

Backup scripts next generation - User Guide for version 1.19.x



Table of Contents

1 Introduction.....	1
1.1 Overview.....	1
1.1.1 Configuration files.....	1
1.1.2 Email.....	1
1.1.3 Email filters.....	1
1.1.4 Logcheck.....	1
1.1.5 Logs.....	1
1.1.6 man pages.....	2
1.1.7 On screen notifications.....	2
1.1.8 Scheduled jobs.....	2
1.1.9 Scripts.....	2
1.2 Related documentation.....	2
1.3 git.....	2
2 Installation.....	3
2.1 Supported environments.....	3
2.2 Pre-requisites.....	3
2.2.1 Required.....	3
2.2.2 Required when bung is configured to mount file system(s).....	3
2.2.3 Optional backup utilities.....	3
2.2.4 Mail utilities.....	3
2.2.5 mysql bu.sh (mysqldump) requirements.....	3
2.2.6 Hotplug backup storage devices' requirements.....	3
2.3 Installation itself.....	3
3 Configuration.....	4
3.1 Overview.....	4
3.1.1 Jobs started by a scheduler.....	4
3.1.2 Jobs started by udev.....	4
3.2 Snapshots and mountpoints.....	4
3.2.1 mlocate's updatedb configuration.....	5
3.3 Creating configuration files.....	5
3.3.1 Do not panic.....	5
3.3.2 Choosing the name.....	5
3.3.3 Examples and man pages.....	5
3.4 Testing configurations.....	5
3.5 super bu.sh configuration.....	5
3.6 hotplug bu.sh configuration.....	6
3.7 check_hotplug_usage.sh configuration.....	6
3.8 MySQL configuration.....	6
3.8.1 MySQL user backup user.....	6
3.8.2 mysql bu.sh configuration.....	7
3.9 PostgreSQL configuration.....	7
3.9.1 pg bu.sh configuration.....	7
3.10 rsync bu.sh configuration.....	8
3.10.1 Exclude files.....	8
3.10.1.1 /etc/mtab.....	8
3.10.1.2 New excludes.....	8
3.10.2 rsync bu.sh configuration for remote source or destination.....	8
3.10.2.1 Populating the local known hosts.....	8
3.10.2.2 Restricting the commands run on the remote host.....	9
3.10.2.3 Compression.....	9
3.10.2.4 Example configuration files.....	9
3.11 sysinfo bu.sh configuration.....	9
3.12 Creating scheduled jobs.....	9
3.12.1 Choosing the scheduling mechanism.....	10
3.12.2 Using /etc/cron.{daily,weekly,monthly}/* jobs.....	10
3.13 Creating udev-initiated jobs.....	10
3.13.1 Creating the udev rule.....	11
3.13.2 Testing.....	13
3.14 Old log removal.....	13
4 Routine operations.....	13
4.1 Monitoring.....	13

4.1.1 Report emails.....	13
4.1.2 Logs.....	14
4.2 Audit.....	14
5 Problem investigation.....	14
5.1 Report email not received.....	14
5.2 bung's logs.....	14
5.2.1 rsync.....	14
5.2.1.1 protocol version mismatch -- is your shell clean?.....	15
5.2.2 Snapshots.....	15
5.2.2.1 Buffer I/O error on device dm-*, logical block *.....	15
5.3 /var/log/syslog.....	15
5.4 bung debugging.....	15
5.4.1 The -d option.....	15
5.4.2 No output?.....	15
5.5 ssh (remote destination or source).....	15
6 Restores.....	16
6.1 mysql bu.sh.....	16
6.2 postgres bu.sh.....	16
6.3 rsync bu.sh.....	16
7 Worked examples.....	16
7.1 rsync bu.sh - backup to local file system, whole tree.....	16
7.1.1 Space calculation.....	16
7.1.2 Create the backup file system.....	17
7.1.3 Create destination directory.....	17
7.1.4 Create snapshot mountpoint.....	17
7.1.5 bung configuration.....	17
7.1.6 Testing.....	18
7.1.7 Create scheduled job.....	18
7.2 rsync bu.sh - backup to local file system, separate /var and /home.....	18
7.2.1 Space calculation.....	18
7.2.2 Create destination directory.....	18
7.2.3 Create snapshot mountpoints.....	19
7.2.4 bung configuration.....	19
7.2.5 Testing.....	20
7.2.6 Create scheduled job.....	20
7.3 rsync bu.sh - backup to hotplug disk.....	20
7.4 rsync bu.sh - backup to remote system (/etc, /home, /root and /var).....	22
7.4.1 Create destination directory.....	22
7.4.2 Create the constant access path.....	22
7.4.3 Create snapshot mountpoints.....	22
7.4.4 Set up ssh.....	22
7.4.5 Configure bung.....	23
7.4.6 Testing.....	24
7.4.7 Create scheduled job.....	25
7.5 rsync bu.sh - backup to remote system (whole tree, no LVM).....	25
7.5.1 Space calculation.....	25
7.5.2 Create destination directory.....	25
7.5.3 Create the constant access path.....	25
7.5.4 Create snapshot mountpoints.....	25
7.5.5 Set up ssh.....	25
7.5.6 Configure bung.....	26
7.5.7 Testing.....	27
7.5.8 Create scheduled job.....	27
7.6 rsync bu.sh - backup from remote system.....	27
7.6.1 Calculate the backup data volume.....	28
7.6.2 Finding space on the destination system.....	28
7.6.3 Create destination directory.....	28
7.6.4 Set up ssh.....	28
7.6.5 Configure bung.....	28
7.6.6 Testing.....	30
7.6.7 Create scheduled job.....	30
7.7 DG-GS1526E bu.sh.....	30

1 Introduction

This is the User Guide for backup scripts next generation (bung) version 1.19.x.

1.1 Overview

backup scripts next generation (bung) is a suite of bash scripts that runs standard backup utilities such as `mysqldump`, `pg_dump`, `rsync`, `slapcat` and `tar`.

`bung` adds configurable support for hotplug storage devices (such as USB disks), LVM snapshots, file system mounting, logging, on-screen notifications and report emails.

`bung` backup jobs are normally started by scheduler (anacron or cron) or by `udev` (by plugging in backup storage devices such as USB HDDs). They may also be started manually.

`bung` comprises ...

1.1.1 Configuration files

Default location `/etc/opt/bung`.

1.1.2 Email

By default a report is emailed to root.

Optionally and additionally, for hotplug storage, the scripts can be configured to send mail when hotplug storage is mounted and unmounted. This can be used to alert the people who unplug the device.

1.1.3 Email filters

Not part of `bung` but a useful adjunct.

For example, for gmail:

1. Create folders "Backup successful", "Backup warning" and "Backup error"
2. Create corresponding filters:
 - i. Matches: `subject:(SUCCESS (bung))`
Do this: Skip Inbox, Apply label "Backup successful", Never send it to Spam
 - ii. Matches: `subject:(WARN (bung))`
Do this: Skip Inbox, Apply label "Backup warning", Never send it to Spam
 - iii. Matches: `subject:(ERROR (bung))`
Do this: Skip Inbox, Apply label "Backup problems", Never send it to Spam

Note: Gmail does not support regular expressions for matching; there's an enhancement request:

<http://code.google.com/p/google-summer-of-code/issues/detail?id=212>

1.1.4 Logcheck

A logcheck filter file filters out routine (not warning and error) messages that `bung` scripts write to `/var/log/syslog`. It is `/etc/logcheck/ignore.d.server/bung`.

1.1.5 Logs

Logs are designed to speed problem diagnosis.

By default, logs are created in `/var/log/bung` with names reflecting the script that created them, the configuration file used and the time the job started:

`/var/log/bung/<script name>.<conf file name>.<timestamp>.log`

By default old logs are removed after 28 days when the same script and configuration file combination is run again.

When backup is to a hotplug device, a further log is created:

/var/log/bung/hotplug/<organisation name>

This contains the day of the backup and the serial number of the device. It is used by the scripts to detect when it is too long since a backup was written to the device or the device was changed.

A weekly cron job removes any /var/log/bung/*.log files more than 122 days (~4 months) old.

1.1.6 man pages

bung includes man pages for its scripts and their configuration.

1.1.7 On screen notifications

Optionally for hotplug storage, the scripts can be configured to display on-screen notifications when a hotplug device is being used and when it is safe to unplug.

If the screen is in graphical mode and not locked, these notifications are pop-up windows, otherwise these notifications are plain text.

1.1.8 Scheduled jobs

Typically in /etc/cron.daily, root's crontab or /etc/cron.d

1.1.9 Scripts

In /opt/bung:

1. **check_hotplug_usage.sh** Checks in case it is too long since a backup was written to the device or the device was changed.
2. **delete_old_bu.sh** In development. Alpha. Deletese select old backups.
3. **DG-GS1526E_bu.sh** Backs up a DG-GS1526E switch.
4. **hotplug_bu.sh** Runs subsidiary jobs if a specific hotplug storage device is plugged in.
5. **hotplug_bu_launcher.sh** Run by udev, it runs hotplug_bu.sh as an "at" job.
6. **mysql_bu.sh** Backs up MySQL database.
7. **opendap_bu.sh** Backs up OpenLDAP
8. **postgres_bu.sh** Backs up Postgres database.
9. **remote_agent.sh** Runs bung commands from remote computers securely.
10. **rsync_bu.sh** Runs an rsync backup.
11. **rsync_restore.sh** Restores from backups made by rsync_bu.sh. Intended for interactive use.
12. **super_bu.sh** A supervisor script. Runs subsidiary scripts which may include hotplug_bu.sh.
13. **sysinfo_bu.sh** Generates reports on hardware, BIOS, storage, networking, groups and users and on packages.
14. **validate_ssh_cmd.sh** Validates bung commands run by remote computers via ssh.

1.2 Related documentation

- "Backup scripts next generation - development log.odt"
- "Backup scripts next generation - issues and TODOs.ods"
- "Backup scripts next generation - Programmer's Guide.odt"
- "Backup scripts next generation - release log.odt"
- "Backup scripts next generation - test suite.odt"

1.3 git

<https://redmine.auroville.org.in/projects/bung/repository>

2 Installation

2.1 Supported environments

Debian 8 Jessie and 9 Stretch. Has been run without modification successfully on Debian 10 Buster. Probably works in other environments, especially Debiana derivatives such as Ubuntu.

2.2 Pre-requisites

2.2.1 Required

bash and logger

2.2.2 Required when bung is configured to mount file system(s)

lsuf. Used only when logging a umount failure.

2.2.3 Optional backup utilities

Depending on which bung scripts are run, one or more backup utilities are required:

- mysql-server
- postgresql
- rsync

2.2.4 Mail utilities

bung uses the mailx command to send report mails; several packages provide that.

When report mails are to be sent from the local computer, an MTA is required; popular choices are exim, postfix and smtp.

2.2.5 mysql_bu.sh (mysqldump) requirements

If bung's mysql_bu.sh is to be used, a compression utility is normally used. The default utility is gzip which is a core package.

Optionally any other compression program that works in a pipeline can be used, for example commands from these packages: bzip2, pbzip2, lbzip2, p7zip, p7zip-full

2.2.6 Hotplug backup storage devices' requirements

When hotplug backup storage devices are to be used, the lsscsi and sdparm utilities are required.

When hotplug backup storage devices are to be used and messages about their status are to be displayed on an X server such as a graphical desktop, yad or zenity is required.

Yad is recommended in preference to zenity for its "always on top" and buttonless functionality. These avoid messages being accidentally closed or hidden behind other windows. Its messages look better too.

2.3 Installation itself

If upgrading, remove the old symlinks and previous versions:

```
rm /opt/bung /usr/share/doc/bung
rm -fr /opt/bung-* /usr/share/doc/bung-*
```

The commands below assume that version 1.19.x is being installed, the tarball is in the current directory and logcheck is in use. The commands in blue may generate error messages if logcheck is not installed; in that case they can safely be ignored.

version=1.19.x

```
rm -f /opt/bung /usr/share/doc/bung
mkdir -p /{etc/opt,var/log}/bung /{opt,tmp,usr/share/doc}/bung-$version \
&& tar -xvf bung-$version.tgz --directory /tmp/bung-$version --group=root \
```

```

--no-same-owner --owner=root \
&& cp --archive --recursive /tmp/bung-$version/doc/* /usr/share/doc/bung-$version \
&& chown root:root --recursive /usr/share/doc/bung-$version \
&& cp --archive /tmp/bung-$version/etc/cron.weekly /etc/cron.weekly/bung \
&& chown root:root /etc/cron.weekly/bung \
&& chmod 744 /etc/cron.weekly/bung \
&& for f in /tmp/bung-$version/man/man5/*.5
do
    gzip --to-stdout $f > /usr/share/man/man5/${f##*/}.gz
done \
&& for f in /tmp/bung-$version/man/man8/*.8
do
    gzip --to-stdout $f > /usr/share/man/man8/${f##*/}.gz
done \
&& cp --archive --recursive /tmp/bung-$version/opt/* /opt/bung-$version \
&& chown -R root:root /opt/bung-$version \
&& find /opt/bung-$version -maxdepth 1 -type f -execdir chmod 755 {} \+ \
&& find /opt/bung-$version/lib -type f -execdir chmod 644 {} \+ \
&& ln -s /opt/bung{-$version,} \
&& cp --archive /tmp/bung-$version/etc/logcheck_filters \
    /etc/logcheck/ignore.d.server/bung \
&& chown root:logcheck /etc/logcheck/ignore.d.server/bung
(cd /tmp && mandb >/dev/null 2>&1) &

```

3 Configuration

3.1 Overview

Backup scripts are started by a scheduler (anacron or cron) or, when a hotplug storage device is plugged in, by the udev daemon. They can also be run manually from the command prompt, most usefully during testing new configurations.

Every backup script needs a configuration file, named on the command line with the `-c` option.

3.1.1 Jobs started by a scheduler

The simplest way is to schedule one of the scripts that runs a backup utility:

- DG-GS1526E_bu.sh
- ldap_bu.sh
- mysql_bu.sh
- postgres_bu.sh
- rsync_bu.sh
- sysinfo_bu.sh

A more sophisticated way is to schedule the supervisor script, `super_bu.sh`, to run a sequence of those, optionally followed by `hotplug_bu.sh`. Running `hotplug_bu.sh` in this way is useful when a hotplug storage device is already plugged in; it can be updated with backup data created by the earlier scripts.

3.1.2 Jobs started by udev

The configuration comprises:

1. A udev rule to run `hotplug_bu_launcher.sh`
2. A configuration file for `hotplug_bu.sh`. `hotplug_bu.sh` is a supervisor script, like `super_bu.sh`. Its configuration file names a sequence of scripts to run.
3. A configuration file for each script that `hotplug_bu.sh` is configured to run.

3.2 Snapshots and mountpoints

File systems are best backed up using snapshots so their files do not change during the backup. Bung supports LVM snapshots. Conventionally snapshots are mounted on `/mnt/snap-*` directories.

Backup file systems are best not normally mounted to minimise the risk of accidental deletion. Conventionally the backup file system is mounted on /mnt/backup.

3.2.1 mlocate's updatedb configuration

If the computer has mlocate installed, by default it will index files in the /mnt tree. This is useless for snapshots and backup file systems because the files are copies. This can be prevented by adding /mnt to PRUNEPATHS in /etc/updatedb.conf.

3.3 Creating configuration files

3.3.1 Do not panic

At first sight the number of configuration options may confuse. Do not panic. Bung's defaults are designed to cover normal usage so typical configuration files have few keywords, some of them with no options.

3.3.2 Choosing the name

Configuration file names appear in the subject of report emails. It helps if they are both short and meaningful.

3.3.3 Examples and man pages

Fully commented examples are available in /usr/share/doc/bung-1.19.x/examples.

Bung man pages can be listed by running `man -k bung`.

3.4 Testing configurations

Configuration files can be tested by running the script at a command line. When doing so:

1. For logging on screen, use `-l /dev/tty`
2. To test the configuration file and exit, use `-c`
3. When testing an `rsync_bu.sh` configuration, for an `rsync` dry-run, use `-r`
4. In case the script starts working as intended and may run for a long, it is convenient to start it using `screen`, `nohup` ... & or similar.
5. It is safe to interrupt with `Ctrl+c`
6. In case you want more information (mostly for developers), use the `-d` option.

For example (would normally be followed by viewing/tailing the log file):

```
nohup /opt/bung/rsync_bu.sh -c all &
```

3.5 super_bu.sh configuration

If you want to sequence a series of bung backups or to run bung backups with non standard priorities (nice and ionice), use `super_bu.sh` to run them. This is especially useful for running scheduled jobs and for minimising the effect of backups on performance.

Man pages:

- `bung.super_bu` (5)
- `bung.common` (5)

Example `super_bu.sh` configuration file with explanatory comments: `/usr/share/doc/bung-1.19.x/examples/fully_commented_confs/sysinfo_bu.sample.conf`

A typical example `super_bu.sh` configuration:

```
Organisation name = "Al \"Scarface\" Capone"
Subsidiary script = mysql_bu.sh mysql
Subsidiary script = rsync_bu.sh server.whole_tree nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh to_remote nice=19 ionice="-c2 -n0"
Subsidiary script = hotplug_bu.sh hotplug.super nice=19 ionice="-c2 -n0"
```

3.6 hotplug_bu.sh configuration

If you want notification that hotplug device is in use – and later not in use – then use hotplug_bu.sh to run the scripts that read or write to it as subsidiary scripts.

Notifications can be configured as on-screen and/or by email to a configurable address.

Man pages:

- bung.hotplug_bu (5)
- bung.common (5)

Example hotplug_bu.sh configuration file with explanatory comments: /usr/share/doc/bung-1.19.x/examples/fully_commented_confs/hotplug_bu.sample.conf.

In case the hotplug device is left plugged in, it is convenient to set up both super_bu.sh and udev to run hotplug_bu.sh with the same configuration file. In that way, whatever backup is run by super_bu.sh can be synchronised to the hotplug device.

3.7 check_hotplug_usage.sh configuration

This is used to warn when there has not been a backup to a hotplug device for more than a configurable period. In case more than one hotplug device is used, it also warns when the hotplug device has not changed for more than a configurable period.

The warning is sent by mail to a configurable email address.

Example hotplug_bu.sh configuration file with explanatory comments: /usr/share/doc/bung-1.19.x/examples/fully_commented_confs/check_hotplug_usage.sample.conf.

The recommended way to run check_hotplug_usage.sh is via "Subsidiary script" in a hotplug_bu.sh and/or super_bu.sh configuration.

3.8 MySQL configuration

3.8.1 MySQL user backup_user

mysql_bu.sh and mysql_binlog_bu.sh need a MySQL user to enumerate the databases and to run mysqldump on each database that is backed up.

This shell command will create a suitable MySQL user. Replace <password> with the password you want to assign to backup_user (twice). In case a password has been set for the MySQL user root (good practice) and a /root/.my.cnf file has not been created with the password, the --password option is also required.

```
mysql --user=root --skip-column-names --execute="
```

```
CREATE USER 'backup_user'@'localhost'
```

```
IDENTIFIED BY '<password>';
```

```
GRANT
```

```
EVENT,
```

```
LOCK TABLES,
```

```
PROCESS,
```

```
RELOAD,
```

```
SELECT,
```

```
SHOW DATABASES,
```

```
SHOW VIEW,
```

```
TRIGGER
```

```
ON *.*
```

```
TO 'backup_user'@'localhost'
```

```
IDENTIFIED BY '<password>'
```

```
WITH
```

```
MAX_QUERIES_PER_HOUR 0
```

```
MAX_CONNECTIONS_PER_HOUR 0
```

```
MAX_UPDATES_PER_HOUR 0
```

```
MAX_USER_CONNECTIONS 0
```

```
;
```

```

    FLUSH PRIVILEGES;
    SHOW GRANTS FOR 'backup_user'@'localhost';
"

```

To remove backup_user:

```

mysql --user=root --execute="
    REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'backup_user'@'localhost';
    DROP USER 'backup_user'@'localhost';
"

```

3.8.2 mysql_bu.sh configuration

Man pages:

- bung.mysql_bu (5)
- bung.common (5)

Example mysql_bu.sh configuration files with explanatory comments are in /usr/share/doc/bung-1.19.x/examples.

Production configuration files with the explanatory comments removed could look like these:

- mysql_bu.sample.conf, typically modified and saved as /etc/opt/bung/mysql, for example:


```

# Configuration file for mysql_bu.sh

# Format references: man pages bung.common (5) and bung.mysql_bu (5)

Organisation name = MyOrganisation
MySQL =

```
- mysql.sample.cnf, typically modified and saved as /etc/opt/bung/mysql.cnf with 600 permissions, for example:


```

# MySQL defaults-file for use with mysql_bu.sh
#
# Format reference: http://dev.mysql.com/doc/refman/5.5/en/option-files.html

[client]
user=backup_user
password=super-secret

```
- mysql-multi.sample.cnf, typically modified and saved as /opt/etc/bung/mysqld<n>.cnf with 600 permissions, for example:


```

# mysql-multi.sample.cnf
#
# Sample MySQL defaults-file for use with mysql_bu.sh when backing up mysqld
# using non standard socket and port (as used by mysqld-multi)
#
# Format reference: http://dev.mysql.com/doc/refman/5.5/en/option-files.html

[client]
user=backup_user
password=super-secret
port=3308
socket=/var/run/mysqld/mysqld2.sock

```

In case mysqld-multi is being used to provide multiple replication slaves, the user and password values are taken from the master and the port and socket values from the slave.

3.9 PostgreSQL configuration

3.9.1 pg_bu.sh configuration

Man pages:

- bung.pg_bu (5)
- bung.common (5)

Example pg_bu.sh configuration files with explanatory comments are in /usr/share/doc/bung-1.19.x/examples.

Production configuration files with the explanatory comments removed could look like these:

- pg_bu.sample.conf, typically modified and saved as /etc/opt/bung/pg, for example:

```
# Configuration file for pg_bu.sh

# Format references: man pages bung.common (5) and bung.pg_bu (5)

Organisation name = MyOrganisation
PostgreSQL =
```

3.10 rsync_bu.sh configuration

Man pages:

- bung.rsync_bu (5)
- bung.common (5)

Example rsync_bu.sh configuration file with explanatory comments: /usr/share/doc/bung-1.19.x/examples/rsync_bu.sample.conf.

3.10.1 Exclude files

Example exclude files are in /usr/share/doc/bung-1.19.x/examples/rsync.excludes

3.10.1.1 /etc/mtab

When /etc/mtab is a file it should be excluded; when it is a symlink it should not be excluded.

3.10.1.2 New excludes

When new excludes are added to an exclude file after it has been used by rsync_bu.sh, the files and directories that are newly excluded are not deleted from the destination directory. The exclude excludes them from being deleted from the destination.

Newly excluded files must be manually deleted from the destination or rsync will generate "cannot delete non-empty directory" messages.

3.10.2 rsync_bu.sh configuration for remote source or destination

When the source or destination directory is of the form name/* then "name" will be resolved according to ssh procedure which commonly means it is an entry in /root/.ssh/config and supplies several other configuration parameters including the private key to be used. The corresponding public key must be in the remote host's /root/.ssh/authorized_keys or authorized_keys2 file. Keys can be created using the ssh-keygen command.

3.10.2.1 Populating the local known_hosts

Caution: if the local /root/.ssh/known_hosts does not have a line for the remote host, ssh will prompt interactively. In this case rsync_bu.sh will hang.

The known_hosts file can be populated at the same time as testing the ssh host definition and the ssh keys by:

```
# ssh -i <path to private key> <ssh hostname> echo -n OK
The authenticity of host '*' can't be established.
ECDSA key fingerprint is *.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '*' (ECDSA) to the list of known hosts.
directory
```

3.10.2.2 Restricting the commands run on the remote host

On the remote host:

1. Install /opt/bung/validate_ssh_cmd.sh. It can be done by installing bung.
2. In the /root/.ssh/authorized_keys or authorized_keys2 file, prefix the entry for the key bung is configured to use with

```
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

3.10.2.3 Compression

Compression reduces network traffic at the cost of CPU cycles. It may or may not speed rsync, depending on circumstances:

1. Unless the network is slow or congested, weak CPUs such as Atoms and ARMs cannot compress fast enough to speed rsync. If there is a weak CPU locally or remotely, use the rsync_bu.sh keyword "nocompress".
2. Powerful CPUs compress fast enough to speed rsync even on high capacity networks.
3. When the CPU(s) are heavily loaded, compression will slow rsync. If the local or remote CPU(s) are normally heavily loaded, use the rsync_bu.sh keyword "nocompress".

Always ensure ssh compression is not used; in the local /root/.ssh/config file, in the Host stanza used for the backup, include "Compression no".

Rationale: rsync man page, section on --compression.

3.10.2.4 Example configuration files

Local to remote:

```
Organisation name = "Al \"Scarface\" Capone"
rsync = /var/backup/ downtown:/var/backup/mob/al/rsync
```

Remote to local with compression disabled:

```
Organisation name = "Al \"Scarface\" Capone"
rsync = uptown:/var/backup/ /var/backup/uptown/rsync nocompress
```

3.11 sysinfo_bu.sh configuration

Man pages:

- bung.sysinfo_bu (5)
- bung.common (5)

Example rsync_bu.sh configuration file with explanatory comments: /usr/share/doc/bung-1.19.x/examples/fully_commented_confs/sysinfo_bu.sample.conf

When using the defaults:

1. Create directory /var/backup/sysinfo
2. Create /etc/opt/bung/sysinfo containing, for example:

```
Organisation name = HOUSING
sysinfo =
```

3.12 Creating scheduled jobs

It can be advantageous to test the backup before scheduling, as described in 3.4 Testing configurations above.

3.12.1 Choosing the scheduling mechanism

Bung scheduled jobs can be scheduled via anacron or cron. Anacron is a good choice when the computer does not run at fixed times.

bung scheduled jobs can be configured using:

- root's crontab
- /etc/cron.d/* files
- /etc/cron.daily/* files
- /etc/cron.hourly/* files
- /etc/cron.monthly/* files
- /etc/cron.weekly/* files
- /etc/crontab

A disadvantage of /etc/cron.{daily,weekly,monthly}/* jobs is a heavy load shortly after booting (the impact can be reduced by using bung's super_bu.sh and configuring nice and ionice).

For computers that do not run at fixed times and are not up for long there is no choice. Backups must be started at boot.

For other computers, there are two further scheduling choices: root's crontab and /etc/cron.d/* jobs. /etc/cron.d/* jobs have the advantage of modularity.

3.12.2 Using /etc/cron.{daily,weekly,monthly}/* jobs

When scheduling bung using files in /etc/cron.{daily,weekly,monthly}/:

- The files must have execute permission for root.
- In case bung jobs take long to run, the script should exit to avoid delaying subsequent jobs.
- In case the scheduler is anacron, the bung script should be run independently of the anacron process

Example:

```
#!/bin/bash
```

```
buf=$(
    echo /opt/bung/super_bu.sh -c all \
        | at now +10 minutes 2>&1 \
        | grep -v '^warning: commands will be executed using /bin/sh' \
        | grep -Ev '^job [[:digit:]]+ at '
)
[[ $buf != '' ]] && echo "$buf"
exit 0
```

3.13 Creating udev-initiated jobs

1. Decide the name of a /dev/ symlink for the device, for example /dev/hotplug
2. Create a mount point for the device, for example /mnt/hotplug
3. Create a rsync_bu.sh configuration file.
4. Create a configuration file for hotplug_bu.sh. It needs to include:
 - a) A "Hotplug device" line for the /dev/ symlink that was decided in step 1.
 - b) A "Subsidiary script" line to run rsync_bu.sh with the configuration file created in step 2.
5. Testing can optionally be done now:
 - a) Plug in the hotplug device
 - b) Identify it using blkid
 - c) Manually create the /dev symlink chosen in step 1.
 - d) Run

```
/opt/bung/hotplug_bu.sh -c <config file>
```

Where <config file> is the one created in step 3. There's more information about running scripts at a command prompt in as described in 3.4 Testing configurations above.

- e) When testing is completed, remove the symlink created in step c).
6. Create a udev rule (details in 3.13.1 Creating the udev rule below) that:
 - a) Creates the /dev/ symlink decided in step 1.
 - b) Runs /opt/bung/hotplug_bu_launcher.sh with these arguments:
hotplug_bu.sh
-c <config file> (where <config file> is the configuration file created in step 3 above)
-u
<further arguments for hotplug_bu.sh> (normally none)
7. Load the new udev rule. This is automatically done on reboot; it can also be done by
udevadm control --reload

3.13.1 Creating the udev rule

Following partitioning and file system creation on the hotplug storage device, plug it in and identify which /dev/sd* it has been assigned. For example:

```
root@CW8:~# blkid
[snip]
/dev/sdcl: LABEL="bung" UUID="757ffdff-3ac6-4191-8a0a-5c7c6b07966b" TYPE="ext4"
[snip]
```

The next step is to find attributes that uniquely identify the device. For example:

```
root@CW8:~# udevadm info -a -p /sys/block/sdc/sdcl
[snip]
looking at device '/block/sdc/sdcl':
    KERNEL=="sdcl"
    SUBSYSTEM=="block"
    DRIVER==" "
    ATTR{partition}=="1"
    ATTR{start}=="245"
    ATTR{size}=="3999499"
    ATTR{ro}=="0"
    ATTR{alignment_offset}=="0"
    ATTR{discard_alignment}=="0"
    ATTR{stat}=="      229      1124      1822      173      0      0      0      0
0      173      173
    ATTR{inflight}=="      0      0"

[snip some parent device stanzas, looking for a stanza with a serial number]
looking at parent device '/devices/pci0000:00/0000:00:1a.0/usb1/1-1/1-1.6':
    KERNELS=="1-1.6"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==" "
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{bMaxPower}=="200mA"
    ATTRS{urbnum}=="1454"
    ATTRS{idVendor}=="0781"
    ATTRS{idProduct}=="5151"
    ATTRS{bcdDevice}=="0010"
    ATTRS{bDeviceClass}=="00"
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{bNumConfigurations}=="1"
```

```

ATTRS{bMaxPacketSize0}=="64"
ATTRS{speed}=="480"
ATTRS{busnum}=="1"
ATTRS{devnum}=="3"
ATTRS{devpath}=="1.6"
ATTRS{version}==" 2.00"
ATTRS{maxchild}=="0"
ATTRS{quirks}=="0x0"
ATTRS{avoid_reset_quirk}=="0"
ATTRS{authorized}=="1"
ATTRS{manufacturer}=="SanDisk Corporation"
ATTRS{product}=="Cruzer Micro"
ATTRS{serial}=="20042203931424B115DC
[snip]

```

For the device itself, the attributes are labelled ATTR{...}. Its parent devices' attributes are labelled ATTRS{...}. In a udev rule attributes can be taken from the device itself and no more than one of its parent devices.

In the example, the attributes in green uniquely identify the device. The size is required because the ATTRS{serial} alone also matches /dev/sdc.

The next step is to create a file of udev rules for the device, one rule that is run when the device is plugged in ("add") and one when it is unmounted ("change"). The "add" rule will run script /opt/bung/hotplug_bu_launcher.sh.

In the examples below the file is 99-bung.rules. You can use any name ending in .rules; the 99- prefix is strongly recommended.

If /etc/udev/rules.d/99-bung.rules does not exist, create it. It should have owner and group root. It needs to be readable by root. rw-r--r-- permissions are convenient.

Add the new rule to 99-bung.rules preceded by an empty line and a comment line. For example:

```

# udev rule(s) for bung

# The USB HDD for backup
KERNEL=="sd*", ACTION=="add", ATTR{size}=="3999499 ", ATTRS{serial}=="20042203931424B115DC",
SYMLINK+="my_hotplug", RUN+="/opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh -c hotplug
-u"
KERNEL=="sd*", ACTION=="change", ATTR{size}=="3999499 ", ATTRS{serial}=="20042203931424B115DC",
SYMLINK+="my_hotplug"

```

Breaking the example into components:

Component	Notes
ATTR{size}=="3999499 "	The size attribute found earlier.
ATTRS{serial}=="20042203931424B115DC "	The serial attribute found earlier.
ACTION=="add"	Required. Signifies the plug in event as opposed to the unplug event.
ACTION==" change"	Required. Preserves the symlink when the file system is unmounted.
SYMLINK+="my_hotplug"	Required for bung. Creates a symlink under /dev/ with constant name although the device itself could become /dev/sdc1, sdf1 etc. depending on which other devices have been connected.
RUN+="/opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh -c hotplug -u"	Required for bung. Only the name of the configuration file, hotplug_bu.sample.conf, needs to be customised.

3.13.2 Testing

1. Load the new rules file:
`udevadm control --reload`
2. Plug in the hotplug device. A `/var/log/bung/hotplug_bu.sh` log file should be created and can be reviewed for warnings and errors.
3. If no `/var/log/bung/hotplug_bu.sh` log file was created, was a `/var/log/bung/hotplug_bu_launcher.sh` log file created which can be reviewed for warnings and errors?
4. If no `/var/log/bung/hotplug_bu_launcher.sh` log file was created, was the `/dev` symlink created? If it has been, the "match" part of the udev rule has worked but the RUN+ part has not. Continue investigation by changing the RUN+ part to `RUN+="/usr/bin/touch /tmp/udev.rule.ran"`, run `udevadm control --reload`, re-plug the hotplug device and check for the presence of `/tmp/udev.rule.ran`.

If `/tmp/udev.rule.ran` was created, try modifying the script as described in 5.4.2 No output? below.

5. If the `/dev` symlink was not created, check the "match" part of the udev rule – SUBSYSTEM, ATTR and ATTRS. Common mistakes are confusing ATTR/ATTRS and using a single = where == is required. Does the

The udev configuration for a particular device can be tested with a command like:

```
udevadm test /sys/block/sdc/sdc1
```

Output is voluminous but should include lines like:

```
parse_file: reading '/etc/udev/rules.d/99-bung.rules' as rules file
udev_rules_apply_to_event: RUN '/opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh
-c hotplug -u' /etc/udev/rules.d/99-bung.rules :1
link_update: creating link '/dev/my-hotplug' to '/dev/sdc1'
run: '/opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh -c hotplug -u'
```

3.14 Old log removal

Normally each bung script and configuration combination removes its own old logs. This allows log retention times to be configured for each combination.

Old logs are not removed when configuration file names are changed or when a job is no longer run. They are removed by `/etc/cron.weekly/bung` which removes files in `/var/log/bung` older than 122 days (that is ~3 months).

4 Routine operations

4.1 Monitoring

4.1.1 Report emails

bung sends a report email, by default to root. Typically the local mail configuration would forward it to somewhere more convenient.

bung's warning and error report emails (with WARN or ERROR in the subject) provide the first routine operational monitoring.

An ERROR means some data was not backed up.

A WARNING means data was backed up but an irregular condition requiring investigation was detected.

When there is a warning or an error, the report emails include a summary and, if it is not too big, the log itself. When the log is too big the report emails include the head and tail of the log and the log's name.

Caution: logs in report emails have their UTF-8 converted to US-ASCII and their lines wrapped at 999 characters so they may not be identical to the original logs.

4.1.2 Logs

Logs generated by the supervisory scripts (super_bu.sh and hotplug_bu.sh) can be very long. A useful technique for stepping through these logs is to search for "Running subsidiary script by command". That takes you to the beginning of each job; any error from the previous job is usually shown in the few lines above.

Key lines from logs can be displayed using this alias and giving it the log file name.

```
alias bls='egrep \'\'(^((cannot delete non-empty directory:|rsync error:|rsync: |IO error encountered))|ERROR:|WARN:|Running rsync command:|Invalid Mount fs_file value .* \((does not exist)\)\'\''
```

4.2 Audit

Report emails cannot be relied on to assure that backups are working:

- The scheduling mechanism could fail.
- The email generation and delivery system could fail.
- Bung may fail to detect and report problems.
- Bung itself surely has some bugs.
- The underlying backup tools may – less likely – have bugs.
- The file systems being backed up could be damaged or unmounted.

For these reasons it is prudent to audit the backups, including checking restores can be done from the backups and backup jobs are running on every system they should be running on.

5 Problem investigation

5.1 Report email not received

In case the email is not received, check /var/log/syslog to see if bung sent it:

```
zgrep '_bu.*\.sh\+.*mailed to ' /var/log/syslog*
```

If bung sent it, further analysis of why it was not received depends on the local mail system so is outside the scope of this document.

5.2 bung's logs

In /var/log/bung

5.2.1 rsync

rsync output can be huge so it is convenient to be able to search for known rsync error messages which begin the line with:

- cannot delete non-empty directory:
- rsync error:
- rsync: connection unexpectedly closed
- rsync: delete_file:
- rsync: rename

This command can be used to search the log for those errors:

```
egrep '^(cannot delete non-empty directory:|rsync error:|rsync: |IO error encountered)'
```

In case there are many rsync jobs in the same log file:

```
egrep '^(cannot delete non-empty directory:|rsync error:|rsync: |IO error encountered))|Running rsync command:'
```

This message is not classified as an error by `rsync_bu.sh` (it is caused by the directory containing an excluded subdirectory or file):

- cannot delete non-empty directory

These messages have been seen when `rsync` is deleting a very large number of files in a single directory:

```
[sender] io timeout after 301 seconds -- exiting
rsync error: timeout in data send/receive (code 30) at io.c(140) [sender=3.0.9]
```

The solution is to increase the timeout value and/or to run the job several times until the files are moved. Long term and generally it is better to avoid having a very large number of files in a single directory.

5.2.1.1 *protocol version mismatch -- is your shell clean?*

Can be caused by `.bashrc` generating output. If the output is wanted during interactive bash initialisation, can be fixed by inserting these lines at the head of `.bashrc`:

```
# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac
```

5.2.2 Snapshots

Snapshots cannot be reliably removed. Sometimes a reboot is required to remove obsolete snapshots. If this is not done the snapshot will eventually fill and no further writes will be possible.

5.2.2.1 *Buffer I/O error on device dm-*, logical block **

Followed by "lost page write due to I/O error on dm-*".

This has been cured by increasing the snapshot size sub-keyword value.

5.3 `/var/log/syslog`

`bung` writes start, warning, error and finish messages to `syslog`.

5.4 `bung` debugging

Only developers should need to debug `bung`. In case it has to be done ...

5.4.1 The `-d` option

To add debugging to the log (to file or to screen), use the `-d` option.

5.4.2 No output?

If running other than at a command prompt (for example started by `cron` or `udev`) and there is no output, it may help to test a modified version of the script with the following lines added immediately after the shebang line:

```
exec 1>>/tmp/${0##*/}.$$
exec 2>>/tmp/${0##*/}.$$
```

5.5 `ssh` (remote destination or source)

If a remote destination or source backup does not work when started by scheduled job or `udev` but does work when tested at the command line after `ssh` logging in, the cause may be that the intended `ssh` keys are defective and your own `ssh-agent` forwarded key is being used when the job is tested at the command line. Unsetting any `SSH_AUTH_SOCK` environment variable will prevent this happening.

6 Restores

6.1 mysql_bu.sh

The backup files are mysqldump output for each database (including mysql) pre-pended with SET FOREIGN_KEY_CHECKS=0 and post-pended with SET FOREIGN_KEY_CHECKS=1 to facilitate restore. By default they are compressed with gzip.

The simplest and most robust way to restore is:

- Ensure an empty database
- Decompress
- Restore from within a mysql shell using the SOURCE command

Example:

```
zcat /var/backup/mysql/mydb/mydb.<timestamp>.sql.gz > /tmp/mydb.<timestamp>.sql
mysql
DROP DATABASE mydb;
CREATE DATABASE mydb;
USE mydb;
SOURCE /tmp/mydb.<timestamp>.sql;
```

6.2 postgres_bu.sh

The backup files are:

- Global data (includes roles and tablespaces)
- A pgdump --format=custom output file for each database

Database restore is most easily done as a PostgreSQL superuser such as postgres, ensuring an empty database then using the Linux pg_restore command.

Example:

```
su - postgres
dropdb mydb
pg_restore --create --dbname=postgres /var/backup/postgres/mydb-<timestamp>.sql
```

6.3 rsync_bu.sh

Commonly the backup tree is simply copied to the restore destination excluding any retention subtree such as "Changed and deleted files".

For approximate "point in time" restores, rsync_restore.sh is provided.

7 Worked examples

7.1 rsync_bu.sh - backup to local file system, whole tree

The scenario was separate /, /home and /var on LVMs; backed up as a whole tree. Some advantages of whole tree backup are easier to restore and the backup being a more useful forensic resource. A disadvantage is using more space.

The source directories are on LVM volumes so snapshots are used.

7.1.1 Space calculation

Found by running `df -hT -x devtmpfs -x nfs -x tmpfs`

That gives more than the size of the initial backup because some files are will be excluded but allows no room for retention and dat growth. It can very roughly be doubled to give the required space.

```
# df -hT -x devtmpfs -x nfs -x tmpfs
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/mapper/gas-root ext4   23G   8.0G   14G   37% /
/dev/mapper/gas-var ext4    28G   3.1G   23G   12% /var
/dev/mapper/gas-home ext4  340G   27G   296G    9% /home
/dev/md0         ext3   922M  204M   671M   24% /boot
```

Say 80 GB required.

7.1.2 Create the backup file system

In this case the backup file system already existed and had more than 80 GB free space. Mounted it on /mnt/backup:

```
# mount UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
```

7.1.3 Create destination directory

Optional; in this case the backup type was chosen (this allows for using more than one backup utility).

```
# mkdir /mnt/backup/rsync
```

7.1.4 Create snapshot mountpoint

```
# mkdir -p /mnt/snap-root
```

7.1.5 bung configuration

Created these configuration files.

```
# for f in super whole_tree whole_tree.exclude; do echo ==== $f ====; cat $f; done
==== super ====
Organisation name = GAS
Subsidiary script = rsync_bu.sh whole_tree nice=19 ionice="-c2 -n0"
==== whole_tree ====
Organisation name = GAS
Snapshot = root snap-root /mnt/snap-root size=2048M
Snapshot = home snap-home /mnt/snap-root/home size=2048M ignore_files_under_fs_file
Snapshot = var snap-var /mnt/snap-root/var size=2048M ignore_files_under_fs_file
Mount = UUID=d0d30ea8-68a4-4329-ae0c-c270e31a3ea3 /mnt/backup
rsync = /mnt/snap-root/ /mnt/backup/rsync --exclude-from=whole_tree.exclude
==== whole_tree.exclude ====
# Configuration file for rsync

# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
# 1. --exclude-from=FILE
# 2. FILTER RULES section (search for "filter rules allow")

# Exclude lines for use when synchronising the whole tree
# * Intended for use when /home and /var are not backed up individually
# * Intended to provide all that is necessary for recovery
# * Intended for use when /home is used by GUI desktop users

/dev/*
/etc/mtab
/home/*/.Dropbox/.dropbox.cache/*
/home/*/.Trash/*
/home/*/.VirtualBox/*
/home/*/.adobe/Flash_Player/AssetCache/*
/home/*/.cache/*
/home/*/.compiz-1/session/*
/home/*/.config/nautilus/saved-session-*
/home/*/.gvfs/
/home/*/.macromedia/*
/home/*/.metacity/sessions/*
/home/*/.mozilla/firefox/*/Cache.Trash/*
/home/*/.mozilla/firefox/*/Cache/*
/home/*/.mozilla/firefox/Crash Reports/*
/home/*/.opera/cache/*
/home/*/.thumbnails/*
/home/*/.thunderbird/**.msf
/home/*/.thunderbird/**/Bin
```

```

/home/*/.thunderbird/**/Bin.*
/home/*/.thunderbird/**/Spam
/home/*/.thunderbird/**/Spam.*
/home/*/.thunderbird/**/Trash
/home/*/.thunderbird/**/Trash.*
/home/*/.thunderbird/**/Cache/*
/home/*/.thunderbird/*/global-messages-db.sqlite
/home/*/.thunderbird/Crash Reports/*
/home/*/Trash/*/*
/home/*/urlclassifier3.sqlite
/home/*/xapiandb/*
/home/lost+found/*
/home/media_no_backup/*
/lost+found/*
/media/*
/mnt/*
/proc/*
/run/*
/sys/*
/tmp/*
/var/cache/*
/var/crash/*
/var/lock/*
/var/run/*
/var/tmp/*

```

7.1.6 Testing

Tested:

```
# /opt/bung/super_bu.sh -c super
```

Reviewed the log, created in /var/log/bung; OK.

7.1.7 Create scheduled job

```

# crontab -l
[snip]
30 12 * * 1-6 /opt/bung/super_bu.sh -c super

```

7.2 rsync_bu.sh - backup to local file system, separate /var and /home

The scenario is backup of ac001.unitypav's /etc, /home, /root and /var to a local file system.

The source directories are on LVM volumes for the /home and /var file systems so snapshots are used.

7.2.1 Space calculation

Found by using du -hs on each backup tree root directory and subtracting similar for the excluded sub-trees.

Backup data volume calculation:

1. /etc: 0 G
2. /home without media_no_backup directories: 113 G
3. /root: 0 G
4. /var: 1.5 G

Total 115 G.

The backup file system had already been created and its UUID found by running the blkid command.

```

root@ac001.unitypav:~# mkdir /mnt/backup
root@ac001.unitypav:~# mount UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
root@ac001.unitypav:~# df /mnt/backup
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/sdc1       ext4  914G  355G  513G  41% /mnt/backup

```

7.2.2 Create destination directory

Optional; in this case the backup type was chosen (this allows for using more than one backup utility).

```
root@ac001.unitypav:~# mkdir /mnt/backup/rsync
```

7.2.3 Create snapshot mountpoints

```
root@ac001.unitypav:~# mkdir /mnt/snap-{root,home,var}
```

7.2.4 bung configuration

Created these configuration files.

```
root@ac001.unitypav:/etc/opt/bung# for f in all etc home{,.exclude} root var{,.exclude}
do
    echo === $f ===
    cat $f
done
=== all ===
Organisation name = Unity Pavilion
Subsidiary script = rsync_bu.sh etc nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh home nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh root nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh var nice=19 ionice="-c2 -n0"
=== etc ===
Organisation name = Unity Pavilion
Snapshot = root snap-root /mnt/snap-root size=2048M
Mount = UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
rsync = /mnt/snap-root/etc/ /mnt/backup/rsync/etc
=== home ===
Organisation name = Unity Pavilion
Snapshot = home snap-home /mnt/snap-home size=2048M
Mount = UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
rsync = /mnt/snap-home/ /mnt/backup/rsync/home --exclude-from=home.exclude
=== home.exclude ===
# Configuration file for rsync

# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
#   1. --exclude-from=FILE
#   2. FILTER RULES section (search for "filter rules allow")

# Example exclusion lines for a /home file system

*/.Trash*/
*/.VirtualBox/
*/.adobe/Flash_Player/AssetCache/
*/.cache/
*/.compiz-1/session/
*/.config/nautilus/saved-session-*
*/.gvfs/
*/.local/
*/.macromedia/
*/.metacity/sessions/
*/.mozilla/firefox/*/Cache.Trash*/
*/.mozilla/firefox/*/Cache/
*/.mozilla/firefox/Crash Reports/
*/.opera/cache/
*/.thumbnails/
*/.thunderbird/*/*/*/*/*.msf
*/.thunderbird/*/*/*/*/*/*.msf
*/.thunderbird/*/*/*/*/*/*/*.msf
*/.thunderbird/*/*/*/*/.Bin
*/.thunderbird/*/*/*/*/.Bin.*
*/.thunderbird/*/*/*/*/.Spam
*/.thunderbird/*/*/*/*/.Spam.*
*/.thunderbird/*/*/*/*/.Trash
*/.thunderbird/*/*/*/*/.Trash.*
*/.thunderbird/*/Cache/
*/.thunderbird/*/global-messages-db.sqlite
*/.thunderbird/Crash Reports/
*/Downloads/
*/Trash*/
*/urlclassifier3.sqlite
*/xapiandb
lost+found/
```

```

media_no_backup/
share/media_no_backup/
=== root ===
Organisation name = Unity Pavilion
Snapshot = root snap-root /mnt/snap-root size=2048M
Mount = UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
rsync = /mnt/snap-root/root/ /mnt/backup/rsync/root
=== var ===
Organisation name = Unity Pavilion
Snapshot = var snap-var /mnt/snap-var size=2048M
Mount = UUID=96658be6-2823-4e7b-bdb1-8040810dd381 /mnt/backup
rsync = /mnt/snap-var/ /mnt/backup/rsync/var --exclude-from=var.exclude
=== var.exclude ===
# Configuration file for rsync

# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
#   1. --exclude-from=FILE
#   2. FILTER RULES section (search for "filter rules allow")

# Example exclude lines for a /var file system

cache/
crash/
games/
lock/
lost+found
run/
tmp/

```

7.2.5 Testing

Tested the smallest backup (because it is the fastest to run):

```

root@ac001.unitypav:~# umount /mnt/backup
root@backupserver.townhall:~# /opt/bung/rsync_bu.sh -c etc -l /dev/tty

```

That ran OK, so ran the complete job. If working remotely this is best done in a way that is robust against the connection being lost, such as dtach, nohup or screen. For example:

```

root@backupserver.townhall:~# nohup /opt/bung/super_bu.sh -c all &

```

When that ran OK, configured a scheduled job ...

7.2.6 Create scheduled job

```

root@ac001.unitypav:~# crontab -l
[snip]
7 19 * * * /opt/bung/super_bu.sh -c all

```

7.3 rsync_bu.sh - backup to hotplug disk

Chose the /dev/ symlink name for the hotplug disk: /dev/hotplug

Created a mountpoint for hotplug disk:

```
# mkdir /mnt/hotplug
```

Created bung configuration files, one for hotplug_bu.sh and several for rsync_bu.sh:

```

/etc/opt/bung# for f in hotplug* ; do echo ==== $f ====; cat $f; echo; done
==== hotplug ====
Organisation name = PTDC
Hotplug device = /dev/hotplug notification_screen
Subsidiary script = rsync_bu.sh hotplug.etc nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh hotplug.home nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh hotplug.root nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh hotplug.var nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh hotplug.monthly nice=19 ionice="-c2 -n0"

==== hotplug.etc ====

```



```

Organisation name = PTDC
Snapshot = root snap-root /mnt/snap-root size=2048M
Mount = /dev/hotplug /mnt/hotplug
rsync = /etc/ /mnt/hotplug/rsync/etc

==== hotplug.home ====
Organisation name = PTDC
Snapshot = home snap-home /mnt/snap-home size=2048M
Mount = /dev/hotplug /mnt/hotplug
rsync = /home/ /mnt/hotplug/rsync/home --exclude-from=home.exclude

==== hotplug.monthly ====
Organisation name = PTDC
Mount = UUID=c3874266-3ddd-47b9-bb72-8da24137fb0d /mnt/backup
Mount = /dev/hotplug /mnt/hotplug
rsync = /mnt/backup/rsync/monthly /mnt/hotplug/rsync/monthly retention=0

==== hotplug.root ====
Organisation name = PTDC
Snapshot = root snap-root /mnt/snap-root size=2048M
Mount = /dev/hotplug /mnt/hotplug
rsync = /root/ /mnt/hotplug/rsync/root

==== hotplug.var ====
Organisation name = PTDC
Snapshot = var snap-var /mnt/snap-var size=2048M
Mount = /dev/hotplug /mnt/hotplug
rsync = /var/ /mnt/hotplug/rsync/var --exclude-from=var.exclude

```

Plugged in the USB disk and identified it:

```

# blkid
...
/dev/sde1: LABEL="Elements" UUID="4E1AEA7B1AEA6007" TYPE="ntfs"

```

Confirmed that sde1 was a whole disk partition.

Found sde1 had been automounted so unmounted it.

Formatted it with ext4, mounted it and create the rsync directory:

```

# mkfs.ext4 -L hotplug_backup /dev/sde1
# mount /dev/sde1 /mnt/hotplug
# mkdir /mnt/hotplug/rsync
# umount /mnt/hotplug

```

Gathered information required for the udev rule:

```

# udevadm info -a -p /sys/block/sde/sde1
...
looking at device '/block/sde/sde1':
...
ATTR{size}=="1953456128"
...
[snip some parent device stanzas, looking for a stanza with a serial number]
...
looking at parent device '/devices/pci0000:00/0000:00:1a.0/usb1/1-1/1-1.2':
...
ATTRS{serial}=="57585131453235434A455250"

```

Created the udev rule and effected it:

```

# cat /etc/udev/rules.d/99-bung.rules
# udev rules file for bung

# Western Digital Elements 10B8
KERNEL=="sd*", ACTION=="add", ATTR{size}=="1953456128"
ATTRS{serial}=="57585131453235434A455250", SYMLINK+="hotplug",
RUN+="/opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh -c hotplug -u -d"
KERNEL=="sd*", ACTION=="change", ATTR{size}=="1953456128",
ATTRS{serial}=="57585131453235434A455250", SYMLINK+="hotplug"

```

```
# udevadm control --reload
```

Tested the bung configuration:

```
# ln -s /dev/sde1 /dev/hotplug
# /opt/bung/hotplug_bu_launcher.sh /opt/bung/hotplug_bu.sh -c hotplug -u
```

Created a udev rule to prevent the USB disk's file system from being automounted:

```
# cat /etc/udev/rules.d/99-local-udisks.rules
# The backup USB HDD
SUBSYSTEM=="block", KERNEL=="sd*", ENV{ID_FS_UUID}=="3162eec4-4ae2-487a-834e-3386c9250c3a",
ENV{UDISKS_PRESENTATION_HIDE}:=1"
```

7.4 rsync_bu.sh - backup to remote system (/etc, /home, /root and /var)

The scenario is backup of ac001.th's /etc, /home, /root and /var to backupserver.townhall (a.k.a. bafi).

It is a "push" backup, that is run by the source directory computer, ac001.th.

All source directories are on LVM volumes so snapshots are used.

7.4.1 Create destination directory

Working on the destination server ...

The file system space required for the backup was calculated, allowing for growth and space needed for 28 days retention.

In this case the /mnt/backup/general file system was chosen. Subdirectories were:

1. Department/organisation name: TH
2. Computer name: ac001
3. Backup type: rsync

```
root@backupserver.townhall:~# mkdir -p /mnt/backup/general/TH/ac001/rsync
```

7.4.2 Create the constant access path

This step was only necessary because the destination system was a backup system with many clients and several file systems.

File systems may change to accommodate growth so a constant access path is used. This allows the backup configurations and the operators' view to remain constant as the file systems change.

```
root@backupserver.townhall:~# ln -s /mnt/backup/general/TH /var/backup/TH
```

7.4.3 Create snapshot mountpoints

Working on the source server ...

```
root@ac001.th:~# mkdir /mnt/snap-{home,root,var}
```

7.4.4 Set up ssh

The private key is required on the computer initiating the backup so that's where to generate the key pair:

```
root@ac001.th:~# cd ~/.ssh && ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ac1.th2bafi
[snip]
```

Add the public key to the destination computer's authorized_keys2 file (copy-and-paste is convenient), pre-pending ...

```
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

... and changing the comment field to reflect the keys' name. For example:

```
root@ac001.th:~/.ssh# cat authorized_keys2
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-
forwarding,no-pty ssh-rsa AAAAB3N...1GzId ac1.th2bafi
```

On the initiating computer, created the ssh Host:

```
root@ac001.th:~# cat ~/.ssh/config
# 3 Feb 2014 Charles for TH-177, added host for backup to bafi
Host ac1.th2bafi
    Hostname bafi.bafi.av
    User root
    IdentityFile /root/.ssh/ac1.th2bafi
```

Populated the local /root/.ssh/known_hosts file:

```
root@ac001.th:~# ssh -i /root/.ssh/ac1.th2bafi ac1.th2bafi stat --format=%F /
directory
```

7.4.5 Configure bung

```
root@ac001.th:/etc/opt/bung# for f in all etc{,.exclude} home{,.exclude} root{,.exclude}
var{,.exclude}; do echo === $f ===; cat $f; done
=== all ===
```

```
Organisation name = TH
Subsidiary script = rsync_bu.sh etc nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh home nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh root nice=19 ionice="-c2 -n0"
Subsidiary script = rsync_bu.sh var nice=19 ionice="-c2 -n0"
=== etc ===
Organisation name = TH
rsync = /mnt/snap-root/etc/ ac1.th2bafi:/var/backup/TH/ac001/rsync/etc --exclude-
from=etc.exclude
Snapshot = root snap-root /mnt/snap-root size=2048M
=== etc.exclude ===
# Configuration file for rsync
```

```
# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
#   1. --exclude-from=FILE
#   2. FILTER RULES section (search for "filter rules allow")
```

Exclude lines for the /etc tree

```
mtab
=== home ===
Organisation name = TH
rsync = /mnt/snap-home/ ac1.th2bafi:/var/backup/TH/ac001/rsync/home --exclude-from=home.exclude
Snapshot = home snap-home /mnt/snap-home size=2048M
=== home.exclude ===
# Configuration file for rsync
```

```
# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
#   1. --exclude-from=FILE
#   2. FILTER RULES section (search for "filter rules allow")
```

Example exclusion lines for a /home file system

```
*/.Trash*/
*/.VirtualBox/
*/.adobe/Flash_Player/AssetCache/
*/.cache/
*/.compiz-1/session/
*/.config/nautilus/saved-session-*
*/.gvfs/
*/.local/
*/.macromedia/
*/.metacity/sessions/
*/.mozilla/firefox/*/Cache.Trash*/
*/.mozilla/firefox/*/Cache/
```


7.4.7 Create scheduled job

```
root@ac001.th:~# crontab -l
# 3 March 2014 Charles for TH-177, created crontab and added super_bu.sh -c all
17 17 * * * /opt/bung/super_bu.sh -c all
```

7.5 rsync_bu.sh - backup to remote system (whole tree, no LVM)

The scenario is backup of blueberry.blue.av's whole file system to bafi.th.av.

It is a "push" backup, that is run by the source directory computer, blueberry.blue.av.

There are no LVM volumes so snapshots cannot be used.

There is no local backup, only the push to remote.

7.5.1 Space calculation

```
root@blueberry.blue:~# df -x devtmpfs -x tmpfs
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/sda1       ext4   19G   13G   4.6G   74% /
```

Allowing space for growth and retention, say 25G.

7.5.2 Create destination directory

On the destination server ...

Chose the /mnt/backup/blue file system was chosen. Created subdirectories for computer name and backup type:

```
root@ac001.bafi:~# mkdir -p /mnt/backup/blue/blueberry/rsync
```

7.5.3 Create the constant access path

To allow the backup server file systems and mount points to be changed without having to reconfigure the clients, a "constant access path" is used:

```
root@backupserver.townhall:~# ln -s /mnt/backup/blue /var/backup/BLUE
```

7.5.4 Create snapshot mountpoints

N/A

7.5.5 Set up ssh

The private key is required on the computer initiating the backup so that's where to generate the key pair:

```
root@blueberry.blue:~# cd ~/.ssh && ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): blueberry_to_bafi
Enter passphrase (empty for no passphrase):
[empty]
```

Add the public key to the destination computer's authorized_keys2 file (copy-and-paste is convenient), pre-pending ...

```
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

... and changing the comment field to reflect the keys' name.

For example:

```
root@blueberry.blue:~/.ssh# cat blueberry_to_bafi.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC6qkGR6daQyqctE9rtNeuXEtct/991vBX5L0ZuB4gppS+xePwrh31wzXj
o1lVfkvfr2oZSCyHpmVjRWgSNHDWuFBBRv5usXgabKa9LFikeLNLfUIV3D12GsAPhGtZK7TyJeEpiSZxUvA6nDt
BSqKoAn0xT9Q0MvstAtD4dS2HqUWXERRou9iYCK+yb810P2d6S+MxFE5VYEeFi9nYXG1+CrS3s6pwbi4cDcrtPs
UDxeC2asxV/NBj6J465r190ethIcqtJkSvDe0I/08ppdRJ53LfNY6HSyOzMnuUZhL4hTaJfeYIm7LeFlrkCAw61
Dfvme+bRzJu7Vizx/LvVydgr root@blueberry
root@ac001.bafi:~/.ssh# grep blueberry authorized_keys2
```

```
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-
forwarding,no-pty ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC6qkGR6daQyqctE9rtNeuXEtCT/991vBX5L0ZuB4gppS+xePwrh31wzXj
o1lVfkvfr2oZSCyHpmVjRWgSNHDWuFBBRv5usXgabKa9LFikeLNLfUIV3D12GsAPhGtZK7TyJeEpiSZxUvA6nDt
BSqKoAn0xT9Q0MvstAtD4dS2HqUWXERRou9iYCK+yb810P2d6S+MxFE5VYEeFi9nYXG1+CrS3s6pwbi4cDcrtPs
UDxeC2asxV/NBj6J465r190ethIcqtJkSvDeOI/08ppdRJ53LfNY6HSyOzZMnUZhL4hTaJfeYIm7LeFlrkCAw61
Dfvme+bRzJu7VIzx/LvVydgr blueberry_to_bafi
```

On the initiating computer, created the ssh Host:

```
root@blueberry.blue:~/.ssh# diff config{.org,}
17a18,22
>
> Host blueberry_to_bafi
>     Hostname bafi.th.av
>     User root
>     IdentityFile /root/.ssh/blueberry_to_bafi
```

Populated the local /root/.ssh/known_hosts file:

```
root@blueberry.blue:~# ssh -i /root/.ssh/blueberry_to_bafi bafi.th.av stat --format=%F /
The authenticity of host 'bafi.th.av (192.168.28.2)' can't be established.
RSA key fingerprint is 91:b0:f7:3b:8f:8b:bc:1e:17:93:6b:ad:82:8e:06:43.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'bafi.th.av,192.168.28.2' (RSA) to the list of known hosts.
directory
```

7.5.6 Configure bung

The whole bung config is shown below, not only the whole tree rsync.

```
root@blueberry.blue:/etc/opt/bung# for f in all sysinfo mysql mysql.cnf whole_tree
whole_tree.exclude ; do echo ==== $f ====; cat $f; done
==== all ====
Organisation name = BLUE.blueberry
Subsidiary script = sysinfo_bu.sh sysinfo nice=19 ionice="-c2 -n0"
Subsidiary script = mysql_bu.sh mysql
Subsidiary script = rsync_bu.sh whole_tree
==== sysinfo ====
Organisation name = BLUE.blueberry
sysinfo =
==== mysql ====
Organisation name = BLUE.blueberry
MySQL = compression=bzip2 dest_dir=/var/backup/mysql
==== mysql.cnf ====
# MySQL defaults-file for use with bung's MySQL backup scripts
#
# Format reference: http://dev.mysql.com/doc/refman/5.5/en/option-files.html

[client]
user=backup_user
password=DBrNRNf7dHe6MeKr
==== whole_tree ====
Organisation name = BLUE.blueberry
rsync = / blueberry_to_bafi:/var/backup/BLUE/blueberry/rsync --exclude-from=whole_tree.exclude
==== whole_tree.exclude ====
# Configuration file for rsync

# * Lines beginning with # and empty lines are ignored.
# * The format is defined in the rsync man page under:
#   1. --exclude-from=FILE
#   2. FILTER RULES section (search for "filter rules allow")

# Exclude lines for use when synchronising the whole tree
# * Intended for use when /home and /var are not backed up individually
# * Intended to provide all that is necessary for recovery
# * Intended for use when /home is used by GUI desktop users

**/.gvfs/
/dev/*
/etc/mtab
```

```

/home/lost+found/*
/home/media_no_backup/*
/home/share/media_no_backup/*
/home/*/Trash*/
/home/*/urlclassifier3.sqlite
/home/*/xapiandb/*
/home/*/.Dropbox/.dropbox.cache/*
/home/*/.Trash*/
/home/*/.VirtualBox/*
/home/*/.adobe/Flash_Player/AssetCache/*
/home/*/.cache/*
/home/*/.compiz-1/session/*
/home/*/.config/nautilus/saved-session-*
/home/*/.local/
/home/*/.macromedia/*
/home/*/.metacity/sessions/*
/home/*/.mozilla/firefox/Crash Reports/*
/home/*/.mozilla/firefox/*/Cache.Trash/*
/home/*/.mozilla/firefox/*/Cache/*
/home/*/.opera/cache/*
/home/*/.thumbnails/*
/home/*/.thunderbird/Crash Reports/*
/home/*/.thunderbird/**.msf
/home/*/.thunderbird/**/Bin
/home/*/.thunderbird/**/Bin.*
/home/*/.thunderbird/**/Spam
/home/*/.thunderbird/**/Spam.*
/home/*/.thunderbird/**/Trash
/home/*/.thunderbird/**/Trash.*
/home/*/.thunderbird/**/Cache/*
/home/*/.thunderbird/**/ImapMail/imap.gmail*.com/*
/home/*/.thunderbird/**/ImapMail/imap.googlemail*.com/*
/home/*/.thunderbird/**/global-messages-db.sqlite
/lost+found/*
/media/*
/mnt/*
/proc/*
/run/*
/sys/*
/tmp/*
/var/cache/*
/var/crash/*
/var/lock/*
/var/run/*
/var/tmp/*

```

7.5.7 Testing

```
root@blueberry.blue:~# /opt/bung/rsync_bu.sh -c all
```

When that ran OK, configured a scheduled job ...

7.5.8 Create scheduled job

Chose a cron.daily job because blueberry.blue.av's up times are irregular:

```
root@blueberry.blue:/etc/cron.daily# cat backup
#!/bin/bash
```

```
# 17 Feb 2016 Charles for BLUE-2147
# * Creation
```

```
(
    sleep 600    # Allow boot initialisation load to finish
    /opt/bung/super_bu.sh -c all
)&    # Background to avoid delaying further cron.daily jobs
```

7.6 rsync_bu.sh - backup from remote system

The scenario is backup of deepam-laptop's /etc, /home, /root and /var to ac001.deepam.

It is a "pull" backup, that is run by the destination directory computer, ac001.deepam.

bung does not support LVM snapshots when the source is remote (that may be a reason to prefer "push" backups over "pull" backups).

7.6.1 Calculate the backup data volume

There were no large directories to be excluded from the backup so a top level measurement was accurate enough.

On the source system:

```
root@deepam-laptop:~# du -hs /etc /home /root /var 2>/dev/null
71M /etc
36G /home
4.3M /root
1.1G /var
```

Say 75 GB allowing room for retention and growth.

7.6.2 Finding space on the destination system

There was a file system already being used for local backups. Checked it had enough space:

```
root@ac001.deepam:~# mount UUID=675f5ec6-7b92-4362-b74a-39128c01853c /mnt/backup
root@ac001.deepam:~# df /mnt/backup
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/sdb3       ext4  312G  200G   97G  68% /mnt/backup
```

That was OK for historical reasons, irrelevant here.

7.6.3 Create destination directory

Following the <hostname>/<backup utility> convention:

```
root@ac001.deepam:~# mkdir -p /mnt/backup/deepam-laptop/rsync
```

7.6.4 Set up ssh

The private key is required on the computer initiating the backup so that's where to generate the key pair:

```
root@ac001.deepam:~# cd ~/.ssh && ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): deepam-laptop2ac001.deepam
[snip]
```

Add the public key to the destination computer's authorized_keys file (copy-and-paste is convenient), pre-pending ...

```
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

... and changing the comment field to reflect the keys' name. For example:

```
root@ac001.deepam:~/.ssh# cat deepam-laptop2ac001.deepam.pub
command="/opt/bung/validate_ssh_cmd.sh",no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty ssh-rsa AAAAB3N...1GzId deepam-laptop2ac001.deepam
```

On the initiating (destination) computer, created the ssh Host:

```
root@ac001.deepam:~/.ssh# cat config
# 4 Mar 2014 Charles for DEEPAM-200, added host for backup from deepam-laptop
Host deepam-laptop2ac001.deepam
    Hostname deepam-laptop.deepam.av
    User root
    IdentityFile /root/.ssh/deepam-laptop2ac001.deepam
```

7.6.5 Configure bung

```
root@ac001.deepam:/etc/opt/bung# for f in laptop.all laptop.etc laptop.home home.exclude
laptop.root laptop.var var.exclude; do echo === $f ===; cat $f; done
=== laptop.all ===
Organisation name = Deepam
Subsidiary script = rsync_bu.sh laptop.etc nice=19 ionice="-c2 -n0"
```



```
# 2. FILTER RULES section (search for "filter rules allow")
```

```
# Example exclude lines for a /var file system
```

```
cache/  
crash/  
games/  
lock/  
lost+found  
run/  
tmp/
```

7.6.6 Testing

Tested the smallest backup (because it is the fastest to run):

```
root@ac001.deepam:~# /opt/bung/rsync_bu.sh -c laptop.etc -l /dev/tty
```

When that ran OK, ran the complete job (under dtach, in case of disconnection during this long-running backup):

```
root@ac001.deepam:~# /opt/bung/super_bu.sh -c laptop.all
```

When that ran OK, configured a scheduled job ...

7.6.7 Create scheduled job

```
root@ac001.deepam:~# crontab -l
```

```
...  
# 5 Mar 2014, Charles for DEEPAM-200, set up deepam-laptop backup  
0 10 * * 1-5 /opt/bung/super_bu.sh -c laptop.all
```

7.7 DG-GS1526E_bu.sh

Setting up the TFTP server is outside the scope of this user guide. DG-GS1526E_bu.sh has been tested with tftpd-hpa 5.2 but any TFTP implementation should work.

Unlike the other bung utilities, DG-GS1526E_bu.sh does not need to be run by root (so should not be).

In case the user that will run DG-GS1526E_bu.sh requires it, ssh into each switch to be backed up to populate the user's known_hosts file.

Create the destination file with whatever ownerships the TFTP server requires. For example:

```
# f=/srv/tftp/192.168.28.4.cfg  
# touch $f  
# chown tftp:tftp $f  
# chmod 202 $f  
# ls -l $f  
--w----w- 1 tftp tftp 0 Jul 20 12:38 /srv/tftp/192.168.28.4.cfg
```

In case the user that will run DG-GS1526E_bu.sh is not root, create log and run directories. For example:

```
# mkdir /var/log/bung/managed_switches  
# chown bl:bl /var/log/bung/managed_switches  
# mkdir -p ~bl/var/run/bung  
# chown bl:bl ~bl/var/run/bung
```

In case the user that will run DG-GS1526E_bu.sh does not have read access to /etc/opt/bung, create a dedicated directory. This was not the case when setting up this example.

When running DG-GS1526E_bu.sh, set environment variables for any non-default directories, for example:

```
$ export BUNG_LOG_DIR=/var/log/bung/managed_switches  
$ export BUNG_PID_DIR=/home/bl/var/run/bung
```