BACKFS/filer

Version 3
Date 2008-02-08

MIZZI Computer Software GmbH
Scheffelstrasse 27b, 68782 Brühl
Tel: 06202/920241, Fax: 06202/920243

Table of Contents

1. Copyright

2. How to use this manual

This manual gives an overview over the new BACKFS backup utility. The description of the tool assures that the reader is familiar with basic backup strategies.

3. Changes

Version 2.2:

Bug fixes in copy mode if user is not root.
Defaults for the version directory in copy, full, and incremental mode.

Version 3.0:

New commands for dump to standard output and restore from standard input.
New command for comparing directories on different volumes.

Version 3.2:

Various bug fixes in acl's for SUSE Sles9.
New Commands -F and -C. -dv now available in disk copy mode

Version 3.4:

New dump format version 2 (incopatible to version 1).
Destination file header check during -r command.

4. Preface

BACKFS is a flexible, easy-to-use backup utility designed to backup and restore data files on a file by file basis. It provides simple copy of a file system, a full backup copy, and an incremental backup strategy and is designed to backup large volumes as well as small directories of only a few hundred files. Therefore it is useful not only for professionals user, BACKFS is nevertheless a perfect solution for small home offices too.

5. Introduction

There are several backup methods used today. One of these is the backup to tape. This operation seems sufficient but during the last 10 years problems appeared with the backup to tape. Disk storage was growing much more than the capacity of the tapes. 30 years ago a single big reel tape (130 MB) has stored the content of a single disk (100 MB). These days a single tape can store about 200 GB but standard disks configured as a RAID set have a size up to 5 TB on a single file system. This requires about 25 tapes to store the content of a single RAID set. Concerning the backup time it is getting worse. 30 years ago 10-15 min were necessary to write one tape today one needs 2.5 hours per tape. This leads to backup times of 50 hours for a single full backup and a tape library is needed in any case. Incremental backup is not such a big problem due to the fact that the number of files which has been changed every day has not been increased in the same order as the size of the disk systems. Also if the backup seems to be possible the restoration of a disk or RAID set seems to be not. If your /home directory crashes the restoration of that disk with a tape library will need 50-100 hours. This is not acceptable in most cases.

What about solutions? Many administrators think that two copies of the same disk should be the best solutions and are using a mirror (RAID1) set. These copies are usually located in two different buildings (to avoid data loss due to fire) and use a simple block copy instead of a file copy. It seems the basic problem is solved, but others are still present! Firstly, there is no backup in case of a file system crash due to software errors because the copied file system will crash also and secondly there is no version control. This means that the user can not retrieve any version of a file which was saved during the last backups. Due to this reasons, a backup to tape is additionally necessary.

All these problems have been driving the development of BACKFS. This utility will create an image of a disk on a file by file basis. BACKFS can spread files over different type of file system and provides version control. In case of a disk crash the backup copy can be mounted immediately and your last backup is available after a few minutes. BACKFS is also much faster than any other incremental backup. Compared to other tape backup utilities, BACKFS is sometimes 20 times faster. Therefore, it is possible to perform an incremental backup many times a day. Our test machine contains a file system with 250000 files and 10 GB data. This file system can be backed up (incrementally) to a NFS-

mounted file system within 20 seconds.

At this time you may think “all of that I can do without BACKFS”, but other tools provide part of the functionality, but none comprise all of them.

“tar” fails to do incremental copies at all (new and old style).

“cpio” does not delete files.

“dump/restore” does not delete files too.

All of them do not detect moved directories and none can move overwritten files to version directories (version control). If the source disk crash while being backed up, the file which is currently copied is lost. If you want to have only all files copied rather than an image of your disk use one of the above programs and wait for the first crash .....

You might think BACKFS is necessary for enterprise computing only, but BACKFS is the ideal backup for your home machine too. If you own a second disk or an external USB Disk you can backup your main disk within of 20 seconds. No burning of CD's or DVD's is required and full version control is available too. Just call BACKFS before you shutdown your PC!

6. Overview

BACKFS was completely rewritten with respect to it's predecessor version. Because BACKFS provides more functionality the command name was changed to "`filer`". It can now make a single copy, a full and incremental backup and is able to count files and sizes of directories (like the UNIX command du). It supports ACL (access control list) on LINUX and a few inconsistencies which were present in the first version were removed. The speed for the incremental backup has been increased. It's now up to six times faster than before.

7. Commands

There is a single command only, it's name is "`filer`". The utility is statically linked and can be copied to any directory of your choice (/usr/bin, /usr/local/bin, /root). The utility may also be used by standard users (not root) but when backing up whole file systems it is strongly recommended to be root. There is no on-line man page but you may get some help about the syntax and the available options if you start "`filer`" with the command "`filer -?`".

filer -? will show the following output:

```
Version:

   BACKFS/filer xxx/xxx/xxx - Copyright MIZZI Computer Software GmbH


Description:

   Creates a backup copy of a directory
   and saves existing files to a version directory

   Displays the file usage of a directory tree


Usage:

   filer [options] pathname1 [pathname2] [pathname3] ...


General and command options:

   -?       show this help

   -v       increase verbose level
   -vl n    set verbose level to n
   -V       enhanced verbosity

   -c       copy pathname1 to pathname2, versions to pathname3
   -f       full backup pathname1 to pathname2, versions to pathname3
   -i       incremental backup pathname1 to pathname2, versions to pathname3
   -C       like -c but status time changes will cause a copy
   -F       like -f but status time changes will cause a copy
   -D       full dump, backup of pathname1 to standard output
   -d       incremental dump, backup of pathname1 to standard output
   -r       restore dump to pathname1, versions to pathname2
   -x       extract from dump to pathname1 and match pathname2
   -l       list content of dump, read from stdin
   -L       same as -l but long format
   -S       print crc summary of pathname1 to stdout
   -s       sum up of files and directories specified by pahtnames

   Defaults: -s -vl 1


Sub-options for the -S command:

   -nocrc   do not calculate content checksum, much faster
```

```
Sub-options for the -s command:

   -a       show allocated sizes
   -k       show sizes in kBytes (1000)
   -K       show sizes in KBytes (1024)
   -m       show sizes in megaBytes (1000*1000)
   -M       show sizes in MegaBytes (1024*1024)
   -md n    print dir/files up to max. depth n (default=1024)
   -nds     do not sum up the size of directories


Sub-options for the -f -i -c -r -d -D commands:

   -md n    print directory logs up to max. depth of n (default=1024)
   -aoe     abort on error
   -coe     continue on error (default)
   -e f     exclude any file with name f in any directory
   -e /f    exclude any path matching exact /f relative to source path
   -acl     copy acl entries (default, acl must be enabled)
   -noacl   do not copy acl entries


Sub-options for the -c -C -f -F -i -r commands:

   -dv      delete updated version files, but hold deleted files


Sub-options for the -f -i -d -D -S commands:

   -t fn    use fn as path for the status/tag file
            default=<pathname2>.backupdate
```

7.1. Command -s (Display file usage)

This -s command is used to sum up the sizes and numbers of files contained in the specified directories:

E.g.: Display (print to stdout) number of files, size, dir/filename

```
>> filer -s .
       5           20 ./Testdir/y
       0            0 ./Testdir/xxx1
       0            0 ./Testdir/xxx4
       8        12308 ./Testdir
       0            0 ./Testdir1
      36      1171609 .
```

Note: No headline is included in the output. Therefore you can pipe the output to any other program.

When using -md 2 the output will be reduced to:

```
>> filer -s -md 2
      10       165476 ./Old
       8        12308 ./Testdir
       0            0 ./Testdir1
      36      1171609 .
```

you may also include all files with -md 0

```
>> filer -s -md 0
       +        12367 ./Old/filetree.c
       +        13123 ./Old/filecopy.c
       +         8255 ./Old/fileusage.c
       +        12224 ./Old/backup.c
       +        12238 ./Old/filer.c.2003-10-11
       +        14434 ./Old/filer.c.2004-04-30
       +        13123 ./Old/filer.c.2004-05-26
       +        21769 ./Old/filer.c.2004-10-01
       +        24906 ./Old/filer.c.2004-10-11
       +        33037 ./Old/filer.c.2004-11-08
      10       165476 ./Old
       +            4 ./Testdir/y/x\07x
       +            4 ./Testdir/y/x\0Ax
       +            4 ./Testdir/y/x\0Dx
       +            4 ./Testdir/y/x x
       +            4 ./Testdir/y/x\\nx
       5           20 ./Testdir/y
       0            0 ./Testdir/xxx1
       .....
```

a + sign marks a plain file, all other entries are directories.

The Suboption -nds can be used if you don't want to add up sizes of directories (good for comparison of two directories trees).

7.2. Command -S (Display CRC summary)

This -S command is used to show all information about files. This includes a file crc summary and ACL information! The output can be used for comparing two directories trees on different volumes or nodes.

E.g.: Display (print to stdout):

```
>>filer -S .
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
. 0 40755 1187616057 1000 1131 0
Old 0 40755 1170242296 1000 1131 0
Old au          0          7 rxw
Old ag          0          5 r-x
Old ao          0          5 r-x
Old du          0          7 rxw
Old dU          0          5 r-x
Old dg          0          5 r-x
Old dm          0          5 r-x
Old do          0          4 r--
Old/filetree.c 12367 100644 1170242743 1000 1131 268fcf33
Old/filecopy.c 13123 100644 1170242306 1000 1131 cae114e
Old/fileusage.c 8255 100644 1170242306 1000 1131 c5269357
Old/backup.c 12224 100644 1170242306 1000 1131 89b89029
...
```

The second parameter contains the size or 0 for a directory or an character a-z to show the ACL entries. If an ordinary file is printed an CRC is printed as last parameters.

The shown entries for files are: file size mode date group owner crc

for ACLs: file acl-type group/user permission permission-text

To compare two directory trees, execute the following commands:

>> filer -S dir1 | sort >dir1.list

>> filer -S dir2 | sort >dir2.list

>> diff dir1.list dir2.list

Sorting is necessary because the listing may be in a different order on different directories. Note that the listings maybe created on different nodes too.

The -S command accepts a tag file option (-t option) to specify the status of the last backup date. If specified all files are show which are NOT modified during the last backup! Otherwise all files are shown.

The -S command accepts an option -nocrc to avoid the calculation of the file content checksum. This is much faster because the header must be read only by filer during the scan and not the complete content of the file.

7.3. Command -c and -C

The -c command creates a single copy of a directory.

```
>> filer -c -vl 2 .  /backup/xyz /backup/xyz.ver/0
```

will copy all files contained in the current directory to the directory /backup/xyz and all files updated in /backup/xyz are moved to /backup/xyz.ver/0

The -c command functions like the BACKFS version 1. It copies all files which are newer in the source than in the destination directory tree. All files which are not longer found in the source directory are deleted from the destination and moved to the version/backup directory. If the version directory is omitted the destination directory with an extension bak is used. This version directory must not exist when the program is started and will be created on demand when the first file must be renamed. The difference between the -c and -C command is that -c is monitoring the file modification time and -C the file status change time. Note that the file modification time can be changed by the user to any time. So -c may fail to find such files. But the -c command is very useful to synchronize directories because you can copy them and back without updating all files. But it may fail to find files which have been changed and the correct modification time was not set.

You can specify a date and time format with the date command e.g.:

```
>> filer -c .  /backup/xyz /backup/xyz.ver/$(date +%d)
```

will create a file with the day of the month as filename. See "`man date`" for a description of the date command.

Note: The -c command is the slowest backup command and it might be disturbed (resulting in a corrupt copy) if files in the destination are newer than in the source.

The output should show something like this:

```
>> filer -f -vl 1 . /home/mizzi/tst/abc1  \
   /home/mizzi/tst/vers/$(date +%s);
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
source path .
destination path /home/mizzi/tst/abc1
backup path /home/mizzi/tst/vers/1195217554
this backup at 1195217554 2007-10-16/13:52:34
last backup at 0 never
total directories scanned 27
total files scanned 97
total files copied 97
total hard links 0
total bytes copied 2560KB/3MB/3MB(allocated)
total time needed 1 sec => 4.72 MB/sec
total number of errors 0
total number of warnings 0
```

7.4. Command -f, -F and -i

The -F, -f and -i command creates a full or incremental copy of a directory

```
>> filer -f -vl 2 .  /backup/xyz /backup/xyz.ver/0
```

will copy all files contained in the current directory to the directory /backup/xyz and all files updated in /backup/xyz are moved to /backup/xyz.ver/0

The difference between the -f and the -c or -C option is that -f writes a status file <destination>.backupdate (can be changed with the -t option) where all information is stored for a further incremental backup. The -f command will also not overwrite any newer file in the destination file system. Therefore you must delete the destination directory to make sure that everything is updated correctly, in case the destination copy was not copied by "filer" before the first full backup. The -i command performs a fast incremental backup of files which has been changed after the last backup by monitoring the file status change time and updates the information in the status file. The difference between the -f and -F command is that -f is monitoring the file modification time and -F the file status change time. Note that the file modification time can be changed by the user to any time. So -f may fail to find such files.

Note that if no status file is found -i is not fully equivalent to -f. The -i command will copy all files and -f or -F only those which has be changed!

E.g.: A simple copy of the current directory:

```
>> filer -f -vl 1 . /home/mizzi/tst/abc1   \
   /home/mizzi/tst/vers/$(date +%s);
BACKFS/filer x.x.x./xx/xxxx - Copyright MIZZI Computer Software GmbH
source path .
destination path /home/mizzi/tst/abc1
backup path /home/mizzi/tst/vers/1195217724
status path /home/mizzi/tst/abc1.backupdate
this backup at 1195217724 2007-10-16/13:55:24
last backup at 0 never
total directories scanned 27
total files scanned 97
total files copied 97
total hard links 0
total bytes copied 2560KB/3MB/3MB(allocated)
total time needed 2 sec => 1.77 MB/sec
total number of errors 0
total number of warnings 0

If you now change a file e.g. filer.c and increase the verbose level to 3
you will get:

>> touch filer.c

>> filer -i -vl 3 . /home/mizzi/tst/abc1  \
   /home/mizzi/tst/vers/$(date +%s);
BACKFS/filer x.x.x/xx/xxx - Copyright MIZZI Computer Software GmbH
source path .
destination path /home/mizzi/tst/abc1
```

```
backup path /home/mizzi/tst/vers/1195217830
status path /home/mizzi/tst/abc1.backupdate
this backup at 1195217830 2007-10-16/13:57:10
last backup at 1195217824 2007-10-16/13:57:04
update file ./filer.c
saving status in /home/mizzi/tst/abc1.backupdate
total directories scanned 27
total files scanned 97
total files copied 1
total hard links 0
total bytes copied 114KB/0MB/0MB(allocated)
total time needed 1 sec => 0.195 MB/sec
total number of errors 0
total number of warnings 0
```

Note that an incremental backup is much faster than a full backup. A full backup can not detect if directories have been changed completely. Thus it is strongly recommended that you use -i instead of -f and you start the copy process with an empty destination directory.

7.5. Command -D, -d, -r and -l, -L

The -D.-d command creates a full or incremental dump to stdout of a directory while the -r restores all files from stdin. E.g.:

```
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
source path ../mcl
status path ../mcl.backupdate
this backup at 1195218004 2007-10-16/14:00:04
last backup at 0 never
total directories scanned 2
total files scanned 41
total files copied 41
total bytes copied 538KB/1MB/1MB(allocated)
total time needed 1 sec => 0.875 MB/sec
total number of errors 0
total number of warnings 0

>> filer -r /tmp/abc  </tmp/abc.dump
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
restore path /tmp/abc
version path /tmp/abc.bak
total directories scanned 2
total files scanned 41
total files restored 41
total files deleted 0
total bytes restored in regular files 538KB/1MB
total time needed 1 sec => 0.875 MB/sec
total number of errors 0
total number of warnings 0
```

Listings can be obtained by the -l command:

```
>> filer -l    </tmp/abc.dump
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
f  100644 ./Old/screen.c.2007-07-28
f  100644 ./Old/screensize.c.2007-07-28
f  100755 ./Old/thread.c.2007-04-03
f  100644 ./Old/chartest.c
f  100644 ./Old/datetest.c
f  100750 ./Old/jobtest.c
f  100644 ./Old/screen.c.2007-08-02
...
d   40755 .
total directories scanned 2
total files scanned 45
total files in dump file  45
total bytes in regular files 604KB/1MB
total time needed 1 sec => 0.982 MB/sec
total number of errors 0
total number of warnings 0
```

Or in long format by the -L command:

```
>> filer -L    </tmp/abc.dump
BACKFS/filer x.x.x/xx/xxxx - Copyright MIZZI Computer Software GmbH
f 0100644 01131 01000 2007-06-28/17:54:02 ./Old/screen.c.2007-07-28
f 0100644 01131 01000 2007-06-28/17:55:40 ./Old/screensize.c.2007-...
f 0100755 01131 01000 2007-03-03/19:54:43 ./Old/thread.c.2007-04-03
f 0100644 01131 01000 2006-10-17/20:50:29 ./Old/chartest.c
f 0100644 01131 01000 2006-10-17/20:59:53 ./Old/datetest.c
f 0100750 01131 01000 2006-11-17/10:14:02 ./Old/jobtest.c
f 0100644 01131 01000 2007-07-02/13:27:40 ./Old/screen.c.2007-08-02
....
f 0100644 01131 01000 2007-00-18/19:53:14 ./guikb.h
d 0040755 01131 01000 2007-07-17/17:02:04 .
total directories scanned 2
total files scanned 45
total files in dump file  45
total bytes in regular files 604KB/1MB
total time needed 2 sec => 0.368 MB/sec
total number of errors 0
total number of warnings 0
```

The commands may be used in a pipe to copy a directory tree. E.g:

```
>> filer -D . | ssh abc filer -r /backup/xxx
```

will copy the current directory to the directory /backup/xxx on node abc. Note that both commands omit the directory names itself in the copy process and uses the directories names as given by the command. The -r command accepts a version/backup directory as an optional parameter to store the files which will be overwritten by the restore operation.

The -r command accepts an option -dv which can be used to delete all ordinary version files. This option may be used if you want to refresh the complete shadow file system but the destination disk can not hold the complete directory tree twice.

8. Backup operation

While scanning the directories, "`filer`" copies all files which are newer than the last backup date. If a directory is found which is newer, "`filer`" updates all attributes of that directory. If the directory was changed completely "`filer`" will move it to the version directory and copies the complete directory tree. If a file is found which was changed since the last backup the file is updated in the destination (existing destination files are moved to the version directory) .

When a file is updated, the following happens. All hardlinks to the files in the same tree will create a hardlink to the same file in the copy. During the copy process sparse data files (files with holes) will be produced. This means that the destination files might be much smaller (in allocation – see `filer -s -a)` than the source files. All attributes are updated (modification time, owner, ...) but the file status on the destination is changed to the backup time. Note that if a file changes during the backup, it is marked that it will be copied during the next backup cycle.

ACL's are updated as long as the -noacl option is not specified . Note that the current NFS versions must not support ACL's and that destination disks must be mounted with ACL support. Otherwise ACL updates will not work!

Note that "`filer`" unlike BACKFS version 1 does not support backup/version directories on file systems other than the destination file system. If you want to do this you must do the following:

```
>> filer -i -vl 3 . /backup/abc /backup/tmp
>> filer -c  /backup/tmp anydestination  ...
>> rm -rf /backup/tmp
```

This is more secure than removing files during a copy process.

9. Excluding files from backups

You may exclude files from update with the -e option.

The -e option has two different exclude formats

-e /filename

and

-e filename

If -e /filename is used, the file or directory relative to the source path is excluded. Note that you must not include the full source path in the exclude filename.

E.g.: create two files abc ....

```
>> touch Testdir/abc
>> touch Testdir1/abc
```

Exclude absolute /Testdir:

```
>>  filer -i -vl 3 -e /Testdir . /home/mizzi-tst/abc1 \
    /home/mizzi-tst/versions/$(date +%s);
...
source path .
destination path /home/mizzi-tst/abc1
backup path /home/mizzi-tst/versions/1101982188
status path /home/mizzi-tst/abc1.backupdate
exclude path /Testdir
this backup at 1101982188 Thu Dec  2 11:09:48 2004
last backup at 1101982114 Thu Dec  2 11:08:34 2004
path excluded Testdir
copy file ./Testdir1/abc
saving status in /home/mizzi-tst/abc1.backupdate
total directories scanned 3
total files scanned 26
total files copied 1
total bytes copied 0KB/0MB/0MB(allocated)
total time needed 1 sec => 0 MB/sec
total number of errors 0
total number of warnings 0
```

Exclude all files named abc:

```
>>  filer -i -vl 3 -e abc . /home/mizzi-tst/abc1 \
    /home/mizzi-tst/versions/$(date +%s);
...
source path .
destination path /home/mizzi-tst/abc1
backup path /home/mizzi-tst/versions/1101982200
status path /home/mizzi-tst/abc1.backupdate
exclude path abc
this backup at 1101982200 Thu Dec  2 11:10:00 2004
last backup at 1101982188 Thu Dec  2 11:09:48 2004
path excluded Testdir/abc
path excluded Testdir1/abc
saving status in /home/mizzi-tst/abc1.backupdate
total directories scanned 7
total files scanned 30
total files copied 0
total bytes copied 0KB/0MB/0MB(allocated)
total time needed 1 sec => 0 MB/sec
total number of errors 0
total number of warnings 0
```

Note: there is currently no way to exclude files with wildcards and/or regular expressions.

10. Errors and Warnings

If an error is encountered during backup and this is also a fatal error, "`filer`" will not stop processing until you specify -aoe (abort on error). Also if you specify this options and a non fatal error (which is reported as ERROR: .... too) occurs the program continues the operation. Some fatal errors if there are not recoverable by the next operations (e.g. no space left on device, I/O error, not existing I/O) will abort the program in any case. This is necessary because the backup file system would be destroyed due to hardware errors.

Warnings like infos are always printed but in most cases you can ignore them. Logging output is written to stderr.

11. Return values

"`filer`" returns an exit status of 0 if no error and no warning has occurred. It returns 1 if some warnings and 2 if an error has occurred. In case of a non-recoverable error an exit status of 3 is returned. Note that "`filer`" logs output with a unique phrase at the beginning at each line and no lines a broken over two or more lines. This is true for filenames containing a newline character too. Thus you can analyze the error log with an external program to retrieve information about errors and warnings.

E.g.: You may select the number of errors with:

```
>> filer ....... 2>filer.log
>> awk  '/^total number of errors/ {print $5}' <filer.log
```

12. Verbose levels

Verbose levels are defined as follows

-vl 0	report nothing but errors
-vl 1	report summary, warnings
-vl 2	report summary, warnings, infos
-vl 3	everything which causes changes
-vl >3	debugging code (up to 99)

Do not use debugging with higher values than 4. You may get more logging messages than you can read in your life!

13. System specific files

Currently all files are copied by “`filer`”. There is no exclusion of quota.user or quota.group. If these files are partially copied during the backup they may be unusable when mounting the copied file system. Thus you must take care about the correct quota files before mounting the copied file system.

If the source directory should be backuped do something like this:

```
>> cd source
>> repquota source >quota.saved.user
>> repquota -g source >quota.saved.group
```

Proceed with the backup and exclude the quota files.

You can use the saved quota to restore the quotas on the destination file system.

Note: If the destination file system is mounted with quotas enabled (not recommended), the quota files must be excluded during the backup.

14. Example

The following shell script will give you an example, how to do an hourly backup. It creates version/backup directories with the name /d32/mizzi/vme.versions/<weekday>-<time> and deletes files on the version/backup directory which are older than seven days. Quota files are saved and a mail is send if an error occurs. A fatal non-recoverable error will stop the script and exits with 1.

```
#!/bin/sh

s=/vme
d=/d32/mizzi/vme.shadow
v=/d32/mizzi/vme.versions
m=mizzi@mpi-k.de

while true
do

w=$(date +%w);
wd=$(echo "(8+$w)%7"|bc)
t=$(date +%T);
wt=$w-$t

# echo new versions $v/$v/$wt
echo deleting versions $v/$wd-*

rm -rf $v/$wd-*
repquota -u  $s >$s/quota.user.saved
repquota -g  $s >$s/quota.group.saved

echo -n doing backup $s $d $v/$wt ... ' '

./filer -i -vl 1 -vm 0 -md 3 $s $d $v/$wt 2>$v/$wt.log
rc=$?

if test $rc -gt 1
then
  echo errors during backup
  mail -s 'filer backup error ' $m <$v/$wt.log
fi

if test $rc = 3
then
  echo abort due to fatal errors
  exit 1
fi

echo done
if test x$1 = "xonce" ; then exit 0; fi
sleep 3600

done
```