

Data management



- [Sharing, copying, and archiving](#)
 - [Archive and compression tools](#)
 - [Copy and synchronization tools](#)
 - [Idioms for the archive](#)
 - [Idioms for the copy](#)
 - [Idioms for the selection of files](#)
 - [Archive media](#)
 - [Removable storage device](#)
 - [Filesystem choice for sharing data](#)
 - [Sharing data via network](#)
- [Backup and recovery](#)
 - [Backup and recovery policy](#)
 - [Backup utility suites](#)
 - [Backup tips](#)
 - [GUI backup](#)
 - [Mount event triggered backup](#)
 - [Timer event triggered backup](#)
- [Data security infrastructure](#)
 - [Key management for GnuPG](#)
 - [Using GnuPG on files](#)
 - [Using GnuPG with Mutt](#)
 - [Using GnuPG with Vim](#)
 - [The MD5 sum](#)
 - [Password keyring](#)
- [Source code merge tools](#)
 - [Extracting differences for source files](#)
 - [Merging updates for source files](#)
 - [Interactive merge](#)

- [Git](#)
 - [Configuration of Git client](#)
 - [Basic Git commands](#)
 - [Git tips](#)
 - [Git references](#)
 - [Other version control systems](#)
-

Tools and tips for managing binary and text data on the Debian system are described.

Sharing, copying, and archiving

⚠ [!WARNING] The uncoordinated write access to actively accessed devices and files from multiple processes must not be done to avoid the [race condition](#). [File locking](#) mechanisms using `flock` (1) may be used to avoid it.

The security of the data and its controlled sharing have several aspects.

- The creation of data archive
- The remote storage access
- The duplication
- The tracking of the modification history
- The facilitation of data sharing
- The prevention of unauthorized file access
- The detection of unauthorized file modification

These can be realized by using some combination of tools.

- Archive and compression tools
- Copy and synchronization tools
- Network filesystems
- Removable storage media
- The secure shell
- The authentication system
- Version control system tools
- Hash and cryptographic encryption tools

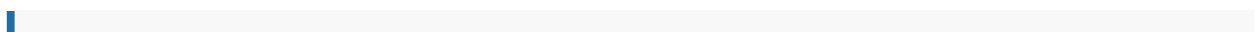
Archive and compression tools

Here is a summary of archive and compression tools available on the Debian system.

package	popcon	size	extension	command	comment
<code>tar</code>	@-@popcon1@-@	@-@psize1@-@	<code>.tar</code>	<code>tar</code> (1)	the standard archiver (de facto standard)
<code>cpio</code>	@-@popcon1@-@	@-@psize1@-@	<code>.cpio</code>	<code>cpio</code> (1)	Unix System V style archiver, use with <code>find</code> (1)
<code>binutils</code>	@-@popcon1@-@	@-@psize1@-@	<code>.ar</code>	<code>ar</code> (1)	archiver for the creation of static libraries
<code>fastjar</code>	@-@popcon1@-@	@-@psize1@-@	<code>.jar</code>	<code>fastjar</code> (1)	archiver for Java (zip like)
<code>pax</code>	@-@popcon1@-@	@-@psize1@-@	<code>.pax</code>	<code>pax</code> (1)	new POSIX standard archiver, compromise between <code>tar</code> and <code>cpio</code>
<code>gzip</code>	@-@popcon1@-@	@-@psize1@-@	<code>.gz</code>	<code>gzip</code> (1), <code>zcat</code> (1), ...	GNU LZ77 compression utility (de facto standard)
<code>bzip2</code>	@-@popcon1@-@	@-@psize1@-@	<code>.bz2</code>	<code>bzip2</code> (1), <code>bzcat</code> (1), ...	Burrows-Wheeler block-sorting compression utility with higher compression ratio than <code>gzip</code> (1) (slower than <code>gzip</code> with similar syntax)
<code>lzma</code>	@-@popcon1@-@	@-@psize1@-@	<code>.lzma</code>	<code>lzma</code> (1)	LZMA compression utility with higher compression ratio than <code>gzip</code> (1) (deprecated)

package	popcon	size	extension	command	comment
xz-utils	@-@popcon1@-@	@-@psize1@-@	.xz	xz (1), xzdec (1), ...	XZ compression utility with higher compression ratio than bzip2 (1) (slower than gzip but faster than bzip2 ; replacement for LZMA compression utility)
zstd	@-@popcon1@-@	@-@psize1@-@	.zstd	zstd (1), zstdcat (1), ...	Zstandard fast lossless compression utility
p7zip	@-@popcon1@-@	@-@psize1@-@	.7z	7zr (1), p7zip (1)	7-Zip file archiver with high compression ratio (LZMA compression)
p7zip-full	@-@popcon1@-@	@-@psize1@-@	.7z	7z (1), 7za (1)	7-Zip file archiver with high compression ratio (LZMA compression and others)
lzop	@-@popcon1@-@	@-@psize1@-@	.lzo	lzop (1)	LZO compression utility with higher compression and decompression speed than gzip (1) (lower compression ratio than gzip with similar syntax)
zip	@-@popcon1@-@	@-@psize1@-@	.zip	zip (1)	InfoZIP : DOS archive and compression tool
unzip	@-@popcon1@-@	@-@psize1@-@	.zip	unzip (1)	InfoZIP : DOS unarchive and decompression tool

List of archive and compression tools



[!WARNING] Do not set the "\$TAPE" variable unless you know what to expect. It changes `tar`(1) behavior.

- The gzipped `tar`(1) archive uses the file extension ".tgz" or ".tar.gz".
- The xz-compressed `tar`(1) archive uses the file extension ".txz" or ".tar.xz".
- Popular compression method in [FOSS](#) tools such as `tar`(1) has been moving as follows:
`gzip` → `bzip2` → `xz`
- `cp`(1), `scp`(1) and `tar`(1) may have some limitation for special files. `cpio`(1) is most versatile.
- `cpio`(1) is designed to be used with `find`(1) and other commands and suitable for creating backup scripts since the file selection part of the script can be tested independently.
- Internal structure of Libreoffice data files are ".jar" file which can be opened also by `unzip`.
- The de-facto cross platform archive tool is `zip`. Use it as "`zip -rX`" to attain the maximum compatibility. Use also the "-s" option, if the maximum file size matters.

Copy and synchronization tools

Here is a summary of simple copy and backup tools available on the Debian system.

package	popcon	size	tool	function
<code>coreutils</code>	@-@popcon1@-@	@-@psize1@-@	GNU cp	locally copy files and directories ("-a" for recursive)
<code>openssh-client</code>	@-@popcon1@-@	@-@psize1@-@	scp	remotely copy files and directories (client, "-r" for recursive)
<code>openssh-server</code>	@-@popcon1@-@	@-@psize1@-@	sshd	remotely copy files and directories (remote server)
<code>rsync</code>	@-@popcon1@-@	@-@psize1@-@		1-way remote synchronization and backup
<code>unison</code>	@-@popcon1@-@	@-@psize1@-@		2-way remote synchronization and backup

List of copy and synchronization tools

Copying files with `rsync`(8) offers richer features than others.

- delta-transfer algorithm that sends only the differences between the source files and the existing files in the destination
- quick check algorithm (by default) that looks for files that have changed in size or in last-modified time
- "`--exclude`" and "`--exclude-from`" options similar to `tar`(1)
- "a trailing slash on the source directory" syntax that avoids creating an additional directory level at the destination.

“ [!TIP] Version control system (VCS) tools in [List of other version control system tools](#) can function as the multi-way copy and synchronization tools.

Idioms for the archive

Here are several ways to archive and unarchive the entire content of the directory "`./source`" using different tools.

GNU `tar`(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

Alternatively, by the following.

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -T -
```

`cpio`(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

Idioms for the copy

Here are several ways to copy the entire content of the directory "`./source`" using different tools.

- Local copy: "`./source`" directory → "`/dest`" directory
- Remote copy: "`./source`" directory at local host → "`/dest`" directory at "`user@host.dom`" host

`rsync`(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

You can alternatively use "a trailing slash on the source directory" syntax.

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

Alternatively, by the following.

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU `cp`(1) and openSSH `scp`(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU `tar`(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvpf - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvpf - )'
```

`cpio`(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

You can substitute "." with "foo" for all examples containing "." to copy files from "./source/foo" directory to "/dest/foo" directory.

You can substitute "." with the absolute path "/path/to/source/foo" for all examples containing "." to drop "cd ./source;". These copy files to different locations depending on tools used as follows.

- "/dest/foo": `rsync`(8), GNU `cp`(1), and `scp`(1)
- "/dest/path/to/source/foo": GNU `tar`(1), and `cpio`(1)

“ [!TIP] `rsync`(8) and GNU `cp`(1) have option "-u" to skip files that are newer on the receiver.

Idioms for the selection of files

`find`(1) is used to select files for archive and copy commands (see [Idioms for the archive](#) and [Idioms for the copy](#)) or for `xargs`(1) (see [???](#)). This can be enhanced by using its command arguments.

Basic syntax of `find`(1) can be summarized as the following.

- Its conditional arguments are evaluated from left to right.
- This evaluation stops once its outcome is determined.
- "Logical **OR**" (specified by "`-o`" between conditionals) has lower precedence than "logical **AND**" (specified by "`-a`" or nothing between conditionals).
- "Logical **NOT**" (specified by "`!`" before a conditional) has higher precedence than "logical **AND**".
- "`-prune`" always returns logical **TRUE** and, if it is a directory, searching of file is stopped beyond this point.
- "`-name`" matches the base of the filename with shell glob (see [???](#)) but it also matches its initial "`.`" with metacharacters such as "`*`" and "`?`". (New [POSIX](#) feature)
- "`-regex`" matches the full path with emacs style **BRE** (see [???](#)) as default.
- "`-size`" matches the file based on the file size (value preceded with "`+`" for larger, preceded with "`-`" for smaller)
- "`-newer`" matches the file newer than the one specified in its argument.
- "`-print0`" always returns logical **TRUE** and print the full filename ([null terminated](#)) on the standard output.

`find`(1) is often used with an idiomatic style as the following.

```
# find /path/to \  
-xdev -regextype posix-extended \  
-type f -regex ".*\.cpio|.*~" -prune -o \  
-type d -regex ".*\/\.git" -prune -o \  
-type f -size +99M -prune -o \  
-type f -newer /path/to/timestamp -print0
```

This means to do following actions.

1. Search all files starting from "`/path/to`"
2. Globally limit its search within its starting filesystem and uses **ERE** (see [???](#)) instead
3. Exclude files matching regex of "`.*\.cpio`" or "`.*~`" from search by stop processing
4. Exclude directories matching regex of "`.*\/\.git`" from search by stop processing
5. Exclude files larger than 99 Megabytes (units of 1048576 bytes) from search by stop processing

6. Print filenames which satisfy above search conditions and are newer than "

```
/path/to/timestamp"
```

Please note the idiomatic use of "-prune -o" to exclude files in the above example.

“ [!NOTE] For non-Debian [Unix-like](#) system, some options may not be supported by `find` (1). In such a case, please consider to adjust matching methods and replace "-print0" with "-print". You may need to adjust related commands too.

Archive media

When choosing [computer data storage media](#) for important data archive, you should be careful about their limitations. For small personal data backup, I use CD-R and DVD-R by the brand name company and store in a cool, shaded, dry, clean environment. (Tape archive media seem to be popular for professional use.)

“ [!NOTE] [A fire-resistant safe](#) are meant for paper documents. Most of the computer data storage media have less temperature tolerance than paper. I usually rely on multiple secure encrypted copies stored in multiple secure locations.

Optimistic storage life of archive media seen on the net (mostly from vendor info).

- 100+ years : Acid free paper with ink
- 100 years : Optical storage (CD/DVD, CD/DVD-R)
- 30 years : Magnetic storage (tape, floppy)
- 20 years : Phase change optical storage (CD-RW)

These do not count on the mechanical failures due to handling etc.

Optimistic write cycle of archive media seen on the net (mostly from vendor info).

- 250,000+ cycles : Harddisk drive
- 10,000+ cycles : Flash memory
- 1,000 cycles : CD/DVD-RW
- 1 cycles : CD/DVD-R, paper

[!CAUTION] Figures of storage life and write cycle here should not be used for decisions on any critical data storage. Please consult the specific product information provided by the manufacture.

“ [!TIP] Since CD/DVD-R and paper have only 1 write cycle, they inherently prevent accidental data loss by overwriting. This is advantage!

“ [!TIP] If you need fast and frequent backup of large amount of data, a hard disk on a remote host linked by a fast network connection, may be the only realistic option.

“ [!TIP] If you use re-writable media for your backups, use of filesystem such as [btrfs](#) or [zfs](#) which supports read-only snapshots may be a good idea.

Removable storage device

Removable storage devices may be any one of the following.

- [USB flash drive](#)
- [Hard disk drive](#)
- [Optical disc drive](#)
- Digital camera
- Digital music player

They may be connected via any one of the following.

- [USB](#)
- [IEEE 1394 / FireWire](#)

Modern desktop environments such as GNOME and KDE can mount these removable devices automatically without a matching "`/etc/fstab`" entry.

- `udisks2` package provides a daemon and associated utilities to mount and unmount these devices.
- [D-bus](#) creates events to initiate automatic processes.

- [PolicyKit](#) provides required privileges.

“ [!TIP] Automounted devices may have the "uhelper=" mount option which is used by `umount` (8).

“ [!TIP] Automounting under modern desktop environment happens only when those removable media devices are not listed in "`/etc/fstab`".

Mount point under modern desktop environment is chosen as "`/media/username/disk_label`" which can be customized by the following.

- `mlabel` (1) for FAT filesystem
- `genisoimage` (1) with "`-v`" option for ISO9660 filesystem
- `tune2fs` (1) with "`-L`" option for ext2/ext3/ext4 filesystem

“ [!TIP] The choice of encoding may need to be provided as mount option (see [???](#)).

“ [!TIP] The use of the GUI menu to unmount a filesystem may remove its dynamically generated device node such as "`/dev/sdc`". If you wish to keep its device node, unmount it with the `umount` (8) command from the shell prompt.

Filesystem choice for sharing data

When sharing data with other system via removable storage device, you should format it with common [filesystem](#) supported by both systems. Here is a list of filesystem choices.

“ [!NOTE] Statements on hard disk ([HDD](#)) are applicable to other storage devices such as [SSD](#) / [USB flash drive](#) / [Memory card](#) / Replace device names in examples such as `/dev/sda` with applicable device names `/dev/nvme0`, `/dev/mmcblk0`,

filesystem name	typical usage scenario
FAT12	cross platform sharing of data on the floppy disk (<32MiB)
FAT16	cross platform sharing of data on the small hard disk like device (<2GiB)
FAT32	cross platform sharing of data on the large hard disk like device (<8TiB, supported by newer than MS Windows95 OSR2)
exFAT	cross platform sharing of data on the large hard disk like device (<512TiB, supported by WindowsXP, Mac OS X Snow Leopard 10.6.5, and Linux kernel since 5.4 release)
NTFS	cross platform sharing of data on the large hard disk like device (supported natively on MS Windows NT and later version, and supported by NTFS-3G via FUSE on Linux)
ISO9660	cross platform sharing of static data on CD-R and DVD+/-R
UDF	incremental data writing on CD-R and DVD+/-R (new)
MINIX	space efficient unix file data storage on the floppy disk
ext2	sharing of data on the hard disk like device with older Linux systems
ext3	sharing of data on the hard disk like device with older Linux systems
ext4	sharing of data on the hard disk like device with current Linux systems
btrfs	sharing of data on the hard disk like device with current Linux systems with read-only snapshots

List of filesystem choices for removable storage devices with typical usage scenarios

“ [!TIP] See [???](#) for cross platform sharing of data using device level encryption.

The FAT filesystem is supported by almost all modern operating systems and is quite useful for the data exchange purpose via removable hard disk like media.

When formatting removable hard disk like devices for cross platform sharing of data with the FAT filesystem, the following should be safe choices.

- Partitioning them with `fdisk`(8), `cfdisk`(8) or `parted`(8) (see [???](#)) into a single primary partition and to mark it as the following.
 - Type "6" for FAT16 for media smaller than 2GB.
 - Type "c" for FAT32 (LBA) for larger media.

- Formatting the primary partition with `mkfs.vfat` (8) with the following.
 - Just its device name, e.g. `"/dev/sda1"` for FAT16
 - The explicit option and its device name, e.g. `"-F 32 /dev/sda1"` for FAT32

When using the FAT or ISO9660 filesystems for sharing data, the following should be the safe considerations.

- Archiving files into an archive file first using `tar` (1), or `cpio` (1) to retain the long filename, the symbolic link, the original Unix file permission and the owner information.
- Splitting the archive file into less than 2 GiB chunks with the `split` (1) command to protect it from the file size limitation.
- Encrypting the archive file to secure its contents from the unauthorized access.

“ [!NOTE] For FAT filesystems by its design, the maximum file size is $(2^{32} - 1)$ bytes = (4GiB - 1 byte). For some applications on the older 32 bit OS, the maximum file size was even smaller $(2^{31} - 1)$ bytes = (2GiB - 1 byte). Debian does not suffer the latter problem.

“ [!NOTE] Microsoft itself does not recommend to use FAT for drives or partitions of over 200 MB. Microsoft highlights its short comings such as inefficient disk space usage in their "[Overview of FAT, HPFS, and NTFS File Systems](#)". Of course, we should normally use the ext4 filesystem for Linux.

“ [!TIP] For more on filesystems and accessing filesystems, please read "[Filesystems HOWTO](#)".

Sharing data via network

When sharing data with other system via network, you should use common service. Here are some hints.

network service	description of typical usage scenario
SMB/CIFS network mounted filesystem with Samba	sharing files via "Microsoft Windows Network", see <code>smb.conf</code> (5) and The Official Samba 3.x.x HOWTO and Reference Guide or the <code>samba-doc</code> package
NFS network mounted filesystem with the Linux kernel	sharing files via "Unix/Linux Network", see <code>exports</code> (5) and Linux NFS-HOWTO

network service	description of typical usage scenario
HTTP service	sharing file between the web server/client
HTTPS service	sharing file between the web server/client with encrypted Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
FTP service	sharing file between the FTP server/client

List of the network service to choose with the typical usage scenario

Although these filesystems mounted over network and file transfer methods over network are quite convenient for sharing data, these may be insecure. Their network connection must be secured by the following.

- Encrypt it with [SSL/TLS](#)
- Tunnel it via [SSH](#)
- Tunnel it via [VPN](#)
- Limit it behind the secure firewall

See also [???](#) and [???](#).

Backup and recovery

We all know that computers fail sometime or human errors cause system and data damages. Backup and recovery operations are the essential part of successful system administration. All possible failure modes hit you some day.

“ [!TIP] Keep your backup system simple and backup your system often. Having backup data is more important than how technically good your backup method is.

Backup and recovery policy

There are 3 key factors which determine actual backup and recovery policy.

1. Knowing what to backup and recover.
 - Data files directly created by you: data in "[~/](#)"

- Data files created by applications used by you: data in `"/var/"` (except `"/var/cache/"`, `"/var/run/"`, and `"/var/tmp/"`)
- System configuration files: data in `"/etc/"`
- Local programs: data in `"/usr/local/"` or `"/opt/"`
- System installation information: a memo in plain text on key steps (partition, ...)
- Proven set of data: confirmed by experimental recovery operations in advance
 - Cron job as a user process: files in `"/var/spool/cron/crontabs"` directory and restart `cron`(8). See ??? for `cron`(8) and `crontab`(1).
 - Systemd timer jobs as user processes: files in `~/.config/systemd/user"` directory. See `systemd.timer`(5) and `systemd.service`(5).
 - Autostart jobs as user processes: files in `~/.config/autostart"` directory. See [Desktop Application Autostart Specification](#).

2. Knowing how to backup and recover.

- Secure storage of data: protection from overwrite and system failure
- Frequent backup: scheduled backup
- Redundant backup: data mirroring
- Fool proof process: easy single command backup

3. Assessing risks and costs involved.

- Risk of data when lost
 - Data should be at least on different disk partitions preferably on different disks and machines to withstand the filesystem corruption. Important data are best stored on a read-only filesystem. [^1]
- Risk of data when breached
 - Sensitive identity data such as `"/etc/ssh/ssh_host_*_key"`, `~/.gnupg/*"`, `~/.ssh/*"`, `~/.local/share/keyrings/*"`, `"/etc/passwd"`, `"/etc/shadow"`, `popularity-contest.conf"`, `"/etc/ppp/pap-secrets"`, and `"/etc/exim4/passwd.client"` should be backed up as encrypted. [^2] (See ???.)
 - Never hard code system login password nor decryption passphrase in any script even on any trusted system. (See [Password keyring](#).)
- Failure mode and their possibility
 - Hardware (especially HDD) will break
 - Filesystem may be corrupted and data in it may be lost
 - Remote storage system can't be trusted for security breaches
 - Weak password protection can be easily compromised
 - File permission system may be compromised
- Required resources for backup: human, hardware, software, ...
 - Automatic scheduled backup with cron job or systemd timer job

“ [!TIP] You can recover debconf configuration data with `"debconf-set-selections debconf-selections"` and dpkg selection data with `"dpkg --set-selection <dpkg-selections.list"`.

[!NOTE] Do not back up the pseudo-filesystem contents found on `/proc`, `/sys`, `/tmp`, and `/run` (see [???](#) and [???](#)). Unless you know exactly what you are doing, they are huge useless data.

“ [!NOTE] You may wish to stop some application daemons such as MTA (see [???](#)) while backing up data.

Backup utility suites

Here is a select list of notable backup utility suites available on the Debian system.

package	popcon	size	description
<code>bacula-common</code>	@-@popcon1@-@	@-@psize1@-@	Bacula : network backup, recovery and verification - common support files
<code>bacula-client</code>	@-@popcon1@-@	@-@psize1@-@	Bacula : network backup, recovery and verification - client meta-package
<code>bacula-console</code>	@-@popcon1@-@	@-@psize1@-@	Bacula : network backup, recovery and verification - text console
<code>bacula-server</code>	@-@popcon1@-@	@-@psize1@-@	Bacula : network backup, recovery and verification - server meta-package
<code>amanda-common</code>	@-@popcon1@-@	@-@psize1@-@	Amanda : Advanced Maryland Automatic Network Disk Archiver (Libs)
<code>amanda-client</code>	@-@popcon1@-@	@-@psize1@-@	Amanda : Advanced Maryland Automatic Network Disk Archiver (Client)
<code>amanda-server</code>	@-@popcon1@-@	@-@psize1@-@	Amanda : Advanced Maryland Automatic Network Disk Archiver (Server)

package	popcon	size	description
<code>backuppc</code>	@-@popcon1@-@	@-@psize1@-@	BackupPC is a high-performance, enterprise-grade system for backing up PCs (disk based)
<code>duplicity</code>	@-@popcon1@-@	@-@psize1@-@	(remote) incremental backup
<code>deja-dup</code>	@-@popcon1@-@	@-@psize1@-@	GUI frontend for duplicity
<code>borgbackup</code>	@-@popcon1@-@	@-@psize1@-@	(remote) deduplicating backup
<code>borgmatic</code>	@-@popcon1@-@	@-@psize1@-@	borgbackup helper
<code>rdiff-backup</code>	@-@popcon1@-@	@-@psize1@-@	(remote) incremental backup
<code>restic</code>	@-@popcon1@-@	@-@psize1@-@	(remote) incremental backup
<code>backupninja</code>	@-@popcon1@-@	@-@psize1@-@	lightweight, extensible meta-backup system
<code>slbackup</code>	@-@popcon1@-@	@-@psize1@-@	(remote) incremental backup
<code>backup-manager</code>	@-@popcon1@-@	@-@psize1@-@	command-line backup tool
<code>backup2l</code>	@-@popcon1@-@	@-@psize1@-@	low-maintenance backup/restore tool for mountable media (disk based)

List of backup suite utilities

Backup tools have their specialized focuses.

- [Mondo Rescue](#) is a backup system to facilitate restoration of complete system quickly from backup CD/DVD etc. without going through normal system installation processes.
- [Bacula](#), [Amanda](#), and [BackupPC](#) are full featured backup suite utilities which are focused on regular backups over network.
- [Duplicity](#), and [Borg](#) are simpler backup utilities for typical workstations.

Backup tips

For a personal workstation, full featured backup suite utilities designed for the server environment may not serve well. At the same time, existing backup utilities for workstations may have some shortcomings.

Here are some tips to make backup easier with minimal user efforts. These techniques may be used with any backup utilities.

For demonstration purpose, let's assume the primary user and group name to be `penguin` and create a backup and snapshot script example `/usr/local/bin/bkss.sh` as:

```
#!/bin/sh -e
SRC="$1" # source data path
DSTFS="$2" # backup destination filesystem path
DSTSV="$3" # backup destination subvolume name
DSTSS="${DSTFS}/${DSTSV}-snapshot" # snapshot destination path
if [ "$(stat -f -c %T "$DSTFS")" != "btrfs" ]; then
    echo "E: $DSTFS needs to be formatted to btrfs" >&2 ; exit 1
fi
MSGID=$(notify-send -p "bkup.sh $DSTSV" "in progress ...")
if [ ! -d "$DSTFS/$DSTSV" ]; then
    btrfs subvolume create "$DSTFS/$DSTSV"
    mkdir -p "$DSTSS"
fi
rsync -aHxS --delete --mkpath "${SRC}/" "${DSTFS}/${DSTSV}"
btrfs subvolume snapshot -r "${DSTFS}/${DSTSV}" "${DSTSS}/${date -u --iso=min}"
notify-send -r "$MSGID" "bkup.sh $DSTSV" "finished!"
```

Here, only the basic tool `rsync` (1) is used to facilitate system backup and the storage space is efficiently used by [Btrfs](#).

“ [!TIP] FYI: This author uses his own similar shell script "[bss: Btrfs Subvolume Snapshot Utility](#)" for his workstation.

GUI backup

Here is an example to setup the single GUI click backup.

- Prepare a USB storage device to be used for backup.
 - Format a USB storage device with one partition in btrfs with its label name as "`BKUP`". This can be encrypted (see [???](#)).
 - Plug this in to your system. The desktop system should automatically mount it as "`/media/penguin/BKUP`".
 - Execute "`sudo chown penguin:penguin /media/penguin/BKUP`" to make it writable by the user.

- Create "`~/.local/share/applications/BKUP.desktop`" following techniques written in [???](#) as:

```
[Desktop Entry]
Name=bkss
Comment=Backup and snapshot of ~/Documents
Exec=/usr/local/bin/bkss.sh /home/penguin/Documents /media/penguin/BKUP Documents
Type=Application
```

For each GUI click, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

Mount event triggered backup

Here is an example to setup for the automatic backup triggered by the mount event.

- Prepare a USB storage device to be used for backup as in [GUI backup](#).
- Create a systemd service unit file "`~/.config/systemd/user/back-BKUP.service`" as:

```
[Unit]
Description=USB Disk backup
Requires=media-%u-BKUP.mount
After=media-%u-BKUP.mount

[Service]
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log

[Install]
WantedBy=media-%u-BKUP.mount
```

- Enable this systemd unit configuration with the following:

```
$ systemctl --user enable bkup-BKUP.service
```

For each mount event, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

Here, names of systemd mount units that systemd currently has in memory can be asked to the service manager of the calling user with "`systemctl --user list-units --type=mount`".

Timer event triggered backup

Here is an example to setup for the automatic backup triggered by the timer event.

- Prepare a USB storage device to be used for backup as in [GUI backup](#).
- Create a systemd timer unit file "`~/.config/systemd/user/snap-Documents.timer`" as:

```
[Unit]
Description=Run btrfs subvolume snapshot on timer
Documentation=man:btrfs(1)

[Timer]
OnStartupSec=30
OnUnitInactiveSec=900

[Install]
WantedBy=timers.target
```

- Create a systemd service unit file "`~/.config/systemd/user/snap-Documents.service`" as:

```
[Unit]
Description=Run btrfs subvolume snapshot
Documentation=man:btrfs(1)

[Service]
Type=oneshot
Nice=15
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
IOSchedulingClass=idle
CPUSchedulingPolicy=idle
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log
```

- Enable this systemd unit configuration with the following:

```
$ systemctl --user enable snap-Documents.timer
```

For each timer event, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

Here, names of systemd timer user units that systemd currently has in memory can be asked to the service manager of the calling user with "`systemctl --user list-units --type=timer`".

For the modern desktop system, this systemd approach can offer more fine grained control than the traditional Unix ones using `at` (1), `cron` (8), or `anacron` (8).

Data security infrastructure

The data security infrastructure is provided by the combination of data encryption tool, message digest tool, and signature tool.

package	popcon	size	command	description
<code>gnupg</code>	@-@popcon1@-@	@-@psize1@-@	<code>gpg</code> (1)	GNU Privacy Guard - OpenPGP encryption and signing tool
<code>gpgv</code>	@-@popcon1@-@	@-@psize1@-@	<code>gpgv</code> (1)	GNU Privacy Guard - OpenPGP signature verification tool
<code>sq</code>	@-@popcon1@-@	@-@psize1@-@	<code>sq</code> (1)	Sequoia-PGP - OpenPGP encryption and signing tool
<code>sqv</code>	@-@popcon1@-@	@-@psize1@-@	<code>sqv</code> (1)	Sequoia-PGP - OpenPGP signature verification tool
<code>paperkey</code>	@-@popcon1@-@	@-@psize1@-@	<code>paperkey</code> (1)	extract just the secret information out of OpenPGP secret keys
<code>cryptsetup</code>	@-@popcon1@-@	@-@psize1@-@	<code>cryptsetup</code> (8), ...	utilities for dm-crypt block device encryption supporting LUKS
<code>coreutils</code>	@-@popcon1@-@	@-@psize1@-@	<code>md5sum</code> (1)	compute and check MD5 message digest
<code>coreutils</code>	@-@popcon1@-@	@-@psize1@-@	<code>sha1sum</code> (1)	compute and check SHA1 message digest
<code>openssl</code>	@-@popcon1@-@	@-@psize1@-@	<code>openssl</code> (1ssl)	compute message digest with " <code>openssl dgst</code> " (OpenSSL)
<code>libsecret-tools</code>	@-@popcon1@-@	@-@psize1@-@	<code>secret-tool</code> (1)	store and retrieve passwords (CLI)
<code>seahorse</code>	@-@popcon1@-@	@-@psize1@-@	<code>seahorse</code> (1)	key management tool (GNOME)

List of data security infrastructure tools

See [???](#) on [dm-crypt](#) and [fscrypt](#) which implement automatic data encryption infrastructure via Linux kernel modules.

Key management for GnuPG

Here are [GNU Privacy Guard](#) commands for the basic key management.

command	description
<code>gpg --gen-key</code>	generate a new key
<code>gpg --gen-revoke my_user_ID</code>	generate revoke key for my_user_ID
<code>gpg --edit-key user_ID</code>	edit key interactively, "help" for help
<code>gpg -o file --export</code>	export all public keys to file
<code>gpg -o file --export-secret-keys</code>	export all private key to file
<code>gpg --import file</code>	import all keys from file
<code>gpg --send-keys user_ID</code>	send key of user_ID to keyserver
<code>gpg --recv-keys user_ID</code>	recv. key of user_ID from keyserver
<code>gpg --list-keys user_ID</code>	list keys of user_ID
<code>gpg --list-sigs user_ID</code>	list sig. of user_ID
<code>gpg --check-sigs user_ID</code>	check sig. of user_ID
<code>gpg --fingerprint user_ID</code>	check fingerprint of user_ID
<code>gpg --refresh-keys</code>	update local keyring

List of GNU Privacy Guard commands for the key management

Here is the meaning of the trust code.

code	description of trust
<code>-</code>	no owner trust assigned / not yet calculated
<code>e</code>	trust calculation failed
<code>q</code>	not enough information for calculation
<code>n</code>	never trust this key
<code>m</code>	marginally trusted
<code>f</code>	fully trusted
<code>u</code>	ultimately trusted

List of the meaning of the trust code

The following generates my key "`9563FC29932C409F1A77F9C1AB8A1522D8234C6A`".

```
$ gpg --gen-key
gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
...
GnuPG needs to construct a user ID to identify your key.

Real name: Osamu Aoki
Email address: osamu@debian.org
You selected this USER-ID:
    "Osamu Aoki <osamu@debian.org>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
...
public and secret key created and signed.

pub   ed25519 2026-02-14 [SC] [expires: 2029-02-13]
       9563FC29932C409F1A77F9C1AB8A1522D8234C6A
uid           Osamu Aoki <osamu@debian.org>
sub   cv25519 2026-02-14 [E] [expires: 2029-02-13]
```

The following uploads my key "`9563FC29932C409F1A77F9C1AB8A1522D8234C6A`" to the popular keyserver "`hkp://keys.gnupg.net`".

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 9563FC29932C409F1A77F9C1AB8A1522D8234C6A
```

A good default keyserver set up in "`~/.gnupg/gpg.conf`" (or old location "`~/.gnupg/options`") contains the following.

```
keyserver hkp://keys.gnupg.net
```

The following obtains unknown keys from the keyserver.

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
    cut -d ':' -f 5 | sort | uniq | xargs gpg --recv-keys
```

There was a bug in [OpenPGP Public Key Server](#) (pre version 0.9.6) which corrupted key with more than 2 sub-keys. The newer `gnupg` (>1.2.1-2) package can handle these corrupted subkeys. See `gpg` (1) under "`--repair-pks-subkey-bug`" option.

Use of SHA-1 for hash has been deprecated. If your old RSA based openPGP key uses SHA-1 for hash, fix it using [FixKeyWithSha1](#).

“ [!TIP] The `sq`(1) and `sqv`(1) commands are an alternative set of openPGP commands. See [sq user documentation](#) and [A Practical Introduction to using sq, Sequoia PGP's CLI](#).

Using GnuPG on files

Here are examples for using [GNU Privacy Guard](#) commands on files.

command	description
<code>gpg -a -s file</code>	sign file into ASCII armored file.asc
<code>gpg --armor --sign file</code>	, ,
<code>gpg --clearsign file</code>	clear-sign message
<code>`gpg --clearsign file</code>	mail foo@example.org`
<code>gpg --clearsign --not-dash-escaped patchfile</code>	clear-sign patchfile
<code>gpg --verify file</code>	verify clear-signed file
<code>gpg -o file.sig -b file</code>	create detached signature
<code>gpg -o file.sig --detach-sign file</code>	, ,
<code>gpg --verify file.sig file</code>	verify file with file.sig
<code>gpg -o crypt_file.gpg -r name -e file</code>	public-key encryption intended for name from file to binary crypt_file.gpg
<code>gpg -o crypt_file.gpg --recipient name --encrypt file</code>	, ,
<code>gpg -o crypt_file.asc -a -r name -e file</code>	public-key encryption intended for name from file to ASCII armored crypt_file.asc
<code>gpg -o crypt_file.gpg -c file</code>	symmetric encryption from file to crypt_file.gpg
<code>gpg -o crypt_file.gpg --symmetric file</code>	, ,
<code>gpg -o crypt_file.asc -a -c file</code>	symmetric encryption intended for name from file to ASCII armored crypt_file.asc
<code>gpg -o file -d crypt_file.gpg -r name</code>	decryption
<code>gpg -o file --decrypt crypt_file.gpg</code>	, ,

List of GNU Privacy Guard commands on files

Using GnuPG with Mutt

Add the following to "`~/.muttrc`" to keep a slow GnuPG from automatically starting, while allowing it to be used by typing "`S`" at the index menu.

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

Using GnuPG with Vim

The `gnupg` plugin let you run GnuPG transparently for files with extension "`.gpg`", "`.asc`", and "`.pgp`".^[^3]

```
$ sudo aptitude install vim-scripts
$ echo "packadd! gnupg" >> ~/.vim/vimrc
```

The MD5 sum

`md5sum`⁽¹⁾ provides utility to make a digest file using the method in [rfc1321](#) and verifying each file with it.

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

“ [!NOTE] The computation for the [MD5](#) sum is less CPU intensive than the one for the cryptographic signature by [GNU Privacy Guard \(GnuPG\)](#). Usually, only the top level digest file is cryptographically signed to ensure data integrity.

Password keyring

On GNOME system, the GUI tool `seahorse` (1) manages passwords and stores them securely in the keyring `~/.local/share/keyrings/*`.

`secret-tool` (1) can store password to the keyring from the command line.

Let's store passphrase used for LUKS/dm-crypt encrypted disk image

```
$ secret-tool store --label='LUKS passphrase for disk.img' LUKS my_disk.img
Password: *****
```

This stored password can be retrieved and fed to other programs, e.g., `cryptsetup` (8).

```
$ secret-tool lookup LUKS my_disk.img | \
  cryptsetup open disk.img disk_img --type luks --keyring -
$ sudo mount /dev/mapper/disk_img /mnt
```

“ [!TIP] Whenever you need to provide password in a script, use `secret-tool` and avoid directly hardcoding the passphrase in it.

Source code merge tools

There are many merge tools for the source code. Following commands caught my eyes.

package	popcon	size	command	description
<code>patch</code>	@-@popcon1@-@	@-@psize1@-@	<code>patch</code> (1)	apply a diff file to an original
<code>vim</code>	@-@popcon1@-@	@-@psize1@-@	<code>vimdiff</code> (1)	compare 2 files side by side in vim
<code>imediff</code>	@-@popcon1@-@	@-@psize1@-@	<code>imediff</code> (1)	interactive full screen 2/3-way merge tool
<code>meld</code>	@-@popcon1@-@	@-@psize1@-@	<code>meld</code> (1)	compare and merge files (GTK)
<code>wiggle</code>	@-@popcon1@-@	@-@psize1@-@	<code>wiggle</code> (1)	apply rejected patches
<code>diffutils</code>	@-@popcon1@-@	@-@psize1@-@	<code>diff</code> (1)	compare files line by line
<code>diffutils</code>	@-@popcon1@-@	@-@psize1@-@	<code>diff3</code> (1)	compare and merges three files line by line

package	popcon	size	command	description
quilt	@-@popcon1@-@	@-@psize1@-@	quilt (1)	manage series of patches
wdiff	@-@popcon1@-@	@-@psize1@-@	wdiff (1)	display word differences between text files
diffstat	@-@popcon1@-@	@-@psize1@-@	diffstat (1)	produce a histogram of changes by the diff
patchutils	@-@popcon1@-@	@-@psize1@-@	combinediff (1)	create a cumulative patch from two incremental patches
patchutils	@-@popcon1@-@	@-@psize1@-@	dehtmldiff (1)	extract a diff from an HTML page
patchutils	@-@popcon1@-@	@-@psize1@-@	filterdiff (1)	extract or excludes diffs from a diff file
patchutils	@-@popcon1@-@	@-@psize1@-@	fixcvsdiff (1)	fix diff files created by CVS that patch (1) mis-interprets
patchutils	@-@popcon1@-@	@-@psize1@-@	flipdiff (1)	exchange the order of two patches
patchutils	@-@popcon1@-@	@-@psize1@-@	grepdiff (1)	show which files are modified by a patch matching a regex
patchutils	@-@popcon1@-@	@-@psize1@-@	interdiff (1)	show differences between two unified diff files
patchutils	@-@popcon1@-@	@-@psize1@-@	lsdiff (1)	show which files are modified by a patch
patchutils	@-@popcon1@-@	@-@psize1@-@	recountdiff (1)	recompute counts and offsets in unified context diffs
patchutils	@-@popcon1@-@	@-@psize1@-@	rediff (1)	fix offsets and counts of a hand-edited diff
patchutils	@-@popcon1@-@	@-@psize1@-@	splitdiff (1)	separate out incremental patches
patchutils	@-@popcon1@-@	@-@psize1@-@	unwrapdiff (1)	demangle patches that have been word-wrapped
dirdiff	@-@popcon1@-@	@-@psize1@-@	dirdiff (1)	display differences and merge changes between directory trees

package	popcon	size	command	description
<code>docdiff</code>	@-@popcon1@-@	@-@psize1@-@	<code>docdiff</code> (1)	compare two files word by word / char by char
<code>makepatch</code>	@-@popcon1@-@	@-@psize1@-@	<code>makepatch</code> (1)	generate extended patch files
<code>makepatch</code>	@-@popcon1@-@	@-@psize1@-@	<code>applypatch</code> (1)	apply extended patch files

List of source code merge tools

Extracting differences for source files

The following procedures extract differences between two source files and create unified diff files "`file.patch0`" or "`file.patch1`" depending on the file location.

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

Merging updates for source files

The diff file (alternatively called patch file) is used to send a program update. The receiving party applies this update to another file by the following.

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

Interactive merge

If you have two versions of a source code, you can perform 2-way merge interactively using `imediff` (1) by the following.

```
$ imediff -o file.merged file.old file.new
```

If you have three versions of a source code, you can perform 3-way merge interactively using `imediff` (1) by the following.

```
$ imediff -o file.merged file.yours file.base file.theirs
```

Git

Git is the tool of choice these days for the [version control system \(VCS\)](#) since Git can do everything for both local and remote source code management.

Debian provides free Git services via [Debian Salsa service](#). Its documentation can be found at <https://wiki.debian.org/Salsa> .

Here are some Git related packages.

package	popcon	size	command	description
<code>git</code>	@-@popcon1@-@	@-@psize1@-@	<code>git</code> (7)	Git, the fast, scalable, distributed revision control system
<code>gitk</code>	@-@popcon1@-@	@-@psize1@-@	<code>gitk</code> (1)	GUI Git repository browser with history
<code>qgit</code>	@-@popcon1@-@	@-@psize1@-@	<code>qgit</code> (1)	GUI Git repository browser with history
<code>git-cola</code>	@-@popcon1@-@	@-@psize1@-@	<code>git-cola</code> (1)	GUI Git repository browser with history
<code>tig</code>	@-@popcon1@-@	@-@psize1@-@	<code>tig</code> (1)	console Git repository browser with history
<code>lazygit</code>	@-@popcon1@-@	@-@psize1@-@	<code>lazygit</code> (1)	console Git repository browser with history
<code>git-gui</code>	@-@popcon1@-@	@-@psize1@-@	<code>git-gui</code> (1)	GUI for Git (No history)
<code>git-email</code>	@-@popcon1@-@	@-@psize1@-@	<code>git-send-email</code> (1)	send a collection of patches as email from the Git
<code>git-buildpackage</code>	@-@popcon1@-@	@-@psize1@-@	<code>git-buildpackage</code> (1)	automate the Debian packaging with the Git
<code>dgit</code>	@-@popcon1@-@	@-@psize1@-@	<code>dgit</code> (1)	git interoperability with the Debian archive
<code>imediff</code>	@-@popcon1@-@	@-@psize1@-@	<code>git-ime</code> (1)	interactive git commit split helper tool
<code>stgit</code>	@-@popcon1@-@	@-@psize1@-@	<code>stg</code> (1)	quilt on top of git (Python)

package	popcon	size	command	description
git-doc	@-@popcon1@-@	@-@psize1@-@	N/A	official documentation for Git
gitmagic	@-@popcon1@-@	@-@psize1@-@	N/A	"Git Magic", easier to understand guide for Git

List of git related packages and commands

Configuration of Git client

You may wish to set several global configuration in "`~/.gitconfig`" such as your name and email address used by Git by the following.

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

You may also customize the Git default behavior by the following.

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

If you are too used to CVS or Subversion commands, you may wish to set several command aliases by the following.

```
$ git config --global alias.ci "commit -a"
$ git config --global alias.co checkout
```

You can check your global configuration by the following.

```
$ git config --global --list
```

Basic Git commands

Git operation involves several data.

- The working tree which holds user facing files and to which you make changes.
 - The changes to be recorded must be explicitly selected and staged to the index. This is `git add` and `git rm` commands.

- The index which holds staged files.
 - Staged files will be committed to the local repository upon the subsequent request. This is `git commit` command.
- The local repository which holds committed files.
 - Git records the linked history of the committed data and organizes them as branches in the repository.
 - The local repository can send data to the remote repository by `git push` command.
 - The local repository can receive data from the remote repository by `git fetch` and `git pull` commands.
 - The `git pull` command performs `git merge` or `git rebase` command after `git fetch` command.
 - Here, `git merge` combines two separate branches of history at the end to a point. (This is default of `git pull` without customization and may be good for upstream people who publish branch to many people.)
 - Here, `git rebase` creates one single branch of sequential history of the remote branch one followed by the local branch one. (This is `pull.rebase true` customization case and may be good for rest of us.)
- The remote repository which holds committed files.
 - The communication to the remote repository uses secure communication protocols such as SSH or HTTPS.

The working tree is files outside of the `.git/` directory. Files inside of the `.git/` directory hold the index, the local repository data, and some git configuration text files.

Here is an overview of main Git commands.

Git command	function
<code>git init</code>	create the (local) repository
<code>git clone URL</code>	clone the remote repository to a local repository with the working tree
<code>git pull origin main</code>	update the local <code>main</code> branch by the remote repository <code>origin</code>
<code>git add .</code>	add file(s) in the working tree to the index for pre-existing files in index only
<code>git add -A .</code>	add file(s) in the working tree to the index for all files including removals
<code>git rm filename</code>	remove file(s) from the working tree and the index
<code>git commit</code>	commit staged changes in the index to the local repository
<code>git commit -a</code>	add all changes in the working tree to the index and commit them to the local repository (add + commit)
<code>git push -u origin branch_name</code>	update the remote repository <code>origin</code> by the local <code>branch_name</code> branch (initial invocation)

Git command	function
<code>git push origin branch_name</code>	update the remote repository <code>origin</code> by the local <code>branch_name</code> branch (subsequent invocation)
<code>git diff treeish1 treeish2</code>	show difference between <i>treeish1</i> commit and <i>treeish2</i> commit
<code>gitk</code>	GUI display of VCS repository branch history tree

Main Git commands

Git tips

Here are some Git tips.

Git command line	function
<code>gitk --all</code>	see complete Git history and operate on them such as resetting HEAD to another commit, cheery-picking patches, creating tags and branches ...
<code>git stash</code>	get the clean working tree without losing data
<code>git remote -v</code>	check settings for remote
<code>git branch -vv</code>	check settings for branch
<code>git status</code>	show working tree status
<code>git config -l</code>	list git settings
<code>git reset --hard HEAD; git clean -x -d -f</code>	revert all working tree changes and clean them up completely
<code>git rm --cached filename</code>	revert staged index changed by <code>git add filename</code>
<code>git reflog</code>	get reference log (useful for recovering commits from the removed branch)
<code>git branch new_branch_name HEAD@{6}</code>	create a new branch from reflog information
<code>git remote add new_remote URL</code>	add a <code>new_remote</code> remote repository pointed by URL
<code>git remote rename origin upstream</code>	rename the remote repository name from <code>origin</code> to <code>upstream</code>
<code>git branch -u upstream/branch_name</code>	set the remote tracking to the remote repository <code>upstream</code> and its branch name <code>branch_name</code> .
<code>git remote set-url origin https://foo/bar.git</code>	change URL of <code>origin</code>
<code>git remote set-url --push upstream DISABLED</code>	disable push to <code>upstream</code> (Edit <code>.git/config</code> to re-enable)
<code>git remote update upstream</code>	fetch updates of all remote branches in the <code>upstream</code> repository

Git command line	function
<code>git fetch upstream foo:upstream-foo</code>	create a local (possibly orphan) <code>upstream-foo</code> branch as a copy of <code>foo</code> branch in the <code>upstream</code> repository
<code>git checkout -b topic_branch ; git push -u topic_branch origin</code>	make a new <code>topic_branch</code> and push it to <code>origin</code>
<code>git branch -m oldname newname</code>	rename local branch name
<code>git push -d origin branch_to_be_removed</code>	remove remote branch (new method)
<code>git push origin :branch_to_be_removed</code>	remove remote branch (old method)
<code>git checkout --orphan unconnected</code>	create a new <code>unconnected</code> branch
<code>git rebase -i origin/main</code>	reorder/drop/squish commits from <code>origin/main</code> to clean branch history
<code>git reset HEAD^; git commit --amend</code>	squash last 2 commits into one
<code>git checkout topic_branch ; git merge --squash topic_branch</code>	squash entire <code>topic_branch</code> into a commit
<code>git fetch --unshallow --update-head-ok origin '+refs/heads/*:refs/heads/*'</code>	convert a shallow clone to the full clone of all branches
<code>git ime</code>	split the last commit into a series of file-by-file smaller commits etc. (<code>imediff</code> package required)
<code>git repack -a -d; git prune</code>	repack the local repository into single pack (this may limit chance of lost data recovery from erased branch etc.)

Git tips

“ [!WARNING] Do not use the tag string with spaces in it even if some tools such as `gitk` (1) allow you to use it. It may choke some other `git` commands.

“ [!CAUTION] If a local branch which has been pushed to remote repository is rebased or squashed, pushing this branch has risks and requires `--force` option. This is usually not an acceptable for `main` branch but may be acceptable for a topic branch before merging to `main` branch.

“ [!CAUTION] Invoking a `git` subcommand directly as "`git-xyz`" from the command line has been deprecated since early 2006.

[!TIP] If there is a executable file `git-foo` in the path specified by `$PATH`, entering "`git foo`" without hyphen to the command line invokes this `git-foo`. This is a feature of the `git` command.

Git references

See the following.

- [manpage: git\(1\)](#) (`/usr/share/doc/git-doc/git.html`)
- [Git User's Manual](#) (`/usr/share/doc/git-doc/user-manual.html`)
- [A tutorial introduction to git](#) (`/usr/share/doc/git-doc/gittutorial.html`)
- [A tutorial introduction to git: part two](#) (`/usr/share/doc/git-doc/gittutorial-2.html`)
- [Everyday GIT With 20 Commands Or So](#) (`/usr/share/doc/git-doc/giteveryday.html`)
- [Git Magic](#) (`/usr/share/doc/gitmagic/html/index.html`)

Other version control systems

The [version control systems \(VCS\)](#) is sometimes known as the revision control system (RCS), or the software configuration management (SCM).

Here is a summary of the notable other non-Git VCS on the Debian system.

package	popcon	size	tool	VCS type	comment
<code>mercurial</code>	@-@popcon1@-@	@-@psize1@-@	Mercurial	distributed	DVCS in Python and some C
<code>darcs</code>	@-@popcon1@-@	@-@psize1@-@	Darcs	distributed	DVCS with smart algebra of patches (slow)
<code>tla</code>	@-@popcon1@-@	@-@psize1@-@	GNU arch	distributed	DVCS mainly by Tom Lord (historic)
<code>bzr</code>	@-@popcon1@-@	@-@psize1@-@	GNU Bazaar	distributed	DVCS influenced by <code>tla</code> written in Python (historic)
<code>subversion</code>	@-@popcon1@-@	@-@psize1@-@	Subversion	remote	"CVS done right", newer standard remote VCS (historic)

package	popcon	size	tool	VCS type	comment
cvcs	@-@popcon1@-@	@-@psize1@-@	CVS	remote	previous standard remote VCS (historic)
tkcvs	@-@popcon1@-@	@-@psize1@-@	CVS, ...	remote	GUI display of VCS (CVS, Subversion, RCS) repository tree
rsc	@-@popcon1@-@	@-@psize1@-@	RCS	local	" Unix SCCS done right" (historic)
cssc	@-@popcon1@-@	@-@psize1@-@	CSSC	local	clone of the Unix SCCS (historic)

List of other version control system tools {#list-of-vcs}

[^1]: A write-once media such as CD/DVD-R can prevent overwrite accidents. (See [???](#) for how to write to the storage media from the shell commandline. GNOME desktop GUI environment gives you easy access via menu: "Places→CD/DVD Creator".)

[^2]: Some of these data can not be regenerated by entering the same input string to the system.

[^3]: If you use "`~/.vimrc`" instead of "`~/.vim/vimrc`", please substitute accordingly.

```

## #2
#  ##  ##  2026-10-31 13:08:08 UTC
#  ##  ##  2026-10-31 13:08:12 UTC

```