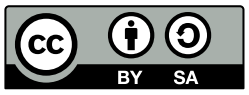
The Leading Open Source Backup Solution

Bacula® Problem Resolution Guide

Kern Sibbald

May 2, 2023

This manual documents Bacula Community Edition 13.0.3 (02 May 2023)



2023-05-02

version 13.0.3

Bacula Community

Contents

List of Figures

Chapter 1

Bacula Frequently Asked Questions

These are questions that have been submitted over time by the Bacula users. The following FAQ is very useful, but it is not always up to date with newer information, so after reading it, if you don't find what you want, you might try the Bacula wiki maintained by Frank Sweetser, which contains more than just a FAQ: wiki.bacula.org or go directly to the FAQ at: wiki.bacula.org/doku.php?id=faq.

Please also see the bugs section of this document for a list of known bugs and solutions.

What is **Bacula**? **Bacula** is a network backup and restore program.

Does Bacula support Windows? Yes, Bacula compiles and runs on Windows machines (Win98, WinMe, WinXP, WinNT, Win2003, and Win2000). We provide a binary version of the Client (`bacula-fd`), but have not tested the Director nor the Storage daemon. Note, Win95 is no longer supported because it doesn't have the `GetFileAttributesExA` API call.

What language is Bacula written in? It is written in `C++`, but it is mostly `C` code using only a limited set of the `C++` extensions over `C`. Thus Bacula is completely compiled using the `C++` compiler. There are several modules, including the Win32 interface, that are written using the object oriented `C++` features. Over time, we are slowly adding a larger subset of `C++`.

On what machines does Bacula run? **Bacula** builds and executes on Red Hat Linux (versions RH7.1-RHEL 4.0, Fedora, SuSE, Gentoo, Debian, Mandriva, ...), FreeBSD, Solaris, Alpha, SGI (client), NetBSD, OpenBSD, Mac OS X (client), and Win32.

Bacula has been my only backup tool for over seven years backing up 8 machines nightly (6 Linux boxes running SuSE, previously Red Hat and Fedora, a WinXP machine, and a WinNT machine).

Is Bacula Stable? Yes, it is remarkably stable, but remember, there are still a lot of unimplemented or partially implemented features. With a program of this size (150,000+ lines of `C++` code not including the SQL programs) there are bound to be bugs. The current test environment (a twisted pair local network and a HP DLT backup tape) is not exactly ideal, so additional testing on other sites is necessary. The File daemon has never crashed – running months at a time with no intervention. The Storage daemon is remarkably stable with most of the problems arising during labeling or switching tapes. Storage daemon crashes are rare but running multiple drives and simultaneous jobs sometimes (rarely) problems. The Director, given the multitude of functions it fulfills is also relatively stable. In a production environment, it rarely if ever crashes. Of the three daemons, the Director is the most prone to having problems. Still, it frequently runs several months with no problems.

There are a number of reasons for this stability.

1. The program is constantly checking the chain of allocated memory buffers to ensure that no overruns have occurred.
2. All memory leaks (orphaned buffers) are reported each time the program terminates.
3. Any signal (segmentation fault, ...) generates a traceback that is emailed to the developer. This permits quick resolution of bugs even if they only show up rarely in a production system.
4. There is a reasonably comprehensive set of regression tests that avoids re-creating the most common errors in new versions of Bacula.

I'm Getting Authorization Errors. What is Going On? For security reasons, Bacula requires that both the File daemon and the Storage daemon know the name of the Director as well as its password. As a consequence, if you change the Director's name or password, you must make the corresponding change in the Storage daemon's and in the File daemon's configuration files.

During the authorization process, the Storage daemon and File daemon also require that the Director authenticates itself, so both ends require the other to have the correct name and password.

If you have edited the conf files and modified any name or any password, and you are getting authentication errors, then your best bet is to go back to the original conf files generated by the Bacula installation process. Make only the absolutely necessary modifications to these files – e.g. add the correct email address. Then follow the instructions in the **Running Bacula** chapter (chapter 20 page 189) of the Bacula Community Edition Main manual. You will run a backup to disk and a restore. Only when that works, should you begin customization of the conf files.

Another reason that you can get authentication errors is if you are running Multiple Concurrent Jobs in the Director, but you have not set them in the File daemon or the Storage daemon. Once you reach their limit, they will reject the connection producing authentication (or connection) errors.

If you are having problems connecting to a Windows machine that previously worked, you might try restarting the Bacula service since Windows frequently encounters networking connection problems.

Some users report that authentication fails if there is not a proper reverse DNS lookup entry for the machine. This seems to be a requirement of gethostbyname(), which is what Bacula uses to translate names into IP addresses. If you cannot add a reverse DNS entry, or you don't know how to do so, you can avoid the problem by specifying an IP address rather than a machine name in the appropriate Bacula conf file.

Here is a picture that indicates what names/passwords in which files/Resources must match up:

In the left column, you will find the Director, Storage, and Client resources, with their names and passwords – these are all in `bacula-dir.conf`. The right column is where the corresponding values should be found in the Console, Storage daemon (SD), and File daemon (FD) configuration files.

Another thing to check is to ensure that the Bacula component you are trying to access has **Maximum Concurrent Jobs** set large enough to handle each of the Jobs and the Console that want to connect simultaneously. Once the maximum connections has been reached, each Bacula component will reject all new connections.

Please also remember that File daemons with later versions than the Director and Storage daemons are not supported and can result in authorization errors.

Finally, make sure you have no `hosts.allow` or `hosts.deny` file that is not permitting access to the site trying to connect.

Bacula Runs Fine but Cannot Access a Client on a Different Machine. Why? There are several reasons why Bacula could not contact a client on a different machine. They are:

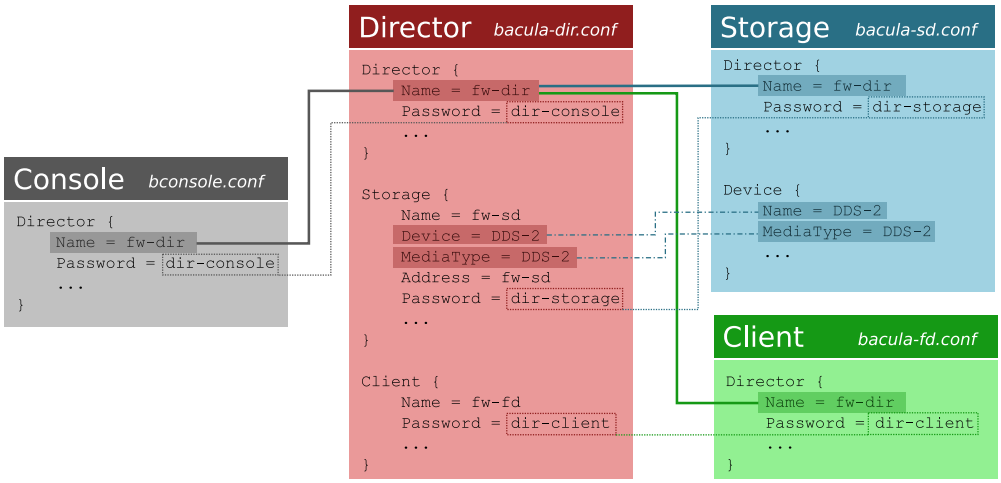


Figure 1.1: Configuration Diagram

- It is a Windows Client, and the client died because of an improper configuration file. Check that the Bacula icon is in the system tray and the the menu items work. If the client has died, the icon will disappear only when you move the mouse over the icon.
- The Client address or port is incorrect or not resolved by DNS. See if you can ping the client machine using the same address as in the Client record.
- You have a firewall, and it is blocking traffic on port 9102 between the Director's machine and the Client's machine (or on port 9103 between the Client and the Storage daemon machines).
- Your password or names are not correct in both the Director and the Client machine. Try configuring everything identical to how you run the client on the same machine as the Director, but just change the Address. If that works, make the other changes one step at a time until it works.
- You may also be having problems between your File daemon and your Storage daemon. The name you use in the Storage resource of your Director's conf file must be known (resolvable) by the File daemon, because it is passed symbolically to the File daemon, which then resolves it to get an IP address used to contact the Storage daemon.
- You may have a `hosts.allow` or `hosts.deny` file that is not permitting access.

My Catalog is Full of Test Runs, How Can I Start Over? If you are using `MySQL`, PostgreSQL, or `SQLite` do the following:

```
cd /opt/bacula/scripts
./drop_bacula_tables
./make_bacula_tables
./grant_bacula_privileges
```

Then write an EOF on each tape you used with **Bacula** using:

```
mt -f /dev/st0 rewind
mt -f /dev/st0 weof
```

where you need to adjust the device name for your system.

I Run a Restore Job and Bacula Hangs. What do I do? On Bacula version 1.25 and prior, it expects you to have the correct tape mounted prior to a restore. On Bacula version 1.26 and higher, it will ask you for the tape, and if the wrong one is mounted, it will inform you.

If you have previously done an `unmount` command, all Storage daemon sessions (jobs) will be completely blocked from using the drive unmounted, so be sure to do a `mount` after your unmount. If in doubt, do a second `mount`, it won't cause any harm.

I Cannot Get My Windows Client to Start Automatically? You are probably having one of two problems: either the Client is dying due to an incorrect configuration file, or you didn't do the Installation commands necessary to install it as a Windows Service.

For the first problem, see the next FAQ question. For the second problem, please review the **Windows Installation Instructions** chapter (chapter 42 page 457) in theBacula Community Edition Main manual.

My Windows Client Immediately Dies When I Start It The most common problem is either that the configuration file is not where it expects it to be, or that there is an error in the configuration file. You must have the configuration file in

```
c:\bacula\bin\bacula-fd.conf
```

To **see** what is going on when the File daemon starts on Windows, do the following:

```
Start a DOS shell Window.
cd c:\bacula\bin
bacula-fd -d100 -c c:\bacula\bin\bacula-fd.conf
```

This will cause the FD to write a file `bacula.trace` in the current directory, which you can examine and thereby determine the problem.

When I Start the Console, the Error Messages Fly By. How can I see them? Either use a shell window with a scroll bar, or use the gnome-console. In any case, you probably should be logging all output to a file, and then you can simply view the file using an editor or the `less` program. To log all output, I have the following in my Director's Message resource definition:

```
append = "/home/kern/bacula/bin/log" = all, !skipped
```

Obviously you will want to change the filename to be appropriate for your system.

I didn't realize that the backups were not working on my Windows Client. What should I do? You should be sending yourself an email message for each job. This will avoid the possibility of not knowing about a failed backup. To do so put something like:

```
Mail = yourname@yourdomain = all, !skipped
```

in your Director's message resource. You should then receive one email for each Job that ran. When you are comfortable with what is going on (it took me 9 months), you might change that to:

```
MailOnError = yourname@yourdomain = all, !skipped
```

then you only get email messages when a Job errors as is the case for your Windows machine.

You should also be logging the Director's messages, please see the previous FAQ for how to do so.

All my Jobs are scheduled for the same time. Will this cause problems? No, not at all. Bacula will schedule all the Jobs at the same time, but will run them one after another unless you have increased the number of simultaneous jobs in the configuration files for the Director, the File daemon, and the Storage daemon. The appropriate configuration record is **Maximum Concurrent Jobs = nn**. At the current time, we recommend that you leave this set to **1** for the Director.

Can Bacula Backup My System To Files instead of Tape? Yes, in principle, Bacula can backup to any storage medium as long as you have correctly defined that medium in the Storage daemon's Device resource. For an example of how to backup to files, please see the **Pruning Example** chapter (chapter 30.7 page 389) of the Bacula Community Edition Main manual. Also, there is a whole chapter devoted to **Basic Volume Management** chapter (chapter 31 page 393) in the Bacula Community Edition Main manual. This chapter was originally written to explain how to write to disk, but was expanded to include volume management. It is, however, still quite a good chapter to read.

Can I use a dummy device to test the backup? Yes, to have a *Virtual* device which just consumes data, you can use a FIFO device (see **Stored configuration** chapter (chapter 24.3 page 323) in the Bacula Community Edition Main manual). It's useful to test a backup.

```
Device {
  Name = NULL
  Media Type = NULL
  Device Type = Fifo
  Archive Device = /dev/null
  LabelMedia = yes
  Random Access = no
  AutomaticMount = no
  RemovableMedia = no
  MaximumOpenWait = 60
  AlwaysOpen = no
}
```

Can Bacula Backup and Restore Files Bigger than 2 Gigabytes? If your operating system permits it, and you are running Bacula version 1.26 or later, the answer is yes. To the best of our knowledge all client system supported by Bacula can handle files bigger 2 Gigabytes.

I Started A Job then Decided I Really Did Not Want to Run It. Is there a better way than `./bacula stop` to stop it? Yes, you normally should use the Console command `cancel` to cancel a Job that is either scheduled or running. If the Job is scheduled, it will be marked for cancellation and will be canceled when it is scheduled to start. If it is running, it will normally terminate after a few minutes. If the Job is waiting on a tape mount, you may need to do a `mount` command before it will be canceled.

Why have You Trademarked the Name Bacula®? We have trademarked the name Bacula to ensure that all media written by any program named Bacula will always be compatible. Anyone may use the name Bacula, even in a derivative product as long as it remains totally compatible in all respects with the program defined here.

Why is the Online Document for Version 1.39 of Bacula when the Current Version is 1.38? As Bacula is being developed, the document is also being enhanced, more often than not it has clarifications of existing features that can be very useful to our users, so we publish the very latest document. Fortunately it is rare that there are confusions with new features.

If you want to read a document that pertains only to a specific version, please use the one distributed in the source code. The web site also has online versions of both the released manual and the current development manual.

How Can I Be Sure that Bacula Really Saves and Restores All Files? It is really quite simple, but took me a while to figure out how to "prove" it. First make a Bacula Rescue disk, see the Disaster Recovery Using Baculamainchapter of the Bacula Community Edition Main manual. Second, you run a full backup of all your files on all partitions. Third, you run an Verify InitCatalog Job on the same FileSet, which effectively makes a record of all the files on your system. Fourth, you run a Verify Catalog job and assure yourself that nothing has changed (well, between an InitCatalog and Catalog one doesn't expect anything). Then do the unthinkable, write zeros on your MBR (master boot record) wiping out your hard disk. Now, restore your whole system using your Bacula Rescue disk and the Full backup you made, and finally re-run the Verify Catalog job. You will see that with the exception of the directory modification and access dates and the files changed during the boot, your system is identical to what it was before you wiped your hard disk. Alternatively you could do the wiping and restoring to another computer of the same type.

I did a Full backup last week, but now in running an Incremental, Bacula says it did not find a FULL backup, so it did a FULL backup. Why? Before doing an Incremental or a Differential backup, Bacula checks to see if there was a prior Full backup of the same Job that terminated successfully. If so, it uses the date that full backup started as the time for comparing if files have changed. If Bacula does not find a successful full backup, it proceeds to do one. Perhaps you canceled the full backup, or it terminated in error. In such cases, the full backup will not be successful. You can check by entering `list jobs` and look to see if there is a prior Job with the same Name that has Level F and JobStatus T (normal termination).

Another reason why Bacula may not find a suitable Full backup is that every time you change the FileSet, Bacula will require a new Full backup. This is necessary to ensure that all files are properly backed up in the case where you have added more files to the FileSet. Beginning with version 1.31, the FileSets are also dated when they are created, and this date is displayed with the name when you are listing or selecting a FileSet. For more on backup levels see below.

See also **Ignore FileSet Changes** in the **FileSet Resource definition** chapter (chapter 22.7 page 244) in the Bacula Community Edition Main manual.

How Can You Claim to Handle Unlimited Path and Filename Lengths when All Other Programs Have Fixed Limits? Most of those other programs have been around for a long time, in fact since the beginning of Unix, which means that they were designed for rather small fixed length path and filename lengths. Over the years, these restrictions have been relaxed allowing longer names. Bacula on the other hand was designed in 2000, and so from the start, Path and Filenames have been kept in buffers that start at 256 bytes in length, but can grow as needed to handle any length. Most of the work is carried out by lower level routines making the coding rather easy.

Note that due to limitations Win32 path and filenames cannot exceed 260 characters. By using Win32 Unicode functions, we will remove this restriction in later versions of Bacula.

What Is the Really Unique Feature of Bacula? Well, it is hard to come up with unique features when backup programs for Unix machines have been around since the 1960s. That said, I believe that Bacula is the first and only program to use a standard SQL interface to catalog its database. Although this adds a bit of complexity and possibly overhead, it provides an amazingly rich set of features that are easy to program and enhance. The current code has barely scratched the surface in this regard (version 1.38).

The second feature, which gives a lot of power and flexibility to Bacula is the Bootstrap record definition.

The third unique feature, which is currently (1.30) unimplemented, and thus can be called vaporware :-), is Base level saves. When implemented, this will enormously reduce tape usage.

If I Run Multiple Simultaneous Jobs, How Can I Force One Particular Job to Run After Another Job? Yes, you can set Priorities on your jobs so that they run in the order you specify. Please see: **the Priority record** chapter (chapter 22.3 page 240) of the Bacula Community Edition Main manual in the Job resource.

I Am Not Getting Email Notification, What Can I Do? The most common problem is that you have not specified a fully qualified email address and your bsmtp server is rejecting the mail. The next most common problem is that your bsmtp server doesn't like the syntax on the From part of the message. For more details on this and other problems, please see the Getting Email Notification to Work section of the Tips chapter of this manual. The section Getting Notified of Job Completion of the Tips chapter may also be useful. For more information on the `bsmtp` mail program, please see **bsmtp** command (command 1.11 page 14) in the Bacula Community Edition Utility programs.

I Change Recycling, Retention Periods, or File Sizes in my Pool Resource and they Still Don't Work. The different variables associated with a Pool are defined in the Pool Resource, but are actually read by Bacula from the Catalog database. On Bacula versions prior to 1.30, after changing your Pool Resource, you must manually update the corresponding values in the Catalog by using the `update pool` command in the Console program. In Bacula version 1.30, Bacula does this for you automatically every time it starts.

When Bacula creates a Media record (Volume), it uses many default values from the Pool record. If you subsequently change the Pool record, the new values will be used as a default for the next Volume that is created, but if you want the new values to apply to existing Volumes, you must manually update the Volume Catalog entry using the `update volume` command in the Console program.

I Have Configured Compression On, But None of My Files Are Compressed. Why? There are two kinds of compression. One is tape compression. This is done by the tape drive hardware,

and you either enable or disable it with system tools such as `mt`. This compression works independently of Bacula, and when it is enabled, you should not use the Bacula software compression.

Bacula also has software compression code in the File daemons, which you normally need to enable only when backing up to file Volumes. There are two conditions necessary to enable the Bacula software compression.

1. You must have the `zip` development libraries loaded on your system when building Bacula and Bacula must find this library, normally `/usr/lib/libz.a`. On Red Hat systems, this library is provided by the `zlib-devel` rpm.
   If the library is found by Bacula during the `./configure` it will be mentioned in the `config.out` line by:
   ```
   ZLIB support:  yes
   ```
2. You must add the **compression=gzip** option on your Include statement in the Director's configuration file.

Bacula is Asking for a New Tape After 2 GB of Data but My Tape holds 33 GB. Why? There are several reasons why Bacula will request a new tape.

- There is an I/O error on the tape. Bacula prints an error message and requests a new tape. Bacula does not attempt to continue writing after an I/O error.
- Bacula encounters and end of medium on the tape. This is not always distinguishable from an I/O error.
- You have specifically set some size limitation on the tape. For example the **Maximum Volume Bytes** or **Maximum Volume Files** in the Director's Pool resource, or **Maximum Volume Size** in the Storage daemon's Device resource.

Bacula is Not Doing the Right Thing When I Request an Incremental Backup. Why? As explained in one of the previous questions, Bacula will automatically upgrade an Incremental or Differential job to a Full backup if it cannot find a prior Full backup or a suitable Full backup. For the gory details on how/when Bacula decides to upgrade levels please see the **Level record** chapter (chapter 22.3 page 223) in the Bacula Community Edition Main manual.

If after reading the above mentioned section, you believe that Bacula is not correctly handling the level (Differential/Incremental), please send us the following information for analysis:

- Your Director's configuration file.
- The output from `list jobs` covering the period where you are having the problem.
- The Job report output from the prior Full save (not critical).
- An `llist jobid=nnn` where `nnn` is the JobId of the prior Full save.
- The Job report output from the save that is doing the wrong thing (not critical).
- An `llist jobid=nnn` where `nnn` is the JobId of the job that was not correct.
- An explanation of what job went wrong and why you think it did.

The above information can allow us to analyze what happened, without it, there is not much we can do.

I am Backing Up an Offsite Machine with an Unreliable Connection. The Director Waits Forever for the Client to Contact the SD. What Can I Do? Bacula was written on the assumption that it will have a good TCP/IP connection between all the daemons. As a consequence, the current Bacula doesn't deal with faulty connections very well. This situation is slowly being corrected over time.

There are several things you can do to improve the situation.

- Upgrade to version 1.32 and use the new SDConnectTimeout record. For example, set:

```
SD Connect Timeout = 5 min
```

in the FileDaemon resource.

- Run these kinds of jobs after all other jobs.

When I ssh into a machine and start Bacula then attempt to exit, ssh hangs forever. This happens because Bacula leaves `stdin`, `stdout`, and `stderr` open for debug purposes. To avoid it, the simplest thing to do is to redirect the output of those files to `/dev/null` or another file in your startup script (the Red Hat autostart scripts do this automatically). For example, you start the Director with:

```
bacula-dir -c bacula-dir.conf \ldots{} >/dev/null 0>\&1 2>\&1
```

and likewise for the other daemons.

I'm confused by the different Retention periods: File Retention, Job Retention, Volume Retention. Why are there so many? Yes, this certainly can be confusing. The basic reason for so many is to allow flexibility. The File records take quite a lot of space in the catalog, so they are typically records you want to remove rather quickly. The Job records, take very little space, and they can be useful even without the File records to see what Jobs actually ran and when. One must understand that if the File records are removed from the catalog, you cannot use the `restore` command to restore an individual file since Bacula no longer knows where it is. However, as long as the Volume Retention period has not expired, the data will still be on the tape, and can be recovered from the tape.

For example, I keep a 30 day retention period for my Files to keep my catalog from getting too big, but I keep my tapes for a minimum of one year, just in case.

Why Does Bacula Ignore the MaxVolumeSize Set in my Pool? The MaxVolumeSize that Bacula uses comes from the Media record, so most likely you changed your Pool, which is used as the default for creating Media records, **after** you created your Volume. Check what is in the Media record by doing:

```
llist Volume=xxx
```

If it doesn't have the right value, you can use:

```
update Volume=xxx
```

to change it.

In connecting to my Client, I get "ERR:Connection Refused. Packet Size too big from File daemon:192.168.1.4:9102" Why? This is typically a communications error resulting from one of the following:

- Old versions of Bacula, usually a Win32 client, where two threads were using the same I/O packet. Fixed in more recent versions. Please upgrade.
- Some other program such as an HP Printer using the same port (9102 in this case).

If it is neither of the above, please submit a bug report at bugs.bacula.org.

Another solution might be to run the daemon with the debug option by:

```
Start a DOS shell Window.
cd c:\bacula\bin
bacula-fd -d100 -c c:\bacula\bin\bacula-fd.conf
```

This will cause the FD to write a file `bacula.trace` in the current directory, which you can examine to determine the problem.

During long running jobs my File daemon dies with Pipe Error, or some other communications error. Why? There are a number of reasons why a connection might break. Most often, it is a router between your two computers that times out inactive lines (not respecting the keepalive feature that Bacula uses). In that case, you can use the **Heartbeat Interval** directive in both the Storage daemon and the File daemon.

In at least one case, the problem has been a bad driver for a Win32 NVidia NForce 3 ethernet card with driver (4.4.2 17/05/2004). In this case, a good driver is (4.8.2.0 06/04/2005). Moral of the story, make sure you have the latest ethernet drivers loaded, or use the following workaround as suggested by Thomas Simmons for Win32 machines:

Browse to:

```
Start > Control Panel > Network Connections
```

Right click the connection for the nvidia adapter and select properties. Under the `General` tab, click "Configure...". Under the `Advanced` tab set "Checksum Offload" to disabled and click OK to save the change.

Lack of communications, or communications that get interrupted can also be caused by Linux firewalls where you have a rule that throttles connections or traffic. For example, if you have:

```
iptables -t filter -A INPUT -m limit --limit 3/second --limit-burst 3 -j DROP
```

you will want to add the following rules **before** the above rule:

```
iptables -t filter -A INPUT --dport 9101 -j ACCEPT
iptables -t filter -A INPUT --dport 9102 -j ACCEPT
iptables -t filter -A INPUT --dport 9103 -j ACCEPT
```

This will ensure that any Bacula traffic will not get terminated because of high usage rates.

I can't figure out how to tell the job which volume to use This is an interesting statement. I now see that a number of people new to Bacula have the same problem as you, probably from using programs like `tar`.

In fact, you do not tell Bacula what tapes to use. It is the inverse. Bacula tells you what tapes it wants. You put tapes at its disposition and it chooses.

Now, if you **really** want to be tricky and try to tell Bacula what to do, it will be reasonable if for example you mount a valid tape that it can use on a drive, it will most likely go ahead and use it. It also has a documented algorithm for choosing tapes – but you are asking for problems ...

So, the trick is to invert your concept of things and put Bacula in charge of handling the tapes. Once you do that, you will be fine. If you want to anticipate what it is going to do, you can generally figure it out correctly and get what you want.

If you start with the idea that you are going to force or tell Bacula to use particular tapes or you insist on trying to run in that kind of mode, you will probably not be too happy.

I don't want to worry about what tape has what data. That is what Bacula is designed for.

If you have an application where you **really** need to remove a tape each day and insert a new one, it can be done the directives exist to accomplish that. In such a case, one little "trick" to knowing what tape Bacula will want at 2am while you are asleep is to run a tiny job at 4pm while you are still at work that backs up say one directory, or even one file. You will quickly find out what tape it wants, and you can mount it before you go home ...

How do I generate a password? Each daemon needs a password. This password occurs in the configuration file for that daemon and in the `bacula-dir.conf` file. These passwords are plain text. There is no special generation procedure. Most people just use random text.

Passwords are never sent over the wire in plain text. They are always encrypted.

Security surrounding these passwords is best left security to your operating system. Passwords are not encrypted within Bacula configuration files.

Chapter 2

Tips and Suggestions

There are a number of example scripts for various things that can be found in the example subdirectory and its subdirectories of the Bacula source distribution.

For additional tips, please see the Bacula wiki.

2.1 Upgrading Bacula Versions

The first thing to do before upgrading from one version to another is to ensure that you don't overwrite or delete your production (current) version of Bacula until you have tested that the new version works.

If you have installed Bacula into a single directory, this is simple: simply make a copy of your Bacula directory.

If you have done a more typical Unix installation where the binaries are placed in one directory and the configuration files are placed in another, then the simplest way is to configure your new Bacula to go into a single file. Alternatively, make copies of all your binaries and especially your conf files.

Whatever your situation may be (one of the two just described), you should probably start with the defaultconf script that can be found in the examples subdirectory. Copy this script to the main Bacula directory, modify it as necessary (there should not need to be many modifications), configure Bacula, build it, install it, then stop your production Bacula, copy all the *.conf files from your production Bacula directory to the test Bacula directory, start the test version, and run a few test backups. If all seems good, then you can proceed to install the new Bacula in place of or possibly over the old Bacula.

When installing a new Bacula you need not worry about losing the changes you made to your configuration files as the installation process will not overwrite them providing that you do not do a make uninstall.

If the new version of Bacula requires an upgrade to the database, you can upgrade it with the script update_bacula_tables, which will be installed in your scripts directory (default /etc/bacula), or alternatively, you can find it in the <bacula-source>/src/cats directory.

2.2 Getting Notified of Job Completion

One of the first things you should do is to ensure that you are being properly notified of the status of each Job run by Bacula, or at a minimum of each Job that terminates with an error.

Until you are completely comfortable with **Bacula**, we recommend that you send an email to yourself for each Job that is run. This is most easily accomplished by adding an email notification address in the **Messages** resource of your Director's configuration file. An email is automatically configured in the default configuration files, but you must ensure that the default `root` address is replaced by your email address.

For additional examples of how to configure a Bacula, please take a look at the `.conf` files found in the `examples` sub-directory. We recommend the following configuration (where you change the paths and email address to correspond to your setup). Note, the **mailcommand** and **operatorcommand** should be on a single line. They were split here for presentation:

```
Messages {
  Name = Standard
  mailcommand = "/home/bacula/bin/bsmtp -h localhost
                -f \"\(Bacula\) %r\"
                -s \"Bacula: %t %e of %c %l\" %r"
  operatorcommand = "/home/bacula/bin/bsmtp -h localhost
                -f \"\(Bacula\) %r\"
                -s \"Bacula: Intervention needed for %j\" %r"
  Mail = your-email-address = all, !skipped, !terminate
  append = "/home/bacula/bin/log" = all, !skipped, !terminate
  operator = your-email-address = mount
  console = all, !skipped, !saved
}
```

You will need to ensure that the `/home/bacula/bin` path on the **mailcommand** and the **operatorcommand** lines point to your **Bacula** binary directory where the `bsmtp` program will be installed. You will also want to ensure that the **your-email-address** is replaced by your email address, and finally, you will also need to ensure that the `/home/bacula/bin/log` points to the file where you want to log all messages.

With the above Messages resource, you will be notified by email of every Job that ran, all the output will be appended to the `log` file you specify, all output will be directed to the console program, and all mount messages will be emailed to you. Note, some messages will be sent to multiple destinations.

The form of the mailcommand is a bit complicated, but it allows you to distinguish whether the Job terminated in error or terminated normally. Please see the **Mail** command (command 26 page 341) in the Bacula Community Edition Main manual for the details of the substitution characters used above.

Once you are totally comfortable with Bacula as I am, or if you have a large number of nightly Jobs as I do (eight), you will probably want to change the **Mail** command to **Mail On Error** which will generate an email message only if the Job terminates in error. If the Job terminates normally, no email message will be sent, but the output will still be appended to the log file as well as sent to the Console program.

2.3 Getting Email Notification to Work

The section above describes how to get email notification of job status. Occasionally, however, users have problems receiving any email at all. In that case, the things to check are the following:

- Ensure that you have a valid email address specified on your **Mail** record in the Director's Messages resource. The email address should be fully qualified. Simply using `root` generally will not work, rather you should use `root@localhost` or better yet your full domain.
- Ensure that you do not have a **Mail** record in the Storage daemon's or File daemon's configuration files. The only record you should have is **director**:

  ```
  director = director-name = all
  ```

- If all else fails, try replacing the **mailcommand** with

```
mailcommand = "mail -s test your@domain.com"
```

- Once the above is working, assuming you want to use bsmtp, submit the desired bsmtp command by hand and ensure that the email is delivered, then put that command into **Bacula**. Small differences in things such as the parenthesis around the word Bacula can make a big difference to some bsmtp programs. For example, you might start simply by using:

```
mailcommand = "/home/bacula/bin/bsmtp -f \"root@localhost\" %r"
```

2.4 Getting Notified that Bacula is Running

If like me, you have setup Bacula so that email is sent only when a Job has errors, as described in the previous section of this chapter, inevitably, one day, something will go wrong and **Bacula** can stall. This could be because Bacula crashes, which is vary rare, or more likely the network has caused **Bacula** to **hang** for some unknown reason.

To avoid this, you can use the **RunAfterJob** command in the Job resource to schedule a Job nightly, or weekly that simply emails you a message saying that Bacula is still running. For example, I have setup the following Job in my Director's configuration file:

```
Schedule {
  Name = "Watchdog"
  Run = Level=Full sun-sat at 6:05
}
Job {
  Name = "Watchdog"
  Type = Admin
  Client=Watchdog
  FileSet="Verify Set"
  Messages = Standard
  Storage = DLTDrive
  Pool = Default
  Schedule = "Watchdog"
  RunAfterJob = "/home/kern/bacula/bin/watchdog %c %d"
}
Client {
  Name = Watchdog
  Address = rufus
  FDPort = 9102
  Catalog = Verify
  Password = ""
  File Retention = 1day
  Job Retention = 1 month
  AutoPrune = yes
}
```

Where I established a schedule to run the Job nightly. The Job itself is type **Admin** which means that it doesn't actually do anything, and I've defined a FileSet, Pool, Storage, and Client, all of which are not really used (and probably don't need to be specified). The key aspect of this Job is the command:

```
  RunAfterJob = "/home/kern/bacula/bin/watchdog %c %d"
```

which runs my "watchdog" script. As an example, I have added the Job codes %c and %d which will cause the Client name and the Director's name to be passed to the script. For example, if the Client's name is **Watchdog** and the Director's name is **main-dir** then referencing $1 in the script would get **Watchdog** and referencing $2 would get **main-dir**. In this case, having the script know the Client and Director's name is not really useful, but in other situations it may be.

You can put anything in the watchdog script. In my case, I like to monitor the size of my catalog to be sure that **Bacula** is really pruning it. The following is my watchdog script:

```
#!/bin/sh
cd /home/kern/mysql/var/bacula
du . * |
/home/kern/bacula/bin/bsmtp  \
   -f "\(Bacula\) abuse@whitehouse.com" -h mail.yyyy.com \
   -s "Bacula running" abuse@whitehouse.com
```

If you just wish to send yourself a message, you can do it with:

```
#!/bin/sh
cd /home/kern/mysql/var/bacula
/home/kern/bacula/bin/bsmtp  \
   -f "\(Bacula\) abuse@whitehouse.com" -h mail.yyyy.com \
   -s "Bacula running" abuse@whitehouse.com <<END-OF-DATA
Bacula is still running!!!
END-OF-DATA
```

2.5 Maintaining a Valid Bootstrap File

By using a **WriteBootstrap** (chapter 22.3 page 227) record in each of your Director's Job resources (Bacula Community Edition Main manual), you can constantly maintain a **bootstrap** (chapter 55 page 645) file that will enable you to recover the state of your system as of the last backup without having the Bacula catalog. This permits you to more easily recover from a disaster that destroys your Bacula catalog.

When a Job resource has a **WriteBootstrap** record, Bacula will maintain the designated file (normally on another system but mounted by NFS) with up to date information necessary to restore your system. For example, in my Director's configuration file, I have the following record:

```
Write Bootstrap = "/mnt/deuter/files/backup/client-name.bsr"
```

where I replace **client-name** by the actual name of the client that is being backed up. Thus, Bacula automatically maintains one file for each of my clients. The necessary bootstrap information is appended to this file during each **Incremental** backup, and the file is totally rewritten during each **Full** backup.

Note, one disadvantage of writing to an NFS mounted volume as I do is that if the other machine goes down, the OS will wait forever on the fopen() call that Bacula makes. As a consequence, Bacula will completely stall until the machine exporting the NFS mounts comes back up. A possible solution to this problem was provided by Andrew Hilborne, and consists of using the **soft** option instead of the **hard** option when mounting the NFS volume, which is typically done in `/etc/fstab/`. The NFS documentation explains these options in detail. However, I found that with the **soft** option NFS disconnected frequently causing even more problems.

If you are starting off in the middle of a cycle (i.e. with Incremental backups) rather than at the beginning (with a Full backup), the **bootstrap** file will not be immediately valid as it must always have the information from a Full backup as the first record. If you wish to synchronize your bootstrap file immediately, you can do so by running a `restore` command for the client and selecting a full restore, but when the `restore` command asks for confirmation to run the restore Job, you simply reply no, then copy the bootstrap file that was written to the location specified on the **Write Bootstrap** record. The restore bootstrap file can be found in `restore.bsr` in the working directory that you defined. In the example given below for the client **rufus**, my input is shown in bold. Note, the JobId output has been partially truncated to fit on the page here:

```
(in the Console program)
*restore
First you select one or more JobIds that contain files
to be restored. You will then be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.
To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of JobIds to select
     4: Enter SQL list command
     5: Select the most recent backup for a client
     6: Cancel
Select item:  (1-6): 5
The defined Client resources are:
     1: Minimatou
     2: Rufus
     3: Timmy
Select Client (File daemon) resource (1-3): 2
The defined FileSet resources are:
     1: Other Files
Item 1 selected automatically.
+-------+------+-------+---------+---------+------+-------+------------+
| JobId | Levl | Files | StrtTim | VolName | File | SesId | VolSesTime |
+-------+------+-------+---------+---------+------+-------+------------+
| 2     | F    | 84    |  ...    | test1   | 0    | 1     | 1035645259 |
+-------+------+-------+---------+---------+------+-------+------------+
You have selected the following JobId: 2
Building directory tree for JobId 2 ...
The defined Storage resources are:
     1: File
Item 1 selected automatically.
You are now entering file selection mode where you add and
remove files to be restored. All files are initially added.
Enter "done" to leave this mode.
cwd is: /
$ done
84 files selected to restore.
Run Restore job
JobName:    kernsrestore
Bootstrap:  /home/kern/bacula/working/restore.bsr
Where:      /tmp/bacula-restores
FileSet:    Other Files
Client:     Rufus
Storage:    File
JobId:      *None*
OK to run? (yes/mod/no): no
quit
(in a shell window)
cp ../working/restore.bsr /mnt/deuter/files/backup/rufus.bsr
```

2.6 Rejected Volumes After a Crash

Bacula keeps a count of the number of files on each Volume in its Catalog database so that before appending to a tape, it can verify that the number of files are correct, and thus prevent overwriting valid data. If the Director or the Storage daemon crashes before the job has completed, the tape will contain one more file than is noted in the Catalog, and the next time you attempt to use the same Volume, Bacula will reject it due to a mismatch between the physical tape (Volume) and the catalog.

The easiest solution to this problem is to label a new tape and start fresh. If you wish to continue appending to the current tape, you can do so by using the `update` command in the console program to change the **Volume Files** entry in the catalog. A typical sequence of events would go like the following:

```
- Bacula crashes
- You restart Bacula
```

Bacula then prints:

```
17-Jan-2003 16:45 rufus-dir: Start Backup JobId 13,
                  Job=kernsave.2003-01-17_16.45.46
17-Jan-2003 16:45 rufus-sd: Volume test01 previously written,
                  moving to end of data.
17-Jan-2003 16:46 rufus-sd: kernsave.2003-01-17_16.45.46 Error:
                  I cannot write on this volume because:
                  The number of files mismatch! Volume=11 Catalog=10
17-Jan-2003 16:46 rufus-sd: Job kernsave.2003-01-17_16.45.46 waiting.
                   Cannot find any appendable volumes.
Please use the "label"  command to create a new Volume for:
    Storage:      SDT-10000
    Media type:   DDS-4
    Pool:         Default
```

(note, lines wrapped for presentation) The key here is the line that reads:

```
   The number of files mismatch! Volume=11 Catalog=10
```

It says that Bacula found eleven files on the volume, but that the catalog says there should be ten. When you see this, you can be reasonably sure that the SD was interrupted while writing before it had a chance to update the catalog. As a consequence, you can just modify the catalog count to eleven, and even if the catalog contains references to files saved in file 11, everything will be OK and nothing will be lost. Note that if the SD had written several file marks to the volume, the difference between the Volume count and the Catalog count could be larger than one, but this is unusual.

If on the other hand the catalog is marked as having more files than Bacula found on the tape, you need to consider the possible negative consequences of modifying the catalog. Please see below for a more complete discussion of this.

Continuing with the example of **Volume = 11 Catalog = 10**, to enable to Bacula to append to the tape, you do the following:

```
update
Update choice:
     1: Volume parameters
     2: Pool from resource
     3: Slots from autochanger
Choose catalog item to update (1-3): 1
Defined Pools:
     1: Default
     2: File
Select the Pool (1-2):
+-------+---------+--------+---------+-----------+------+----------+------+-----+
| MedId | VolName | MedTyp | VolStat | VolBytes  | Last | VolReten | Recy | Slt |
+-------+---------+--------+---------+-----------+------+----------+------+-----+
| 1     | test01  | DDS-4  | Error   | 352427156 | ...  | 31536000 | 1    | 0   |
+-------+---------+--------+---------+-----------+------+----------+------+-----+
Enter MediaId or Volume name: 1
```

(note table output truncated for presentation) First, you chose to update the Volume parameters by entering a **1**. In the volume listing that follows, notice how the VolStatus is **Error**. We will correct that after changing the Volume Files. Continuing, you respond 1,

```
Updating Volume "test01"
Parameters to modify:
     1: Volume Status
     2: Volume Retention Period
     3: Volume Use Duration
     4: Maximum Volume Jobs
     5: Maximum Volume Files
     6: Maximum Volume Bytes
```

```
     7: Recycle Flag
     8: Slot
     9: Volume Files
    10: Pool
    11: Done
Select parameter to modify (1-11): 9
Warning changing Volume Files can result
in loss of data on your Volume
Current Volume Files is: 10
Enter new number of Files for Volume: 11
New Volume Files is: 11
Updating Volume "test01"
Parameters to modify:
     1: Volume Status
     2: Volume Retention Period
     3: Volume Use Duration
     4: Maximum Volume Jobs
     5: Maximum Volume Files
     6: Maximum Volume Bytes
     7: Recycle Flag
     8: Slot
     9: Volume Files
    10: Pool
    11: Done
Select parameter to modify (1-10): 1
```

Here, you have selected **9** in order to update the Volume Files, then you changed it from **10** to **11**, and you now answer **1** to change the Volume Status.

```
Current Volume status is: Error
Possible Values are:
     1: Append
     2: Archive
     3: Disabled
     4: Full
     5: Used
     6: Read-Only
Choose new Volume Status (1-6): 1
New Volume status is: Append
Updating Volume "test01"
Parameters to modify:
     1: Volume Status
     2: Volume Retention Period
     3: Volume Use Duration
     4: Maximum Volume Jobs
     5: Maximum Volume Files
     6: Maximum Volume Bytes
     7: Recycle Flag
     8: Slot
     9: Volume Files
    10: Pool
    11: Done
Select parameter to modify (1-11): 11
Selection done.
```

At this point, you have changed the Volume Files from **10** to **11** to account for the last file that was written but not updated in the database, and you changed the Volume Status back to **Append**.

This was a lot of words to describe something quite simple.

The **Volume Files** option exists only in version 1.29 and later, and you should be careful using it. Generally, if you set the value to that which Bacula said is on the tape, you will be OK, especially if the value is one more than what is in the catalog.

Now lets consider the case:

```
The number of files mismatch! Volume=10 Catalog=12
```

Here the Bacula found fewer files on the volume than what is marked in the catalog. Now, in this case, you should hesitate a lot before modifying the count in the catalog, because if you force the catalog from 12 to 10, Bacula will start writing after the file 10 on the tape, possibly overwriting valid data, and if you ever try to restore any of the files that the catalog has marked as saved on Files 11 and 12, all chaos will break out. In this case, you will probably be better off using a new tape. In fact, you might want to see what files the catalog claims are actually stored on that Volume, and back them up to another tape and recycle this tape.

2.7 Security Considerations

Only the File daemon needs to run with root permission (so that it can access all files). As a consequence, you may run your Director, Storage daemon, and MySQL or PostgreSQL database server as non-root processes. Version 1.30 has the **-u** and the **-g** options that allow you to specify a userid and groupid on the command line to be used after Bacula starts.

As of version 1.33, thanks to Dan Langille, it is easier to configure the Bacula Director and Storage daemon to run as non-root.

You should protect the Bacula port addresses (normally 9101, 9102, and 9103) from outside access by a firewall or other means of protection to prevent unauthorized use of your daemons.

You should ensure that the configuration files are not world readable since they contain passwords that allow access to the daemons. Anyone who can access the Director using a console program can restore any file from a backup Volume.

You should protect your Catalog database. If you are using SQLite, make sure that the working directory is readable only by root (or your Bacula userid), and ensure that **bacula.db** has permissions **-rw-r--r--** (i.e. 640) or more strict. If you are using MySQL or PostgreSQL, please note that the Bacula setup procedure leaves the database open to anyone. At a minimum, you should assign the user `bacula` a userid and add it to your Director's configuration file in the appropriate Catalog resource.

If you use the make_catalog_backup script provided by Bacula, remember that you should take care when supplying passwords on the command line. Read the Backing Up Your Bacula Database — Security Considerations section for more information.

2.8 Creating Holiday Schedules

If you normally change tapes every day or at least every Friday, but Thursday is a holiday, you can use a trick proposed by Lutz Kittler to ensure that no job runs on Thursday so that you can insert Friday's tape and be sure it will be used on Friday. To do so, define a **RunJobBefore** script that normally returns zero, so that the Bacula job will normally continue. You can then modify the script to return non-zero on any day when you do not want Bacula to run the job.

2.9 Automatic Labeling Using Your Autochanger

If you have an autochanger but it does not support barcodes, using a "trick" you can make Bacula automatically label all the volumes in your autochanger's magazine.

First create a file containing one line for each slot in your autochanger that has a tape to be labeled. The line will contain the slot number a colon (:) then the Volume name you want to use. For example, create a file named **volume-list**, which contains:

```
1:Volume001
2:TestVolume02
5:LastVolume
```

The records do not need to be in any order and you don't need to mention all the slots. Normally, you will have a consistent set of Volume names and a sequential set of numbers for each slot you want labeled. In the example above, I've left out slots 3 and 4 just as an example. Now, modify your mtx-changer script and comment out all the lines in the **list)** case by putting a # in column 1. Then add the following two lines:

```
cat <absolute-path>/volume-list
exit 0
```

so that the whole case looks like:

```
list)
#
# commented out lines
cat <absolute-path>/volume-list
exit 0
;;
```

where you replace <absolute-path> with the full path to the volume-list file. Then using the console, you enter the following command:

```
label barcodes
```

and Bacula will proceed to mount the autochanger Volumes in the list and label them with the Volume names you have supplied. Bacula will think that the list was provided by the autochanger barcodes, but in reality, it was you who supplied the <barcodes>.

If it seems to work, when it finishes, enter:

```
list volumes
```

and you should see all the volumes nicely created.

2.10 Backing Up Portables Using DHCP

You may want to backup laptops or portables that are not always connected to the network. If you are using DHCP to assign an IP address to those machines when they connect, you will need to use the Dynamic Update capability of DNS to assign a name to those machines that can be used in the Address field of the Client resource in the Director's conf file.

2.11 Going on Vacation

At some point, you may want to be absent for a week or two and you want to make sure Bacula has enough tape left so that the backups will complete. You start by doing a **list volumes** in the Console program:

```
list volumes
```

```
Using default Catalog name=BackupDB DB=bacula
Pool: Default
+---------+---------------+-----------+-----------+----------------+-
| MediaId | VolumeName    | MediaType | VolStatus |       VolBytes |
+---------+---------------+-----------+-----------+----------------+-
|      23 | DLT-30Nov02   | DLT8000   | Full      | 54,739,278,128 |
|      24 | DLT-21Dec02   | DLT8000   | Full      | 56,331,524,629 |
|      25 | DLT-11Jan03   | DLT8000   | Full      | 67,863,514,895 |
|      26 | DLT-02Feb03   | DLT8000   | Full      | 63,439,314,216 |
|      27 | DLT-03Mar03   | DLT8000   | Full      | 66,022,754,598 |
|      28 | DLT-04Apr03   | DLT8000   | Full      | 60,792,559,924 |
|      29 | DLT-28Apr03   | DLT8000   | Full      | 62,072,494,063 |
|      30 | DLT-17May03   | DLT8000   | Full      | 65,901,767,839 |
|      31 | DLT-07Jun03   | DLT8000   | Used      | 56,558,490,015 |
|      32 | DLT-28Jun03   | DLT8000   | Full      | 64,274,871,265 |
|      33 | DLT-19Jul03   | DLT8000   | Full      | 64,648,749,480 |
|      34 | DLT-08Aug03   | DLT8000   | Full      | 64,293,941,255 |
|      35 | DLT-24Aug03   | DLT8000   | Append    |  9,999,216,782 |
+---------+---------------+-----------+-----------+----------------+
```

Note, I have truncated the output for presentation purposes. What is significant, is that I can see that my current tape has almost 10 Gbytes of data, and that the average amount of data I get on my tapes is about 60 Gbytes. So if I go on vacation now, I don't need to worry about tape capacity (at least not for short absences).

Equally significant is the fact that I did go on vacation the 28th of June 2003, and when I did the `list volumes` command, my current tape at that time, DLT-07Jun03 MediaId 31, had 56.5 Gbytes written. I could see that the tape would fill shortly. Consequently, I manually marked it as **Used** and replaced it with a fresh tape that I labeled as DLT-28Jun03, thus assuring myself that the backups would all complete without my intervention.

2.12 Exclude Files on Windows Regardless of Case

This tip was submitted by Marc Brueckner who wasn't sure of the case of some of his files on Win32, which is case insensitive. The problem is that Bacula thinks that **/UNIMPORTANT FILES** is different from **/Unimportant Files**. Marc was aware that the file exclusion permits wild-cards. So, he specified:

```
"/[Uu][Nn][Ii][Mm][Pp][Oo][Rr][Tt][Aa][Nn][Tt]  [Ff][Ii][Ll][Ee][Ss]"
```

As a consequence, the above exclude works for files of any case.

Please note that this works only in Bacula Exclude statement and not in Include.

2.13 Executing Scripts on a Remote Machine

This tip also comes from Marc Brueckner. (Note, this tip is probably outdated by the addition of **ClientRunBeforJob** and **ClientRunAfterJob** Job records, but the technique still could be useful.) First I thought the "Run Before Job" statement in the Job-resource is for executing a script on the remote machine (the machine to be backed up). (Note, this is possible as mentioned above by using **ClientRunBeforJob** and **ClientRunAfterJob**). It could be useful to execute scripts on the remote machine e.g. for stopping databases or other services while doing the backup. (Of course I have to start the services again when the backup has finished) I found the following solution: Bacula could execute scripts on the remote machine by using ssh. The authentication is done automatically using a private key. First you have to generate a keypair. I've done this by:

```
ssh-keygen -b 4096 -t dsa -f Bacula_key
```

This statement may take a little time to run. It creates a public/private key pair with no passphrase. You could save the keys in /etc/bacula. Now you have two new files : Bacula_key which contains the private key and Bacula_key.pub which contains the public key.

Now you have to append the Bacula_key.pub file to the file authorized_keys in the /root/.ssh directory of the remote machine. Then you have to add (or uncomment) the line

```
AuthorizedKeysFile          %h/.ssh/authorized_keys
```

to the sshd_config file on the remote machine. Where the %h stands for the home-directory of the user (root in this case).

Assuming that your sshd is already running on the remote machine, you can now enter the following on the machine where Bacula runs:

```
ssh -i Bacula_key  -l root <machine-name-or-ip-address> "ls -la"
```

This should execute the ls -la command on the remote machine.

Now you could add lines like the following to your Director's conf file:

```
...
Run Before Job = ssh -i /etc/bacula/Bacula_key 192.168.1.1 \
                 "/etc/init.d/database stop"
Run After Job = ssh -i /etc/bacula/Bacula_key 192.168.1.1 \
                 "/etc/init.d/database start"
...
```

Even though Bacula version 1.32 and later has a ClientRunBeforeJob, the ssh method still could be useful for updating all the Bacula clients on several remote machines in a single script.

2.14 Recycling All Your Volumes

This tip comes from Phil Stracchino.

If you decide to blow away your catalog and start over, the simplest way to re-add all your prelabeled tapes with a minimum of fuss (provided you don't care about the data on the tapes) is to add the tape labels using the console add command, then go into the catalog and manually set the VolStatus of every tape to **Recycle**.

The SQL command to do this is very simple, either use your vendor's command line interface (MySQL, PostgreSQL, SQLite ,...) or use the sql command in the Bacula console:

```
update Media set VolStatus='Recycle';
```

Bacula will then ignore the data already stored on the tapes and just re-use each tape without further objection.

2.15 Backing up ACLs on ext3 or XFS filesystems

This tip comes from Volker Sauer.

Note, this tip was given prior to implementation of ACLs in Bacula (version 1.34.5). It is left here because dumping/displaying ACLs can still be useful in testing/verifying that Bacula is

backing up and restoring your ACLs properly. Please see the aclsupport FileSet option in the configuration chapter of this manual.

For example, you could dump the ACLs to a file with a script similar to the following:

```
#!/bin/sh
BACKUP_DIRS="/foo /bar"
STORE_ACL=/root/acl-backup
umask 077
for i in $BACKUP_DIRS; do
 cd $i /usr/bin/getfacl -R --skip-base .>$STORE_ACL/${i//\//_}
done
```

Then use Bacula to backup /root/acl-backup.

The ACLs could be restored using Bacula to the /root/acl-backup file, then restored to your system using:

```
setfacl --restore/root/acl-backup
```

2.16 Total Automation of Bacula Tape Handling

This tip was provided by Alexander Kuehn.

http://www.bacula.org/[1] is a really nice backup program except that the manual tape changing requires user interaction with the bacula console.

Fortunately I can fix this. NOTE!!! This suggestion applies for people who do **not** have tape autochangers and must change tapes manually.!!!!!

Bacula supports a variety of tape changers through the use of mtx-changer scripts/programs. This highly flexible approach allowed me to create http://www.bacula.org/en/rel-manual/mtx-changer.txt[2] which does the following: Whenever a new tape is required it sends a mail to the operator to insert the new tape. Then it waits until a tape has been inserted, sends a mail again to say thank you and let's bacula continue its backup. So you can schedule and run backups without ever having to log on or see the console. To make the whole thing work you need to create a Device resource which looks something like this ("Archive Device", "Maximum Changer Wait", "Media Type" and "Label media" may have different values):

```
Device {
   Name=DDS3
   Archive Device = # use yours not mine! ;)/dev/nsa0
   Changer Device = # not really required/dev/nsa0
   Changer Command = "# use this (maybe change the path)!
         /opt/bacula/scripts/mtx-changer %o %a %S"
   Maximum Changer Wait = 3d          # 3 days in seconds
   AutomaticMount = yes;              # mount on start
   AlwaysOpen = yes;                  # keep device locked
   Media Type = DDS3                  # it's just a name
   RemovableMedia = yes;              #
   Offline On Unmount = Yes;          # keep this too
   Label media = Yes;                 #
}
```

As the script has to emulate the complete wisdom of a mtx-changer it has an internal "database" containing where which tape is stored, you can see this on the following line:

[1] Bacula
[2] this shell script

```
labels="VOL-0001 VOL-0002 VOL-0003 VOL-0004 VOL-0005 VOL-0006
VOL-0007 VOL-0008 VOL-0009 VOL-0010 VOL-0011 VOL-0012"
```

The above should be all on one line, and it effectively tells Bacula that volume "VOL-0001" is located in slot 1 (which is our lowest slot), that volume "VOL-0002" is located in slot 2 and so on. The script also maintains a logfile (/var/log/mtx.log) where you can monitor its operation.

2.17 Running Concurrent Jobs

Bacula can run multiple concurrent jobs, but the default configuration files do not enable it. Using the **Maximum Concurrent Jobs** directive, you can configure how many and which jobs can be run simultaneously. The Director's default value for **Maximum Concurrent Jobs** is 1.

To initially setup concurrent jobs you need to define **Maximum Concurrent Jobs** in the Director's configuration file (bacula-dir.conf) in the Director, Job, Client, and Storage resources.

Additionally the File daemon, and the Storage daemon each have their own **Maximum Concurrent Jobs** directive that sets the overall maximum number of concurrent jobs the daemon will run. The default for both the File daemon and the Storage daemon is 20.

For example, if you want two different jobs to run simultaneously backing up the same Client to the same Storage device, they will run concurrently only if you have set **Maximum Concurrent Jobs** greater than one in the Director resource, the Client resource, and the Storage resource in bacula-dir.conf.

We recommend that you read the Data Spooling of this manual first, then test your multiple concurrent backup including restore testing before you put it into production.

Below is a super stripped down bacula-dir.conf file showing you the four places where the the file must be modified to allow the same job **NightlySave** to run up to four times concurrently. The change to the Job resource is not necessary if you want different Jobs to run at the same time, which is the normal case.

```
#
# Bacula Director Configuration file -- bacula-dir.conf
#
Director {
  Name = rufus-dir
  Maximum Concurrent Jobs = 4
  ...
}
Job {
  Name = "NightlySave"
  Maximum Concurrent Jobs = 4
  Client = rufus-fd
  Storage = File
  ...
}
Client {
  Name = rufus-fd
  Maximum Concurrent Jobs = 4
  ...
}
Storage {
  Name = File
  Maximum Concurrent Jobs = 4
  ...
}
```

Chapter 3

Testing Your Tape Drive With Bacula

This chapter is concerned with testing and configuring your tape drive to make sure that it will work properly with Bacula using the btape program.

3.1 Get Your Tape Drive Working

In general, you should follow the following steps to get your tape drive to work with Bacula. Start with a tape mounted in your drive. If you have an autochanger, load a tape into the drive. We use /dev/nst0 as the tape drive name, you will need to adapt it according to your system.

Do not proceed to the next item until you have succeeded with the previous one.

1. Make sure that Bacula (the Storage daemon) is not running or that you have unmounted the drive you will use for testing.
2. Use tar to write to, then read from your drive:

```
mt -f /dev/nst0 rewind
tar cvf /dev/nst0 .
mt -f /dev/nst0 rewind
tar tvf /dev/nst0
```

3. Make sure you have a valid and correct Device resource corresponding to your drive. For Linux users, generally, the default one works. For FreeBSD users, there are two possible Device configurations (see below). For other drives and/or OSes, you will need to first ensure that your system tape modes are properly setup (see below), then possibly modify you Device resource depending on the output from the btape program (next item). When doing this, you should consult the **Storage Daemon Configuration** chapter (chapter 24 page 315) of the Bacula Community Edition Main manual.
4. If you are using a Fibre Channel to connect your tape drive to Bacula, please be sure to disable any caching in the Network Storage Router (NSR) (NSR is a Fibre Channel to SCSI converter).
5. Run the btape test command:

```
./btape -c bacula-sd.conf /dev/nst0
test
```

It isn't necessary to run the autochanger part of the test at this time, but do not go past this point until the basic test succeeds. If you do have an autochanger, please be sure to read the **Autochanger chapter** chapter (chapter 36 page 429) of the Bacula Community Edition Main manual.

6. Run the btape `fill` command, preferably with two volumes. This can take a long time. If you have an autochanger and it is configured, Bacula will automatically use it. If you do not have it configured, you can manually issue the appropriate `mtx` command, or press the autochanger buttons to change the tape when requested to do so.

7. FreeBSD users, if you have a pre-5.0 system run the `tapetest` program, and make sure your system is patched if necessary. The tapetest program can be found in the platform/freebsd directory. The instructions for its use are at the top of the file.

8. Run Bacula, and backup a reasonably small directory, say 60 Megabytes. Do three successive backups of this directory.

9. Stop Bacula, then restart it. Do another full backup of the same directory. Then stop and restart Bacula.

10. Do a restore of the directory backed up, by entering the following restore command, being careful to restore it to an alternate location:

```
restore select all done
yes
```

    Do a `diff` on the restored directory to ensure it is identical to the original directory. If you are going to backup multiple different systems (Linux, Windows, Mac, Solaris, FreeBSD, ...), be sure you test the restore on each system type.

11. If you have an autochanger, you should now go back to the `btape` program and run the `autochanger` test:

```
./btape -c bacula-sd.conf /dev/nst0
auto
```

    Adjust your autochanger as necessary to ensure that it works correctly. See the Autochanger chapter of this manual for a complete discussion of testing your autochanger.

12. We strongly recommend that you use a dedicated SCSI controller for your tape drives. Scanners are known to induce serious problems with the SCSI bus, causing it to reset. If the SCSI bus is reset while Bacula has the tape drive open, it will most likely be fatal to your tape since the drive will rewind. These kinds of problems show up in the system log. For example, the following was most likely caused by a scanner:

```
Feb 14 17:29:55 epohost kernel: (scsi0:A:2:0): No or incomplete CDB sent to device.
Feb 14 17:29:55 epohost kernel: scsi0: Issued Channel A Bus Reset. 1 SCBs aborted
```

If you have reached this point, you stand a good chance of having everything work. If you get into trouble at any point, **carefully** read the documentation given below. If you cannot get past some point, ask the **bacula-users** email list, but specify which of the steps you have successfully completed. In particular, you may want to look at the Tips for Resolving Problems section below.

3.1.1 Problems When no Tape in Drive

When Bacula was first written the Linux 2.4 kernel permitted opening the drive whether or not there was a tape in the drive. Thus the Bacula code is based on the concept that if the drive cannot be opened, there is a serious problem, and the job is failed.

With version 2.6 of the Linux kernel, if there is no tape in the drive, the OS will wait two minutes (default) and then return a failure, and consequently, Bacula version 1.36 and below will fail the job. This is important to keep in mind, because if you use an option such as **Offline**

on Unmount = yes, there will be a point when there is no tape in the drive, and if another job starts or if Bacula asks the operator to mount a tape, when Bacula attempts to open the drive (about a 20 minute delay), it will fail and Bacula will fail the job.

In version 1.38.x, the Bacula code partially gets around this problem – at least in the initial open of the drive. However, functions like Polling the drive do not work correctly if there is no tape in the drive. Providing you do not use **Offline on Unmount = yes**, you should not experience job failures as mentioned above. If you do experience such failures, you can also increase the **Maximum Open Wait** time interval, which will give you more time to mount the next tape before the job is failed.

3.1.2 Specifying the Configuration File

Starting with version 1.27, each of the tape utility programs including the `btape` program requires a valid Storage daemon configuration file (actually, the only part of the configuration file that `btape` needs is the **Device** resource definitions). This permits `btape` to find the configuration parameters for your archive device (generally a tape drive). Without those parameters, the testing and utility programs do not know how to properly read and write your drive. By default, they use `bacula-sd.conf` in the current directory, but you may specify a different configuration file using the **-c** option.

3.1.3 Specifying a Device Name For a Tape

`btape` **device-name** where the Volume can be found. In the case of a tape, this is the physical device name such as `/dev/nst0` or `/dev/rmt/0ubn` depending on your system that you specify on the Archive Device directive. For the program to work, it must find the identical name in the Device resource of the configuration file. If the name is not found in the list of physical names, the utility program will compare the name you entered to the Device names (rather than the Archive device names).

When specifying a tape device, it is preferable that the "non-rewind" variant of the device file name be given. In addition, on systems such as Sun, which have multiple tape access methods, you must be sure to specify to use Berkeley I/O conventions with the device. The **b** in the Solaris (Sun) archive specification `/dev/rmt/0mbn` is what is needed in this case. Bacula does not support SysV tape drive behavior.

See below for specifying Volume names.

3.1.4 Specifying a Device Name For a File

If you are attempting to read or write an archive file rather than a tape, the **device-name** should be the full path to the archive location including the filename. The filename (last part of the specification) will be stripped and used as the Volume name, and the path (first part before the filename) must have the same entry in the configuration file. So, the path is equivalent to the archive device name, and the filename is equivalent to the volume name.

3.2 btape

This program permits a number of elementary tape operations via a `tty` command interface. The `test` command, described below, can be very useful for testing tape drive compatibility problems. Aside from initial testing of tape drive compatibility with **Bacula**, `btape` will be mostly used by developers writing new tape drivers.

btape can be dangerous to use with existing **Bacula** tapes because it will relabel a tape or write on the tape if so requested regardless of whether or not the tape contains valuable data, so please be careful and use it only on blank tapes.

To work properly, btape needs to read the Storage daemon's configuration file. As a default, it will look for bacula-sd.conf in the current directory. If your configuration file is elsewhere, please use the **-c** option to specify where.

The physical device name or the Device resource name must be specified on the command line, and this same device name must be present in the Storage daemon's configuration file read by btape

```
Usage: btape [options] device_name
       -b <file>   specify bootstrap file
       -c <file>   set configuration file to file
       -d <nn>     set debug level to nn
       -p          proceed inspite of I/O errors
       -s          turn off signals
       -v          be verbose
       -?          print this message.
```

3.2.1 Using btape to Verify your Tape Drive

An important reason for this program is to ensure that a Storage daemon configuration file is defined so that Bacula will correctly read and write tapes.

It is highly recommended that you run the test command before running your first Bacula job to ensure that the parameters you have defined for your storage device (tape drive) will permit **Bacula** to function properly. You only need to mount a blank tape, enter the command, and the output should be reasonably self explanatory. For example:

```
(ensure that Bacula is not running)
./btape -c /usr/bin/bacula/bacula-sd.conf /dev/nst0
```

The output will be:

```
Tape block granularity is 1024 bytes.
btape: btape.c:376 Using device: /dev/nst0
*
```

Enter the test command:

```
test
```

The output produced should be something similar to the following: I've cut the listing short because it is frequently updated to have new tests.

```
=== Append files test ===
This test is essential to Bacula.
I'm going to write one record  in file 0,
                   two records in file 1,
              and three records in file 2
btape: btape.c:387 Rewound /dev/nst0
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
btape: btape.c:410 Wrote EOF to /dev/nst0
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
```

```
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
btape: btape.c:410 Wrote EOF to /dev/nst0
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
btape: btape.c:855 Wrote one record of 64412 bytes.
btape: btape.c:857 Wrote block to device.
btape: btape.c:410 Wrote EOF to /dev/nst0
btape: btape.c:387 Rewound /dev/nst0
btape: btape.c:693 Now moving to end of media.
btape: btape.c:427 Moved to end of media
We should be in file 3. I am at file 3. This is correct!
Now the important part, I am going to attempt to append to the tape.
...
=== End Append files test ===
```

If you do not successfully complete the above test, please resolve the problem(s) before attempting to use **Bacula**. Depending on your tape drive, the test may recommend that you add certain records to your configuration. We strongly recommend that you do so and then re-run the above test to insure it works the first time.

Some of the suggestions it provides for resolving the problems may or may not be useful. If at all possible avoid using fixed blocking. If the test suddenly starts to print a long series of:

```
Got EOF on tape.
Got EOF on tape.
...
```

then almost certainly, you are running your drive in fixed block mode rather than variable block mode. See below for more help of resolving fix versus variable block problems.

It is also possible that you have your drive set in SysV tape drive mode. The drive must use BSD tape conventions. See the section above on setting your **Archive device** correctly.

For FreeBSD users, please see the notes below for doing further testing of your tape drive.

3.2.2 Testing tape drive speed

To determine the best configuration of your tape drive, you can run the `speed` command available in the `btape` program.

This command can have the following arguments:

`file_size=n` Specify the Maximum File Size for this test (between 1 and 5GB). This counter is in GB.

`nb_file=n` Specify the number of file to be written. The amount of data should be greater than your memory ($file\_size * nb\_file$).

`skip_zero` This flag permits to skip tests with constant data.

`skip_random` This flag permits to skip tests with random data.

`skip_raw` This flag permits to skip tests with raw access.

`skip_block` This flag permits to skip tests with Bacula block access.

```
*speed file_size=3 skip_raw
btape.c:1078 Test with zero data and bacula block structure.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
```

```
++++++++++++++++++++++++++++++++++++++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 44.128 MB/s
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 43.531 MB/s

btape.c:1090 Test with random data, should give the minimum throughput.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
++++++++++++++++++++++++++++++++++++++++++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 7.271 MB/s
++++++++++++++++++++++++++++++++++++++++++++
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 7.365 MB/s
```

When using compression, the random test will give your the minimum throughput of your drive . The test using constant string will give you the maximum speed of your hardware chain. (cpu, memory, scsi card, cable, drive, tape).

You can change the block size in the Storage Daemon configuration file.

3.2.3 Linux SCSI Tricks

You can find out what SCSI devices you have by doing:

```
lsscsi
```

Typical output is:

```
[0:0:0:0]    disk    ATA      ST3160812AS      3.AD  /dev/sda
[2:0:4:0]    tape    HP       Ultrium 2-SCSI   F6CH  /dev/st0
[2:0:5:0]    tape    HP       Ultrium 2-SCSI   F6CH  /dev/st1
[2:0:6:0]    mediumx OVERLAND LXB              0107  -
[2:0:9:0]    tape    HP       Ultrium 1-SCSI   E50H  /dev/st2
[2:0:10:0]   mediumx OVERLAND LXB              0107  -
```

There are two drives in one autochanger: /dev/st0 and /dev/st1 and a third tape drive at /dev/st2. For using them with Bacula, one would normally reference them as /dev/nst0 . . . /dev/nst2. Not also, there are two different autochangers identified as "mediumx OVERLAND LXB". They can be addressed via their /dev/sgN designation, which can be obtained by counting from the beginning as 0 to each changer. In the above case, the two changers are located on /dev/sg3 and /dev/sg5. The one at /dev/sg3, controls drives /dev/nst0 and /dev/nst1; and the one at /dev/sg5 controles drive /dev/nst2.

If you do not have the lsscsi command, you can obtain the same information as follows:

```
cat /proc/scsi/scsi
```

For the above example with the three drives and two autochangers, I get:

```
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3160812AS      Rev: 3.AD
  Type:   Direct-Access                    ANSI SCSI revision: 05
Host: scsi2 Channel: 00 Id: 04 Lun: 00
  Vendor: HP       Model: Ultrium 2-SCSI   Rev: F6CH
  Type:   Sequential-Access                ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 05 Lun: 00
  Vendor: HP       Model: Ultrium 2-SCSI   Rev: F6CH
```

```
  Type:   Sequential-Access               ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 06 Lun: 00
  Vendor: OVERLAND Model: LXB             Rev: 0107
  Type:   Medium Changer                  ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 09 Lun: 00
  Vendor: HP       Model: Ultrium 1-SCSI  Rev: E50H
  Type:   Sequential-Access               ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 10 Lun: 00
  Vendor: OVERLAND Model: LXB             Rev: 0107
  Type:   Medium Changer                  ANSI SCSI revision: 02
```

As an additional example, I get the following (on a different machine from the above example):

```
Attached devices:
Host: scsi2 Channel: 00 Id: 01 Lun: 00
  Vendor: HP       Model: C5713A          Rev: H107
  Type:   Sequential-Access               ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 04 Lun: 00
  Vendor: SONY     Model: SDT-10000       Rev: 0110
  Type:   Sequential-Access               ANSI SCSI revision: 02
```

The above represents first an autochanger and second a simple tape drive. The HP changer (the first entry) uses the same SCSI channel for data and for control, so in Bacula, you would use:

```
Archive Device = /dev/nst0
Changer Device = /dev/sg0
```

If you want to remove the SDT-10000 device, you can do so as `root` with:

```
echo "scsi remove-single-device 2 0 4 0">/proc/scsi/scsi
```

and you can put add it back with:

```
echo "scsi add-single-device 2 0 4 0">/proc/scsi/scsi
```

where the 2 0 4 0 are the Host, Channel, Id, and Lun as seen on the output from `cat /proc/scsi/scsi`. Note, the Channel must be specified as numeric.

Below is a slightly more complicated output, which is a single autochanger with two drives, and which operates the changer on a different channel from from the drives:

```
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: WDC WD1600JD-75H Rev: 08.0
  Type:   Direct-Access                   ANSI SCSI revision: 05
Host: scsi2 Channel: 00 Id: 04 Lun: 00
  Vendor: HP       Model: Ultrium 2-SCSI  Rev: F6CH
  Type:   Sequential-Access               ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 05 Lun: 00
  Vendor: HP       Model: Ultrium 2-SCSI  Rev: F6CH
  Type:   Sequential-Access               ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 06 Lun: 00
  Vendor: OVERLAND Model: LXB             Rev: 0106
  Type:   Medium Changer                  ANSI SCSI revision: 02
```

The above tape drives are accessed on `/dev/nst0` and `/dev/nst1`, while the control channel for those two drives is `/dev/sg3`.

3.3 Tips for Resolving Problems

3.3.1 Bacula Saves But Cannot Restore Files

If you are getting error messages such as:

```
Volume data error at 0:1! Wanted block-id: "BB02", got "". Buffer discarded
```

It is very likely that Bacula has tried to do block positioning and ended up at an invalid block. This can happen if your tape drive is in fixed block mode while Bacula's default is variable blocks. Note that in such cases, Bacula is perfectly able to write to your Volumes (tapes), but cannot position to read them.

There are two possible solutions.

1. The first and best is to always ensure that your drive is in variable block mode. Note, it can switch back to fixed block mode on a reboot or if another program uses the drive. So on such systems you need to modify the Bacula startup files to explicitly set:

   ```
   mt -f /dev/nst0 defblksize 0
   ```

   or whatever is appropriate on your system. Note, if you are running a Linux system, and the above command does not work, it is most likely because you have not loaded the appropriate `mt` package, which is often called `mt_st`, but may differ according to your distribution.

2. The second possibility, especially, if Bacula wrote while the drive was in fixed block mode, is to turn off block positioning in Bacula. This is done by adding:

   ```
   Block Positioning = no
   ```

   to the Device resource. This is not the recommended procedure because it can enormously slow down recovery of files, but it may help where all else fails. This directive is available in version 1.35.5 or later (and not yet tested).

If you are getting error messages such as:

```
Volume data error at 0:0!
Block checksum mismatch in block=0 len=32625 calc=345678 blk=123456
```

You are getting tape read errors, and this is most likely due to one of the following things:

1. An old or bad tape.
2. A dirty drive that needs cleaning (particularly for DDS drives).
3. A loose SCSI cable.
4. Old firmware in your drive. Make sure you have the latest firmware loaded.
5. Computer memory errors.
6. Over-clocking your CPU.
7. A bad SCSI card.

3.3.2 Bacula Cannot Open the Device

If you get an error message such as:

```
dev open failed: dev.c:265 stored: unable to open
device /dev/nst0:> ERR=No such device or address
```

the first time you run a job, it is most likely due to the fact that you specified the incorrect device name on your **Archive Device**.

If Bacula works fine with your drive, then all off a sudden you get error messages similar to the one shown above, it is quite possible that your driver module is being removed because the kernel deems it idle. This is done via `crontab` with the use of `rmmod -a`. To fix the problem, you can remove this entry from `crontab`, or you can manually `modprob` your driver module (or add it to the local startup script). Thanks to Alan Brown for this tip.

Sometimes the tape handler gets confused. This can happen if your tape drive door is locked (Bacula locks it when writing a tape), and Bacula crashes or some other problem occurs. After this kind of a problem, the kernel driver will not recognize that a tape is loaded and you may see messages such as:

```
19-Sep 16:39 groschat-sd JobId 16072: Warning: mount.c:219 Open of tape
device "LTO4-Drive" (/dev/tape/by-id/scsi-3500143801403cf22-nst) Volume
"AAJ372" failed: ERR=tape_dev.c:162 Unable to open device "LTO4-Drive"
(/dev/tape/by-id/scsi-3500143801403cf22-nst): ERR=No medium found

19-Sep 16:39 groschat-sd JobId 16072: Please mount append Volume "AAJ372"
or label a new one for:
    Job:          Rufus.2016-09-19_16.34.22_03
    Storage:      "LTO4-Drive" (/dev/tape/by-id/scsi-3500143801403cf22-nst)
    Pool:         Default
    Media type:   LTO-4
```

in that case, the best thing to do is unload the tape by hand with:

```
(stop Bacula)
mtx -f /dev/sgx unload
(restart Bacula)
```

where you replace /dev/sgx by your scsi control device name.

3.3.3 Incorrect File Number

When Bacula moves to the end of the medium, it normally uses the **ioctl(MTEOM)** function. Then Bacula uses the **ioctl(MTIOCGET)** function to retrieve the current file position from the **mt_fileno** field. Some SCSI tape drivers will use a fast means of seeking to the end of the medium and in doing so, they will not know the current file position and hence return a **-1**. As a consequence, if you get **"This is NOT correct!"** in the positioning tests, this may be the cause. You must correct this condition in order for Bacula to work.

There are two possible solutions to the above problem of incorrect file number:

- Figure out how to configure your SCSI driver to keep track of the file position during the MTEOM request. This is the preferred solution.
- Modify the **Device** resource of your `bacula-sd.conf` file to include:

  ```
  Hardware End of File = no
  ```

  This will cause Bacula to use the MTFSF request to seek to the end of the medium, and Bacula will keep track of the file number itself.

3.3.4 Incorrect Number of Blocks or Positioning Errors

Bacula's preferred method of working with tape drives (sequential devices) is to run in variable block mode, and this is what is set by default. You should first ensure that your tape drive is set for variable block mode (see below).

If your tape drive is in fixed block mode and you have told Bacula to use different fixed block sizes or variable block sizes (default), you will get errors when Bacula attempts to forward space to the correct block (the kernel driver's idea of tape blocks will not correspond to Bacula's).

All modern tape drives support variable tape blocks, but some older drives (in particular the QIC drives) as well as the ATAPI ide-scsi driver run only in fixed block mode. The Travan tape drives also apparently must run in fixed block mode (to be confirmed).

Even in variable block mode, with the exception of the first record on the second or subsequent volume of a multi-volume backup, Bacula will write blocks of a fixed size. However, in reading a tape, Bacula will assume that for each read request, exactly one block from the tape will be transferred. This the most common way that tape drives work and is well supported by **Bacula**.

Drives that run in fixed block mode can cause serious problems for Bacula if the drive's block size does not correspond exactly to **Bacula's** block size. In fixed block size mode, drivers may transmit a partial block or multiple blocks for a single read request. From **Bacula's** point of view, this destroys the concept of tape blocks. It is much better to run in variable block mode, and almost all modern drives (the OnStream is an exception) run in variable block mode. In order for Bacula to run in fixed block mode, you must include the following records in the Storage daemon's Device resource definition:

```
Minimum Block Size = nnn
Maximum Block Size = nnn
```

where **nnn** must be the same for both records and must be identical to the driver's fixed block size.

We recommend that you avoid this configuration if at all possible by using variable block sizes.

If you must run with fixed size blocks, make sure they are not 512 bytes. This is too small and the overhead that Bacula has with each record will become excessive. If at all possible set any fixed block size to something like 64,512 bytes or possibly 32,768 if 64,512 is too large for your drive. See below for the details on checking and setting the default drive block size.

To recover files from tapes written in fixed block mode, see below.

3.3.5 Ensuring that the Tape Modes Are Properly Set – Linux Only

If you have a modern SCSI tape drive and you are having problems with the `test` command as noted above, it may be that some program has set one or more of your SCSI driver's options to non-default values. For example, if your driver is set to work in SysV manner, Bacula will not work correctly because it expects BSD behavior. To reset your tape drive to the default values, you can try the following, but **ONLY** if you have a SCSI tape drive on a **Linux** system:

```
become super user
mt -f /dev/nst0 rewind
mt -f /dev/nst0 stoptions buffer-writes async-writes read-ahead
```

The above commands will clear all options and then set those specified. None of the specified options are required by Bacula, but a number of other options such as SysV behavior must not be set. Bacula does not support SysV tape behavior. On systems other than Linux, you will need to consult your `mt` man pages or documentation to figure out how to do the same thing.

This should not really be necessary though – for example, on both Linux and Solaris systems, the default tape driver options are compatible with Bacula. On Solaris systems, you must take care to specify the correct device name on the **Archive device** directive. See above for more details.

You may also want to ensure that no prior program has set the default block size, as happened to one user, by explicitly turning it off with:

```
mt -f /dev/nst0 defblksize 0
```

If you are running a Linux system, and the above command does not work, it is most likely because you have not loaded the appropriate `mt` package, which is often called `mt_st`, but may differ according to your distribution.

If you would like to know what options you have set before making any of the changes noted above, you can now view them on Linux systems, thanks to a tip provided by Willem Riede. Do the following:

```
become super user
mt -f /dev/nst0 stsetoptions 0
grep st0 /var/log/messages
```

and you will get output that looks something like the following:

```
kernel: st0: Mode 0 options: buffer writes: 1, async writes: 1, read ahead: 1
kernel: st0:    can bsr: 0, two FMs: 0, fast mteom: 0, auto lock: 0,
kernel: st0:    defs for wr: 0, no block limits: 0, partitions: 0, s2 log: 0
kernel: st0:    sysv: 0 nowait: 0
```

Note, I have chopped off the beginning of the line with the date and machine name for presentation purposes.

Some people find that the above settings only last until the next reboot, so please check this otherwise you may have unexpected problems.

Beginning with Bacula version 1.35.8, if Bacula detects that you are running in variable block mode, it will attempt to set your drive appropriately. All OSes permit setting variable block mode, but some OSes do not permit setting the other modes that Bacula needs to function properly.

3.3.6 Tape Hardware Compression and Blocking Size

You should be able to verify the tape compression status with sysfs on Linux.

```
cat /sys/class/scsi_tape/nst0/default_compression
```

You can, turn it on by using (on Linux):

```
become super user
mt -f /dev/nst0 defcompression 1
```

and of course, if you use a zero instead of the one at the end, you will turn it off.

If you have built the `mtx` program in the **depkgs** package, you can use `tapeinfo` to get quite a bit of information about your tape drive even if it is not an autochanger. This program is called using the SCSI control device. On Linux for tape drive `/dev/nst0`, this is usually `/dev/sg0`, while on FreeBSD for `/dev/nsa0`, the control device is often `/dev/pass2`. For example on my DDS-4 drive (`/dev/nst0`), I get the following:

```
tapeinfo -f /dev/sg0
Product Type: Tape Drive
Vendor ID: 'HP      '
Product ID: 'C5713A          '
Revision: 'H107'
Attached Changer: No
MinBlock:1
MaxBlock:16777215
SCSI ID: 5
SCSI LUN: 0
Ready: yes
BufferedMode: yes
Medium Type: Not Loaded
Density Code: 0x26
BlockSize: 0
```

where the **DataCompEnabled: yes** means that tape hardware compression is turned on. You can turn it on and off (yes|no) by using the `mt` commands given above. Also, this output will tell you if the **BlockSize** is non-zero and hence set for a particular block size. Bacula is not likely to work in such a situation because it will normally attempt to write blocks of 64,512 bytes, except the last block of the job which will generally be shorter. The first thing to try is setting the default block size to zero using the `mt -f /dev/nst0 defblksize 0` command as shown above. On FreeBSD, this would be something like: `mt -f /dev/nsa0 blocksize 0`.

On some operating systems with some tape drives, the amount of data that can be written to the tape and whether or not compression is enabled is determined by the density usually the `mt -f /dev/nst0 setdensity xxx` command. Often `mt -f /dev/nst0 status` will print out the current density code that is used with the drive. Most systems, but unfortunately not all, set the density to the maximum by default. On some systems, you can also get a list of all available density codes with: `mt -f /dev/nst0 densities` or a similar `mt` command. Note, for DLT and SDLT devices, no-compression versus compression is very often controlled by the density code. On FreeBSD systems, the compression mode is set using `mt -f /dev/nsa0 comp xxx` where xxx is the mode you want. In general, see `man mt` for the options available on your system.

Note, some of the above `mt` commands may not be persistent depending on your system configuration. That is they may be reset if a program other than Bacula uses the drive or, as is frequently the case, on reboot of your system.

If your tape drive requires fixed block sizes (very unusual), you can use the following records:

```
Minimum Block Size = nnn
Maximum Block Size = nnn
```

in your Storage daemon's Device resource to force Bacula to write fixed size blocks (where you sent nnn to be the same for both of the above records). This should be done only if your drive does not support variable block sizes, or you have some other strong reasons for using fixed block sizes. As mentioned above, a small fixed block size of 512 or 1024 bytes will be very inefficient. Try to set any fixed block size to something like 64,512 bytes or larger if your drive will support it.

Also, note that the **Medium Type** field of the output of `tapeinfo` reports **Not Loaded**, which is not correct. As a consequence, you should ignore that field as well as the **Attached Changer** field.

To recover files from tapes written in fixed block mode, see below.

3.3.7 Tape Modes on FreeBSD

On most FreeBSD systems such as 4.9 and most tape drives, Bacula should run with:

```
mt  -f  /dev/nsa0  seteotmodel  2
mt  -f  /dev/nsa0  blocksize   0
mt  -f  /dev/nsa0  comp  enable
```

You might want to put those commands in a startup script to make sure your tape driver is properly initialized before running Bacula, because depending on your system configuration, these modes may be reset if a program other than Bacula uses the drive or when your system is rebooted.

Then according to what the `btape test` command returns, you will probably need to set the following (see below for an alternative):

```
Hardware End of Medium = no
BSF at EOM = yes
Backward Space Record = no
Backward Space File = no
Fast Forward Space File = no
TWO EOF = yes
```

Then be sure to run some append tests with Bacula where you start and stop Bacula between appending to the tape, or use `btape` version 1.35.1 or greater, which includes simulation of stopping/restarting Bacula.

Please see the file `platforms/freebsd/pthreads-fix.txt` in the main Bacula directory concerning **important** information concerning compatibility of Bacula and your system. A much more optimal Device configuration is shown below, but does not work with all tape drives. Please test carefully before putting either into production.

Note, for FreeBSD 4.10-RELEASE, using a Sony TSL11000 L100 DDS4 with an autochanger set to variable block size and DCLZ compression, Brian McDonald reports that to get Bacula to append correctly between Bacula executions, the correct values to use are:

```
mt  -f  /dev/nsa0  seteotmodel  1
mt  -f  /dev/nsa0  blocksize  0
mt  -f /dev/nsa0  comp  enable
```

and

```
Hardware End of Medium = no
BSF at EOM = no
Backward Space Record = no
Backward Space File = no
Fast Forward Space File = yes
TWO EOF = no
```

This has been confirmed by several other people using different hardware. This configuration is the preferred one because it uses one EOF and no backspacing at the end of the tape, which works much more efficiently and reliably with modern tape drives.

Finally, here is a Device configuration that Danny Butroyd reports to work correctly with the Overland Powerloader tape library using LT0-2 and FreeBSD 5.4-Stable:

```
# Overland Powerloader LT02 - 17 slots single drive
Device {
  Name = Powerloader
  Media Type = LT0-2
  Archive Device = /dev/nsa0
  AutomaticMount = yes;
  AlwaysOpen = yes;
  RemovableMedia = yes;
  RandomAccess = no;
```

```
  Changer Command = "/opt/bacula/scripts/mtx-changer %c %o %S %a %d"
  Changer Device = /dev/pass2
  AutoChanger = yes
  Alert Command = "sh -c 'tapeinfo -f %c |grep TapeAlert|cat'"

  # FreeBSD Specific Settings
  Offline On Unmount = no
  Hardware End of Medium = no
  BSF at EOM = yes
  Backward Space Record = no
  Fast Forward Space File = no
  TWO EOF = yes
}
```

The following Device resource works fine with Dell PowerVault 110T and 120T devices on both FreeBSD 5.3 and on NetBSD 3.0. It also works with Sony AIT-2 drives on FreeBSD.

```
Device {
  ...
  # FreeBSD/NetBSD Specific Settings
  Hardware End of Medium = no
  BSF at EOM = yes
  Backward Space Record = no
  Fast Forward Space File = yes
  TWO EOF = yes
}
```

On FreeBSD version 6.0, it is reported that you can even set Backward Space Record = yes.

3.3.8 Finding your Tape Drives and Autochangers on FreeBSD

On FreeBSD, you can do a `camcontrol devlist` as `root` to determine what drives and autochangers you have. For example,

```
undef# camcontrol devlist
    at scbus0 target 2 lun 0 (pass0,sa0)
    at scbus0 target 4 lun 0 (pass1,sa1)
    at scbus0 target 4 lun 1 (pass2)
```

from the above, you can determine that there is a tape drive on `/dev/sa0` and another on `/dev/sa1` in addition since there is a second line for the drive on `/dev/sa1`, you know can assume that it is the control device for the autochanger (i.e. `/dev/pass2`). It is also the control device name to use when invoking the tapeinfo program. E.g.

```
tapeinfo -f /dev/pass2
```

3.3.9 Using the OnStream driver on Linux Systems

Bacula version 1.33 (not 1.32x) is now working and ready for testing with the OnStream kernel osst driver version 0.9.14 or above. Osst is available from: sourceforge.net/projects/osst.

To make Bacula work you must first load the new driver then, as `root`, do:

```
mt -f /dev/nosst0 defblksize 32768
```

Also you must add the following to your Device resource in your Storage daemon's conf file:

```
Minimum Block Size = 32768
Maximum Block Size = 32768
```

Here is a Device specification provided by Michel Meyers that is known to work:

```
Device {
  Name = "Onstream DI-30"
  Media Type = "ADR-30"
  Archive Device = /dev/nosst0
  Minimum Block Size = 32768
  Maximum Block Size = 32768
  Hardware End of Medium = yes
  BSF at EOM = no
  Backward Space File = yes
  Fast Forward Space File = yes
  Two EOF = no
  AutomaticMount = yes
  AlwaysOpen = yes
  Removable Media = yes
}
```

3.4 Hardware Compression on EXB-8900

To active, check, or disable the hardware compression feature on an EXB-8900, use the exabyte MammothTool. You can get it here: www.exabyte.com/support/online/downloads/index.cfm. There is a Solaris version of this tool. With option -C 0 or 1 you can disable or activate compression. Start this tool without any options for a small reference.

3.4.1 Using btape to Simulate Filling a Tape

Because there are often problems with certain tape drives or systems when end of tape conditions occur, `btape` has a special command `fill` that causes it to write random data to a tape until the tape fills. It then writes at least one more Bacula block to a second tape. Finally, it reads back both tapes to ensure that the data has been written in a way that Bacula can recover it. Note, there is also a single tape option as noted below, which you should use rather than the two tape test. See below for more details.

This can be an extremely time consuming process (here it is about 6 hours) to fill a full tape. Note, that btape writes random data to the tape when it is filling it. This has two consequences:

1. it takes a bit longer to generate the data, especially on slow CPUs.
2. the total amount of data is approximately the real physical capacity of your tape, regardless of whether or not the tape drive compression is on or off. This is because random data does not compress very much.

To begin this test, you enter the `fill` command and follow the instructions. There are two options: the simple single tape option and the multiple tape option. Please use only the simple single tape option because the multiple tape option still doesn't work totally correctly. If the single tape option does not succeed, you should correct the problem before using Bacula.

3.5 Recovering Files Written With Fixed Block Sizes

If you have been previously running your tape drive in fixed block mode (default 512) and Bacula with variable blocks (default), then in version 1.32f-x and 1.34 and above, Bacula will fail to

recover files because it does block spacing, and because the block sizes don't agree between your tape drive and Bacula it will not work.

The long term solution is to run your drive in variable block mode as described above. However, if you have written tapes using fixed block sizes, this can be a bit of a pain. The solution to the problem is: while you are doing a restore command using a tape written in fixed block size, ensure that your drive is set to the fixed block size used while the tape was written. Then when doing the `restore` command in the Console program, do not answer the prompt **yes/mod/no**. Instead, edit the bootstrap file (the location is listed in the prompt) using any ASCII editor. Remove all **VolBlock** lines in the file. When the file is re-written, answer the question, and Bacula will run without using block positioning, and it should recover your files.

3.6 Tape Blocking Modes

SCSI tapes may either be written in **variable** or **fixed** block sizes. Newer drives support both modes, but some drives such as the QIC devices always use fixed block sizes. Bacula attempts to fill and write complete blocks (default 65K), so that in normal mode (variable block size), Bacula will always write blocks of the same size except the last block of a Job. If Bacula is configured to write fixed block sizes, it will pad the last block of the Job to the correct size. Bacula expects variable tape block size drives to behave as follows: Each write to the drive results in a single record being written to the tape. Each read returns a single record. If you request less bytes than are in the record, only those number of bytes will be returned, but the entire logical record will have been read (the next read will retrieve the next record). Thus data from a single write is always returned in a single read, and sequentially written records are returned by sequential reads.

Bacula expects fixed block size tape drives to behave as follows: If a write length is greater than the physical block size of the drive, the write will be written as two blocks each of the fixed physical size. This single write may become multiple physical records on the tape. (This is not a good situation). According to the documentation, one may never write an amount of data that is not the exact multiple of the blocksize (it is not specified if an error occurs or if the the last record is padded). When reading, it is my understanding that each read request reads one physical record from the tape. Due to the complications of fixed block size tape drives, you should avoid them if possible with Bacula, or you must be ABSOLUTELY certain that you use fixed block sizes within Bacula that correspond to the physical block size of the tape drive. This will ensure that Bacula has a one to one correspondence between what it writes and the physical record on the tape.

Please note that Bacula will not function correctly if it writes a block and that block is split into two or more physical records on the tape. Bacula assumes that each write causes a single record to be written, and that it can sequentially recover each of the blocks it has written by using the same number of sequential reads as it had written.

3.7 Details of Tape Modes

Rudolf Cejka has provided the following information concerning certain tape modes and MTEOM.

Tape level It is always possible to position filemarks or blocks, whereas positioning to the end-of-data is only optional feature, however it is implemented very often. SCSI specification also talks about optional sequential filemarks, setmarks and sequential setmarks, but these are not implemented so often. Modern tape drives keep track of file positions in built-in chip (AIT, LTO) or at the beginning of the tape (SDLT), so there is not any speed difference, if end-of-data or filemarks is used (I have heard, that LTO-1 from all 3 manufacturers do not use its chip for file locations, but a tape as in SDLT case, and I'm not sure about LTO-2 and LTO-3 case). However there is a big difference, that end-of-data ignores file position,

whereas filemarks returns the real number of skipped files, so OS can track current file number just in filemarks case.

OS level Solaris does use just SCSI SPACE Filemarks, it does not support SCSI SPACE End-of-data. When MTEOM is called, Solaris does use SCSI SPACE Filemarks with count = 1048576 for fast mode, and combination of SCSI SPACE Filemarks with count = 1 with SCSI SPACE Blocks with count = 1 for slow mode, so EOD mark on the tape on some older tape drives is not skipped. File number is always tracked for MTEOM.

Linux does support both SCSI SPACE Filemarks and End-of-data: When MTEOM is called in MT_ST_FAST_MTEOM mode, SCSI SPACE End-of-data is used. In the other case, SCSI SPACE Filemarks with count = 8388607 is used. There is no real slow mode like in Solaris - I just expect, that for older tape drives Filemarks may be slower than End-of-data, but not so much as in Solaris slow mode. File number is tracked for MTEOM just without MT_ST_FAST_MTEOM - when MT_ST_FAST_MTEOM is used, it is not.

FreeBSD does support both SCSI SPACE Filemarks and End-of-data, but when MTEOD (MTEOM) is called, SCSI SPACE End-of-data is always used. FreeBSD never use SCSI SPACE Filemarks for MTEOD. File number is never tracked for MTEOD.

Bacula level When **Hardware End of Medium = Yes** is used, MTEOM is called, but it does not mean, that hardware End-of-data must be used. When Hardware End of Medium = No, if Fast Forward Space File = Yes, MTFSF with count = 32767 is used, else Block Read with count = 1 with Forward Space File with count = 1 is used, which is really very slow.

Hardware End of Medium = Yes|No The name of this option is misleading and is the source of confusion, because it is not the hardware EOM, what is really switched here.

If I use Yes, OS must not use SCSI SPACE End-of-data, because Bacula expects, that there is tracked file number, which is not supported by SCSI specification. Instead, the OS have to use SCSI SPACE Filemarks.

If I use No, an action depends on Fast Forward Space File.

When I set **Hardware End of Medium = no** and **Fast Forward Space File = no** file positioning was very slow on my LTO-3 (about ten to 100 minutes), but

with **Hardware End of Medium = no** and **Fast Forward Space File = yes**, the time is ten to 100 times faster (about one to two minutes).

3.8 Tape Performance Problems

If you have LTO-3 or LTO-4 drives, you should be able to fairly good transfer rates; from 60 to 150 MB/second, providing you have fast disks; GigaBit Ethernet connections (probably 2); you are running multiple simultaneous jobs; you have Bacula data spooling enabled; your tape block size is set to 131072 or 262144; and you have set **Maximum File Size = 5G**.

If you are not getting good performance, consider some of the following suggestions from the Allen Balck on the Bacula Users email list:

1. You are using an old HBA (i.e. SCSI-1, which only does 5 MB/s)
2. There are other, slower, devices on the SCSI bus. The HBA will negotiate the speed of every device down to the speed of the slowest.
3. There is a termination problem on the bus (either too much or too little termination). The HBA will drop the bus speed in an attempt to increase the reliability of the bus.
4. Loose or damaged cabling - this will probably make the HBA "think" you have a termination problem and it will react as in 3 above.

See if /var/adm/messages (or /var/log/messages) tells you what the sync rate of the SCSI devices/bus are. Also, the next time you reboot, the BIOS may be able to tell you what the rate of each device is.

3.9 Autochanger Errors

If you are getting errors such as:

```
3992 Bad autochanger "load slot 1, drive 1": ERR=Child exited with code 1.
```

and you are running your Storage daemon as non-root, then most likely you are having permissions problems with the control channel. Running as root, set permissions on /dev/sgX so that the userid and group of your Storage daemon can access the device. You need to ensure that you all access to the proper control device, and if you don't have any SCSI disk drives (including SATA drives), you might want to change the permissions on /dev/sg*.

3.10 Syslog Errors

If you are getting errors such as:

```
: kernel: st0: MTSETDRVBUFFER only allowed for root
```

you are most likely running your Storage daemon as non-root, and Bacula is attempting to set the correct OS buffering to correspond to your Device resource. Most OSes allow only root to issue this ioctl command. In general, the message can be ignored providing you are sure that your OS parameters are properly configured as described earlier in this manual. If you are running your Storage daemon as root, you should not be getting these system log messages, and if you are, something is probably wrong.

Chapter 4

Dealing with Firewalls

If you have a firewall or a DMZ installed on your computer, you may experience difficulties contacting one or more of the Clients to back them up. This is especially true if you are trying to backup a Client across the Internet.

4.1 Technical Details

If you are attempting to do this, the sequence of network events in Bacula to do a backup are the following:

```
Console -> DIR:9101
DIR     -> SD:9103
DIR     -> FD:9102
FD      -> SD:9103
```

Where hopefully it is obvious that DIR represents the Director, FD the File daemon or client, and SD the Storage daemon. The numbers that follow those names are the standard ports used by Bacula, and the $\rightarrow$ represents the left side making a connection to the right side (i.e. the right side is the "server" or is listening on the specified port), and the left side is the "client" that initiates the conversation.

Note, port 9103 serves both the Director and the File daemon, each having its own independent connection.

If you are running `iptables`, you might add something like:

```
-A FW-1-INPUT -m state --state NEW -m tcp -p tcp --dport 9101:9103 -j ACCEPT
```

on your server, and

```
-A FW-1-INPUT -m state --state NEW -m tcp -p tcp --dport 9102 -j ACCEPT
```

on your client. In both cases, I assume that the machine is allowed to initiate connections on any port. If not, you will need to allow outgoing connections on ports 9102 and 9103 on your server and 9103 on your client. Thanks to Raymond Norton for this tip.

4.2 A Concrete Example

The following discussion was originally written by Jesse Guardiani because he has "internal" and "external" requiring the Director and the Client to use different IP addresses. His original solution was to define two different Storage resources in the Director's conf file each pointing to the same Storage daemon but with different IP addresses. In Bacula 1.38.x this no longer works, because Bacula makes a one-to-one association between a Storage daemon resource and a Device (such as an Autochanger). As a consequence, I have modified his original text to a method that I believe will work, but is as of yet untested (KES - July 2006).

My bacula server is on the `192.168.1.0/24` network at IP address `192.168.1.52`. For the sake of discussion we will refer to this network as the "internal" network because it connects to the internet through a NAT'd firewall. We will call the network on the public (internet) side of the NAT'd firewall the "external" network. Also, for the sake of discussion we will call my Bacula server:

```
server.int.mydomain.tld
```

when a fully qualified domain name is required, or simply:

```
server
```

if a hostname is adequate. We will call the various bacula daemons running on the `server.int.mydomain.tld` machine:

```
server-fd
server-sd
server-dir
```

In addition, I have two clients that I want to back up with Bacula. The first client is on the internal network. Its fully qualified domain name is:

```
private1.int.mydomain.tld
```

And its hostname is:

```
private1
```

This machine is a client and therefore runs just one bacula daemon:

```
private1-fd
```

The second client is on the external network. Its fully qualified domain name is:

```
public1.mydomain.tld
```

And its hostname is:

```
public1
```

This machine also runs just one bacula daemon:

```
public1-fd
```

Finally, I have a NAT firewall/gateway with two network interfaces. The first interface is on the internal network and serves as a gateway to the internet for all the machines attached to the internal network (For example, `server.int.mydomain.tld` and `private1.int.mydomain.tld`). The second interface is on the external (internet) network. The external interface has been assigned the name:

```
firewall.mydomain.tld
```

Remember:

```
*.int.mydomain.tld = internal network
*.mydomain.tld = external network
```

4.2.1 The Bacula Configuration Files for the Above

`server-sd` manages a 4 tape AIT autoloader. All of my backups are written to `server-sd`. I have just **one** Device resource in my `server-sd.conf` file:

```
Autochanger {
  Name = "autochanger1"
  Device = Drive0
  Changer Device = /dev/ch0
  Changer Command = "/usr/local/sbin/chio-bacula %c %o %S %a"
}
Device {
  Name = Drive0
  DriveIndex = 0
  Media Type = AIT-1;
  Archive Device = /dev/nrsa1
  Label Media = yes
  AutoChanger = yes
  AutomaticMount = yes               # when device opened, read it
  AlwaysOpen = yes
  Hardware End of Medium = No
  Fast Forward Space File = No
  BSF at EOM = yes
}
```

(note, please see the Tape Testing chapter of this manual for important FreeBSD information.) However, unlike previously, there is only one Storage definition in my `server-dir.conf` file:

```
Storage {
  Name = "autochanger1"    # Storage device for backing up
  Address = Storage-server
  SDPort = 9103
  Password = "mysecretpassword"
  Device = "autochanger1"
  Media Type = AIT-1
  Autochanger = yes
}
```

Note that the Storage resource uses neither of the two addresses to the Storage daemon – neither `server.int.mydomain.tld` nor `firewall.mydomain.tld`, but instead uses the address Storage-server.

What is key is that in the internal net, Storage-server is resolved to `server.int.mydomain.tld`, either with an entry in `/etc/hosts`, or by creating and appropriate DNS entry, and on the external net (the Client machine), Storage-server is resolved to `firewall.mydomain.tld`.

In addition to the above, I have two Client resources defined in `server-dir.conf`:

```
Client {
  Name = private1-fd
  Address = private1.int.mydomain.tld
  FDPort = 9102
  Catalog = MyCatalog
  Password = "mysecretpassword"       # password for FileDaemon
}
Client {
  Name = public1-fd
  Address = public1.mydomain.tld
  FDPort = 9102
  Catalog = MyCatalog
  Password = "mysecretpassword"       # password for FileDaemon
}
```

And finally, to tie it all together, I have two Job resources defined in `server-dir.conf`:

```
Job {
  Name = "Private1-Backup"
  Type = Backup
  Client = private1-fd
  FileSet = "Private1"
  Schedule = "WeeklyCycle"
  Storage = "autochanger1-int"
  Messages = Standard
  Pool = "Weekly"
  Write Bootstrap = "/var/db/bacula/Private1-Backup.bsr"
  Priority = 12
}
Job {
  Name = "Public1-Backup"
  Type = Backup
  Client = public1-fd
  FileSet = "Public1"
  Schedule = "WeeklyCycle"
  Storage = "autochanger1-ext"
  Messages = Standard
  Pool = "Weekly"
  Write Bootstrap = "/var/db/bacula/Public1-Backup.bsr"
  Priority = 13
}
```

It is important to notice that because the "Private1-Backup" Job is intended to back up a machine on the internal network so it resolves Storage-server to contact the Storage daemon via the internal net. On the other hand, the "Public1-Backup" Job is intended to back up a machine on the external network, so it resolves Storage-server to contact the Storage daemon via the external net.

I have left the Pool, Catalog, Messages, FileSet, Schedule, and Director resources out of the above `server-dir.conf` examples because they are not pertinent to the discussion.

4.2.2 How Does It Work?

If I want to run a backup of `private1.int.mydomain.tld` and store that backup using `server-sd` then my understanding of the order of events is this:

1. I execute my Bacula `bconsole` command on `server.int.mydomain.tld`.
2. console connects to `server-dir`.
3. I tell console to `run` backup Job "Private1-Backup".
4. console relays this command to `server-dir`.

5. server-dir connects to private1-fd at private1.int.mydomain.tld:9102
6. server-dir tells private1-fd to start sending the files defined in the "Private1-Backup" Job's FileSet resource to the Storage resource "autochanger1", which we have defined in server-dir.conf as having the address:port of Storage-server, which is mapped by DNS to server.int.mydomain.tld.
7. private1-fd connects to server.int.mydomain.tld:9103 and begins sending files.

Alternatively, if I want to run a backup of public1.mydomain.tld and store that backup using server-sd then my understanding of the order of events is this:

1. I execute my Bacula bconsole command on server.int.mydomain.tld.
2. console connects to server-dir.
3. I tell console to run backup Job "Public1-Backup".
4. console relays this command to server-dir.
5. server-dir connects, through the NAT'd firewall, to public1-fd at public1.mydomain.tld:9102
6. server-dir tells public1-fd to start sending the files defined in the "Public1-Backup" Job's FileSet resource to the Storage resource "autochanger1", which we have defined in server-dir.conf as having the same address:port as above of Storage-server, but which on this machine is resolved to firewall.mydomain.tld:9103.
7. public1-fd connects to firewall.mydomain.tld:9103 and begins sending files.

4.2.3 Important Note

In order for the above "Public1-Backup" Job to succeed, firewall.mydomain.tld:9103 **must** be forwarded using the firewall's configuration software to server.int.mydomain.tld:9103. Some firewalls call this "Server Publication". Others may call it "Port Forwarding".

4.2.4 Firewall Problems

Either a firewall or a router may decide to timeout and terminate open connections if they are not active for a short time. By Internet standards the period should be two hours, and should be indefinitely extended if KEEPALIVE is set as is the case by Bacula. If your firewall or router does not respect these rules, you may find Bacula connections terminated. In that case, the first thing to try is turning on the **Heart Beat Interval** both in the File daemon and the Storage daemon and set an interval of say five minutes.

Also, if you have denial of service rate limiting in your firewall, this too can cause Bacula disconnects since Bacula can at times use very high access rates. To avoid this, you should implement default accept rules for the Bacula ports involved before the rate limiting rules.

Finally, if you have a Windows machine, it will most likely by default disallow connections to the Bacula Windows File daemon. See the Windows chapter of this manual for additional details.

Chapter 5

What To Do When Bacula Crashes (Kaboom)

If you are running on a Linux system, and you have a set of working configuration files, it is very unlikely that **Bacula** will crash. As with all software, however, it is inevitable that someday, it may crash, particularly if you are running on another operating system or using a new or unusual feature.

This chapter explains what you should do if one of the three **Bacula** daemons (Director, File, Storage) crashes. When we speak of crashing, we mean that the daemon terminates abnormally because of an error. There are many cases where Bacula detects errors (such as PIPE errors) and will fail a job. These are not considered crashes. In addition, under certain conditions, Bacula will detect a fatal in the configuration, such as lack of permission to read/write the working directory. In that case, Bacula will force itself to crash with a SEGFAULT. However, before crashing, Bacula will normally display a message indicating why. For more details, please read on.

5.1 Traceback

Each of the three Bacula daemons has a built-in exception handler which, in case of an error, will attempt to produce a traceback. If successful the traceback will be emailed to you.

For this to work, you need to ensure that a few things are setup correctly on your system:

1. You must have a version of Bacula built with debug information turned on and not stripped of debugging symbols.
2. You must have an installed copy of `gdb` (the GNU debugger), and it must be on **Bacula's** path. On some systems such as Solaris, `gdb` may be replaced by `dbx`.
3. The Bacula installed script file `btraceback` must be in the same directory as the daemon which dies, and it must be marked as executable.
4. The script file `btraceback.gdb` must have the correct path to it specified in the `btraceback` file.
5. You must have a `mail` program which is on **Bacula's** path. By default, this `mail` program is set to `bsmtp`, so it must be correctly configured.
6. If you run either the Director or Storage daemon under a non-root userid, you will most likely need to modify the `btraceback` file to do something like `sudo` (raise to root priority) for the call to `gdb` so that it has the proper permissions to debug Bacula.

If all the above conditions are met, the daemon that crashes will produce a traceback report and email it to you. If the above conditions are not true, you can either run the debugger by hand as described below, or you may be able to correct the problems by editing the btraceback file. I recommend not spending too much time on trying to get the traceback to work as it can be very difficult.

The changes that might be needed are to add a correct path to the gdb program, correct the path to the btraceback.gdb file, change the mail program or its path, or change your email address. The key line in the btraceback file is:

```
gdb -quiet -batch -x /home/kern/bacula/bin/btraceback.gdb \
     $1 $2 2>\&1 | bsmtp -s "Bacula traceback" your-address@xxx.com
```

Since each daemon has the same traceback code, a single btraceback file is sufficient if you are running more than one daemon on a machine.

5.2 Testing The Traceback

To "manually" test the traceback feature, you simply start **Bacula** then obtain the **PID** of the main daemon thread (there are multiple threads). The output produced here will look different depending on what OS and what version of the kernel you are running. Unfortunately, the output had to be split to fit on this page:

```
[kern@rufus kern]$ ps fax --columns 132 | grep bacula-dir
 2103 ?        S      0:00 /home/kern/bacula/k/src/dird/bacula-dir -c
                                 /home/kern/bacula/k/src/dird/dird.conf
 2104 ?        S      0:00  \_ /home/kern/bacula/k/src/dird/bacula-dir -c
                                 /home/kern/bacula/k/src/dird/dird.conf
 2106 ?        S      0:00      \_ /home/kern/bacula/k/src/dird/bacula-dir -c
                                 /home/kern/bacula/k/src/dird/dird.conf
 2105 ?        S      0:00      \_ /home/kern/bacula/k/src/dird/bacula-dir -c
                                 /home/kern/bacula/k/src/dird/dird.conf
```

which in this case is 2103. Then while Bacula is running, you call the program giving it the path to the Bacula executable and the **PID**. In this case, it is:

```
./btraceback /home/kern/bacula/k/src/dird 2103
```

It should produce an email showing you the current state of the daemon (in this case the Director), and then exit leaving **Bacula** running as if nothing happened. If this is not the case, you will need to correct the problem by modifying the btraceback script.

Typical problems might be that gdb or dbx for Solaris is not on the default path. Fix this by specifying the full path to it in the btraceback file. Another common problem is that you haven't modified the script so that the bsmtp program has an appropriate smtp server or the proper syntax for your SMTP server. If you use the mail program and it is not on the default path, it will also fail. On some systems, it is preferable to use Mail rather than mail.

5.3 Getting A Traceback On Other Systems

It should be possible to produce a similar traceback on systems other than Linux, either using gdb or some other debugger. Solaris with dbx loaded works quite fine. On other systems, you will need to modify the btraceback program to invoke the correct debugger, and possibly correct the btraceback.gdb script to have appropriate commands for your debugger. If anyone

succeeds in making this work with another debugger, please send us a copy of what you modified. Please keep in mind that for any debugger to work, it will most likely need to run as root, so you may need to modify the btraceback script accordingly.

5.4 Manually Running Bacula Under The Debugger

If for some reason you cannot get the automatic traceback, or if you want to interactively examine the variable contents after a crash, you can run Bacula under the debugger. Assuming you want to run the Storage daemon under the debugger (the technique is the same for the other daemons, only the name changes), you would do the following:

1. Start the Director and the File daemon. If the Storage daemon also starts, you will need to find its PID as shown above (ps fax | grep bacula-sd) and kill it with a command like the following:

   ```
   kill -15 PID
   ```

   where you replace **PID** by the actual value.
2. At this point, the Director and the File daemon should be running but the Storage daemon should not.
3. cd to the directory containing the Storage daemon
4. Start the Storage daemon under the debugger:

   ```
   gdb ./bacula-sd
   ```

5. Run the Storage daemon:

   ```
   run -s -f -c ./bacula-sd.conf
   ```

   You may replace the ./bacula-sd.conf with the full path to the Storage daemon's configuration file.
6. At this point, Bacula will be fully operational.
7. In another shell command window, start the Console program and do what is necessary to cause Bacula to die.
8. When Bacula crashes, the gdb shell window will become active and gdb will show you the error that occurred.
9. To get a general traceback of all threads, issue the following command:

   ```
   thread apply all bt
   ```

   After that you can issue any debugging command.

5.5 Getting Debug Output from Bacula

Each of the daemons normally has debug compiled into the program, but disabled. There are two ways to enable the debug output. One is to add the **-d nnn** option on the command line when starting the debugger. The **nnn** is the debug level, and generally anything between 50 and 200 is reasonable. The higher the number, the more output is produced. The output is written to standard output.

The second way of getting debug output is to dynamically turn it on using the Console using the setdebug command. The full syntax of the command is:

```
setdebug level=nnn client=client-name storage=storage-name dir
```

If none of the options are given, the command will prompt you. You can selectively turn on/off debugging in any or all the daemons (i.e. it is not necessary to specify all the components of the above command).

Appendices

Appendix A

Acronyms

ACL Access Control List

API Application Programming Interface

DHCP Dynamic Host Configuration Protocole

DMZ Demilitarized Zone

DNS Domain Name Service

EOF End Of File

FAQ Frequently Asked Questions

GNU Gnu is not Unix

NAT Network Address Translation

NFS Network File System

NSR Network Storage Router

SCSI Small Computer System Interface

SMTP Simple Mail Transfer Protocol

SQL Structured Query Language

Index

F

G

H

I

J

K

L

M

N

O

P

Q

R

S