Good stewardship: the preservation of data in the face of adversity

Part of the first imperative of system administration is preservation of the important data that you are the administrator of.

It means that you must have redundancy in terms of the data. The easy way to do this is to have multiple copies of the data that you are responsible for. The prices of non-volatile media falling (you can buy a terabyte harddrive

for around $200 currently), and the density is such that you can comfortably carry a terabyte drive in one hand.

Two key concepts for data recovery:

- RTO – Recovery time objective. How long will it take me to recover whatever data that I am trying to recover?
- RPO – Recovery point objective. What time points will I be able to recover back to?

A second pair concepts are those of "archival" material versus ordinary backups. With archival material, you are

creating a historical snapshot. For instance, here in the computer science department, we create archival backups at the end of each semester as a snapshot.

Backup tapes

Traditionally, data redundancy was achieved via tape, and was called a backup tape.

Backup tapes traditionally were made generally once per day, giving you an RPO generally of sometime in the early morning. Usually the scheme involved making a "full" backup only occasionally (perhaps once a week, once a month, or even (page 180 of USAH) once a year!) with one or more levels of "incremental" backups which

recorded only changed files and directories.

These backups were made on a per machine basis (though today such backups are generally made with a backup server which uses ip to transfer the data from the client machine to the backup server.)

The upsides of such traditional schemes are:

☞ It is an efficient use of tape.

☞ It is efficient with respect to the amount of time spent writing to tape.

☞ Tapes are generally easy to mark up and store off-site.

The downsides of such traditional schemes are:

☞ It is slow to recover from, giving you a long RTO; if you have a full with two levels of incrementals, you have to restore. from the full and both levels of incrementals

☞ If tapes are unencrypted (as they generally are), then their loss can be a serious breach of security. This has happened numerous times.

☞ Tape failures do occur, as we will discuss in a moment.

Such a scheme is intolerant of tape failures. If you have full backup that fails, you generally will miss your RPO objectives.

What's wrong with tapes?

☞ Tapes historically have suffered from many problems, from feeding problems with 9 track (1/2 inch) tapes to generally high rates of failure in reused tape media, often with large implications.

☞ A single read failure early on in a serially accessed medium such as magnetic tape (as opposed to a random access device such as a disk drive) often means that the

remainder of a tape cannot even be used — and with incremental tape schemes, a single tape failure can cascade into days (or even weeks) worth of data loss.

☞ Problems with tape

- ➳ simple media failure (particularly reused media)
- ➳ complex and slow restoration for large recovery (especially in disaster recovery situations);
- ➳ slowness relative to other data movement and to the growth of data, and thus you see backups that start to span the amount of all available time (also known

as the backup window problem)

- enormous technological turnover requiring large amounts of media refreshment as we advance in technology;
- archival data on tape from various archiving vendors is usually stored in formats that are not close to native file systems, and these formats usually require additional software to recover from; this software must be available at recovery sites to even begin recovery from a disaster.
- Tape is relatively expensive compared to disk (disk

prices are running around 20 cents per gigabyte; tape is about 10 cents per (uncompressed) gigabyte; tapes can be used for less hours than disk drives).

➳ Most tape solutions do not have incremental growth paths; every few years, you usually end up having to buy a complete new solution, and this has to be done on regular basis simply to keep up with tape technology. For instance, from the 1980s until now, we have gone in this department used at various times 1/2 inch tape, 1/4 inch cartridge tape, 8mm Exabyte, 4mm Exabyte, AIT, DLT-7000, DLT IV,

LTO-2, and LTO-3. However, over the same span, for disk technology we only saw a few connectivity types: SCSI (which we still use), IDE/ATA, and SATA, and all of which settled on logical block assignment long ago, thus eliminating the vast majority of compatibility problems with the old Cylinder/Head/Sector schemes.

- Single points of failure are rife: the tape drive itself, the software processes themselves, the network between the two.
- More bottlenecks, especially as the network as a shared resource.

In the Unix world, the pair of programs **dump** and **restore** have been our mainstays for doing backups. On a machine with a locally attached tape drive, these still make a lot of sense. Another program that has been used is **tar** (short for **t**ape **ar**chive) which is also used for creating general aggregates of files and directories. Another related program is **cpio**. All of these can also create a normal file instead of a tape. **dump** and **tar** allow you to also create compressed images, although with today's tape compression at the hardware level, this is probably not of as much interest as it was in the past.

While all of these allow use to make backups either locally or remotely, using **dump** and **restore** allows one to make incremental backups. (I don't recommend using **tar**'s incremental scheme. Also, many default versions of **tar** have in the past only allowed up to 100 characters in a pathname.)

restore also you to do interactive recovery of data from tape or dumpfiles.

USAH recommends using a single machine for tape backups if feasible by using the network to transfer data

to the backup server. I have seen that advice combined with the idea of local backups (in order to have more redundancy) where tapes were made **both** on the local machine and remotely. (In that case, the remote tape backups were also highly available. The backup server had a tape silo, and you could literally pull any recent file for any machine that was backed up from tapes that were in the tape silo.)

Label your tapes

One of the worst stories that I have ever heard is the story of going into a server room and finding on top of each tape drive 4 tapes labelled "A", "B", "D", and "E" – and no key to explain what the contents of each might be (although it would be a very good guess that "C" is currently in the drive.)

If you find that you are doing such schemes where tapes do not have labels that are fully descriptive of the

contents, while this is okay if this is because you are using a tape silo that creates tape sets where the contents are essentially stored in an effectively random access fashion, this is probably not okay if it is because you are simply recycling the tapes so frequently that it is inconvenient to relabel them. Tape reuse is the number cause of tape failure.

Tape management

While with some tape silos it is impossible not to span tapes in what are called "tape sets", it is generally not a good idea to span tapes if you are doing individual backups. In addition to the simple hassle of changing tapes when making them, both the tasks of labelling and recovery from multiple tapes is not much fun.

One part of recovery planning has to include that of disaster scenarios. Generally, tapes are a good backup

mechanism for such scenarios since they are easy to transport offsite — though that itself engenders risk if the tapes are not encrypted!

Update: Lost backup tapes prompt IT changes at NY bank

Additional measures, including encryption, will protect tapes transported to off-site facilities

By Brian Fonseca

June 2, 2008 (Computerworld) Bank of New York Mellon Corp. late last week said it has launched a new policy to encrypt data held on storage devices and to limit the amount of confidential client data stored on tape drives. The policy was launched after unencrypted backup data tapes were lost twice by third-party couriers this year.

The bank would not disclose its past storage policies.

The company announced the new policy just days after disclosing that one of 10 boxes of storage tapes being delivered to an off-site facility by storage firm Archive America was

lost in February and that another courier, which it did not identify, lost a storage tape during transit in April.

Combined, the two data breaches exposed sensitive information of more than 4.5 million people and 747 companies, according to BNY Mellon officials.

In the February incident, the tapes were being transported from the bank's Mellon Shareowner Services facility in Jersey City, N.J. Those missing backup tapes include names, birth dates, Social Security numbers, and other information from customers of BNY Mellon and the People's United Bank in Bridgeport, Conn.

The tape lost in April was in transit from BNY Mellon's Working Capital Solutions operations in Philadelphia to a branch office in Pittsburgh. That backup tape included images of scanned checks and other documents relating to payments made by BNY Mellon clients, said company officials.

The company late last week also announced that it will provide two years of free credit monitoring, credit-freeze benefits and a $25,000 identity theft insurance policy to those affected by the missing tapes.

"We deeply regret that this occurred and sincerely apologize to all of those impacted," said Todd Gibbons, chief risk officer at BNY Mellon, in a statement. Gibbons said there

is no indication that data on the missing tapes has been misused or inappropriately accessed.

To bolster its security controls, the bank said it will now require that any confidential data written on tapes or CDs for transport must be encrypted or transported with undisclosed additional data protections. Further, when "technically feasible," the bank will demand that encrypted confidential data be delivered to off-site facilities electronically, noted Gibbons.

BNY Mellon has ended its relationship with Archive America and is cooperating with law enforcement agencies and state and federal officials in the investigation of the incident. A spokesman for the financial institution refused to comment on any details surrounding the circumstances of how the tapes disappeared. Archive America officials declined comment.

Connecticut Attorney General Richard Blumenthal said that the missing BNY Mellon computer tapes have put the personal identities of 497,333 state residents at risk. In a statement released late last week, Blumenthal and the state's Department of Consumer Protection listed 25 companies with Connecticut-based customers affected by the breach. The list includes People's United Financial Inc., John Hancock Financial Services Inc., The Walt Disney Co. and TD Bank Financial Group.

Blumenthal said he is still waiting for answers from the bank about how the data

> breach occurred, who is responsible for the crime and why BNY waited months to notify customers about the incident. "We haven't completed our investigation and there are still some important questions that have to be answered," he remarked.

(This appeared in *Computer World* at `http://computerworld.com/action/article.do?command`

on June 2, 2008.)

Iron Mountain Loses More Tapes

Backup tapes from City National Bank were lost in April, but there's no evidence the data has been compromised, the bank says.

By Steven Marlin
InformationWeek

Jul 8, 2005 03:00 PM

City National Bank has become the second company in two months to experience a loss of backup tapes in transit by Iron Mountain Inc. The Los Angeles-based bank disclosed Thursday that two tapes containing sensitive data, including Social Security numbers, account numbers, and other customer information, were lost during transport to a secure storage facility.

The bank said the data was formatted to make the tapes difficult to read without highly specialized skills, but declines to say if they were encrypted. It said there's no evidence that data on the tapes has been compromised or misused.

...

(This appeared in *Information Week* at http://informationweek.com/story/showArticle.jhtml?articleI

back in July of 2005.)

Stolen University of Utah medical files recovered

By PAUL FOY 07.03.08, 10:58 AM ET
SALT LAKE CITY

Three people involved in the theft of millions of medical records didn't have the ability, knowledge or equipment to decode personal information from the backup tapes, authorities said.

Salt Lake County sheriff's investigators said Wednesday a caller led detectives to the University of Utah billing records that were burglarized last month from a courier's vehicle.

...

The backup tapes contained various combinations of social security and driver's license numbers, birth dates, doctors' names, insurance providers and medical procedure codes on about 1.5 million patients who visited the hospital over the past 16 years, a hospital executive said. The hospital originally said the records contained confidential information on 2.2 million patients, but later corrected the figure.

Detectives said the three people didn't have the specialized equipment needed to run or decode the encrypted tapes.

...

The courier faces no charges and was a victim of the crime, he said.

The courier was supposed to drive the tapes to a storage vault burrowed into the granite of Little Cottonwood Canyon. He went home instead.

Detectives believe the car was a random target of burglary and that at first, the thief didn't realize what was inside a metal canister or on the backup tapes.

With the tapes in safe hands at an FBI lab, Entwistle said there was no merit to a proposed class action lawsuit filed Monday against the hospital on behalf of patients whose information may have been at risk.

> ...
>
> Many companies use a secure Internet line to electronically back up business records at a database center, and Entwistle said the hospital is looking at ways to eliminate the risk of deploying couriers to move information.

(This appeared in *Forbes.com* at http://www.forbes.com/feeds/ap/2008/07/03/ap5181656.html on July 3rd, 2008.)

Tape management continued

While I have seen two sites that had adequate on-site vaulting, that is enormously expensive compared to simply storing them offsite, such as with Iron Mountain.

When you are making backups, it is advisable to try to have quiescent filesytems. Back in the 1980s, I used to take all of one facility's machines down to single-user state to make backups on Friday evening.

Some people use three-way disk mirrors, and then break

the second mirror off to ensure that the filesystem is quiescent.

However, it is more common to make backups on relatively live machines, and generally there aren't that many obstacles — except for databases. If you need to back up a database on a live filesystem, the simplest thing to do is probably to create a snapshot file (such as with **mysqldump** or **pg_dump**), and then – **once you are certain that file has been made** – make your backup tape.

Tape management, continued

Checking your tapes: This is so important that in some regulated industries it is mandatory.

It is best practice to check your tapes on a regular basis. Either you or a user should periodically choose a file or directory to attempt to recover, and if at all possible, then try to recover that on a different tape drive than the original one. (Some tape units come go out of alignment; while they may be able to read their own tapes, other

drives may have trouble with those tapes.)

Refreshing media: Tape technology has very rapid turnover, and technical obsolence is a constant problem. If at all possible, you should try to move your old tapes when you introduce a new tape technology. This is expensive (especially if you have a considerable collection of archives), but paying a recovery service is even more expensive if your tapes fall too far behind the technology curve. Just look at pp. 170-175 in USAH and you can see the technology curve is all too apparent. Only DLT is in all that common usage, and 12 MB/s is no longer

"blindingly fast".

Incremental schedules

One possibility is to simply do a full dump each day of every machine. That provides a lot of redundancy, and generally will keep you close to your RPO. This is quite common in smaller establishments, or in ones where data preservation is an especially high priority.

However, some establishments have lesser criteria for data preservation. In those, it is common to schemes such as that on page 180 of USAH:

☞ Do a level 9 every day.

☞ Once a week, do a level 5.

☞ Once a month, do a level 0.

(I haven't seen the suggestion on page 180 implemented that the monthlies be actually level 3s, and reserve level 0s for once per year. That's putting an awful lot of faith in that level 0 — if it fails, you could be looking at going back to level 0 from two years prior!)

Recovery from tape

When you are trying to recover a single file or directory, the first order of business is to find the right tape. If a user comes to you about a file that he has not thought about in a while, you will have to try to ascertain which tapes to go to find the file. Once you have found an appropriate tape, if you are using **restore**, you can run it in interactive mode instead of trying to extract the whole tape to disk.

If instead you want to recover an entire filesystem, you need to make sure that you aren't simply compounding whatever problem you had to begin with. If, for instance, a disk drive or controller is acting flaky, simply recovering back to the same unit isn't going to do you much good.

For instance, as USAH has it on page 183, if you are using the scheme of levels 0, 3, 5, and 9 as mentioned above, you need to first recover from the most recent level 0 tape, and then from the most recent level 3, then the most recent level 5, and finally, the most recent level 9.

Another reason to make backups which we haven't touched on is when you are doing system upgrades. System upgrades have the distinct possibility of rendering your system unuseable. In that case, convenience of recovery is a strong goal, and backing up to disk rather than tape should be a strong consideration.

Tape management – the mt program (USAH pp. 186-187)

The program **mt** lets one manipulate tapes that contain more than a single "EOF" written to it.

The main commands for **mt** are

☞ **rew** – rewind the tape

☞ **offline** – rewind and unload the tape (USAH has this

as "offl", which I don't believe that I have ever seen that name used.)

- **status** – gives the current status of the tape (tape loaded, where the tape pointer is, etc.)

- **erase** – erase the tape

- **retension** – retension the tape by first rewinding it and then winding back out to the current location (not an option that you ever hope to use! It's generally one of the final resorts on reading a balky tape.)

- **fsf** – skip forward over items on the tape
- **bsf** – skip back over items on the tape

Getting organized: scripts and Amanda

Commonly, you will find that backups are done by $ad\ hoc$ scripts. However, there is a comprehensive suite of backup control called "AMANDA" ("Advanced Maryland Automatic Network Disk Archiver").

As described on pages 187–201 in USAH, AMANDA has some very nice features and capabilities. One of the nicest features is the header that it writes to each tape, and its systematic organization of your entire backup structure.

AMANDA is based on the idea of a centralized backup server; i.e., it expects to be the repository of data from a number of computers in a network.

An example amand.conf from CentOS 3.5:

```
#
# amanda.conf - sample Amanda configuration file.  This started off life as
#               the actual config file in use at CS.UMD.EDU.
#
# If your configuration is called, say, "csd", then this file normally goes
# in /etc/amanda/csd/amanda.conf.
#

org "DailySet1"                 # your organization name for reports
mailto "amanda"                 # space separated list of operators at your site
dumpuser "amanda"         # the user to run dumps under
```

```
inparallel 4                 # maximum dumpers that will run in parallel
netusage  600 Kbps          # maximum net bandwidth for Amanda, in KB per sec

dumpcycle 4 weeks          # the number of days in the normal dump cycle
runspercycle 4 weeks     # the number of amdump runs in dumpcycle days
tapecycle 25 tapes         # the number of tapes in rotation
                         # 4 weeks (dumpcycle) times 5 tapes per week (just
                         # the weekdays) plus a few to handle errors that
                         # need amflush and so we do not overwrite the full
                         # backups performed at the beginning of the previous
                         # cycle
### ### ###
# WARNING: don't use 'inf' for tapecycle, it's broken!
### ### ###

bumpsize 20 Mb                  # minimum savings (threshold) to bump level 1 -> 2
bumpdays 1                 # minimum days at each level
bumpmult 4                 # threshold = bumpsize * bumpmult^(level-1)
```

```
etimeout 300                    # number of seconds per filesystem for estimates.
#etimeout -600                   # total number of seconds for estimates.
# a positive number will be multiplied by the number of filesystems on
# each host; a negative number will be taken as an absolute total time-out.
# The default is 5 minutes per filesystem.


# Specify tape device and/or tape changer.  If you don't have a tape
# changer, and you don't want to use more than one tape per run of
# amdump, just comment out the definition of tpchanger.

# Some tape changers require tapedev to be defined; others will use
# their own tape device selection mechanism.  Some use a separate tape
# changer device (changerdev), others will simply ignore this
# parameter.  Some rely on a configuration file (changerfile) to
# obtain more information about tape devices, number of slots, etc;
# others just need to store some data in files, whose names will start
# with changerfile.  For more information about individual tape
# changers, read docs/TAPE.CHANGERS.
```

```
# At most one changerfile entry must be defined; select the most
# appropriate one for your configuration.  If you select man-changer,
# keep the first one; if you decide not to use a tape changer, you may
# comment them all out.

runtapes 1                 # number of tapes to be used in a single run of amdump
#tpchanger "chg-manual"         # the tape-changer glue script
#tapedev "/dev/null"          # the no-rewind tape device to be used
#rawtapedev "/dev/null"         # the raw device to be used (ftape only)
#changerfile "/var/lib/amanda/DailySet1/changer"
#changerfile "/var/lib/amanda/DailySet1/changer-status"
#changerfile "/etc/amanda/DailySet1/changer.conf"
#changerdev "/dev/null"

tapetype HP-DAT                # what kind of tape it is (see tapetypes below)
labelstr "^DailySet1[0-9][0-9]*$"        # label constraint regex: all tapes must ma

# Specify holding disks.  These are used as a temporary staging area for
# dumps before they are written to tape and are recommended for most sites.
# The advantages include: tape drive is more likely to operate in streaming
```

```
# mode (which reduces tape and drive wear, reduces total dump time); multiple
# dumps can be done in parallel (which can dramatically reduce total dump time.
# The main disadvantage is that dumps on the holding disk need to be flushed
# (with amflush) to tape after an operating system crash or a tape failure.
# If no holding disks are specified then all dumps will be written directly
# to tape.  If a dump is too big to fit on the holding disk than it will be
# written directly to tape.  If more than one holding disk is specified then
# they will all be used round-robin.

holdingdisk hd1 {
    comment "main holding disk"
    directory "/var/tmp"        # where the holding disk is
    use 290 Mb                  # how much space can we use on it
                        # a negative value mean:
                        #        use all space except that value
#    chunksize 2 Gb          # size of chunk if you want big dump to be
                        # dumped on multiple files on holding disks
                        #  N Kb/Mb/Gb split disks in chunks of size N
                        #  0          split disks in INT_MAX/1024 Kb chunks
                        # -N Kb/Mb/Gb dont split, dump larger
```

```
                        #               filesystems directly to tape
                        #               (example: -2 Gb)
    }
#holdingdisk hd2 {
#    directory "/dumps2/amanda"
#    use 1000 Mb
#    }
#holdingdisk hd3 {
#    directory "/mnt/disk4"
#    use 1000 Mb
#    }


# If amanda cannot find a tape on which to store backups, it will run
# as many backups as it can to the holding disks.  In order to save
# space for unattended backups, by default, amanda will only perform
# incremental backups in this case, i.e., it will reserve 100% of the
# holding disk space for the so-called degraded mode backups.
# However, if you specify a different value for the 'reserve'
# parameter, amanda will not degrade backups if they will fit in the
```

```
# non-reserved portion of the holding disk.

# reserve 30 # percent


# This means save at least 30% of the holding disk space for degraded
# mode backups.

# Amanda needs a few Mb of diskspace for the log and debug files,
# as well as a database.  This stuff can grow large, so the conf directory
# isn't usually appropriate.  Some sites use /usr/local/var and some /usr/adm.
# Create an amanda directory under there.  You need a separate infofile and
# logdir for each configuration, so create subdirectories for each conf and
# put the files there.  Specify the locations below.

infofile "/var/lib/amanda/DailySet1/curinfo"         # database filename
logdir   "/var/lib/amanda/DailySet1"                 # log directory
indexdir "/var/lib/amanda/DailySet1/index"         # index directory
#tapelist "/var/lib/amanda/DailySet1/tapelist"         # list of used tapes
# tapelist is stored, by default, in the directory that contains amanda.conf
```

```
# tapetypes

# Define the type of tape you use here, and use it in "tapetype"
# above.  Some typical types of tapes are included here.  The tapetype
# tells amanda how many MB will fit on the tape, how big the filemarks
# are, and how fast the tape device is.

# A filemark is the amount of wasted space every time a tape section
# ends.  If you run ‘make tapetype’ in tape-src, you’ll get a program
# that generates tapetype entries, but it is slow as hell, use it only
# if you really must and, if you do, make sure you post the data to
# the amanda mailing list, so that others can use what you found out
# by searching the archives.

# For completeness Amanda should calculate the inter-record gaps too,
# but it doesn’t.  For EXABYTE and DAT tapes this is ok.  Anyone using
# 9 tracks for amanda and need IRG calculations?  Drop me a note if
# so.
```

```
# If you want amanda to print postscript paper tape labels
# add a line after the comment in the tapetype of the form
#    lbl-templ "/path/to/postscript/template/label.ps"

# if you want the label to go to a printer other than the default
# for your system, you can also add a line above for a different
# printer. (i usually add that line after the dumpuser specification)

# dumpuser "operator"     # the user to run dumps under
# printer "mypostscript"  # printer to print paper label on

# here is an example of my definition for an EXB-8500

# define tapetype EXB-8500 {
# ...
#     lbl-templ "/usr/local/amanda/config/lbl.exabyte.ps"
# }
```

```
define tapetype QIC-60 {
    comment "Archive Viper"
    length 60 mbytes
    filemark 100 kbytes                # don't know a better value
    speed 100 kbytes                 # dito
}

define tapetype DEC-DLT2000 {
    comment "DEC Differential Digital Linear Tape 2000"
    length 15000 mbytes
    filemark 8 kbytes
    speed 1250 kbytes
}

# goluboff@butch.Colorado.EDU
# in amanda-users (Thu Dec 26 01:55:38 MEZ 1996)
define tapetype DLT {
    comment "DLT tape drives"
    length 20000 mbytes                # 20 Gig tapes
    filemark 2000 kbytes         # I don't know what this means
```

```
    speed 1536 kbytes                # 1.5 Mb/s
}

define tapetype SURESTORE-1200E {
    comment "HP AutoLoader"
    length 3900 mbytes
    filemark 100 kbytes
    speed 500 kbytes
}

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

define tapetype EXB-8200 {
    comment "Exabyte EXB-8200 drive on decent machine"
    length 2200 mbytes
```

```
    filemark 2130 kbytes
    speed 240 kbytes
}

define tapetype HP-DAT {
    comment "DAT tape drives"
    # data provided by Rob Browning <rlb@cs.utexas.edu>
    length 1930 mbytes
    filemark 111 kbytes
    speed 468 kbytes
}

define tapetype DAT {
    comment "DAT tape drives"
    length 1000 mbytes                # these numbers are not accurate
    filemark 100 kbytes                # but you get the idea
    speed 100 kbytes
}

define tapetype MIMSY-MEGATAPE {
```

```
    comment "Megatape (Exabyte based) drive through Emulex on Vax 8600"
    length 2200 mbytes
    filemark 2130 kbytes
    speed 170 kbytes                # limited by the Emulex bus interface, ugh
}


# dumptypes
#
# These are referred to by the disklist file.  The dumptype specifies
# certain parameters for dumping including:
#   auth        - authentication scheme to use between server and client.
#                 Valid values are "bsd" and "krb4".  Default: [auth bsd]
#   comment       - just a comment string
#   comprate       - set default compression rate.  Should be followed by one or
#                 two numbers, optionally separated by a comma.  The 1st is
#                 the full compression rate; the 2nd is the incremental rate.
#                 If the second is omitted, it is assumed equal to the first.
#                 The numbers represent the amount of the original file the
#                 compressed file is expected to take up.
```

```
#                   Default: [comprate 0.50, 0.50]
#   compress         - specify compression of the backed up data.  Valid values are:
#                   "none"         - don't compress the dump output.
#                   "client best" - compress on the client using the best (and
#                                    probably slowest) algorithm.
#                   "client fast" - compress on the client using fast algorithm.
#                   "server best" - compress on the tape host using the best (and
#                                    probably slowest) algorithm.
#                   "server fast" - compress on the tape host using a fast
#                                    algorithm.  This may be useful when a fast
#                                    tape host is backing up slow clients.
#                   Default: [compress client fast]
#   dumpcycle         - set the number of days in the dump cycle, ie, set how often a
#                   full dump should be performed.  Default: from DUMPCYCLE above
#   exclude         - specify files and directories to be excluded from the dump.
#                   Useful with gnutar only; silently ignored by dump and samba.
#                   Valid values are:
#                   "pattern"       - a shell glob pattern defining which files
#                                     to exclude.
#                                     gnutar gets --exclude="pattern"
```

```
#                   list "filename" - a file (on the client!) containing patterns
#                                     re's (1 per line) defining which files to
#                                     exclude.
#                                     gnutar gets --exclude-from="filename"
#                   Note that the 'full pathname' of a file within its
#                   filesystem starts with './', because of the way amanda runs
#                   gnutar: 'tar -C $mountpoint -cf - --lots-of-options .' (note
#                   the final dot!)  Thus, if you're backing up '/usr' with a
#                   diskfile entry like ''host /usr gnutar-root', but you don't
#                   want to backup /usr/tmp, your exclude list should contain
#                   the pattern './tmp', as this is relative to the '/usr' above.
#                   Please refer to the man-page of gnutar for more information.
#                   Default: include all files
#   holdingdisk         - should the holding disk be used for this dump.  Useful for
#                   dumping the holding disk itself.  Default: [holdingdisk yes]
#   ignore         - do not back this filesystem up.  Useful for sharing a single
#                   disklist in several configurations.
#   index         - keep an index of the files backed up.  Default: [index no]
#   kencrypt         - encrypt the data stream between the client and server.
#                   Default: [kencrypt no]
```

```
#   maxdumps         - max number of concurrent dumps to run on the client.
#                   Default: [maxdumps 1]
#   priority         - priority level of the dump.  Valid levels are "low", "medium"
#                   or "high".  These are really only used when Amanda has no
#                   tape to write to because of some error.  In that "degraded
#                   mode", as many incrementals as will fit on the holding disk
#                   are done, higher priority first, to insure the important
#                   disks are at least dumped.  Default: [priority medium]
#   program         - specify the dump system to use.  Valid values are "DUMP" and
#                   "GNUTAR".  Default: [program "DUMP"].
#   record          - record the dump in /etc/dumpdates.  Default: [record yes]
#   skip-full         - skip the disk when a level 0 is due, to allow full backups
#                   outside Amanda, eg when the machine is in single-user mode.
#   skip-incr         - skip the disk when the level 0 is NOT due.  This is used in
#                   archive configurations, where only full dumps are done and
#                   the tapes saved.
#   starttime         - delay the start of the dump?  Default: no delay
#   strategy         - set the dump strategy.  Valid strategies are currently:
#                   "standard" - the standard one.
#                   "nofull"   - do level 1 dumps every time.  This can be used,
```

```
#                                 for example, for small root filesystems that
#                                 only change slightly relative to a site-wide
#                                 prototype.  Amanda then backs up just the
#                                 changes.
#                   "noinc"     - do level 0 dumps every time.
#                                 Unfortunately, this is not currently
#                                 implemented.  Use 'dumpcycle 0'
#                                 instead.
#                   "skip"      - skip all dumps.  Useful for sharing a single
#                                 disklist in several configurations.
#                   Default: [strategy standard]
#
# Note that you may specify previously defined dumptypes as a shorthand way
# of defining parameters.

define dumptype global {
    comment "Global definitions"
    # This is quite useful for setting global parameters, so you don't have
    # to type them everywhere.  All dumptype definitions in this sample file
    # do include these definitions, either directly or indirectly.
```

```
    # There's nothing special about the name 'global'; if you create any
    # dumptype that does not contain the word 'global' or the name of any
    # other dumptype that contains it, these definitions won't apply.
    # Note that these definitions may be overridden in other
    # dumptypes, if the redefinitions appear *after* the 'global'
    # dumptype name.
    # You may want to use this for globally enabling or disabling
    # indexing, recording, etc.  Some examples:
    # index yes
    # record no
}

define dumptype always-full {
    global
    comment "Full dump of this filesystem always"
    compress none
    priority high
    dumpcycle 0
}
```

```
define dumptype root-tar {
    global
    program "GNUTAR"
    comment "root partitions dumped with tar"
    compress none
    index
    exclude list "/usr/local/lib/amanda/exclude.gtar"
    priority low
}

define dumptype user-tar {
    root-tar
    comment "user partitions dumped with tar"
    priority medium
}

define dumptype high-tar {
    root-tar
    comment "partitions dumped with tar"
    priority high
```

```
}

define dumptype comp-root-tar {
    root-tar
    comment "Root partitions with compression"
    compress client fast
}

define dumptype comp-user-tar {
    user-tar
    compress client fast
}

define dumptype holding-disk {
    global
    comment "The master-host holding disk itself"
    holdingdisk no # do not use the holding disk
    priority medium
}
```

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

define dumptype nocomp-user {
    comp-user
    comment "Non-root partitions on slow machines"
    compress none
}

define dumptype comp-root {
    global
    comment "Root partitions with compression"
    compress client fast
    priority low
}
```

```
define dumptype nocomp-root {
    comp-root
    comment "Root partitions without compression"
    compress none
}

define dumptype comp-high {
    global
    comment "very important partitions on fast machines"
    compress client best
    priority high
}

define dumptype nocomp-high {
    comp-high
    comment "very important partitions on slow machines"
    compress none
}

define dumptype nocomp-test {
```

```
    global
    comment "test dump without compression, no /etc/dumpdates recording"
    compress none
    record no
    priority medium
}

define dumptype comp-test {
    nocomp-test
    comment "test dump with compression, no /etc/dumpdates recording"
    compress client fast
}

# network interfaces
#
# These are referred to by the disklist file.  They define the attributes
# of the network interface that the remote machine is accessed through.
# Notes: - netusage above defines the attributes that are used when the
#          disklist entry doesn't specify otherwise.
#        - the values below are only samples.
```

```
#        - specifying an interface does not force the traffic to pass
#          through that interface.  Your OS routing tables do that.  This
#          is just a mechanism to stop Amanda trashing your network.
# Attributes are:
#        use                - bandwidth above which amanda won't start
#                           backups using this interface.  Note that if
#                           a single backup will take more than that,
#                           amanda won't try to make it run slower!

define interface local {
    comment "a local disk"
    use 1000 kbps
}

define interface eth0 {
    comment "10 Mbps ethernet"
    use 400 kbps
}

# You may include other amanda configuration files, so you can share
```

```
# dumptypes, tapetypes and interface definitions among several
# configurations.

#includefile "/usr/local/amanda.conf.main"
```