     1029    1.21 %
  3 VTBL alpha sda2 0           00:00:00 03/15/98     1030     1906    1.62 %
  4 VTBL alpha sda6 0           00:00:00 03/15/98     1907     8092   11.45 %
  5 VTBL Amanda Tape End        01:45:15 03/16/98     8093     8094    0.00 %
```

With lvtblc, currently available with the libvtblc library, you can list the complete label strings (44 characters, not only the first 22 characters as with vtblc):

```
prompt: lvtblc -l

 Nr  Id          Label                                          Date

  0 VTBL TEST-VOL2                                        23:08:06 03/15/98
  1 VTBL gamma //beta/C 0                                 00:00:00 03/15/98
  2 VTBL gamma sda2 0                                     00:00:00 03/15/98
  3 VTBL alpha sda2 0                                     00:00:00 03/15/98
  4 VTBL alpha sda6 0                                     00:00:00 03/15/98
  5 VTBL Amanda Tape End                                  01:45:15 03/16/98
```

Note on datestamps: volume 0 (*Amanda* label): reflects the time of starting the backup volume i (backup files): *Amanda* datestamps of the backup files last volume (end marker) : reflects the time of finishing the backup

I tested this on a Linux machine (P90 / 96Mb RAM / 256 Mb swap) with two other clients (Linux / WfW) using

- Linux Kernel 2.0.33,
- ftape 3.04d,
- *Amanda* 2.4.0b6p4

with an internal Iomega Ditto 3200 (TR-3) drive attached to an Iomega Ditto Dash controller (at 2000 Kbps). My tapetype follows:

```
define tapetype DITTO-TR3 {
    comment "Iomega DITTO 3200 Travan 3 tape drives"
    length 1500 mbytes          #
    filemark 29 kbytes          # ???
    speed 256 kbytes            # = 2000 Kbit/s ?
}
```

NOTE

Filemarks are not written to the tape (they are written to the header segment and use there 128 byte), so their size could even be 0. But a tape segment takes at least 29 kb (+3 kb ecc-code = 32 kb), so in the worst case an eof will waste 29 kb.

NOTE

Refer to <http://www.amanda.org/docs/zftape.html> for the current version of this document.

Part VII

Appendixes

Chapter 36

THE *AMANDA* MANUAL PAGES.

This chapter contains the manual pages from the official *Amanda* distribution.

amadmin

Name

amadmin — administrative interface to control *Amanda* backups

Synopsis

`amadmin` *`config command`* `[`*`command_options...`*`]` `[-o`*`configoption`*`]...`

DESCRIPTION

Amadmin performs various administrative tasks on the *config Amanda* configuration.

See the amanda(8) man page for more details about *Amanda*.

COMMANDS

Commands that take a *hostname* [*disks*] parameter pair operate on all disks in the *disklist* for that *hostname* if no disks are specified. Where *hostname* is also marked as being optional, the command operates on all hosts and disks in the *disklist*. Both *hostname* and *disks* are special expressions; see the "HOST & DISK EXPRESSION" section of amanda(8) for a description.

Commands that take one or more *dumpspec* parameters operate on the set of dumps specified by all of the expressions. See the "DUMP SPECIFICATIONS" section of amanda(8) for a description.

version Show the current version and some compile time and runtime parameters. The *config* parameter must be present but is ignored.

force-bump [***hostname*** [***disks***]*]+ Force the *disks* on *hostname* to bump to a new incremental level during the next *Amanda* run.

force-no-bump [***hostname*** [***disks***]*]+ Force the *disks* on *hostname* to not bump to a new incremental level during the next *Amanda* run.

unforce-bump [***hostname*** [***disks***]*]+ Undo a previous *force-bump* or *force-no-bump* command.

force [***hostname*** [***disks***]*]+ Force the *disks* on *hostname* to do a full (level 0) backup during the next *Amanda* run.

unforce [***hostname*** [***disks***]*]+ Undo a previous *force* command.

reuse tapelabel [***...***] The tapes listed will be available for reuse at their point in the tape cycle.

no-reuse tapelabel [***...***] The tapes listed will not be reused when their turn comes up again in the tape cycle. Note that if this causes the number of reusable tapes to drop below the *amanda.conf tapecycle* value, *Amanda* will request new tapes until the count is satisfied again.

Tape marked *no-reuse* are available for recovery, marking them *no-reuse* is a security to be sure amanda will not overwrite them.

due [***hostname*** [***disks***]*]* Show when the next full dump is due.

find [`--sort` *`hkdlpb`*] [***hostname*** [***disks***]*]* Display all backups currently on tape or in the holding disk. The tape label or holding disk filename, file number, and status are displayed.

The `--sort` option changes the sort order using the following flags:

```
h: host name
k: disk name
d: dump date
l: backup level
p: dump part
b: tape label
```

An uppercase letter reverses the sort order for that key. The default sort order is *hkdlpb*.

holding delete hostname [***disk*** [***datestamp*** [**..**]]] Delete holding files matching the given specification. At least a hostname must be provided.

holding list [**-l**] [***hostname*** [***disk*** [***datestamp*** [**..**]]]] List holding files matching the given specification, or all holding files if no specification is provided. With '-l', additional information (size and level) is provided.

delete [***hostname*** [***disks***]*]+ Delete the specified *disks* on *hostname* from the *Amanda* database.

> NOTE
>
> 
>
> If you do not also remove the disk from the *disklist* file, *Amanda* will treat it as a new disk during the next run.

tape Display the tape(s) *Amanda* expects to write to during the next run. See also amcheck(8).

bumpsize Display the current bump threshold parameters, calculated for all backup levels.

balance [`--days` <*num*>] Display the distribution of full backups throughout the dump schedule.

export [***hostname*** [***disks***]*]* Convert records from the *Amanda* database to a text format that may be transmitted to another *Amanda* machine and *import*ed.

import Convert *export*ed records read from standard input to a form *Amanda* uses and insert them into the database on this machine.

disklist [***hostname*** [***disks***]*]* Display the *disklist* information for each of the *disks* on *hostname* (or all hosts). Mostly used for debugging.

info [***hostname*** [***disks***]*]* Display the database record for each of the *disks* on *hostname* (or all hosts). Mostly used for debugging.

-o *configoption* See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLES

Request three specific file systems on *machine-a* get a full level 0 backup during the next *Amanda* run.

```
$ amadmin daily force machine-a / /var /usr
amadmin: machine-a:/ is set to a forced level 0 tonight.
amadmin: machine-a:/var is set to a forced level 0 tonight.
amadmin: machine-a:/usr is set to a forced level 0 tonight.
```

Request all file systems on *machine-b* get a full level 0 backup during the next *Amanda* run.

```
$ amadmin daily force machine-b
amadmin: machine-b:/ is set to a forced level 0 tonight.
amadmin: machine-b:/var is set to a forced level 0 tonight.
amadmin: machine-b:/usr is set to a forced level 0 tonight.
amadmin: machine-b:/home is set to a forced level 0 tonight.
```

Undo the previous *force* request for `/home` on *machine-b*. The other file systems will still get a full level 0 backup.

```
$ amadmin daily unforce machine-b /home
amadmin: force command for machine-b:/home cleared.
```

Locate backup images of `/var` from *machine-c*. The *tape or file* column displays either a tape label or a filename depending on whether the image is on tape or is still in the holding disk. If the image is on tape, the *file* column tells you which file on the tape has the image (file number zero is a tape label). This column shows zero and is not meaningful if the image is still in the holding disk. The *status* column tells you whether the backup was successful or had some type of error.

```
$ amadmin daily find machine-c /var
date        host      disk lv tape or file                 file part  status
2000-11-09  machine-c /var  0 000110                       9   --  OK
2000-11-08  machine-c /var  2 000109                       2   --  OK
2000-11-07  machine-c /var  2 /amanda/20001107/machine-c._var.2  0 OK
2000-11-06  machine-c /var  2 000107                       2   --  OK
2000-11-05  machine-c /var  2 000106                       3   --  OK
2000-11-04  machine-c /var  2 000105                       2   --  OK
2000-11-03  machine-c /var  2 000104                       2   --  OK
2000-11-02  machine-c /var  2 000103                       2   --  OK
2000-11-01  machine-c /var  1 000102                       5   --  OK
2000-10-31  machine-c /var  1 000101                       3   --  OK
```

Forget about the `/workspace` disk on *machine-d*. If you do not also remove the disk from the *disklist* file, *Amanda* will treat it as a new disk during the next run.

```
$ amadmin daily delete machine-d /workspace
amadmin: machine-d:/workspace deleted from database.
amadmin: NOTE: you'll have to remove these from the disklist yourself.
```

Find the next tape *Amanda* will use (in this case, `123456`).

```
$ amadmin daily tape
The next Amanda run should go onto tape 123456 or a new tape.
```

Show how well full backups are balanced across the dump cycle. The *due-date* column is the day the backups are due for a full backup. *#fs* shows the number of filesystems doing full backups that night, and *orig KB* and *out KB* show the estimated total size of the backups before and after any compression, respectively.

The *balance* column shows how far off that night's backups are from the average size (shown at the bottom of the balance column). *Amanda* tries to keep the backups within +/- 5%, but since the amount of data on each filesystem is always changing, and *Amanda* will never delay backups just to rebalance the schedule, it is common for the schedule to fluctuate by larger percentages. In particular, in the case of a tape or backup failure, a bump will occur the following night, which will not be smoothed out until the next pass through the schedule.

The last line also shows an estimate of how many *Amanda* runs will be made between full backups for a file system. In the example, a file system will probably have a full backup done every eight times *Amanda* is run (e.g. every eight days).

```
$ amadmin daily balance
 due-date  #fs   orig KB    out KB  balance
----------------------------------------
11/10 Mon   21    930389    768753    +5.1%
11/11 Tue   29   1236272    733211    +0.2%
11/12 Wed   31   1552381    735796    +0.6%
11/13 Thu   23   1368447    684552    -6.4%
11/14 Fri   32   1065603    758155    +3.6%
11/15 Sat   14   1300535    738430    +0.9%
11/16 Sun   31   1362696    740365    +1.2%
11/17 Mon   30   1427936    773397    +5.7%
11/18 Tue   11   1059191    721786    -1.3%
11/19 Wed   19   1108737    661867    -9.5%
----------------------------------------
TOTAL      241  12412187   7316312   731631  (estimated 8 runs per dumpcycle)
```

FILES

/usr/local/etc/amanda/*config*/amanda.conf

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amcheck(8), amdump(8), amrestore(8), amfetchdump(8)

amaespipe

Name

amaespipe — wrapper program for aespipe

Synopsis

`amaespipe`

DESCRIPTION

amaespipe requires *aespipe*, *uuencode* and *gpg* to work. Aespipe is available from <http://loop-aes.sourceforge.net>

amaespipe will search for the aespipe program in the following directories: /usr/bin:/usr/local/bin:/sbin

amaespipe is called by **amcrypt** for *Amanda* data encryption.

amaespipe is based on aespipe's bzaespipe program. It calls aespipe to encrypt data using AES256 as the encryption and SHA256 as the hash function. GPG key should be stored in $AMANDA_HOME/.gnupg/am_key.gpg. **amaespipe** reads passphrase from file descriptor 3. During decryption, **amaespipe** autodects encryption type and hash function from the encrypted image.

SEE ALSO

amanda(8), amanda.conf(5), aespipe(1), amcrypt(8), gpg(1)

amanda

Name

amanda — Advanced Maryland Automatic Network Disk Archiver

Synopsis

amadmin *config command* [*options*]

amcheck [*options*] *config*

amcheckdb *config*

amcleanup *config*

amcrypt

amdd [*options*]

amdump *config*

amaespipe

amflush [-f] *config*

amgetconf [*config*] *parameter*

amlabel *config label* [*slot slot*]

ammt [*options*]

amoverview *config* [*options*]

amplot [*options*] *amdump-files*

amrecover [*config*] [*options*]

amreport [*config*] [*options*]

amrestore [*options*] *tapedevice* [*hostname* [*diskname*]]

amfetchdump [*options*] *config* [*hostname* [*diskname* [*date* [level]]]]

amrmtape [*options*] *config label*

amstatus *config* [*options*]

amtape *config command* [*options*]

amtapetype [*options*]

amtoc [*options*] *logfile*

amverify *config*

amverifyrun *config*

DESCRIPTION

Amanda is the "Advanced Maryland Automatic Network Disk Archiver". This manual page gives an overview of the *Amanda* commands and configuration files for quick reference.

Here are all the *Amanda* commands. Each one has its own manual page. See them for all the gory details.

amdump Take care of automatic *Amanda* backups. This is normally executed by *cron* on a computer called the *tape server host* and requests backups of file systems located on *backup clients*. *Amdump* backs up all disks in the *disklist* file (discussed below) to tape or, if there is a problem, to a special *holding disk*. After all backups are done, **amdump** sends mail reporting failures and successes.

amflush Flush backups from the holding disk to tape. *Amflush* is used after **amdump** has reported it could not write backups to tape for some reason. When this happens, backups stay in the holding disk. Run *amflush* after the tape problem is corrected to write backups from the holding disk to tape.

amcleanup Clean up after an interrupted **amdump**. This command is only needed if **amdump** was unable to complete for some reason, usually because the tape server host crashed while **amdump** was running.

amrecover Provides an interactive interface to browse the *Amanda* index files (backup image catalogues) and select which tapes to recover files from. It can also run *amrestore* and a restore program (e.g. *tar*) to actually recover the files.

amrestore Read an *Amanda* tape, searching for requested backups. *Amrestore* is suitable for everything from interactive restores of single files to a full restore of all partitions on a failed disk.

amfetchdump Performs *Amanda* tape restoration, similar to *amrestore*. Additional capabilities include "hands-off" searching of multiple tapes, automatic retrieval of specific dump files based on dump logs, and assembly of tape-spanning split dump files.

amlabel Write an *Amanda* format label onto a tape. All *Amanda* tapes must be labeled with *amlabel*. *Amdump* and *amflush* will not write to an unlabeled tape (see TAPE MANAGEMENT below).

amcheck Verify the correct tape is mounted and all file systems on all backup client systems are ready to be backed up. Often run by *cron* before **amdump** to generate a mail warning that backups might fail unless corrective action is taken.

amadmin Take care of administrative tasks like finding out which tapes are needed to

restore a filesystem, forcing hosts to do full backups of selected disks and looking at schedule balance information.

amtape Take care of tape changer control operations like loading particular tapes, ejecting tapes and scanning the tape storage slots.

amverify Check *Amanda* backup tapes for errors.

amrmtape Delete a tape from the *Amanda* databases.

amstatus Report the status of a running or completed **amdump**.

amoverview Display a chart of hosts and file systems backed up every run.

amplot Generate utilization plots of *Amanda* runs for performance tuning.

amreport Generate an *Amanda* summary E-mail report.

amtoc Generate table of content files for *Amanda* tapes.

amcheckdb Verify every tape *Amanda* knows about is consistent in the database.

amgetconf Look up parameters in the *Amanda* configuration file.

amtapetype Generate a tapetype definition.

amaespipe Wrapper program from aespipe (data encryption utility)

amcrypt Reference encryption program for Amanda symmetric data encryption

CONFIGURATION

There are three user-editable files that control the behavior of *Amanda*.

The first is *amanda.conf*, the main configuration file. It contains parameters to customize *Amanda* for the site. Refer to the amanda.conf(5), manpage for details on *Amanda* configuration parameters.

Second is the *disklist* file, which lists hosts and disk partitions to back up.

Third is the *tapelist* file, which lists tapes that are currently active. These files are described in more detail in the following sections.

All files are stored in individual configuration directories under `/usr/local/etc/amanda/` . A site will often have more than one configuration. For example, it might have a *normal* configuration for everyday backups and an *archive* configuration for infrequent full archival backups. The configuration files would be stored under directories `/usr/local/etc/amanda/normal/` and `/usr/local/etc/amanda/archive/`, respectively. Part of the job of an *Amanda* administrator is to create, populate and maintain these directories.

All log and database files generated by *Amanda* go in corresponding directories somewhere. The exact location is controlled by entries in *amanda.conf*. A typical location would be under `/var/adm/amanda`. For the above example, the files might go in `/var/adm/amanda/normal/` and `/var/adm/amanda/archive/`.

As log files are no longer needed (no longer contain relevant information), *Amanda* cycles them out in various ways, depending on the type of file.

Detailed information about **amdump** runs are stored in files named *amdump.NN* where *NN* is a sequence number, with 1 being the most recent file. *Amdump* rotates these files each run, keeping roughly the last *tapecycle* (see below) worth of them.

The file used by *amreport* to generate the mail summary is named *log.YYYYMMDD.NN* where *YYYYMMDD* is the datestamp of the start of the **amdump** run and *NN* is a sequence number started at 0. At the end of each **amdump** run, log files for runs whose tapes have been reused are renamed into a subdirectory of the main log directory (see the *logdir* parameter below) named *oldlog*. It is up to the *Amanda* administrator to remove them from this directory when desired.

Index (backup image catalogue) files older than the full dump matching the oldest backup image for a given client and disk are removed by **amdump** at the end of each run.

DISKLIST FILE

The *disklist* file determines which disks will be backed up by *Amanda*. The file usually contains one line per disk:

```
hostname diskname [diskdevice] dumptype [spindle [interface] ]
```

All pairs [*hostname diskname*] must be unique.

Lines starting with # are ignored, as are blank lines. The fields have the following meanings:

hostname The name of the host to be backed up. If *diskdevice* refers to a PC share, this is the host *Amanda* will run the Samba *smbclient* program on to back up the share.

diskname The name of the disk (a label). In most case, you set your *diskname* to the *diskdevice* and you don't set the *diskdevice*. If you want multiple entries with the same *diskdevice*, you must set a different *diskname* for each entry. It's the *diskname* that you use on the commandline for any *Amanda* command. Look at the example/disklist file for example.

diskdevice Default: same as diskname. The name of the disk device to be backed up. It may be a full device name, a device name without the `/dev/` prefix, e.g. *sd0a*, or a mount point such as `/usr`.

It may also refer to a PC share by starting the name with two (forward) slashes, e.g. `//some-pc/home`. In this case, the *program* option in the associated *dumptype* must be entered as *GNUTAR*. It is the combination of the double slash disk name and *program* *GNUTAR* in the *dumptype* that triggers the use of Samba.

dumptype Refers to a *dumptype* defined in the *amanda.conf* file. *Dumptype*s specify backup related parameters, such as whether to compress the backups, whether to record backup results in `/etc/dumpdates`, the disk's relative priority, etc.

spindle Default: `-1`. A number used to balance backup load on a host. *Amanda* will not run multiple backups at the same time on the same spindle, unless the spindle number is -1, which means there is no spindle restriction.

interface Default: *local*. The name of a network interface definition in the *amanda.conf* file, used to balance network load.

Instead of naming a *dumptype*, it is possible to define one in-line, enclosing *dumptype* options within curly braces, one per line, just like a *dumptype* definition in *amanda.conf*. Since pre-existing *dumptype*s are valid option names, this syntax may be used to customize *dumptype*s for particular disks.

A line break *must* follow the left curly bracket.

For instance, if a *dumptype* named *normal* is used for most disks, but use of the holding disk needs to be disabled for the file system that holds it, this would work instead of defining a new dumptype:

```
hostname diskname [ diskdevice ] {
  normal
  holdingdisk never
} [ spindle [ interface ] ]
```

TAPE MANAGEMENT

The *tapelist* file contains the list of tapes in active use. This file is maintained entirely by *Amanda* and should not be created or edited during normal operation. It contains lines of the form:

```
YYYYMMDD label flags
```

Where *YYYYMMDD* is the date the tape was written, *label* is a label for the tape as written by *amlabel* and *flags* tell *Amanda* whether the tape may be reused, etc (see the *reuse* options of *amadmin*).

Amdump and *amflush* will refuse to write to an unlabeled tape, or to a labeled tape that is considered active. There must be more tapes in active rotation (see the *tapecycle* option) than there are runs in the backup cycle (see the *dumpcycle* option) to prevent overwriting a backup image that would be needed to do a full recovery.

OUTPUT DRIVERS

The normal value for the *tapedev* parameter, or for what a tape changer returns, is a full path name to a non-rewinding tape device, such as `/dev/nst0` or `/dev/rmt/0mn` or `/dev/nst0.1` or whatever conventions the operating system uses. *Amanda* provides additional application level drivers that support non-traditional tape-simulations or features. To access a specific output driver, set *tapedev* (or configure your changer to return) a string of the form *driver*:*driver-info* where *driver* is one of the supported drivers and *driver-info* is optional additional information needed by the driver.

The supported drivers are:

tape This is the default driver. The *driver-info* is the tape device name. Entering

```
tapedev /dev/rmt/0mn
```

is really a short hand for

```
tapedev tape:/dev/rmt/0mn
```

null This driver throws away anything written to it and returns EOF for any reads except a special case is made for reading a label, in which case a "fake" value is returned that *Amanda* checks for and allows through regardless of what you have set in *labelstr*. The *driver-info* field is not used and may be left blank:

```
tapedev null:
```

The *length* value from the associated *tapetype* is used to limit the amount of data written. When the limit is reached, the driver will simulate end of tape.

NOTE

This driver should only be used for debugging and testing, and probably only with the *record* option set to *no*.

rait *R*edundant *A*rray of *I*nexpensive (?) *T*apes. Reads and writes tapes mounted on multiple drives by spreading the data across N-1 drives and using the last drive for a checksum. See docs/RAIT for more information.

The *driver-info* field describes the devices to use. Curly braces indicate multiple replacements in the string. For instance:

```
tapedev rait:/dev/rmt/tps0d{4,5,6}n
```

would use the following devices:

```
/dev/rmt/tps0d4n /dev/rmt/tps0d5n /dev/rmt/tps0d6n
```

file This driver emulates a tape device with a set of files in a directory. The *driver-info* field must be the name of an existing directory. The driver will test for a subdirectory of that named *data* and return *offline* until it is present. When present, the driver uses two files in the *data* subdirectory for each tape file. One contains the actual data. The other contains record length information.

The driver uses a file named *status* in the *file* device directory to hold driver status information, such as tape position. If not present, the driver will create it as though the device is rewound.

The *length* value from the associated *tapetype* is used to limit the amount of data written. When the limit is reached, the driver will simulate end of tape.

One way to use this driver with a real device such as a CD-writer is to create a directory for the *file* device and one or more other directories for the actual data. Create a symlink named *data* in the *file* directory to one of the data directories. Set the *tapetype* length to whatever the medium will hold.

When *Amanda* fills the *file* device, remove the symlink and (optionally) create a new symlink to another data area. Use a CD writer software package to burn the image from the first data area.

To read the CD, mount it and create the *data* symlink in the *file* device directory.

AUTHORIZATION

Amanda processes on the tape server host run as the *dumpuser* user listed in *amanda.conf*. When they connect to a backup client, they do so with an *Amanda*-specific protocol. They do not, for instance, use *rsh* or *ssh* directly.

On the client side, the *amandad* daemon validates the connection using one of several methods, depending on how it was compiled and on options it is passed:

.rhosts Even though *Amanda* does not use *rsh*, it can use .rhosts-style authentication and a .rhosts file.

.amandahosts This is essentially the same as .rhosts authentication except a different file, with almost the same format, is used. This is the default mechanism built into *Amanda*.

The format of the `.amandahosts` file is:

hostname [*username* [*service*]*]

If *username* is ommitted, it defaults to the user running *amandad*, i.e. the user listed in the *inetd* or *xinetd* configuration file.

The *service* is a list of the service the client is authorized to execute: *amdump*, *noop*, *selfcheck*, *sendsize*, *sendbackup*, *amindexd*, *amidxtaped*. *amdump* is a shortcut for "noop selfcheck sendsize sendbackup"

Kerberos *Amanda* may use the Kerberos authentication system. Further information is in the *docs/KERBEROS* file that comes with an *Amanda* distribution.

For Samba access, *Amanda* needs a file on the Samba server (which may or may not also be the tape server) named `/etc/amandapass` with share names, (clear text) passwords and (optional) domain names, in that order, one per line, whitespace separated. By default, the user used to connect to the PC is the same for all PC's and is compiled into *Amanda*. It may be changed on a host by host basis by listing it first in the password field followed by a percent sign and then the password. For instance:

```
//some-pc/home normalpw
//another-pc/disk otheruser%otherpw
```

With clear text passwords, this file should obviously be tightly protected. It only needs to be readable by the *Amanda*-user on the Samba server.

You can find further information in the *docs/SAMBA* file that comes with an *Amanda* distribution.

HOST & DISK EXPRESSION

All host and disk arguments to programs are special expressions. The command applies to all disks that match your arguments. This section describes the matcher.

The matcher matches by word, each word is a glob expression, words are separated by the separator '.' for host and '/' for disk. You can anchor the expression at left with a '^'. You can anchor the expression at right with a '$'. The matcher is case insensitive for host but is

case sensitive for disk. A match succeeds if all words in your expression match contiguous words in the host or disk.

.	word separator for a host
/	word separator for a disk
^	anchor at left
$	anchor at right
?	match exactly one character except the separator
*	match zero or more characters except the separator
**	match zero or more characters including the separator

Some examples:

EXPRESSION	WILL MATCH	WILL NOT MATCH
hosta	hosta	hostb
	hoSTA.dOMAIna.ORG	
	foo.hosta.org	
host	host	hosta
host?	hosta	host
	hostb	
ho*na	hoina	ho.aina.org
ho**na	hoina	
	ho.aina.org	
^hosta	hosta	foo.hosta.org
sda*	/dev/sda1	
	/dev/sda12	
/opt	opt (disk)	opt (host)
.opt.	opt (host)	opt (disk)
/	/	any other disk
/usr	/usr	
	/usr/opt	
/usr$	/usr	/usr/opt

DATESTAMP EXPRESSION

A *datestamp* expression is a range expression where we only match the prefix. Leading ^ is removed. Trailing $ forces an exact match.

20001212-14	match all dates beginning with 20001212, 20001213 or 20001214
20001212-4	same as previous
20001212-24	match all dates between 20001212 and 20001224
2000121	match all dates that start with 2000121 (20001210-20001219)
2	match all dates that start with 2 (20000101-29991231)
2000-10	match all dates between 20000101-20101231
200010$	match only 200010

DUMP SPECIFICATIONS

A dump specification selects one or more dumps. It has the form *[host][:disk][@datestamp]*, where each component is a pattern as described above. If a component is missing, it is treated as a wildcard. The characters ':', '@', and '\' may be escaped within any component by preceding them with a '\'.

Some examples:

DUMPSPEC	DESCRIPTION
client17	all dumps of client17
@20080615	All dumps on with datestamps matching 20080615
webserver:/var/www	All dumps of /var/www on host webserver
webserver:/var/www@200806150317	The dump of webserver with datestamp 200806150317
:/var/www	All dumps of /var/www on any host

CONFIGURATION OVERRIDE

Most commands allow the override of specific configuration options on the command line, using the -o option. This option has the form -o *name*=*value*. An optional space is allowed after the -o . Each configuration option should be specified in a separate command-line option.

For global options, *name* is simply the name of the option, e.g.,

```
amdump -oruntapes=2
```

For options in a named section of the configuration, *name* has the form *SECTION*:*section_name*:*name*, where *SECTION* is one of TAPETYPE, DUMPTYPE, HOLDINGDISK, or INTERFACE, and *section_name* is the name of the tapetype, dumptype, holdingdisk, or interface. Examples:

```
amdump -o TAPETYPE:HP-DAT:length=2000m
amdump -o DUMPTYPE:no-compress:compress="server fast"
amdump -o HOLDINGDISK:hd1:use="-100 mb"
amdump -o INTERFACE:local:use="2000 kbps"
```

Note that configuration overrides are not effective for tape changers, which supply a tapedev based on their own configuration. In order to override *tapedev*, you must also disable any changer:

```
amdump -otapedev=/dev/nst1 -otpchanger=''
```

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion, major update

SEE ALSO

amadmin(8), amanda.conf(5), amanda-client.conf(5), amcheck(8), amcheckdb(8), amcleanup(8), amdd(8), amdump(8), amfetchdump(8) amflush(8), amgetconf(8), amlabel(8), ammt(8), amoverview(8), amplot(8), amrecover(8), amreport(8), amrestore(8), amrmtape(8), amstatus(8), amtape(8), amtapetype(8), amtoc(8), amverify(8), amverifyrun(8)

amanda.conf

Name

amanda.conf — Main configuration file for *Amanda*, the Advanced Maryland Automatic Network Disk Archiver

DESCRIPTION

`amanda.conf` is the main configuration file for *Amanda*. This manpage lists the relevant sections and parameters of this file for quick reference.

The file *<CONFIG_DIR>/<config>/amanda.conf* is loaded.

PARAMETERS

There are a number of configuration parameters that control the behavior of the *Amanda* programs. All have default values, so you need not specify the parameter in *amanda.conf* if the default is suitable.

Lines starting with # are ignored, as are blank lines. Comments may be placed on a line with a directive by starting the comment with a #. The remainder of the line is ignored.

Keywords are case insensitive, i.e. *mailto* and *MailTo* are treated the same.

Integer arguments may have one of the following (case insensitive) suffixes, some of which have a multiplier effect:

POSSIBLE SUFFIXES

b byte bytes Some number of bytes.

bps Some number of bytes per second.

k kb kbyte kbytes kilobyte kilobytes Some number of kilobytes (bytes*1024).

kps kbps Some number of kilobytes per second (bytes*1024).

m mb meg mbyte mbytes megabyte megabytes Some number of megabytes (bytes*1024*1024).

mps mbps Some number of megabytes per second (bytes*1024*1024).

g gb gbyte gbytes gigabyte gigabytes Some number of gigabytes (bytes*1024*1024*1024).

tape tapes Some number of tapes.

day days Some number of days.

week weeks Some number of weeks (days*7).

NOTE

The value *inf* may be used in most places where an integer is expected to mean an infinite amount.

Boolean arguments may have any of the values *y*, *yes*, *t*, *true* or *on* to indicate a true state, or *n*, *no*, *f*, *false* or *off* to indicate a false state. If no argument is given, *true* is assumed.

PARAMETERS

org string Default: *daily*. A descriptive name for the configuration. This string appears in the Subject line of mail reports. Each *Amanda* configuration should have a different string to keep mail reports distinct.

mailto string Default: *operators*. A space separated list of recipients for mail reports.

dumpcycle int Default: *10 days*. The number of days in the backup cycle. Each disk will get a full backup at least this often. Setting this to zero tries to do a full backup each run.

> NOTE
>
> 
>
> This parameter may also be set in a specific *dumptype* (see below). This value sets the default for all *dumptype*s so must appear in *amanda.conf* before any *dumptype*s are defined.

runspercycle int Default: *same as dumpcycle*. The number of amdump runs in *dumpcycle* days. A value of 0 means the same value as *dumpcycle*. A value of -1 means guess the number of runs from the *tapelist* file, which is the number of tapes used in the last *dumpcycle* days / *runtapes*.

tapecycle int Default: *15 tapes*. Typically tapes are used by *Amanda* in an ordered rotation. The *tapecycle* parameter defines the size of that rotation. The number of tapes in rotation must be larger than the number of tapes required for a complete dump cycle (see the *dumpcycle* parameter).

This is calculated by multiplying the number of **amdump** runs per dump cycle (*runspercycle* parameter) times the number of tapes used per run (*runtapes* parameter). Typically two to four times this calculated number of tapes are in rotation. While *Amanda* is always willing to use a new tape in its rotation, it refuses to reuse a tape until at least '*tapecycle* -1' number of other tapes have been used.

It is considered good administrative practice to set the *tapecycle* parameter slightly lower than the actual number of tapes in rotation. This allows the administrator to more easily cope with damaged or misplaced tapes or schedule adjustments that call for slight adjustments in the rotation order.

usetimestamps bool Default: *No*. By default, Amanda can only track at most one run per calendar day. When this option is enabled, however, Amanda can track as many runs as you care to make.

WARNING: This option is not backward-compatible. Do not enable it if you intend to downgrade your server installation to Amanda community edition 2.5.0

label_new_tapes string Default: not set. When set, this directive will cause *Amanda* to automatically write an *Amanda* tape label to any blank tape she encounters. This option is DANGEROUS because when set, *Amanda* will ERASE any non-*Amanda* tapes you may have, and may also ERASE any near-failing tapes. Use with caution.

When using this directive, specify the template for new tape labels. The template should contain some number of contiguous '%' characters, which will be replaced with a

generated number. Be sure to specify enough '%' characters that you do not run out of tape labels. Example: label_*newtapes*"$DailySet1-Default : amanda.TheloginnameAmandausest$

***dumpuser stpinter* string** Printer to use when doing tape labels. See the *lbl-templ tapetype* option.

***tapedev* string** Default: `null:`. The path name of the non-rewinding tape device. Non-rewinding tape device names often have an 'n' in the name, e.g. `/dev/rmt/0mn`, however this is operating system specific and you should consult that documentation for detailed naming information.

If a tape changer is configured (see the *tpchanger* option), this option might not be used.

If the *null* output driver is selected (see the section OUTPUT DRIVERS in the amanda(8) manpage for more information), programs such as **amdump** will run normally but all images will be thrown away. This should only be used for debugging and testing, and probably only with the *record* option set to *no*.

***rawtapedev* string** Default: `null:`. The path name of the raw tape device. This is only used if *Amanda* is compiled for Linux machines with floppy tapes and is needed for QIC volume table operations.

***tpchanger* string** Default: *none*. The name of the tape changer. If a tape changer is not configured, this option is not used and should be commented out of the configuration file.

If a tape changer is configured, choose one of the changer scripts (e.g. *chg-scsi*) and enter that here.

***changerdev* string** Default: `/dev/null`. A tape changer configuration parameter. Usage depends on the particular changer defined with the *tpchanger* option.

***changerfile* string** Default: `/usr/adm/amanda/log/changer-status`. A tape changer configuration parameter. Usage depends on the particular changer defined with the *tpchanger* option.

***runtapes* int** Default: `1`. The maximum number of tapes used in a single run. If a tape changer is not configured, this option is not used and should be commented out of the configuration file.

If a tape changer is configured, this may be set larger than one to let *Amanda* write to more than one tape.

Note that this is an upper bound on the number of tapes, and *Amanda* may use less.

Also note that as of this release, *Amanda* does not support true tape overflow. When it reaches the end of one tape, the backup image *Amanda* was processing starts over again on the next tape.

***maxdumpsize* int** Default: *runtapes*tape_length*. Maximum number of bytes the planner will schedule for a run.

***taperalgo* [first|firstfit|largest|largestfit|smallest|last]** Default: *first*. The algorithm used to choose which dump image to send to the taper.

first First in, first out.

firstfit The first dump image that will fit on the current tape.

largest The largest dump image.

largestfit The largest dump image that will fit on the current tape.

smallest The smallest dump image.

last Last in, first out.

labelstr string Default: *.**. The tape label constraint regular expression. All tape labels generated (see amlabel(8)) and used by this configuration must match the regular expression. If multiple configurations are run from the same tape server host, it is helpful to set their labels to different strings (for example, "DAILY[0-9][0-9]*" vs. "ARCHIVE[0-9][0-9]*") to avoid overwriting each other's tapes.

tapetype string Default: *EXABYTE*. The type of tape drive associated with *tapedev* or *tpchanger*. This refers to one of the defined *tapetype*s in the config file (see below), which specify various tape parameters, like the *length*, *filemark* size, and *speed* of the tape media and device.

First character of a *tapetype* string must be an alphabetic character

***ctimeout* int** Default: *30 seconds*. Maximum amount of time that *amcheck* will wait for each client host.

***dtimeout* int** Default: *1800 seconds*. Amount of idle time per disk on a given client that a *dumper* running from within **amdump** will wait before it fails with a data timeout error.

***etimeout* int** Default: *300 seconds*. Amount of time per disk on a given client that the *planner* step of **amdump** will wait to get the dump size estimates. For instance, with the default of 300 seconds and four disks on client A, *planner* will wait up to 20 minutes for that machine. A negative value will be interpreted as a total amount of time to wait per client instead of per disk.

***connect_tries* int** Default: *3*. How many times the server will try a connection.

***req_tries* int** Default: *3*. How many times the server will resend a REQ packet if it doesn't get the ACK packet.

***netusage* int** Default: *8000 Kbps*. The maximum network bandwidth allocated to *Amanda*, in Kbytes per second. See also the *interface* section.

***inparallel* int** Default: `10`. The maximum number of backups that *Amanda* will attempt to run in parallel. *Amanda* will stay within the constraints of network bandwidth and holding disk space available, so it doesn't hurt to set this number a bit high. Some contention can occur with larger numbers of backups, but this effect is relatively small on most systems.

***displayunit* "k|m|g|t"** Default: `"k"`. The unit used to print many numbers, k=kilo, m=mega, g=giga, t=tera.

***dumporder* string** Default: *tttTTTTTTT*. The priority order of each dumper:

```
s: smallest size
S: largest size
t: smallest time
T: largest time
b: smallest bandwidth
B: largest bandwidth
```

***maxdumps* int** Default: `1`. The maximum number of backups from a single host that *Amanda* will attempt to run in parallel. See also the *inparallel* option.

Note that this parameter may also be set in a specific *dumptype* (see below). This value sets the default for all *dumptype*s so must appear in *amanda.conf* before any *dumptype*s are defined.

***bumpsize* int** Default: *10 Mbytes*. The minimum savings required to trigger an automatic bump from one incremental level to the next, expressed as size. If *Amanda* determines that the next higher backup level will be this much smaller than the current level, it will do the next level. The value of this parameter is used only if the parameter *bumppercent* is set to 0.

The global setting of this parameter can be overwritten inside of a dumptype-definition.

See also the options *bumppercent*, *bumpmult* and *bumpdays*.

***bumppercent* int** Default: *0 percent*. The minimum savings required to trigger an automatic

bump from one incremental level to the next, expressed as percentage of the current size of the DLE (size of current level 0). If *Amanda* determines that the next higher backup level will be this much smaller than the current level, it will do the next level.

If this parameter is set to 0, the value of the parameter *bumpsize* is used to trigger bumping.

The global setting of this parameter can be overwritten inside of a dumptype-definition.

See also the options *bumpsize*, *bumpmult* and *bumpdays*.

bumpmult float Default: `1.5`. The bump size multiplier. *Amanda* multiplies *bumpsize* by this factor for each level. This prevents active filesystems from bumping too much by making it harder to bump to the next level. For example, with the default *bumpsize* and *bumpmult* set to 2.0, the bump threshold will be 10 Mbytes for level one, 20 Mbytes for level two, 40 Mbytes for level three, and so on.

The global setting of this parameter can be overwritten inside of a dumptype-definition.

bumpdays int Default: *2 days*. To insure redundancy in the dumps, *Amanda* keeps filesystems at the same incremental level for at least *bumpdays* days, even if the other bump threshold criteria are met.

The global setting of this parameter can be overwritten inside of a dumptype-definition.

diskfile string Default: *disklist*. The file name for the *disklist* file holding client hosts, disks and other client dumping information.

infofile string Default: `/usr/adm/amanda/curinfo`. The file or directory name for the historical information database. If *Amanda* was configured to use DBM databases, this is the base file name for them. If it was configured to use text formated databases (the default), this is the base directory and within here will be a directory per client, then a directory per disk, then a text file of data.

logdir string Default: `/usr/adm/amanda`. The directory for the **amdump** and *log* files.

indexdir string Default `/usr/adm/amanda/index`. The directory where index files (backup image catalogues) are stored. Index files are only generated for filesystems whose *dumptype* has the *index* option enabled.

tapelist string Default: *tapelist*. The file name for the active *tapelist* file. *Amanda* maintains this file with information about the active set of tapes.

tapebufs int Default: `20`. The number of buffers used by the *taper* process run by **amdump** and *amflush* to hold data as it is read from the network or disk before it is written to tape. Each buffer is a little larger than 32 KBytes and is held in a shared memory region.

reserve number Default: `100`. The part of holding-disk space that should be reserved for incremental backups if no tape is available, expressed as a percentage of the available holding-disk space (0-100). By default, when there is no tape to write to, degraded mode (incremental) backups will be performed to the holding disk. If full backups should also be allowed in this case, the amount of holding disk space reserved for incrementals should be lowered.

autoflush bool Default: *off*. Whether an amdump run will flush the dumps from holding disk to tape.

amrecover_do_fsf bool Default: *on*. Amrecover will call amrestore with the -f flag for faster positioning of the tape.

amrecover_check_label bool Default: *on*. Amrecover will call amrestore with the -l flag to check the label.

amrecover_changer string Default: ". Amrecover will use the changer if you use 'settape <string>' and that string is the same as the amrecover_changer setting.

columnspec string Defines the width of columns *amreport* should use. *String* is a comma (',') separated list of triples. Each triple consists of three parts which are separated by a equal sign ('=') and a colon (':') (see the example). These three parts specify:

1. the name of the column, which may be:

```
Compress (compression ratio)
Disk (client disk name)
DumpRate (dump rate in KBytes/sec)
DumpTime (total dump time in hours:minutes)
HostName (client host name)
Level (dump level)
OrigKB (original image size in KBytes)
OutKB (output image size in KBytes)
TapeRate (tape writing rate in KBytes/sec)
TapeTime (total tape time in hours:minutes)
```

2. the amount of space to display before the column (used to get whitespace between columns).
3. the width of the column itself. If set to a negative value, the width will be calculated on demand to fit the largest entry in this column.

Here is an example:

```
columnspec "Disk=1:18,HostName=0:10,OutKB=1:7"
```

The above will display the disk information in 18 characters and put one space before it. The hostname column will be 10 characters wide with no space to the left. The output KBytes column is seven characters wide with one space before it.

includefile string Default: *none*. The name of an *Amanda* configuration file to include within the current file. Useful for sharing dumptypes, tapetypes and interface definitions among several configurations.

***debug_auth* int** Default: *0*. Debug level of the auth module

***debug_event* int** Default: *0*. Debug level of the event module

***debug_holding* int** Default: *0*. Debug level of the holdingdisk module

***debug_protocol* int** Default: *0*. Debug level of the protocol module

***debug_planner* int** Default: *0*. Debug level of the planner process

***debug_driver* int** Default: *0*. Debug level of the driver process

***debug_dumper* int** Default: *0*. Debug level of the dumper process

***debug_chunker* int** Default: *0*. Debug level of the chunker process

***debug_taper* int** Default: *0*. Debug level of the taper process

***reserved-udp-port* int,int** Default: –with-udpportrange or *512,1023*. Reserved udp port that will be used (bsd, bsdudp)

***reserved-tcp-port* int,int** Default: –with-low-tcpportrange or *512,1023*. Reserved tcp port that will be used (bsdtcp)

***unreserved-tcp-port* int,int** Default: –with-tcpportrange or *1025,65536*. Unreserved tcp port that will be used (bsd, bsdudp)

HOLDINGDISK SECTION

The *amanda.conf* file may define one or more holding disks used as buffers to hold backup images before they are written to tape. The syntax is:

```
holdingdisk name {
    holdingdisk-option holdingdisk-value
    {\verb ...}
}
```

Name is a logical name for this holding disk.

The options and values are:

comment string Default: *none*. A comment string describing this holding disk.

directory disk Default: `/dumps/amanda`. The path to this holding area.

use int Default: *0 Gb*. Amount of space that can be used in this holding disk area. If the value is zero, all available space on the file system is used. If the value is negative, *Amanda* will use all available space minus that value.

chunksize int Default: *1 Gb*. Holding disk chunk size. Dumps larger than the specified size will be stored in multiple holding disk files. The size of each chunk will not exceed the specified value. However, even though dump images are split in the holding disk, they are concatenated as they are written to tape, so each dump image still corresponds to a single continuous tape section.

If 0 is specified, *Amanda* will create holding disk chunks as large as ((INT_MAX/1024)-64) Kbytes.

Each holding disk chunk includes a 32 Kbyte header, so the minimum chunk size is 64 Kbytes (but that would be really silly).

Operating systems that are limited to a maximum file size of 2 Gbytes actually cannot handle files that large. They must be at least one byte less than 2 Gbytes. Since *Amanda* works with 32 Kbyte blocks, and to handle the final read at the end of the chunk, the chunk size should be at least 64 Kbytes (2 * 32 Kbytes) smaller than the maximum file size, e.g. 2047 Mbytes.

DUMPTYPE SECTION

The `amanda.conf` file may define multiple sets of backup options and refer to them by name from the `disklist` file. For instance, one set of options might be defined for file systems that can benefit from high compression, another set that does not compress well, another set for file systems that should always get a full backup and so on.

A set of backup options are entered in a *dumptype* section, which looks like this:

```
define dumptype name {
    dumptype-option dumptype-value
```

```
    {\verb ...}
}
```

Name is the name of this set of backup options. It is referenced from the `disklist` file.

Some of the options in a *dumptype* section are the same as those in the main part of `amanda.conf`. The main option value is used to set the default for all *dumptype* sections. For instance, setting *dumpcycle* to 50 in the main part of the config file causes all following *dumptype* sections to start with that value, but the value may be changed on a section by section basis. Changes to variables in the main part of the config file must be done before (earlier in the file) any *dumptype*s are defined.

The dumptype options and values are:

auth string Default: *bsd*. Type of authorization to perform between tape server and backup client hosts.

bsd, bsd authorization with udp initial connection and one tcp connection by data stream.

bsdtcp, bsd authorization but use only one tcp connection.

bsdudp, like bsd, but will use only one tcp connection for all data stream.

krb4 to use Kerberos-IV authorization.

krb5 to use Kerberos-V authorization.

rsh to use rsh authorization.

ssh to use OpenSSH authorization.

amandad_path string Default: *$libexec/amandad*. Specify the amandad path of the client, only use with rsh/ssh authentification.

client_username string Default: *CLIENT_LOGIN*. Specify the username to connect on the client, only use with rsh/ssh authentification.

***bumpsize* int** Default: *10 Mbytes*. The minimum savings required to trigger an automatic bump from one incremental level to the next, expressed as size. If *Amanda* determines that the next higher backup level will be this much smaller than the current level, it will do the next level. The value of this parameter is used only if the parameter *bumppercent* is set to 0.

See also the options *bumppercent*, *bumpmult* and *bumpdays*.

***bumppercent* int** Default: *0 percent*. The minimum savings required to trigger an automatic bump from one incremental level to the next, expressed as percentage of the current size of the DLE (size of current level 0). If *Amanda* determines that the next

higher backup level will be this much smaller than the current level, it will do the next level.

If this parameter is set to 0, the value of the parameter *bumpsize* is used to trigger bumping.

See also the options *bumpsize*, *bumpmult* and *bumpdays*.

bumpmult float Default: `1.5`. The bump size multiplier. *Amanda* multiplies *bumpsize* by this factor for each level. This prevents active filesystems from bumping too much by making it harder to bump to the next level. For example, with the default *bumpsize* and *bumpmult* set to 2.0, the bump threshold will be 10 Mbytes for level one, 20 Mbytes for level two, 40 Mbytes for level three, and so on.

bumpdays int Default: *2 days*. To insure redundancy in the dumps, *Amanda* keeps filesystems at the same incremental level for at least *bumpdays* days, even if the other bump threshold criteria are met.

comment string Default: *none*. A comment string describing this set of backup options.

comprate float [**,** ***float***] Default: `0.50`, `0.50`. The expected full and incremental compression factor for dumps. It is only used if *Amanda* does not have any history information on compression rates for a filesystem, so should not usually need to be set. However, it may be useful for the first time a very large filesystem that compresses very little is backed up.

compress [client|server] string Default: *client fast*. If *Amanda* does compression of the backup images, it can do so either on the backup client host before it crosses the network or on the tape server host as it goes from the network into the holding disk or to tape. Which place to do compression (if at all) depends on how well the dump image usually compresses, the speed and load on the client or server, network capacity, holding disk capacity, availability of tape hardware compression, etc.

For either type of compression, *Amanda* also allows the selection of three styles of compression. *Best* is the best compression available, often at the expense of CPU overhead. *Fast* is often not as good a compression as *best*, but usually less CPU overhead. Or to specify *Custom* to use your own compression method. (See dumptype custom-compress in example/amanda.conf for reference)

So the *compress* options line may be one of:

compress none

compress client fast

compress client best

compress client custom Specify *client_custom_compress* "PROG"

PROG must not contain white space and it must accept -d for uncompress.

compress server fast

compress server best

compress server custom Specify *server_custom_compress* "PROG"

PROG must not contain white space and it must accept -d for uncompress.

Note that some tape devices do compression and this option has nothing to do with whether that is used. If hardware compression is used (usually via a particular tape device name or *mt* option), *Amanda* (software) compression should be disabled.

dumpcycle int Default: *10 days*. The number of days in the backup cycle. Each disk using this set of options will get a full backup at least this of ten. Setting this to zero tries to do a full backup each run.

encrypt [none|client|server] Default: *none*. To encrypt backup images, it can do so either on the backup client host before it crosses the network or on the tape server host as it goes from the network into the holding disk or to tape.

So the *encrypt* options line may be one of:

encrypt none

encrypt client Specify client_encrypt "PROG"

PROG must not contain white space.

Specify client_decrypt_option "decryption-parameter" Default: "-d"

decryption-parameter must not contain white space.

(See dumptype server-encrypt-fast in example/amanda.conf for reference)

encrypt server Specify server_encrypt "PROG"

PROG must not contain white space.

Specify server_decrypt_option "decryption-parameter" Default: "-d"

decryption-parameter must not contain white space.

(See dumptype client-encrypt-nocomp in example/amanda.conf for reference)

Note that current logic assumes compression then encryption during backup(thus decrypt then uncompress during restore). So specifying client-encryption AND server-compression is not supported. *amcrypt* which is a wrapper of *aespipe* is provided as a reference symmetric encryption program.

estimate client|calcsize|server Default: *client*. Determine the way *Amanda* does it's estimate.

client Use the same program as the dumping program, this is the most accurate way to do estimates, but it can take a long time.

calcsize Use a faster program to do estimates, but the result is less accurate.

server Use only statistics from the previous run to give an estimate, it takes only a few seconds but the result is not accurate if your disk usage changes from day to day.

exclude [**list|file**][**[optional]**[**append**][***string***]+] Default: *file*. There are two exclude lists, *exclude file* and *exclude list*. With *exclude file* , the *string* is a **GNU-tar** exclude expression. With *exclude list* , the *string* is a file name on the client containing **GNU-tar** exclude expressions. The path to the specified exclude list file, if present (see description of 'optional' below), must be readable by the *Amanda* user.

All exclude expressions are concatenated in one file and passed to **GNU-tar** as an `--exclude-from` argument.

Exclude expressions must always be specified as relative to the head directory of the DLE.

With the *append* keyword, the *string* is appended to the current list, without it, the *string* overwrites the list.

If *optional* is specified for *exclude list*, then amcheck will not complain if the file doesn't exist or is not readable.

For *exclude list*, if the file name is relative, the disk name being backed up is prepended. So if this is entered:

```
exclude list ".amanda.excludes"
```

the actual file used would be `/var/.amanda.excludes` for a backup of `/var`, `/usr/local/.amanda.excludes` for a backup of `/usr/local`, and so on.

holdingdisk [**never|auto|required**] Default: *auto*. Whether a holding disk should be used for these backups or whether they should go directly to tape. If the holding disk is a portion of another file system that *Amanda* is backing up, that file system should

refer to a dumptype with *holdingdisk* set to *never* to avoid backing up the holding disk into itself.

***never*|no|false|off** Never use a holdingdisk, the dump will always go directly to tape. There will be no dump if you have a tape error.

***auto*|yes|true|on** Use the holding disk, unless there is a problem with the holding disk, the dump won't fit there or the medium doesn't require spooling (e.g., VFS device)

required Always dump to holdingdisk, never directly to tape. There will be no dump if it doesn't fit on holdingdisk

ignore boolean Default: *no*. Whether disks associated with this backup type should be backed up or not. This option is useful when the *disklist* file is shared among several configurations, some of which should not back up all the listed file systems.

include [**list**|**file**][[**optional**][**append**][***string***]+] Default: *file* ".". There are two include lists, *include file* and *include list.* With *include file* , the *string* is a glob expression. With *include list* , the *string* is a file name on the client containing glob expressions.

All include expressions are expanded by *Amanda*, concatenated in one file and passed to **GNU-tar** as a `--files-from` argument. They must start with "./" and contain no other "/".

Include expressions must always be specified as relative to the head directory of the DLE.

NOTE

For globbing to work at all, even the limited single level, the top level directory of the DLE must be readable by the *Amanda* user.

With the *append* keyword, the *string* is appended to the current list, without it, the *string* overwrites the list.

If *optional* is specified for *include list,* then amcheck will not complain if the file doesn't exist or is not readable.

For *include list*, If the file name is relative, the disk name being backed up is prepended.

index boolean Default: *no*. Whether an index (catalogue) of the backup should be generated and saved in *indexdir*. These catalogues are used by the *amrecover* utility.

kencrypt boolean Default: *no*. Whether the backup image should be encrypted by Kerberos as it is sent across the network from the backup client host to the tape server host.

maxdumps int Default: `1`. The maximum number of backups from a single host that *Amanda* will attempt to run in parallel. See also the main section parameter *inparallel*.

maxpromoteday int Default: `10000`. The maximum number of day for a promotion, set it 0 if you don't want promotion, set it to 1 or 2 if your disks get overpromoted.

priority string Default: *medium*. When there is no tape to write to, *Amanda* will do incremental backups in priority order to the holding disk. The priority may be high (2), medium (1), low (0) or a number of your choice.

program string Default: *DUMP*. The type of backup to perform. Valid values are *DUMP* for the native operating system backup program, and *GNUTAR* to use **GNU-tar** or to do PC backups using Samba.

record boolean Default: *yes*. Whether to ask the backup program to update its database (e.g. `/etc/dumpdates` for DUMP or `/usr/local/var/amanda/gnutar-lists` for GNUTAR) of time stamps. This is normally enabled for daily backups and turned off for periodic archival runs.

skip-full boolean Default: *no*. If *true* and *planner* has scheduled a full backup, these disks will be skipped, and full backups should be run off-line on these days. It was reported that *Amanda* only schedules level 1 incrementals in this configuration; this is probably a bug.

skip-incr boolean Default: *no*. If *true* and *planner* has scheduled an incremental backup, these disks will be skipped.

starttime int Default: *none*. Backups will not start until after this time of day. The value should be hh*100+mm, e.g. 6:30PM (18:30) would be entered as `1830`.

strategy string Default: *standard*. Strategy to use when planning what level of backup to run next. Values are:

standard The standard *Amanda* schedule.

nofull Never do full backups, only level 1 incrementals.

noinc Never do incremental backups, only full dumps.

skip Never do backups (useful when sharing the *disklist* file).

incronly Only do incremental dumps. **amadmin force** should be used to tell *Amanda* that a full dump has been performed off-line, so that it resets to level 1. It is similar to skip-full, but with incronly full dumps may be scheduled manually. Unfortunately, it appears that *Amanda* will perform full backups with this configuration, which is probably a bug.

tape_splitsize int Default: *none*. Split dump file on tape into pieces of a specified size. This allows dumps to be spread across multiple tapes, and can potentially make more efficient use of tape space. Note that if this value is too large (more than half the size of the average dump being split), substantial tape space can be wasted. If too small, large dumps will be split into innumerable tiny dumpfiles, adding to restoration complexity. A good rule of thumb, usually, is 1/10 of the size of your tape.

split_diskbuffer string Default: *none*. When dumping a split dump in PORT-WRITE mode (usually meaning "no holding disk"), buffer the split chunks to a file in the directory specified by this option.

fallback_splitsize int Default: *10M*. When dumping a split dump in PORT-WRITE mode, if no split_diskbuffer is specified (or if we somehow fail to use our split_diskbuffer), we must buffer split chunks in memory. This specifies the maximum size split chunks can be in this scenario, and thus the maximum amount of memory consumed for in-memory splitting. The size of this buffer can be changed from its (very conservative) default to a value reflecting the amount of memory that each taper process on the dump server may reasonably consume.

The following *dumptype* entries are predefined by *Amanda*:

```
define dumptype no-compress {
    compress none
}
define dumptype compress-fast {
    compress client fast
}
define dumptype compress-best {
    compress client best
}
define dumptype srvcompress {
    compress server fast
}
```

```
define dumptype bsd-auth {
    auth bsd
}
define dumptype krb4-auth {
    auth krb4
}
define dumptype no-record {
    record no
}
define dumptype no-hold {
    holdingdisk no
}
define dumptype no-full {
    skip-full yes
}
```

In addition to options in a *dumptype* section, one or more other *dumptype* names may be entered, which make this *dumptype* inherit options from other previously defined *dumptype*s. For instance, two sections might be the same except for the *record* option:

```
define dumptype normal {
    comment "Normal backup, no compression, do indexing"
    no-compress
    index yes
    maxdumps 2
}
define dumptype testing {
    comment "Test backup, no compression, do indexing, no recording"
    normal
    record no
}
```

Amanda provides a *dumptype* named *global* in the sample *amanda.conf* file that all *dumptype*s should reference. This provides an easy place to make changes that will affect every *dumptype*.

TAPETYPE SECTION

The *amanda.conf* file may define multiple types of tape media and devices. The information is entered in a *tapetype* section, which looks like this in the config file:

```
define tapetype name {
    tapetype-option tapetype-value
    {\verb ...}
}
```

Name is the name of this type of tape medium/device. It is referenced from the *tapetype* option in the main part of the config file.

The tapetype options and values are:

comment string Default: *none*. A comment string describing this set of tape information.

filemark int Default: *1 kbytes*. How large a file mark (tape mark) is, measured in kbytes. If the size is only known in some linear measurement (e.g. inches), convert it to kbytes using the device density.

length int Default: *2000 kbytes*. How much data will fit on a tape.

Note that this value is only used by *Amanda* to schedule which backups will be run. Once the backups start, *Amanda* will continue to write to a tape until it gets an error, regardless of what value is entered for *length* (but see the section OUTPUT DRIVERS in the amanda(8) manpage for exceptions).

blocksize int Default: *32 kbytes*. How much data will be written in each tape record expressed in KiloBytes. The tape record size (= blocksize) can not be reduced below the default 32 KBytes. The parameter blocksize can only be raised if *Amanda* was compiled with the configure option –with-maxtapeblocksize=N set with "N" greater than 32 during **configure**.

readblocksize int Default: (*from configure –with-maxtapeblocksize*). How much data will be read in each tape record expressed in KiloBytes. Some hardware require a value not too large, and some require it to be equal to the blocksize. It is useful if you configured amanda with a big –with-maxtapeblocksize and your hardware don't work with a value that big.

file-pad boolean Default: *true*. If true, every record, including the last one in the file, will have the same length. This matches the way *Amanda* wrote tapes prior to the availability of this parameter. It may also be useful on devices that only support a fixed blocksize.

Note that the last record on the tape probably includes trailing null byte padding, which will be passed back to *gzip*, *compress* or the restore program. Most programs just ignore this (although possibly with a warning).

If this parameter is false, the last record in a file may be shorter than the block size. The file will contain the same amount of data the dump program generated, without trailing null byte padding. When read, the same amount of data that was written will be returned.

speed int Default: *200 bps*. How fast the drive will accept data, in bytes per second. This parameter is NOT currently used by *Amanda*.

lbl-templ string A PostScript template file used by *amreport* to generate labels. Several sample files are provided with the *Amanda* sources in the *example* directory. See the amreport(8) man page for more information.

In addition to options, another *tapetype* name may be entered, which makes this *tapetype* inherit options from another *tapetype*. For instance, the only difference between a DLT4000 tape drive using Compact-III tapes and one using Compact-IV tapes is the length of the tape. So they could be entered as:

```
define tapetype DLT4000-III {
    comment "DLT4000 tape drives with Compact-III tapes"
    length 12500 mbytes         # 10 Gig tapes with some compression
    filemark 2000 kbytes
    speed 1536 kps
}
define tapetype DLT4000-IV {
    DLT4000-III
    comment "DLT4000 tape drives with Compact-IV tapes"
    length 25000 mbytes         # 20 Gig tapes with some compression
}
```

INTERFACE SECTION

The *amanda.conf* file may define multiple types of network interfaces. The information is entered in an *interface* section, which looks like this:

```
define interface name {
    interface-option interface-value
    {\verb ...}
}
```

name is the name of this type of network interface. It is referenced from the *disklist* file.

Note that these sections define network interface characteristics, not the actual interface that will be used. Nor do they impose limits on the bandwidth that will actually be taken up by *Amanda*. *Amanda* computes the estimated bandwidth each file system backup will take based on the estimated size and time, then compares that plus any other running backups with the limit as another of the criteria when deciding whether to start the backup. Once a backup starts, *Amanda* will use as much of the network as it can leaving throttling up to the operating system and network hardware.

The interface options and values are:

comment string Default: *none*. A comment string describing this set of network information.

use int Default: *8000 Kbps*. The speed of the interface in Kbytes per second.

In addition to options, another *interface* name may be entered, which makes this *interface* inherit options from another *interface*. At the moment, this is of little use.

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org>: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion, major update, splitting

SEE ALSO

amanda(8), amanda-client.conf(5), amcrypt(8), aespipe(1),

amanda-client.conf

Name

amanda-client.conf — Client configuration file for *Amanda*, the Advanced Maryland Automatic Network Disk Archiver

DESCRIPTION

`amanda-client.conf` is the client configuration file for *Amanda*. This manpage lists the relevant sections and parameters of this file for quick reference.

The files *<CONFIG_DIR>/amanda-client.conf* and *<CONFIG_DIR>/<config>/amanda-client.conf* are loaded.

PARAMETERS

There are a number of configuration parameters that control the behavior of the *Amanda* programs. All have default values, so you need not specify the parameter in *amanda-client.conf* if the default is suitable.

Lines starting with # are ignored, as are blank lines. Comments may be placed on a line with a directive by starting the comment with a #. The remainder of the line is ignored.

Keywords are case insensitive, i.e. *auth* and *Auth* are treated the same.

Integer arguments may have one of the following (case insensitive) suffixes, some of which have a multiplier effect:

POSSIBLE SUFFIXES

b byte bytes Some number of bytes.

bps Some number of bytes per second.

k kb kbyte kbytes kilobyte kilobytes Some number of kilobytes (bytes*1024).

kps kbps Some number of kilobytes per second (bytes*1024).

m mb meg mbyte mbytes megabyte megabytes Some number of megabytes (bytes*1024*1024).

mps mbps Some number of megabytes per second (bytes*1024*1024).

g gb gbyte gbytes gigabyte gigabytes Some number of gigabytes (bytes*1024*1024*1024).

tape tapes Some number of tapes.

day days Some number of days.

week weeks Some number of weeks (days*7).

NOTE

The value *inf* may be used in most places where an integer is expected to mean an infinite amount.

Boolean arguments may have any of the values *y*, *yes*, *t*, *true* or *on* to indicate a true state, or *n*, *no*, *f*, *false* or *off* to indicate a false state. If no argument is given, *true* is assumed.

PARAMETERS

conf string Default: *Set by configure.* The conf use by amrecover.

index_server string Default: *Set by configure.* The amindexd server amrecover will connect to.

tape_server string Default: *Set by configure*. The amidxtaped server amrecover will connect to.

tapedev string Default: *Set by configure*. The tapedev amrecover will use.

auth string Default: *bsd*. Type of authorization to perform between tape server and backup client hosts.

bsd, bsd authorization with udp initial connection and one tcp connection by data stream.

bsdtcp, bsd authorization but use only one tcp connection.

bsdudp, like bsd, but will use only one tcp connection for all data stream.

krb4 to use Kerberos-IV authorization.

krb5 to use Kerberos-V authorization.

rsh to use rsh authorization.

ssh to use OpenSSH authorization.

ssh_keys string Default: *No default*. The key file the ssh auth will use, it must be the private key. If this parameter is not specified, then the deafult ssh key will be used.

gnutar_list_dir string Default from configure *–with-gnutar-listdir=DIR*. The directory where gnutar keep its state file.

amandates string Default: */etc/amandates*. The file where amanda keep the last date of each dumplevel.

***connect_tries* int** Default: *3*. How many times the server will try a connection.

***rep_tries* int** Default: *5*. How many times amandad will resend a REP packet if it doesn't get the ACK packet.

***debug_amandad* int** Default: *0*. Debug level of the amandad process

***debug_amidxtaped* int** Default: *0*. Debug level of the amidxtaped process

***debug_amindexd* int** Default: *0*. Debug level of the amindexd process

***debug_amrecover* int** Default: *0*. Debug level of the amrecover process

debug_auth **int** Default: *0*. Debug level of the auth module

debug_event **int** Default: *0*. Debug level of the event module

debug_holding **int** Default: *0*. Debug level of the holdingdisk module

debug_protocol **int** Default: *0*. Debug level of the protocol module

debug_selfcheck **int** Default: *0*. Debug level of the selfcheck process

debug_sendsize **int** Default: *0*. Debug level of the sendsize process

debug_sendbackup **int** Default: *0*. Debug level of the sendbackup process

reserved-udp-port **int,int** Default: –with-udpportrange or *512,1023*. Reserved udp port that will be used (amrecover with bsd or bsdudp)

reserved-tcp-port **int,int** Default: –with-low-tcpportrange or *512,1023*. Reserved tcp port that will be used (amrecover with bsdtcp)

unreserved-tcp-port **int,int** Default: –with-tcpportrange or *1025,65536*. Unreserved tcp port that will be used (bsd, bsdudp)

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org>: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion, major update, splitting

SEE ALSO

amanda(8), amanda.conf(5), amcrypt(8), aespipe(1),

amcheck

Name

amcheck — run *Amanda* self-checks

Synopsis

amcheck [-am] [-w] [-sclt] [-M *address*] *config* [*host* [*disk*...]...]
[-o*configoption*]...

DESCRIPTION

Amcheck runs a number of self-checks on both the *Amanda* tape server host and the *Amanda* client hosts.

On the tape server host, **amcheck** can go through the same tape checking used at the start of the nightly *amdump* run to verify the correct tape for the next run is mounted.

Amcheck can also do a self-check on all client hosts to make sure each host is running and that permissions on filesystems to be backed up are correct.

You can specify many host/disk expressions, only disks that match an expression will be checked. All disks are checked if no expressions are given.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

-s Run the tape server local and tape checks (same as -lt).

-c Run the client host checks. Multiple specific clients can be checked by specifying the client name.

-l Run the local tests (e.g. permissions) on the server host.

-t Run the tape tests on the server host.

-w Enables a DESTRUCTIVE check for write-protection on the tape (which would otherwise cause the subsequent *amdump* to fail). If the tape is writable, this check causes all data after the tape label to be erased. If the label_new_tapes option is enabled, this check may ERASE any non-Amanda tape in the drive or changer. The check enable the tape tests on the server host and is only made if the tape is otherwise correct.

-m Nothing is printed, but mail is sent if any errors are detected. The mail goes to the *mailto* address specified in the *amanda.conf* file or the *address* value if -M is set.

-a Like -m but the mail is always sent.

-M *address* Mail the report to *address* instead of the *mailto* value from *amanda.conf*. Implies -m.

host [***disk***]* Specify the host and disk on which the command will work.

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

The default is `-cs`.

EXAMPLES

In this example, both the tape server and client tests are run. The results are displayed on standard output.

```
% amcheck daily
Amanda Tape Server Host Check
-----------------------------
/amanda2/amanda/work: 911475 KB disk space available, that's plenty.
NOTE: skipping tape-writable test.
Tape VOL10 label ok.
Server check took 34.966 seconds.

Amanda Backup Client Hosts Check
--------------------------------
WARNING: northstar: selfcheck request timed out.  Host down?
WARNING: drinkme: selfcheck request timed out.  Host down?
WARNING: scruffy: selfcheck request timed out.  Host down?
Client check: 136 hosts checked in 51.945 seconds, 3 problems found.

(brought to you by Amanda 2.5.0)
```

In this example, if the line *mailto csd-amanda* is in *amanda.conf*, mail will be sent to *csd-amanda* if the server check returns an error.

```
% amcheck -s -m daily
```

MESSAGES

fatal slot *slot*: *error message* (error) The tape changer detected some kind of fatal error while trying to load slot *slot*.

slot *slot*: *error message* (warning) The tape changer detected some kind of non-fatal error (e.g. an empty slot was detected) while trying to load slot *slot*, or an error was detected trying to read the tape label.

slot *slot*: date *YYYYMMDD* label *label* (*result*) (info) Tape *label* in slot *slot* was loaded and found to have been last written on *YYYYMMDD*. If the tape is new, the date field will be an *X*. The *result* may be one of:

exact label match This is the expected tape.

no match This label does not match the *labelstr* pattern in *amanda.conf*. Tape scanning will continue.

active tape This tape is still active and cannot be overwritten. Tape scanning will continue.

first labelstr match This tape is the first one that matches the *labelstr* pattern in *amanda.conf*. Tape scanning will continue if necessary.

labelstr match This tape is the next one that matches the *labelstr* pattern in *amanda.conf*. Tape scanning will continue.

ERROR: cannot look up dump user *user* (error) Dump user *user* from *amanda.conf* could not be found in the system password information.

ERROR: cannot look up my own uid (*uid*) (error) User id *uid* running **amcheck** could not be found in the system password information.

ERROR: running as user *runuser* instead of *dumpuser* (error) *Amcheck* should be run as the dump user *dumpuser* from *amanda.conf* instead of *runuser*.

ERROR: program dir *directory*: not accessible (error) The directory *Amanda* expects to find its auxiliary programs in, *directory*, is not accessible.

ERROR: program *program*: does not exist (error) Program *program* needed on the tape server could not be found.

ERROR: program *program*: not a file (error) Program *program* needed on the tape server exists but is not a file.

ERROR: program *program*: not executable (error) Program *program* needed on the tape server exists but is not executable.

WARNING: program *program*: not setuid-root (warning) Program *program* needed on the tape server exists but should be owned by user "root" and setuid.

ERROR: *XXX* dir *directory*: not writable (error) Directory *directory* is either not writable, i.e. the dump user will not be able to create or remove files, or cannot be accessed, perhaps because a parent directory does not allow search permission. The *XXX* may be:

log for the *Amanda* log directory (see *logdir* in *amanda.conf*)

oldlog for the directory that holds the old log files (see *logdir* in *amanda.conf*)

info for an *Amanda* database information directory (see *curinfo* in *amanda.conf*) or

index for an *Amanda* index directory (see *indexdir* in *amanda.conf*)

tapelist for the *Amanda* tapelist directory (see *tapelist* in *amanda.conf*)

NOTE: *XXX* dir *directory*: does not exist (info) A database (info) or index directory does not exist or cannot be accessed. This might just mean this is a new client or disk, but if that is not the case, this should be treated as an error.

NOTE: it will be created on the next run (info) This indicates the info directory listed in the previous message will be created on the next run.

ERROR: *XXX* dir *name*: not a directory (error) *Amcheck* expected *name* to be a directory, but it is something else (e.g. file).

WARNING: info file `file`: does not exist (warning) File *file* does not exist in the text format database. Since the parent directories do exist, the file should already have been created.

ERROR: info file `name`: not a file (error) *Amcheck* expected *name* to be a file, but it is something else (e.g. file).

ERROR: info file `file`: not readable (error) The text format database file *file* is not readable.

ERROR: log file `file`: not writable (error) Log file *file* (file *log* in *logdir* from *amanda.conf*) is either not writable, or cannot be accessed, perhaps because a parent directory does not allow search permission.

ERROR: tape list *tapelist*: not writable (error) *Amanda* tape list file *tapelist* (see *tapelist* in *amanda.conf*) is not writable or was not found.

ERROR: tape list *tapelist*: parse error (error) *Amanda* tape list file *tapelist* (see *tapelist* in *amanda.conf*) could not be read or parsed.

WARNING: tapedev is /dev/null, dumps will be thrown away (warning) The *tapedev* parameter in *amanda.conf* is set to `/dev/null` and *Amanda* uses that when debugging to throw all the dump images away.

WARNING: hold file `file` exists (info) Hold file *file* exists and will cause *amdump* to pause at the beginning until it is removed.

ERROR: holding disk *disk*: statfs: *error message* (error) An error was returned from the *statfs* system call on holding disk *disk* (maybe because it does not exist).

ERROR: holding disk *disk*: not writable (error) Holding disk *disk*, is not writable, probably because the caller does not have write permission or a parent directory does not allow search permission.

WARNING: holding disk *disk*: available space unknown *N* KB requested. (warning) *Amcheck* could not determine the amount of available space on holding disk *disk* to see if there were at least *N* KBytes available.

WARNING: holding disk *disk*: only *F* KB free (*R* KB requested). (warning) *amanda.conf* requested *R* KBytes of free space on holding disk *disk*, but only *F* KBytes were available. 10 MBytes is subtracted for each backup process (see the *inparallel amanda.conf* option) to allow for unexpected overruns.

NOTE

Even though this message is listed as a warning, it causes **amcheck** to exit with a non-zero status.

Holding disk *disk*: *N* KB disk space available, that's plenty. (info) There was sufficient free space on holding disk *disk*.

WARNING: holding disk *disk*: only *F* KB free, using nothing (warning) Holding

disk *disk* has *F* KBytes of free space, but that is not enough for what is requested in *amanda.conf*.

Holding disk *disk*: *F* KB disk space available, using *U* KB (info) Holding disk *disk* has *F* KBytes of free space and *Amanda* will be using up to *U* Kbytes.

WARNING: if a tape changer is not available, runtapes must be set to 1. (warning) The *runtapes amanda.conf* option must be set to 1 if the *tpchanger amanda.conf* option is not set.

ERROR: *error message*. (error) An error was detected while initializing the tape changer.

ERROR: *tape device*: *error message*. (error) An error was detected while processing the tape label.

ERROR: cannot overwrite active tape *label*. (error) Tape *label* is still active and cannot be used.

ERROR: label *label* doesn't match labelstr *pattern* . (error) The label on tape *label* does not match the *labelstr amanda.conf* option.

(expecting a new tape) (info) The tape is not OK and a new tape was expected.

(expecting tape *label* or a new tape) (info) The tape is not OK and either tape *label* or a new tape was expected.

ERROR: tape *label* label ok, but is not writable. (error) Tape *label* is OK, but the write enable test failed.

Tape *label* is writable. (info) Tape *label* is OK and the write enable test succeeded.

NOTE: skipping tape-writable test. (info) The tape write test (see the `-w` option) was not enabled.

WARNING: skipping tape test because amdump or amflush seem to be running, WARNI (warning) It looked to **amcheck** like either *amdump* or *amflush* were running because a log file or amdump file exists. If they are not running, you probably need to run *amcleanup* to clear up a previous failure. Otherwise, you need to wait until they complete before running **amcheck**.

NOTE: skipping tape checks (info) The tape tests are being skipped because you used the `-t` command line option.

WARNING: *compress* is not executable, server-compression and indexing will not work (warning) Compression program *compress* is not executable, so compression on the tape server host and creating index files will not work.

Tape *label* label ok. (info) Tape *label* is OK for the next run.

Server check took *S* seconds. (info) Reports how long the tape server host checks took.

ERROR: *host*: could not resolve hostname (error) Could not look up client hostname *host*.

Client check: *H* hosts checked in *S* seconds, *N* problems found. (info) Reports the number of client hosts checked, how long it took and the number of errors detected.

WARNING: *host*: selfcheck request timed out. Host down? (warning) There was no response from *host* when trying to do the client checks. The host might really be down or it might not be configured properly.

ERROR: *host* NAK: *message* (error) *Host* reported a negative acknowledgment error of *message* to the status check request.

ERROR: *host* NAK: [NAK parse failed] (error) *Amcheck* could not parse the negative acknowledgment error from *host*. There might be an *Amanda* version mismatch between the host running **amcheck** and *host*.

ERROR: *host* [mutual-authentication failed] (error) Kerberos authentication failed while contacting *host*.

ERROR: *host*: *message* (error) Error *message* was reported by the status check on *host*.

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amdump(8)

amcheckdb

Name

amcheckdb — check *Amanda* database for tape consistency

Synopsis

`amcheckdb` *`config`*

DESCRIPTION

Amcheckdb verifies that every tape mentioned in the *Amanda* database is still valid in the *tapelist* file.

See the amanda(8) man page for more details about *Amanda*.

EXAMPLE

This shows a normal response:

```
# amcheckdb daily
Ready.
```

This shows tape *DMP014* is still listed in the database but is no longer listed in the *tapelist* file:

```
# amcheckdb daily
Tape DMP014 missing in /usr/local/etc/amanda//daily/tapelist
Ready.
```

AUTHOR

Adrian T. Filipi-Martin <atf3r@cs.virginia.edu>: Original text

Stefan G. Weichinger, sgw@amanda.org <`mailto:sgw@amanda.org`>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amadmin(8), amrmtape(8), amanda(8)

amcleanup

Name

amcleanup — run the *Amanda* cleanup process after a failure

Synopsis

`amcleanup` *`config`*

DESCRIPTION

Amcleanup generates the Amanda *Mail Report* and updates the *Amanda* databases after a system failure on a tape server host. This cleanup process is normally done automatically as part of the *amdump* program, but if *amdump* cannot complete for some reason (usually because of a tape server host crash), **amcleanup** must be run some time later (usually during system boot).

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

`-k` Kill all Amanda processes.

EXAMPLES

This example runs the *Amanda* cleanup process by hand after a failure.

```
% amcleanup daily
```

Putting the following line in a system boot script (e.g. `/etc/rc.local`) runs the *Amanda* cleanup process as part of the reboot, eliminating the need to run it by hand.

```
/usr/local/sbin/amcleanup daily
```

If nothing needs to be done, **amcleanup** exits normally with the message:

```
amcleanup: no unprocessed logfile to clean up.
```

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org>: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amdump(8)

amcrypt

Name

amcrypt — reference crypt program for *Amanda* symmetric data encryption

Synopsis

`amcrypt`

DESCRIPTION

amcrypt requires *aespipe*, *uuencode* and *gpg* to work. Aespipe is available from <http://loop-aes.sourceforge.net>

amcrypt will search for the aespipe program in the following directories: /usr/bin:/usr/local/bin:/sbin:/

amcrypt calls **amaespipe** and pass the *passphrase* through file descriptor 3. The passphrase should be stored in ~amanda/.am_passphrase.

How to create encryption keys for amcrypt

1. Create 65 random encryption keys and encrypt those keys using gpg. Reading from /dev/random may take indefinitely long if kernel's random entropy pool is empty. If that happens, do some other work on some other console (use keyboard, mouse and disks).

head -c 2925 /dev/random | uuencode -m - | head -n 66 | tail -n 65 \| gpg –symmetric -a > ~amanda/.gnupg/am_key.gpg

This will ask for a passphrase. Remember this passphrase as you will need it in the next step.

2. Store the passphrase inside the home-directory of the AMANDA-user and protect it with proper permissions:

```
echo my_secret_passphrase > ~amanda/.am_passphrase
chown amanda:disk ~amanda/.am_passphrase
chmod 700 ~amanda/.am_passphrase
```

Key and Passphrase

amcrypt uses the same key to encrypt and decrypt data.

It is very important to store and protect the key and the passphrase properly. Encrypted backup data can *only* be recovered with the correct key and passphrase.

SEE ALSO

amanda(8), amanda.conf(5), aespipe(1), amaespipe(8), gpg(1)

amcrypt-ossl

Name

amcrypt-ossl — crypt program for *Amanda* symmetric data encryption using OpenSSL

Synopsis

```
amcrypt-ossl [-d]
```

DESCRIPTION

amcrypt-ossl uses *OpenSSL* to encrypt and decrypt data. OpenSSL is available from www.openssl.org <`http://www.openssl.org/`>. OpenSSL offers a wide variety of cipher choices (**amcrypt-ossl** defaults to 256-bit AES) and can use hardware cryptographic accelerators on several platforms.

amcrypt-ossl will search for the OpenSSL program in the following directories: /bin:/usr/bin:/usr/local/

PASSPHRASE MANAGEMENT

amcrypt-ossl uses the same pass phrase to encrypt and decrypt data. It is very important to store and protect the pass phrase properly. Encrypted backup data can *only* be recovered with the correct passphrase.

OpenSSL's key derivation routines use a salt to guard against dictionary attacks on the pass phrase; still it is important to pick a pass phrase that is hard to guess. The Diceware method (see www.diceware.com <`http://www.diceware.com/`>) can be used to create passphrases that are difficult to guess and easy to remember.

FILES

/var/lib/amanda/.am_passphrase File containing the pass phrase. It should not be readable by any user other than the *Amanda* user.

SEE ALSO

amanda(8), amanda.conf(5), openssl(1), amcrypt-ossl-asym(8)

amcrypt-ossl-asym

Name

amcrypt-ossl-asym — crypt program for *Amanda* asymmetric data encryption using OpenSSL

Synopsis

```
amcrypt-ossl-asym [-d]
```

DESCRIPTION

amcrypt-ossl-asym uses *OpenSSL* to encrypt and decrypt data. OpenSSL is available from www.openssl.org <`http://www.openssl.org/`>. OpenSSL offers a wide variety of cipher choices (**amcrypt-ossl-asym** defaults to 256-bit AES) and can use hardware cryptographic accelerators on several platforms.

amcrypt-ossl-asym will search for the OpenSSL program in the following directories: /bin:/usr/bin:/usr/local/bin:/usr/ssl/bin:/usr/local/ssl/bin.

GENERATING PUBLIC AND PRIVATE KEYS

RSA keys can be generated with the standard OpenSSL commands, e.g.:

```
$ cd /var/lib/amanda
$ openssl genrsa -aes128 -out backup-privkey.pem 1024
Generating RSA private key, 1024 bit long modulus
[...]
Enter pass phrase for backup-privkey.pem: ENTER YOUR PASS PHRASE
Verifying - Enter pass phrase for backup-key.pem: ENTER YOUR PASS PHRASE

$ openssl rsa -in backup-privkey.pem -pubout -out backup-pubkey.pem
Enter pass phrase for backup-privkey.pem: ENTER YOUR PASS PHRASE
Writing RSA key
```

To generate a private key without a passphrase, omit the `-aes128` option. See openssl_genrsa(1) for more key generation options.

Note that it is always possible to generate the public key from the private key.

KEY AND PASSPHRASE MANAGEMENT

amcrypt-ossl-asym uses the *public key* to encrypt data. The security of the data does not depend on the confidentiality of the public key. The *private key* is used to decrypt data, and must be protected. Encrypted backup data cannot be recovered without the private key. The private key may optionally be encrypted with a passphrase.

While the public key must be online at all times to perorm backups, the private key and optional passphrase are only needed to restore data. It is recommended that the latter be stored offline all other times. For example, you could keep the private key on removable media, and copy it into place for a restore; or you could keep the private key online, encrypted with a passphrase that is present only for a restore.

OpenSSL's key derivation routines use a salt to guard against dictionary attacks on the pass phrase; still it is important to pick a pass phrase that is hard to guess. The Diceware method (see www.diceware.com <`http://www.diceware.com/`>) can be used to create passphrases that are difficult to guess and easy to remember.

FILES

/var/lib/amanda/backup-privkey.pem File containing the RSA private key. It should not be readable by any user other than the *Amanda* user.

/var/lib/amanda/backup-pubkey.pem File containing the RSA public key.

/var/lib/amanda/.am_passphrase File containing the passphrase. It should not be readable by any user other than the *Amanda* user.

SEE ALSO

amanda(8), amanda.conf(5), openssl(1), amcrypt-ossl(8)

amdd

Name

amdd — *Amanda* version of dd

Synopsis

`amdd [-d] [if=input] [of=output] [bs=blocksize] [skip=count] [count=count]`

DESCRIPTION

Amdd provides just enough of the standard UNIX *dd* command for the needs of *Amanda*. This is handy when doing a full restore and the standard *dd* program has not yet been found.

Amdd also provides access to the *Amanda* output drivers that support various tape simulations. This may be used for debugging or to convert from one format to another.

See the amanda(8) man page for more details about *Amanda*. See the *OUTPUT DRIVERS* section of *amanda(8)* for more information on the *Amanda* output drivers.

OPTIONS

`-d` Turn on debugging output.

`-l`***length*** Set the output length. If the output driver limits the output size, this controls when end of tape will be simulated.

Length may have a multiplier suffix:

k -¿ 1024 (Kilobytes)
b -¿ 512 (Blocks)
M -¿ 1024*1024 (Megabytes)

The default is no multiplier (bytes).

if=input Input to *dd*. Default is stdin.

of=output Where to send the output of *dd*. Default is stdout.

bs=blocksize Size of each record. Input records smaller than this will *not* be padded. Output records will be the same size as the corresponding input record. Default is 512 bytes.

Blocksize may have a multiplier suffix:

k -¿ 1024 (Kilobytes)
b -¿ 512 (Blocks)
M -¿ 1024*1024 (Megabytes)

The default is no multiplier (bytes).

count=count Number of records to copy. Default is all records until end of file.

skip=count Number of records to skip before copying input to output. Default is zero.

AUTHOR

Marc Mengel mengel@fnal.gov <mailto:mengel@fnal.gov>, John R. Jackson jrj@purdue.edu <mailto:jrj@purdue.edu> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8)

amdump

Name

amdump — back up all disks in an *Amanda* configuration

Synopsis

`amdump` *`config`* `[` *`host`* `[`*`disk...`*`]...]` `[-o`*`configoption`*`]...`

DESCRIPTION

Amdump switches to the appropriate *Amanda* configuration directory, e.g. /usr/local/etc/amanda/*config* then attempts to back up every disk specified by the *amanda.conf* file. *Amdump* is normally run by *cron*.

You can specify many host/disk expressions, only disks that match an expression will be dumped. All disks are dumped if no expressions are given.

If the file /usr/local/etc/amanda/*config*/hold exists, **amdump** will wait until it is removed before starting the backups. This allows scheduled backups to be delayed when circumstances warrant, for example, if the tape device is being used for some other purpose. While waiting, **amdump** checks for the hold file every minute.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

host [***disk***]* Specify the host and disk on which the command will work.

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLE

Here is a typical crontab entry. It runs **amdump** every weeknight at 1 a.m. as user *bin*:

```
0 1 * * 1-5 bin /usr/local/sbin/amdump daily
```

Please see the crontab(5) or crontab(1) manual page for the correct crontab format for your system.

MESSAGES

amdump: waiting for hold file to be removed The "hold" file exists and **amdump** is waiting for it to be removed before starting backups.

amdump: amdump or amflush is already running, or you must run amcleanup Amdump detected another **amdump** or **amflush** running, or the remains of a previous incomplete **amdump** or **amflush** run. This run is terminated. If the problem is caused by the remains of a previous run, you must execute amcleanup(8) and then rerun **amdump**.

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amcheck(8), amcleanup(8), amrestore(8), amflush(8), cron(8)

amfetchdump

Name

amfetchdump — extract backup images from multiple *Amanda* tapes.

Synopsis

```
amfetchdump [-pcClawns] [-d device] [-O directory] [-i logfile] [-b
     blocksize] config hostname [ disk [ date [ level [ hostname
     [...]]]]] [-oconfigoption]...
```

DESCRIPTION

Amfetchdump pulls one or more matching dumps from tape or from the holding disk, handling the reassembly of multi-tape split dump files as well as any tape autochanger operations.

It will automatically use the logs created by amdump(8) to locate available dumps on tape, in the same way that the *find* feature of amadmin(8) lists available dumps. If these logs are unavailable, it can search tape-by-tape to find what it needs, and can generate new logs to serve as an emergency tape inventory.

The *hostname*, *diskname*, *datestamp*, and *level* dump pattern-matching works as in amrestore(8), with the added requirement that at minimum a *hostname* must be specified when not in inventory mode.

Unless `-p` is used, backup images are extracted to files in the current directory named:

hostname.diskname.datestamp.dumplevel

OPTIONS

`-p` Pipe exactly one complete dump file to *stdout*, instead of writing the file to disk. This will restore only the first matching dumpfile (where "first" is determined by the dump log search facility).

`-d` *device* Restore from this tape device instead of the default.

`-O` *directory* Output restored files to this directory, instead of to the current working directory.

`-c` Compress output, fastest method available.

`-C` Compress output, smallest file size method available.

`-l` Leave dumps in the compressed/uncompressed state in which they were found on tape. By default, *amfetchdump* will automatically uncompress when restoring.

`-a` Assume that all tapes are already available, via tape changer or otherwise, instead of prompting the operator to ensure that all tapes are loaded.

`-i` *`filename`* Generate an inventory of all dumps "seen" on the tapes we search, for later use as a log.

`-w` Wait to put split dumps together until all chunks have been restored. Normally, *amfetchdump* will attempt to read pieces of a split file from tape in order, so that it can assemble them simply by appending each file to the first. This option disables the appending behavior, and instead restores each piece as an individual file and reassembles them only after all have been restored.

> NOTE
>
> 
>
> This requires at least double the size of your dump in free disk space, in order to build the final assembled dumpfile.

This behavior is implicitly invoked in circumstances where knowing the location of all dumps on tape in advance is not possible, such as when you are restoring without log files.

`-n` Do not reassemble split dump files at all, just restore each piece as an individual file.

`-s` Do not fast-forward straight to needed files on tape. This will slow down most restores substantially. Only use this option if your tape drive does not properly support the fast-forward operation.

`-b` *`blocksize`* Force a particular block size when reading from tapes. This value will usually be autodetected, and should not normally need to be set.

`-o` *`configoption`* See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLES

All the examples here assume your configuration is called *`SetA`*.

Here's a simple case, restoring all known dumps of the host vanya to the current working directory.

```
$ amfetchdump SetA vanya
```

A more likely scenario involves restoring a particular dump from a particular date. We'll pipe this one to **GNU-tar** as well, to automatically extract the dump.

$ amfetchdump -p SetA vanya /home 20051020 — gtar -xvpf -

In a situation where all of our dump logs have been wiped out, we could also use amfetchdump to inventory our tapes and recreate an imitation of those logs, which we'll send to *stdout* for casual perusal.

$ amfetchdump -i - SetA

Note that you can specify a restore while in inventory mode, and **amfetchdump** will continue searching for more dumps from this host even after successfully restoring a dump, inventorying all the while. If your backup searcher has been trashed, this is a handy way to recover what you have.

$ amfetchdump -i /var/amanda/log SetA backupserver

CAVEATS

Amfetchdump is dependent on accessing your server's config, tape changer, and (normally) dump logs. As such, it's not necessarily the most useful tool when those have all been wiped out and you desperately need to pull things from your tape. Pains have been taken to make it as capable as possible, but for seriously minimialist restores, look to amrestore(8) or dd(8) instead.

AUTHOR

John Stange, building@nap.edu <mailto:building@nap.edu>, National Academies Press

Ian Turner, ian@zmanda.com <mailto:ian@zmanda.com>: XML-conversion

SEE ALSO

amanda(8), amadmin(8), amrestore(8), tar(1) restore(8)

amflush

Name

amflush — flush *Amanda* backup files from holding disk to tape

Synopsis

```
amflush [-b] [-f] [-s] [-D datestamp] config [ host [disk...]...]
     [-oconfigoption]...
```

DESCRIPTION

Amflush writes *Amanda* backups from the holding disks to tape, and updates the *Amanda* info database and tapelist accordingly. Backups may stay in a holding disk when something is wrong with the tape at the time *amdump* is run. When this happens, the problem must be corrected and **amflush** run by hand.

OPTIONS

`-b` Run **amflush** in batch mode. All datestamps are selected unless specified. The flush is started without confirmation.

`-f` Run **amflush** in foreground. *Amflush* normally detaches itself from the tty and runs as a background process. With the `-f` option, **amflush** stays in the foreground. This is useful if **amflush** is run as part of another script that, for example, advances the tape after the flush is completed.

`-s` Write log to stdout/stderr instead of the amflush log file. Requires the `-f` option.

`-D datestamp` specify a datestamp expression you want to flush, see the "DATESTAMP EXPRESSION" section of amanda(8) for a description. `-D 20001225-7` will flush all dumps from 25 december 2000 to 27 december 2000.

host [*disk*]* Specify the host and disk on which the command will work.

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

You can specify many host/disk expressions, only disks that match an expression will be flushed. All disks are flushed if no expressions are given. see the "HOST & DISK EXPRESSION" section of amanda(8) for a description.

Amflush will look in the holding disks specified by the *amanda.conf* file in /usr/local/etc/amanda/*config* for any non-empty *Amanda* work directories. It then prompts you to select a directory or to process all of the directories. The work directories in the holding disks are named by the date at the time *amdump* was run, e.g. `19910215`.

See the amanda(8) man page for more details about *Amanda*.

EXAMPLE

Amflush will search for holding areas associated with the *daily* configuration. After you select which holding area to flush, **amflush** writes the data to tape, updates the databases and sends a mail report similar to amdump(8).

```
% amflush daily
Scanning /amanda-hold...
  20001113: found Amanda directory.
  20001114: found Amanda directory.

Multiple Amanda directories, please pick one by letter:
  A. 20001113
  B. 20001114
Select directories to flush [A..B]: [ALL] all

Flushing dumps in 20001113, 20001114,
today: 20001117
to tape drive /dev/rmt/0mn.
Expecting tape DMP014 or a new tape.  (The last dumps were to tape DMP013)
Are you sure you want to do this? yes
Running in background, you can log off now.
You'll get mail when amflush is finished.
```

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amdump(8)

amgetconf

Name

amgetconf — look up amanda.conf variables

Synopsis

`amgetconf [`*`config`*`] [`*`--list`*`] `*`parameter`*` [-o`*`configoption`*`]...`

DESCRIPTION

Amgetconf looks up parameters in *amanda.conf*, the *Amanda* configuration file, or from the build and runtime environment, and returns their corresponding value.

If *config* is not specified, **amgetconf** assumes it is being run from the configuration directory and that *amanda.conf* is present.

If *parameter* begins with *build.*, the (case insensitive) string following the period is a build environment variable. Variables without a value (e.g. *XFSDUMP* on a system that does not support that type of file system) will not report an error and will return an empty string as the value. Flag variables (e.g. *USE_AMANDAHOSTS*) will return `1` if the flag is set or an empty string if it is not.

If *parameter* begins with *dbopen.*, the string following the period is a program name and an *Amanda* debug file will be created for the caller. The name of the file is returned.

If *parameter* begins with *dbclose.*, the string following the period is a program name previously used with *dbopen.*, followed by a colon (:) and the previously opened file name.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

–list The parameter must be 'tapetype', 'dumptype, 'holdingdisk' or 'interface'. It will output, one by line, the list of identifier for the parameter.

–list tapetype Output the list of tapetype, one by line.

–list dumptype Output the list of dumptype, one by line.

–list holdingdisk Output the list of holdingdisk, one by line.

–list interface Output the list of interface, one by line.

parameter It could be one of the below format:

runtapes

DUMPTYPE:no-compress:compress

TAPETYPE:HP-DAT:length

INTERFACE:local:use

HOLDINGDISK:hd1:use

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLE

Find out the path to the log file directory:

% amgetconf daily logdir
/usr/local/etc/amanda//daily

Find out the current tape type:

% amgetconf daily tapetype
DLT4000-IV

Find out the default configuration directory:

% amgetconf daily build.CONFIG˙DIR
/usr/local/etc/amanda/

Create, use and close a debug file in a script:

% set debug˙file = ‘amgetconf daily dbopen.myscript‘
% echo debug information ¿¿ $debug˙file
% amgetconf daily dbclose.myscript:$debug˙file

MESSAGES

amgetconf: no such parameter *param* Parameter *param* is not a known keyword (e.g. not a valid *amanda.conf* keyword).

SEE ALSO

amanda(8)

amlabel

Name

amlabel — label an *Amanda* tape

Synopsis

`amlabel [-f] config label [slot slot] [-oconfigoption]...`

DESCRIPTION

All *Amanda* tapes must be pre-labeled before they are used. *Amanda* verifies the label in *amdump* and *amflush* before writing to make sure the proper tape is loaded.

Amlabel writes an *Amanda* label on the tape in the device specified by the *amanda.conf* file in /usr/local/etc/amanda/*config*. *Label* may be any string that does not contain whitespace and that matches the *amanda.conf labelstr* regular expression option. It is up to the system administrator to define a naming convention.

Amlabel appends the new tape to the *tapelist* file so it will be used by *Amanda* before it reuses any other tapes. When you **amlabel** multiple tapes, they will be used in the order you **amlabel** them.

Amlabel will not write the label if the tape contains an active *Amanda* tape or if the label specified is on an active tape. The `-f` (force) flag bypasses these verifications.

An optional *slot* may be specified after the tape label. If a tape changer is in use, **amlabel** will label the tape in the specified slot instead of the currently loaded tape.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLE

Write an *Amanda* label with the string "DMP000" on the tape loaded in the device named in the *tapedev* option in /usr/local/etc/amanda/daily/amanda.conf:

```
% amlabel daily DMP000
```

Label the tape in slot 3 of the currently configured tape changer with the string "DMP003":

```
% amlabel daily DMP003 slot 3
```

MESSAGES

label *label* doesn't match labelstr *str* Label *label* on the command line does not match the *labelstr* regular expression *str* from *amanda.conf*.

label *label* already on a tape Label *label* is already listed as an active *Amanda* tape.

no tpchanger specified in *path* , so slot command invalid The command line has the *slot* parameter but the *amanda.conf* file in *path* does not have a tape changer configured.

reading label *label*, tape is in another amanda configuration This tape appears to be a valid *Amanda* tape, but label does not match *labelstr* for this configuration so it is probably part of a different *Amanda* configuration.

reading label *label*, tape is active Tape *label* appears to already be part of this *Amanda* configuration and active, i.e. has valid data on it.

no label found, are you sure *tape* is non-rewinding? While checking that the label was written correctly, **amlabel** got an error that might be caused by mis-configuring *Amanda* with a rewinding tape device name instead of a non-rewinding device name for *tape*.

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org>: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8) amdump(8) amflush(8)

ammt

Name

ammt — *Amanda* version of mt

Synopsis

`ammt [-d] [-f | -t | device] command [count]`

DESCRIPTION

Ammt provides just enough of the standard UNIX *mt* command for the needs of *Amanda*. This is handy when doing a full restore and the standard *mt* program has not yet been found.

Ammt also provides access to the *Amanda* output drivers that support various tape simulations.

See the amanda(8) man page for more details about *Amanda*. See the *OUTPUT DRIVERS* section of *amanda(8)* for more information on the *Amanda* output drivers.

OPTIONS

`-d` Turn on debugging output.

`-f`*device* Access tape device *device*. If not specified, the *TAPE* environment variable is used.

`-t`*device* Same as `-f`.

commandcount Which command to issue, and an optional count of operations.

COMMANDS

Each command may be abbreviated to whatever length makes it unique.

eof|weofcount Write *count* (default: 1) end of file marks (tapemarks).

fsfcount Skip forward *count* (default: 1) files.

bsfcount Skip backward *count* (default: 1) files.

asfcount Position to file number *count* (default: 0) where zero is beginning of tape. This is the same as a *rewind* followed by a *fsf count*.

rewind Rewind to beginning of tape.

offline|rewofflf Rewind to beginning of tape and unload the tape from the drive.

status Report status information about the drive. Which data reported, and what it means, depends on the underlying operating system, and may include:

ONLINE Indicates the drive is online and ready.

OFFLINE Indicates the drive is offline or not ready.

BOT Indicates the drive is at beginning of tape.

EOT Indicates the drive is at end of tape.

PROTECTED Indicates the tape is write protected.

ds Device status.

er Error register.

fileno Current tape file number.

blkno Current tape block number file.

NOTE

Many systems only report good data when a tape is in the drive and ready.

AUTHOR

Marc Mengel mengel@fnal.gov <mailto:mengel@fnal.gov>, John R. Jackson jrj@purdue.edu <mailto:jrj@purdue.edu>: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8)

amoverview

Name

amoverview — display file systems processed by *Amanda* over time

Synopsis

```
amoverview [[-config] config ] [-hostwidth width] [-diskwidth width]
     [-skipmissed] [-last] [-num0] [-togo0] [-verbose]
```

DESCRIPTION

Amoverview displays a chart showing hosts and file systems processed by *Amanda* along with the backup level performed each day.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

`-config` *`config`* Use configuration *config* instead of configuration daily.

`-hostwidth` *`width`* Set *host* field column width to *width* characters instead of 8.

`-diskwidth` *`width`* Set *disk* field column width to *width* characters instead of 20.

`-skipmissed` Compacts the output by only printing stats for the days *Amanda* actually ran.

`-last` Outputs the last status of each disk at the start. Useful for long tapecycles and/or sparse reports.

`-num0` Outputs the number of level 0 dumps for each disk.

`-togo0` Outputs the number of runs until the last level 0 dump is overwritten.

`-verbose` *Amoverview* can take a long while on large systems. This option reports intermediate steps while it is working.

RESULTS

amoverview is a summary of the output of " *amadmin <config> find* ". When the last column of *amadmin find* contains anything other than "OK", amoverview translates this into "E" for that day.

A number indicates the level of backup and it succeeded. An "E" indicates an error for that day. You get an "E" for all errors, like failed to connect, datatimeout, computer crashed, etc, but also for failing to write to tape.

You can have an "E" followed by a number if a filesystem ran into end-of-tape once (gives an "E", and later that day, you flush it to a second tape (a number: the level, indicating success). If the flush failed too, you get a double "EE" for that day.

You can also have a double code if you have two tapes in the changer and *Amanda* failed to write to tape the first time because it hit end of tape (resulting in "E0", for a full, "E1" for an incremental etc.) or twice with error ("EE"), and may a successful flush afterwards giving maybe "EE0". (Only the latest 2 characters are printed).

EXAMPLE

This shows the `/home` file system on *host2* was backed up at level 3 on the 8th, 9th and 10th of December, had a full backup on the 11th, a level 1 on the 12th and a level 2 on the 13th.

```
# amoverview
                         date 12 12 12 12 12 12
host     disk                 08 09 10 11 12 13

host1    /                     0  1  1  1  1  1
host1    /var                  0  1  1  1  1  1
host2    /                     1  1  1  1  1  0
host2    /home                 3  3  3  0  1  2
host2    /opt                  1  1  1  1  1  1
host2    /var                  1  1  0  1  1  1
```

SEE ALSO

amadmin(8), amanda(8)

amplot

Name

amplot — visualize the behavior of *Amanda*

Synopsis

`amplot [-b] [-c] [-e] [-g] [-l] [-p] [ -t` *T* `] amdump_files`

DESCRIPTION

Amplot reads an *amdump* output file that *Amanda* generates each run (e.g. *amdump.1*) and translates the information into a picture format that may be used to determine how your installation is doing and if any parameters need to be changed. *Amplot* also prints out *amdump* lines that it either does not understand or knows to be warning or error lines and a summary of the start, end and total time for each backup image.

Amplot is a shell script that executes an *awk* program (*amplot.awk*) to scan the *amdump* output file. It then executes a *gnuplot* program (*amplot.g*) to generate the graph. The *awk* program is written in an enhanced version of awk, such as GNU awk (*gawk* version 2.15 or later) or *nawk*.

During execution, **amplot** generates a few temporary files that *gnuplot* uses. These files are deleted at the end of execution.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

`-b` Generate b/w postscript file (need `-p`).

`-c` Compress *amdump_files* after plotting.

`-e` Extend the X (time) axis if needed.

`-g` Direct *gnuplot* output directly to the X11 display (default).

`-p` Direct postscript output to file *YYYYMMDD.ps* (opposite of `-g`).

`-l` Generate landscape oriented output (needs `-p`).

`-t`***T*** Set the right edge of the plot to be *T* hours.

The *amdump_files* may be in various compressed formats (*compress*, *gzip*, *pact*, *compact*).

INTERPRETATION

The figure is divided into a number of regions. There are titles on the top that show important statistical information about the configuration and from this execution of *amdump*. In

the figure, the X axis is time, with 0 being the moment *amdump* was started. The Y axis is divided into 5 regions:

QUEUES: How many backups have not been started, how many are waiting on space in the holding disk and how many have been transferred successfully to tape.

%BANDWIDTH: Percentage of allowed network bandwidth in use.

HOLDING DISK: The higher line depicts space allocated on the holding disk to backups in progress and completed backups waiting to be written to tape. The lower line depicts the fraction of the holding disk containing completed backups waiting to be written to tape including the file currently being written to tape. The scale is percentage of the holding disk.

TAPE: Tape drive usage.

%DUMPERS: Percentage of active dumpers.

The idle period at the left of the graph is time *amdump* is asking the machines how much data they are going to dump. This process can take a while if hosts are down or it takes them a long time to generate estimates.

AUTHOR

Olafur Gudmundsson ogud@tis.com <mailto:ogud@tis.com>, Trusted Information Systems, formerly at University of Maryland, College Park: Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

BUGS

Reports lines it does not recognize, mainly error cases but some are legitimate lines the program needs to be taught about.

SEE ALSO

amanda(8), amdump(8), gawk(1), nawk(1), awk(1), gnuplot(1), sh(1), compress(1), gzip(1)

amrecover

Name

amrecover — *Amanda* index database browser

Synopsis

```
amrecover [-C config] [-s index-server] [-t tape-server] [-d tape-device]
     [-oconfigoption]...
```

DESCRIPTION

Amrecover browses the database of *Amanda* index files to determine which tapes contain files to recover. Furthermore, it is able to recover files.

In order to restore files in place, you must invoke **amrecover** from the root of the backed up filesystem, or use *lcd* to move into that directory, otherwise a directory tree that resembles the backed up filesystem will be created in the current directory. See the examples below for details.

Amrecover will read the *amanda-client.conf* file and the *config/amanda-client.conf* file.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

NOTE

The Default values are those set at compile-time. Use **amrestore** to recover client-encrypted or client-custom-compressed tapes.

*[**-C**] **config*** *Amanda* configuration.

`-s index-server` Host that runs the index daemon.

`-t tape-server` Host that runs the tape server daemon.

`-d tape-device` Tape device to use on the tape server host.

-o *clientconfigoption* See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

COMMANDS

Amrecover connects to the index server and then presents a command line prompt. Usage is similar to an ftp client. The GNU readline library is used to provide command line history and editing if it was built in to **amrecover**.

The purpose of browsing the database is to build up a *restore list* of files to be extracted from the backup system. The following commands are available:

sethost hostname Specifies which host to look at backup files for (default: the local host).

setdate YYYY-MM-DD-HH-MM[-SS] | YYYY-MM-DD Set the restore time (default: now). File listing commands only return information on backup images for this day, for the day before with the next lower dump level, and so on, until the most recent level 0 backup on or before the specified date is encountered.

For example, if:

1996-07-01 was a level 0 backup
1996-07-02 through 1996-07-05 were level 1 backups
1996-07-06 through 1997-07-08 were level 2 backups

then the command *setdate 1997-07-08-00* would yield files from the following days:

1997-07-08 (the latest level 2 backup)
1997-07-05 (the latest level 1 backup)
1997-07-01 (the latest level 0 backup)

Only the most recent version of a file will be presented.

The following abbreviated date specifications are accepted:

`--MM-DD` dates in the current year

`---DD` dates in the current month of the current year

setdisk diskname [***mountpoint***] Specifies which disk to consider (default: the disk holding the working directory where **amrecover** is started). It can only be set after the host is set with *sethost*. *Diskname* is the device name specified in the *amanda.conf* or *disklist* configuration file. The disk must be local to the host. If *mountpoint* is not specified, all pathnames will be relative to the (unknown) mount point instead of full pathnames.

listhost [***diskdevice***] List all *host*

listdisk [***diskdevice***] List all *diskname*

settape [[***server***]**:**][***tapedev***|***default***] Specifies the host to use as the tape server, and which of its tape devices to use. If the server is omitted, but the colon is not, the server name reverts to the configure-time default. If the tape device is omitted, it remains unchanged. To use the default tape device selected by the tape server, the word *default* must be specified. If no argument is specified, or the argument is an empty string, no changes occur, and the current settings are displayed.

If you want amrecover to use your changer, the *tapedev* must be equal to the amrecover_changer setting on the server.

If you need to change the protocol (tape:, rait:, file:, null:) then you must specify the hostname.

```
settape 192.168.0.10:file:/file1
```

You can change the tape device when amrecover ask you to load the tape:

```
Load tape DMP014 now
Continue? [Y/n/t]: t
Tape device: server2:/dev/nst2
Continue? [Y/n/t]: Y
Using tape /dev/nst2 from server server2.
```

setmode mode Set the extraction mode for Samba shares. If *mode* is *smb*, shares are sent to the Samba server to be restored back onto the PC. If *mode* is *tar*, they are extracted on the local machine the same way tar volumes are extracted.

mode Displays the extracting mode for Samba shares.

history Show the backup history of the current host and disk. Dates, levels, tapes and file position on tape of each backup are displayed.

pwd Display the name of the current backup working directory.

cd dir Change the backup working directory to *dir*. If the mount point was specified with *setdisk*, this can be a full pathname or it can be relative to the current backup working directory. If the mount point was not specified, paths are relative to the mount point if they start with "/", otherwise they are relative to the current backup working directory. The *dir* can be a shell style wildcards.

cdx dir Like the *cd* command but allow regular expression.

lpwd Display the **amrecover** working directory. Files will be restored under this directory, relative to the backed up filesystem.

lcd path Change the **amrecover** working directory to *path*.

ls List the contents of the current backup working directory. See the description of the

setdate command for how the view of the directory is built up. The backup date is shown for each file.

add item1 item2 ... Add the specified files or directories to the restore list. Each item may have shell style wildcards.

addx item1 item2 ... Add the specified files or directories to the restore list. Each item may be a regular expression.

delete item1 item2 ... Delete the specified files or directories from the restore list. Each item may have shell style wildcards.

deletex item1 item2 ... Delete the specified files or directories from the restore list. Each item may be a regular expression.

list file Display the contents of the restore list. If a file name is specified, the restore list is written to that file. This can be used to manually extract the files from the *Amanda* tapes with *amrestore*.

clear Clear the restore list.

quit Close the connection to the index server and exit.

exit Close the connection to the index server and exit.

extract Start the extract sequence (see the examples below). Make sure the local working directory is the root of the backed up filesystem, or another directory that will behave like that. Use *lpwd* to display the local working directory, and *lcd* to change it.

help Display a brief list of these commands.

EXAMPLES

The following shows the recovery of an old *syslog* file.

```
# cd /var/log
# ls -l syslog.7
syslog.7: No such file or directory
# amrecover
AMRECOVER Version 2.4.2. Contacting server on oops ...
220 oops Amanda index server (2.4.2) ready.
Setting restore date to today (1997-12-09)
```

```
200 Working date set to 1997-12-09.
200 Config set to daily.
200 Dump host set to this-host.some.org.
$CWD '/var/log' is on disk '/var' mounted at '/var'.
200 Disk set to /var.
/var/log
WARNING: not on root of selected filesystem, check man-page!
amrecover> ls
1997-12-09 daemon.log
1997-12-09 syslog
1997-12-08 authlog
1997-12-08 sysidconfig.log
1997-12-08 syslog.0
1997-12-08 syslog.1
1997-12-08 syslog.2
1997-12-08 syslog.3
1997-12-08 syslog.4
1997-12-08 syslog.5
1997-12-08 syslog.6
1997-12-08 syslog.7
amrecover> add syslog.7
Added /log/syslog.7
amrecover> lpwd
/var/log
amrecover> lcd ..
/var
amrecover> extract

Extracting files using tape drive /dev/nst0 on host 192.168.0.10

The following tapes are needed: DMP014

Restoring files into directory /var
Continue? [Y/n]: y

Load tape DMP014 now
Continue? [Y/n/t]: y
set owner/mode for '.'? [yn] n
amrecover> quit
200 Good bye.
# ls -l syslog.7
total 26
-rw-r--r--   1 root     other      12678 Oct 14 16:36 syslog.7
```

If you do not want to overwrite existing files, create a subdirectory to run **amrecover** from and then move the restored files afterward.

```
# cd /var
# (umask 077 ; mkdir .restore)
# cd .restore
# amrecover
AMRECOVER Version 2.4.2. Contacting server on oops ...
...
amrecover> cd log
/var/log
amrecover> ls
...
amrecover> add syslog.7
Added /log/syslog.7
amrecover> lpwd
/var/.restore
amrecover> extract

Extracting files using tape drive /dev/nst0 on host 192.168.0.10
...
amrecover> quit
200 Good bye.
# mv -i log/syslog.7 ../log/syslog.7-restored
# cd ..
# rm -fr .restore
```

If you need to run *amrestore* by hand instead of letting **amrecover** control it, use the *list* command after browsing to display the needed tapes.

```
# cd /var/log
# amrecover
AMRECOVER Version 2.4.2. Contacting server on oops ...
...
amrecover> ls
...
amrecover> add syslog syslog.6 syslog.7
Added /log/syslog
Added /log/syslog.6
Added /log/syslog.7
amrecover> list
TAPE DMP014 LEVEL 0 DATE 1997-12-08
        /log/syslog.7
        /log/syslog.6
TAPE DMP015 LEVEL 1 DATE 1997-12-09
        /log/syslog
amrecover> quit
```

The *history* command shows each tape that has a backup of the current disk along with the date of the backup, the level, the tape label and the file position on the tape. All active tapes are listed, not just back to the most recent full dump.

Tape file position zero is a label. The first backup image is in file position one.

```
# cd /var/log
# amrecover
AMRECOVER Version 2.4.2. Contacting server on oops ...
...
amrecover> history
200- Dump history for config "daily" host "this-host.some.org" disk "/var"
201- 1997-12-09 1 DMP015 9
201- 1997-12-08 1 DMP014 11
201- 1997-12-07 0 DMP013 22
201- 1997-12-06 1 DMP012 16
201- 1997-12-05 1 DMP011 9
201- 1997-12-04 0 DMP010 11
201- 1997-12-03 1 DMP009 7
201- 1997-12-02 1 DMP008 7
201- 1997-12-01 1 DMP007 9
201- 1997-11-30 1 DMP006 6
...
amrecover> quit
```

ENVIRONMENT

`PAGER` The *ls* and *list* commands will use $PAGER to display the file lists. Defaults to *more* if PAGER is not set.

`AMANDA_SERVER` If set, $AMANDA_SERVER will be used as index-server. The value will take precedence over the compiled default, but will be overridden by the -s switch.

`AMANDA_TAPE_SERVER` If set, $AMANDA_TAPE_SERVER will be used as tape-server. The value will take precedence over the compiled default, but will be overridden by the -t switch.

AUTHOR

Alan M. McIvor alan@kauri.auck.irl.cri.nz <mailto:alan@kauri.auck.irl.cri.nz> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amanda-client.conf(5), amrestore(8), amfetchdump(8), readline(3)

amreport

Name

amreport — generate a formatted output of statistics for an *Amanda* run

Synopsis

`amreport [`*`config`*`] [-i] [-M `*`address`*`] [-l `*`logfile`*`] [-f `*`outputfile`*`] [-p `*`postscriptfile`*`] [-o`*`configoption`*`]...`

DESCRIPTION

Amreport generates a summary report of an amanda(8) backup run. If no configuration name is specified, amanda.conf is read from the current directory.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

config Name of the configuration to process.

`-i` Don't email the report.

`-M` ***`address`*** Mail the report to *address* instead of the *mailto* value from *amanda.conf*.

`-l` ***`logfile`*** Name of the log file to parse to generate the report. If a log file is not specified, it defaults to the file:

logdir/log

where *logdir* is the log directory defined in amanda.conf.

`-f` ***`outputfile`*** Normally, **amreport** sends the report via e-mail to the *mailto* user as defined in the amanda.conf file. If *outputfile* is specified, then the report is put in *outputfile*.

`-p` ***`postscriptfile`*** Send the postscript output to the file *postscriptfile* instead of to the lpr(1) command. This option has an effect only if the *lbl-templ* directive is specified in amanda.conf.

`-o configoption` See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

LABEL PRINTING

Amanda can print postscript labels describing the contents of tape(s) written in a run. The labels are designed to be folded and inserted into the tape case along with the tape or hole punched and put in a 3-ring binder. Various label templates are provided to format data for different tape sizes.

The information printed varies slightly between label templates due to size constraints. Labels contain one line for each host/file-system pair and may also contain the file number on the tape, the level of the dump, the original size of the dump and the size of the (possibly compressed) tape file.

Add the *lbl-templ* parameter to the tapetype definition in amanda.conf to enable labels. If you don't add this line to your tapetype definition, **amreport** will not print tape labels.

You may use the *remap='I'>printer* keyword in amanda.conf to print to other than the system default printer.

TEMPLATES

Amanda provides label templates for the following tape types. These are pretty generic labels and should be easy to customize for other tape types or particular site needs.

* ExaByte 8mm tapes
* DAT 4mm tapes
* DLT tapes
* 3-ring binder

The 3-ring binder type is the most generic. It may be used to make a hardcopy log of the tapes.

SEE ALSO

amanda(8), amflush(8)

amrestore

Name

amrestore — extract backup images from an *Amanda* tape

Synopsis

```
amrestore [-r | -c | -C] [-b | blocksize] [-f | fileno] [-l | label] [-p]
     [-h] tapedevice | holdingfile [hostname [diskname [datestamp
     [hostname [diskname [datestamp | ...]]]]]]
```

DESCRIPTION

Amrestore extracts backup images from the tape mounted on *tapedevice* or from the holding disk file *holdingfile* that match *hostname*, *diskname* and *datestamp* patterns given on the command line. The tape or holding file must be in a format written by the *amdump* or *amflush* program.

If *diskname* is not specified, all backups on the tape for the previous *hostname* are candidates. If *datestamp* is not specified, all backups on the tape for the previous *hostname* and *diskname* are candidates. If no *hostname*, *diskname* or *datestamp* are specified, every backup on the tape is a candidate.

Hostname and *diskname* are special expressions described in the "HOST & DISK EXPRESSION" section of amanda(8). *Datestamp* are special expression described in the "DATESTAMP EXPRESSION" section of amanda(8). For example, if *diskname* is "rz[23]a", it would match disks *rz2a* and *rz3a*.

Datestamp is useful if *amflush* writes multiple backup runs to a single tape.

Unless `-p` is used, candidate backup images are extracted to files in the current directory named:

hostname.diskname.datestamp.dumplevel

Amrestore doesn't use a changer, it restore from the tape already loaded in the *tapedevice.*

OPTIONS

`-b` Set the blocksize used to read the tape or holding file. All holding files must be read with a blocksize of 32 KBytes. *Amrestore* should normally be able to determine the blocksize for tapes on its own and not need this parameter.

The default is 32 KBytes.

`-f` Do a rewind followed by a fsf <fileno> before trying to restore an image.

`-l` Check if we restoring from the tape with the right *label*

`-p` Pipe output. The first matching backup image is sent to standard output, which is normally a pipe to *restore* or *tar*, then **amrestore** quits. It may be run again to continue selecting backups to process. Make sure you specify the no-rewind *tapedevice* when doing this.

Note: *restore* may report "short read" errors when reading from a pipe. Most versions of *restore* support a blocking factor option to let you set the read block size, and you should set it to 2. See the example below.

`-c` Compress output using the fastest method the compression program provides. *Amrestore* normally writes output files in a format understood by *restore* or *tar*, even if the

backups on the tape are compressed. With the `-c` or `-C` option, **amrestore** writes all files in compressed format, even if the backups on the tape are not compressed. Output file names will have a .Z or .gz extension depending on whether *compress* or *gzip* is the preferred compression program. This option is useful when the current directory disk is small.

`-C` Compress output using the best method the compression program provides (may be very CPU intensive). See the notes above about the `-c` option.

`-r` Raw output. Backup images are output exactly as they are on the tape, including the *amdump* headers. Output file names will have a .RAW extension. This option is only useful for debugging and other strange circumstances.

`-h` Header output. The tape header block is output at the beginning of each file. This is like `-r` except `-c` or `-C` may also be used to compress the result. *Amrecover* uses the header to determine the restore program to use.

If a header is written (-r or -h), only 32 KBytes are output regardless of the tape blocksize. This makes the resulting image usable as a holding file.

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

EXAMPLES

The following does an interactive restore of disk *rz3g* from host *seine*, to restore particular files. Note the use of the *b* option to *restore*, which causes it to read in units of two 512-byte blocks (1 Kbyte) at a time. This helps keep it from complaining about short reads.

% amrestore -p /dev/nrmt9 seine rz3g — restore -ivbf 2 -

The next example extracts all backup images for host *seine*. This is the usual way to extract all data for a host after a disk crash.

% amrestore /dev/nrmt9 seine

If the backup datestamp in the above example is `19910125` and *seine* has level 0 backups of disks *rz1a* and *rz1g* on the tape, these files will be created in the current directory:

seine.rz1a.19910125.0
seine.rz1g.19910125.0

You may also use **amrestore** to extract a backup image from a holding disk file that has not yet been flushed to tape:

% amrestore -p /amanda/20001119/seine.rz1a.2 — restore -ivbf 2 -

Amrestore may be used to generate a listing of images on a tape:

% mt -f /dev/nrmt9 rewind
% amrestore -p /dev/nrmt9 no-such-host ¿ /dev/null

This asks **amrestore** to find images for host *no-such-host*. It will not find any entries that match, but along the way will report each image it skips.

CAVEATS

GNU-tar must be used to restore files from backup images created with the GNUTAR dumptype. Vendor tar programs sometimes fail to read GNU tar images.

AUTHOR

James da Silva, jds@amanda.org <`mailto:jds@amanda.org`>, University of Maryland, College Park: Original text

Stefan G. Weichinger, sgw@amanda.org <`mailto:sgw@amanda.org`>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8), amdump(8), amflush(8), tar(1) restore(8)

amrmtape

Name

amrmtape — remove a tape from the *Amanda* database

Synopsis

`amrmtape [-n] [-v] [-q] [-d] config label`

DESCRIPTION

Amrmtape invalidates the contents of an existing backup tape in the configuration database. This is meant as a recovery mechanism when a good backup is damaged either by faulty hardware or user error, e.g. the tape is eaten by the drive or is overwritten.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

`-n` Generate new *tapelist* and database files with *label* removed, but leave them in `/tmp` and do not update the original copies.

`-v` List backups of hosts and disks that are being discarded. Enabled by default.

`-q` Opposite of `-v`.

`-d` Run in debugging mode so all executed commands are shown.

EXAMPLE

Remove tape labeled *DAILY034* from the *DailySet1* configuration.

```
# amrmtape DailySet1 DAILY034
```

AUTHOR

Adrian T. Filipi-Martin atf3r@cs.virginia.edu <`mailto:atf3r@cs.virginia.edu`>: Original text

Stefan G. Weichinger, sgw@amanda.org <`mailto:sgw@amanda.org`>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amadmin(8), amanda(8)

amstatus

Name

amstatus — display the state of an *Amanda* run

Synopsis

```
amstatus [--config] config [ --file amdumpfile ] [--summary] [--dumping]
     [--waitdumping] [--waittaper] [--dumpingtape] [--writingtape]
     [--finished] [--failed] [--estimate] [--gestimate] [--stats]
```

DESCRIPTION

Amstatus gives the current state of the *Amanda* run specified by the *config* configuration. If there is no active *Amanda* running, it summarizes the result of the last run. It may also be used to summarize the results of a previous run.

See the amanda(8) man page for more details about *Amanda*.

OPTIONS

All options may be abbreviated to the shortest non-ambiguous sub-string. If no options are given, everything is displayed.

[–config] config Specify the *Amanda* configuration you want to display the state for.

`--file amdumpfile` Specify an alternate file instead of the *amdump* or *amflush* file.

`--summary` Display a summary of the state of the run.

`--dumping` Display all partitions that are dumping.

`--waitdumping`|`wdumping` Display all partitions that are waiting to be dumped.

`--waittaper`|`wtaper` Display all partitions dumped that are waiting to be written to tape.

`--dumpingtape`|`dtape` Display all partitions that are dumping directly to tape.

`--writingtape`|`wtape` Display all partitions that are writing to tape.

`--finished` Display all partitions that are dumped and written to tape.

`--failed`|`error` Display all partitions that failed.

`--estimate` Display all partitions whose estimate is finished. Works only during the estimate phase.

`--gestimate`|`gettingestimate` Display all partitions whose estimate is not finished. Works only during the estimate phase.

`--stats`|`statistics` Display statistics about active-time of taper and dumpers.

SEE ALSO

amanda(8), amcheck(8), amdump(8), amrestore(8), amadmin(8)

amtape

Name

amtape — user interface to *Amanda* tape changer controls

Synopsis

amtape *config command* [*command_options...*] [-o*configoption*]...

DESCRIPTION

Amtape performs tape changer control operations. It uses the underlying tape changer script defined by the *tpchanger* option for a particular *Amanda* configuration as specified by the *config* argument.

Tape changers maintain a notion of the *current* and *next* slot for each configuration. These may or may not correspond to an actual physical state of the device, but do tend to minimize searching through the tape storage slots. If the desired tape is in the current slot, it is likely the next tape needed is in the next slot rather than at some random position in the storage slots.

See the amanda(8) man page for more details about *Amanda*.

COMMANDS

reset Reset the tape changer to a known state. The *current* slot is set to the *first* slot. Other device-specific side effects may occur. Some gravity stackers need to be reset to the top position by hand. This command notifies *Amanda* the stacker is back in that position.

eject If a tape is loaded in the drive, it is ejected and returned to the slot from which it was loaded.

clean If a cleaning tape is defined for the changer, it is used to clean the drive.

show Show the contents of all slots. This can be slow.

label label Search for and load the *Amanda* tape with label *label*.

taper Perform the *taper* scan algorithm. Load the next tape in the configuration's tape sequence, or a fresh tape with a suitable label.

device Display the name of the current tape device on *stdout*.

current Display the current slot.

update Update the changer label database, if it has one, to match the tapes now available.

slot slot Eject any tape in the drive and put it away, then load the tape from slot *slot* and reset *current*.

slot current Eject any tape in the drive and put it away, then load the tape from the current slot.

slot prev Eject any tape in the drive and put it away, then load the tape from the previous slot and reset *current*.

slot next Eject any tape in the drive and put it away, then load the tape from the next slot and reset *current*.

slot first Eject any tape in the drive and put it away, then load the tape from the first slot and reset *current*.

slot last Eject any tape in the drive and put it away, then load the tape from the last slot and reset *current*.

slot advance Eject any tape in the drive and put it away. Advance *current* to the next tape, but do not load it.

-o configoption See the "*CONFIGURATION OVERRIDE*" section in amanda(8).

This is useful with non-gravity stackers to unload the last tape used and set up *Amanda* for the next run. If you just use *eject*, the current tape will be mounted again in the next run, where it will be rejected as being still in use, ejected and the next tape requested. Using *slot next* followed by *eject* does an unnecessary mount.

Note: most changers optimize the *slot* commands to not eject the loaded tape if it is the one being requested.

AUTHOR

James da Silva, jds@amanda.org <mailto:jds@amanda.org> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amanda(8)

amtapetype

Name

amtapetype — generate a tapetype definition.

Synopsis

`amtapetype [-h] [-c] [-o] [-b` *`blocksize`*`] -e` *`estsize`* `[-f` *`tapedev`*`] [-t` *`typename`*`]`

DESCRIPTION

amtapetype generates a tapetype entry for *Amanda*.

OPTIONS

`-h` Display an help message.

`-c` Run only the hardware compression detection heuristic test and stop. This takes a few minutes only.

`-o` Overwrite the tape, even if it's an *Amanda* tape.

`-b`*`blocksize`* record block size (default: 32k)

`-e`*`estsize`* estimated tape size (No default!)

`-f`*`tapedev`* tape device name (default: $TAPE) The device to perform the test.

`-t`*`typename`* tapetype name (default: unknown-tapetype)

EXAMPLE

Generate a tapetype definition for your tape device:

% amtapetype -f /dev/nst0 -e 150G

NOTES

Hardware compression is detected by measuring the writing speed difference of the tape drive when writing an amount of compressable and uncompresseable data. It does not rely on the status bits of the tape drive or the OS parameters. If your tape drive has very large buffers or is very fast, the program could fail to detect hardware compression status reliably.

During the first pass, it writes files that are estimated to be 1% of the expected tape capacity. It gets the expected capacity from the -e command line flag, or defaults to 1 GByte. In a perfect world (which means there is zero chance of this happening with tapes :-), there would be 100 files and 100 file marks.

During the second pass, the file size is cut in half. In that same fairyland world, this means 200 files and 200 file marks.

In both passes the total amount of data written is summed as well as the number of file marks written. At the end of the second pass, quoting from the code:

* Compute the size of a filemark as the difference in data written between pass 1 and pass 2 divided by the difference in number of file marks written between pass 1 and pass 2. ... *

So if we wrote 1.0 GBytes on the first pass and 100 file marks, and 0.9 GBytes on the second pass with 200 file marks, those additional 100 file marks in the second pass took 0.1 GBytes and therefor a file mark is 0.001 GBytes (1 MByte).

Note that if the estimated capacity is wrong, the only thing that happens is a lot more (or less, but unlikely) files, and thus, file marks, get written. But the math still works out the same. The -e flag is there to keep the number of file marks down because they can be slow (since they force the drive to flush all its buffers to physical media).

All sorts of things might happen to cause the amount of data written to vary enough to generate a big file mark size guess. A little more "shoe shining" because of the additional file marks (and flushes), dirt left on the heads from the first pass of a brand new tape, the temperature/humidity changed during the multi-hour run, a different amount of data was written after the last file mark before EOT was reported, etc.

Note that the file mark size might really be zero for whatever device this is, and it was just the measured capacity variation that caused **amtapetype** to think those extra file marks in pass 2 actually took up space.

It also explains why **amtapetype** used to sometimes report a negative file mark size if the math happened to end up that way. When that happens now we just report it as zero.

SEE ALSO

amanda(8)

amtoc

Name

amtoc — generate TOC (Table Of Contents) for an *Amanda* run

Synopsis

`amtoc [-a] [-i] [-t] [ -f file ] [ -s subs ] [-w] [--] logfile`

DESCRIPTION

Amtoc generates a table of contents for an *Amanda* run. It's a perl script (if you don't have perl, install it first!).

OPTIONS

`-a` The output file name will be *label-of-the-tape*.toc in the same directory as *logfile*.

`-i` Display help about **amtoc**.

`-t` Generate the output in tabular form.

`-f file` Write the output to a file ('-' for stdout).

`-s subs` Evaluate the output file name from *subs*, with $_ set to *label-of-the-tape*. The `-a` option is equivalent to `-s 's/$_/.toc/'`.

`-w` Separate tapes with form-feeds and display blank lines before totals.

`--` Marks the last option so the next parameter is the *logfile*.

logfile (use '-' for stdin)

OUTPUT FORMAT

The standard output has five fields separated by two spaces:

```
#  Server:/partition  date  level  size[Kb]
0  daily-05:  19991005  -  -
1  cuisun15:/cuisun15/home  19991005  1  96
2  cuinfs:/export/dentiste  19991005  1  96
  ...
103  cuisg11:/  19991005  0  4139136
103  total:  -  -  16716288
```

In tabular format (-t), this would look like:

```
  #  Server:/partition            date      lev  size[Kb]
  0  daily-05:                  19991005    -        -
  1  cuisun15:/cuisun15/home    19991005    1        96
  2  cuinfs:/export/dentiste    19991005    1        96
  ...
103  cuisg11:/                   19991005    0   4139136
103  total:                     -           -  16716288
```

USAGE

The easiest way to use it is to run **amtoc** right after *amdump* in the *cron job:*

amdump daily ; logdir=‘amgetconf daily logdir‘ ; log=‘ls -1t $logdir/log.*.[0-9] — head -1‘ ; amtoc -a $

which will generate /usr/local/etc/amanda//daily/*tape_label*.toc. You may also want to call **amtoc** after an *amflush*.

SEE ALSO

amanda(8), amdump(8), amflush(8), amgetconf(8), cron, perl

AUTHOR

Nicolas Mayencourt Nicolas.Mayencourt@cui.unige.ch <mailto:Nicolas.Mayencourt@cui.unige.ch>, University of Geneva/Switzerland : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

amverify

Name

amverify — check an Amanda tape for errors

Synopsis

amverify *config* [*slot* [*runtapes*]]

DESCRIPTION

Amverify reads an Amanda format tape and makes sure each backup image can be processed by *amrestore* and, if possible, the appropriate restore program (e.g. *tar*).

Amverify runs *amrestore* on each file of the tape and pipes the output to a restore program (if available) with an option to create a catalogue of the backup. The catalogue itself is discarded. Only the success or failure of the operation itself is reported.

If the backup image cannot be processed by the restore program, e.g. if it was written on a different operating system, the image is sent through *dd* to /dev/null. This still determines if the tape is readable, but does not do any internal consistency check on the image.

If *config* is set up to use a tape changer, the *slot* argument may be used to choose the first tape to process. Otherwise, the *current* slot is used.

The *runtapes* configuration parameter determines how many tapes are processed unless it is specified on the command line.

See the amanda(8) man page for more details about Amanda.

AUTHOR

Axel Zinser fifi@icem.de <mailto:fifi@icem.de> : Original text

Stefan G. Weichinger, sgw@amanda.org <mailto:sgw@amanda.org>, maintainer of the *Amanda*-documentation: XML-conversion

SEE ALSO

amrestore(8), amanda(8), amverifyrun(8)

amverifyrun

Name

amverifyrun — check the tapes written by the last *Amanda* run

Synopsis

`amverifyrun` *`config`*

DESCRIPTION

Amverifyrun read the log from the last *Amanda* run to find the slot of the first tape used and the number of tapes used. It call *amverify* with these argument.

SEE ALSO

amanda(8), amverify(8)

NOTE

Refer to <`http://www.amanda.org/docs/manpages.html`> for the current version of this document.

Chapter 37

WEB RESSOURCES

See some original *Amanda*-papers by James da Silva and Olafur Gundmundsson here:

- The presentation of *Amanda*:

  Postscript: <http://www.amanda.org/docs/lisa-vii.ps>

  PDF: <http://www.amanda.org/docs/lisa-vii.pdf>

- A document about the performance of *Amanda*:

  Postscript: <http://www.amanda.org/docs/usenix92.ps>

  PDF: <http://www.amanda.org/docs/usenix92.pdf>

NOTE

Refer to <http://www.amanda.org/docs/links.html> for the current version of this document.

SUBJECT INDEX