# **User management**

User management is a critical part of maintaining a secure system. Ineffective user and privilege management often leads to a system being compromised. Therefore, it is important that you understand how to protect your server through simple and effective user account management techniques.

## **Where is root?**

Ubuntu developers decided to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted, or that it may not be accessed. Instead, it has been given a password hash that matches no possible value, and so may not log in directly by itself.

Instead, the sudo utility (“superuser do”) is used to carry out system administrative duties. sudo allows an authorised user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account. This provides accountability for all user actions, and gives the administrator control over which actions a user can perform with said privileges.

### **Enabling the root account**

If for some reason you wish to enable the root account, you will need to give it a password:

sudo passwd

sudo will prompt you for your password, and then ask you to supply a new password for root as shown below:

[sudo] password for username: (enter your own password)

Enter new UNIX password: (enter a new password for root)

Retype new UNIX password: (repeat new password for root)

passwd: password updated successfully

### **Disabling the root account password**

To disable the root account password, use the following passwd syntax:

sudo passwd -l root

You can learn more about sudo by reading the [sudo man page](https://manpages.ubuntu.com/manpages/man8/sudo.8.html): man sudo

By default, the initial user created by the Ubuntu installer is a member of the group sudo which is added to the file /etc/sudoers as an authorised sudo user. To give any other account full root access through sudo, add them to the sudo group.

## **Adding and deleting users**

Managing local users and groups differs very little from most other [GNU](https://documentation.ubuntu.com/server/reference/glossary/#term-GNU)/Linux operating systems. Ubuntu and other Debian-based distributions encourage the use of the adduser package for account management.

### **Add a user**

To add a user account, use the following syntax, and follow the prompts to give the account a password and identifiable characteristics, such as a full name, phone number, etc:

sudo adduser username

### **Delete a user**

To delete a user account and its primary group, use the following syntax:

sudo deluser username

Deleting an account does not remove their respective home folder. You must decide whether or not to delete the folder manually, or to keep it according to your desired retention policies.

Remember, any user added later with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.

You may want to change the UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts:

sudo chown -R root:root /home/username/

sudo mkdir /home/archived\_users/

sudo mv /home/username /home/archived\_users/

### **Lock or unlock a password**

To temporarily lock a user password, use the following syntax:

sudo passwd -l username

Similarly, to unlock a user password:

sudo passwd -u username

### **Add or delete a group**

To add or delete a personalised group, use the following syntax, respectively:

sudo addgroup groupname

sudo delgroup groupname

### **Add a user to a group**

To add a user to a group, use the following syntax:

sudo adduser username groupname

## **User profile security**

When a new user is created, the adduser utility creates a brand new home directory named /home/username. The default profile is modelled after the contents found in the directory of /etc/skel, which includes all profile basics.

If your server will be home to multiple users, you should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read/execute permissions. This means that all users can browse and access the contents of other users home directories, which may not be suitable for your environment.

To verify your current user home directory permissions, use the following syntax:

ls -ld /home/username

The following output shows that the directory /home/username has world-readable permissions:

drwxr-xr-x 2 username username 4096 2007-10-02 20:03 username

You can remove the world readable-permissions using the following command:

sudo chmod 0750 /home/username

Note

Some people use the recursive option (-R) indiscriminately, which modifies all child folders and files. However, this is not necessary and may have undesirable/unintended consequences. Modifying only the parent directory is enough to prevent unauthorised access to anything below the parent.

A more efficient approach would be to modify the adduser global default permissions when creating user home folders. To do this, edit the /etc/adduser.conf file and modify the DIR\_MODE variable to something appropriate, so that all new home directories will receive the correct permissions.

DIR\_MODE=0750

After correcting the directory permissions using any of the previously mentioned techniques, verify the results as follows:

ls -ld /home/username

The output below shows that world-readable permissions have been removed:

drwxr-x--- 2 username username 4096 2007-10-02 20:03 username

## **Password policy**

A strong password policy is one of the most important aspects of your security posture. Many successful security breaches involve simple **brute force** and **dictionary** attacks against weak passwords.

If you intend to offer any form of remote access involving your local password system, make sure you address minimum password complexity requirements, maximum password lifetimes, and frequent audits of your authentication systems.

### **Minimum password length**

By default, Ubuntu requires a minimum password length of 6 characters, as well as some basic entropy checks. These values are controlled in the file /etc/pam.d/common-password, which is outlined below.

password [success=1 default=ignore] pam\_unix.so obscure sha512

To adjust the minimum length to 8 characters, change the appropriate variable to minlen=8. The modification is outlined below:

password [success=1 default=ignore] pam\_unix.so obscure sha512 minlen=8

Note

Basic password entropy checks and minimum length rules do not apply to the administrator using sudo-level commands to setup a new user.

### **Password expiration**

When creating user accounts, you should make it a policy to have a minimum and maximum password age, forcing users to change their passwords when they expire.

To view the current status of a user account:

sudo chage -l username

The output below shows interesting facts about the user account, namely that there are no policies applied:

Last password change : Jan 20, 2015

Password expires : never

Password inactive : never

Account expires : never

Minimum number of days between password change : 0

Maximum number of days between password change : 99999

Number of days of warning before password expires : 7

To set any of these values, use the chage (“change age”) command, and follow the interactive prompts:

sudo chage username

The following is also an example of how you can manually change the explicit expiration date (-E) to 01/31/2015, minimum password age (-m) of 5 days, maximum password age (-M) of 90 days, inactivity period (-I) of 30 days after password expiration, and a warning time period (-W) of 14 days before password expiration:

sudo chage -E 01/31/2015 -m 5 -M 90 -I 30 -W 14 username

To verify changes, use the same syntax as mentioned previously:

sudo chage -l username

The output below shows the new policies that have been established for the account:

Last password change : Jan 20, 2015

Password expires : Apr 19, 2015

Password inactive : May 19, 2015

Account expires : Jan 31, 2015

Minimum number of days between password change : 5

Maximum number of days between password change : 90

Number of days of warning before password expires : 14

## **Other security considerations**

Many applications use alternate authentication mechanisms that can be easily overlooked by even experienced system administrators. Therefore, it is important to understand and control how users authenticate and gain access to services and applications on your server.

### **SSH access by disabled passwords**

Disabling or locking a user password will not prevent a user from logging into your server remotely if they have previously set up SSH public key authentication. They will still be able to gain shell access to the server, without the need for any password. Remember to check the user’s home directory for files that will allow for this type of authenticated SSH access, e.g. /home/username/.ssh/authorized\_keys.

Remove or rename the directory .ssh/ in the user’s home folder to prevent further SSH authentication access.

Be sure to check for any established SSH connections by the disabled account, as it is possible they may have existing inbound or outbound connections – then pkill any that are found.

who | grep username (to get the pts/# terminal)

sudo pkill -f pts/#

Restrict SSH access to only user accounts that should have it. For example, you may create a group called sshlogin and add the group name as the value associated with the AllowGroups variable located in the file /etc/ssh/sshd\_config:

AllowGroups sshlogin

Then add your permitted SSH users to the group sshlogin, and restart the SSH service.

sudo adduser username sshlogin

sudo systemctl restart ssh.service

### **External user database authentication**

Most enterprise networks require centralised authentication and access controls for all system resources. If you have configured your server to authenticate users against external databases, be sure to disable the user accounts both externally and locally. This way you ensure that local [fallbacks](https://documentation.ubuntu.com/server/reference/glossary/#term-fallbacks) authentication is not possible.

## **Understanding and Using File Permissions**

In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Devices are usually referred to as a node; however, they are still files. All of the files on a system have permissions that allow or prevent others from viewing, modifying or executing. If the file is of type Directory then it restricts different actions than files and device nodes. The super user "root" has the ability to access any file on the system. Each file has access restrictions with permissions, user restrictions with owner/group association. Permissions are referred to as bits.

To change or edit files that are owned by root, **sudo** must be used - please see [RootSudo](https://help.ubuntu.com/community/RootSudo) for details.

If the owner read & execute bit are on, then the permissions are:

-r-x------

There are three types of access restrictions:

| **Permission** | **Action** | **chmod option** |
| --- | --- | --- |
| read | (view) | r or 4 |
| write | (edit) | w or 2 |
| execute | (execute) | x or 1 |

There are also three types of user restrictions:

| **User** | ***ls* output** |
| --- | --- |
| owner | -rwx------ |
| group | ----rwx--- |
| other | -------rwx |

**Note:** The restriction type scope is not inheritable: the file owner will be unaffected by restrictions set for his group or everybody else.

## **Folder/Directory Permissions**

Directories have directory permissions. The directory permissions restrict different actions than with files or device nodes.

| **Permission** | **Action** | **chmod option** |
| --- | --- | --- |
| read | (view contents, i.e. ls command) | r or 4 |
| write | (create or remove files from dir) | w or 2 |
| execute | (cd into directory) | x or 1 |

* read restricts or allows viewing the directories contents, i.e. *ls* command
* write restricts or allows creating new files or deleting files in the directory. (Caution: **write access for a directory allows deleting of files in the directory even if the user does not have write permissions for the file!**)
* execute restricts or allows changing into the directory, i.e. *cd* command

Info <!> **Folders (directories) must have 'execute' permissions set (x or 1), or folders (directories) will NOT FUNCTION as folders (directories) and WILL DISAPPEAR from view in the file browser (Nautilus).**

## **Permissions in Action**

user@host:/home/user$ ls -l /etc/hosts

-rw-r--r-- 1 root root 288 2005-11-13 19:24 /etc/hosts

user@host:/home/user$

Using the example above we have the file "/etc/hosts" which is owned by the user root and belongs to the root group.

What are the permissions from the above /etc/hosts ls output?

-rw-r--r--

owner = Read & Write (rw-)

group = Read (r--)

other = Read (r--)

## **Changing Permissions**

The command to use when modifying permissions is chmod. There are two ways to modify permissions, with numbers or with letters. Using letters is easier to understand for most people. When modifying permissions be careful not to create security problems. Some files are configured to have very restrictive permissions to prevent unauthorized access. For example, the /etc/shadow file (file that stores all local user passwords) does not have permissions for regular users to read or otherwise access.

user@host:/home/user# ls -l /etc/shadow

-rw-r----- 1 root shadow 869 2005-11-08 13:16 /etc/shadow

user@host:/home/user#

Permissions:

owner = Read & Write (rw-)

group = Read (r--)

other = None (---)

Ownership:

owner = root

group = shadow

### **chmod with Letters**

Usage: chmod {options} filename

| **Options** | **Definition** |
| --- | --- |
| u | owner |
| g | group |
| o | other |
| a | all (same as ugo) |
| x | execute |
| w | write |
| r | read |
| + | add permission |
| - | remove permission |
| = | set permission |

Here are a few examples of chmod usage with letters (try these out on your system).

First create some empty files:

user@host:/home/user$ touch file1 file2 file3 file4

user@host:/home/user$ ls -l

total 0

-rw-r--r-- 1 user user 0 Nov 19 20:13 file1

-rw-r--r-- 1 user user 0 Nov 19 20:13 file2

-rw-r--r-- 1 user user 0 Nov 19 20:13 file3

-rw-r--r-- 1 user user 0 Nov 19 20:13 file4

Add owner execute bit:

user@host:/home/user$ chmod u+x file1

user@host:/home/user$ ls -l file1

-rwxr--r-- 1 user user 0 Nov 19 20:13 file1

Add other write & execute bit:

user@host:/home/user$ chmod o+wx file2

user@host:/home/user$ ls -l file2

-rw-r--rwx 1 user user 0 Nov 19 20:13 file2

Remove group read bit:

user@host:/home/user$ chmod g-r file3

user@host:/home/user$ ls -l file3

-rw----r-- 1 user user 0 Nov 19 20:13 file3

Add read, write and execute to everyone:

user@host:/home/user$ chmod ugo+rwx file4

user@host:/home/user$ ls -l file4

-rwxrwxrwx 1 user user 0 Nov 19 20:13 file4

user@host:/home/user$

### **chmod with Numbers**

Usage: chmod {options} filename

| **Options** | **Definition** |
| --- | --- |
| #-- | owner |
| -#- | group |
| --# | other |
| 1 | execute |
| 2 | write |
| 4 | read |

Owner, Group and Other is represented by three numbers. To get the value for the options determine the type of access needed for the file then add.

For example if you want a file that has -rw-rw-rwx permissions you will use the following:

| **Owner** | **Group** | **Other** |
| --- | --- | --- |
| read & write | read & write | read, write & execute |
| 4+2=6 | 4+2=6 | 4+2+1=7 |

user@host:/home/user$ chmod 667 filename

Another example if you want a file that has --w-r-x--x permissions you will use the following:

| **Owner** | **Group** | **Other** |
| --- | --- | --- |
| write | read & execute | execute |
| 2 | 4+1=5 | 1 |

user@host:/home/user$ chmod 251 filename

Here are a few examples of chmod usage with numbers (try these out on your system).

First create some empty files:

user@host:/home/user$ touch file1 file2 file3 file4

user@host:/home/user$ ls -l

total 0

-rw-r--r-- 1 user user 0 Nov 19 20:13 file1

-rw-r--r-- 1 user user 0 Nov 19 20:13 file2

-rw-r--r-- 1 user user 0 Nov 19 20:13 file3

-rw-r--r-- 1 user user 0 Nov 19 20:13 file4

Add owner execute bit:

user@host:/home/user$ chmod 744 file1

user@host:/home/user$ ls -l file1

-rwxr--r-- 1 user user 0 Nov 19 20:13 file1

Add other write & execute bit:

user@host:/home/user$ chmod 647 file2

user@host:/home/user$ ls -l file2

-rw-r--rwx 1 user user 0 Nov 19 20:13 file2

Remove group read bit:

user@host:/home/user$ chmod 604 file3

user@host:/home/user$ ls -l file3

-rw----r-- 1 user user 0 Nov 19 20:13 file3

Add read, write and execute to everyone:

user@host:/home/user$ chmod 777 file4

user@host:/home/user$ ls -l file4

-rwxrwxrwx 1 user user 0 Nov 19 20:13 file4

user@host:/home/user$

### **chmod with sudo**

Changing permissions on files that you do not have ownership of: (**Note** that changing permissions the wrong way on the wrong files can quickly mess up your system a great deal! Please be careful when using **sudo**!)

user@host:/home/user$ ls -l /usr/local/bin/somefile

-rw-r--r-- 1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile

user@host:/home/user$

user@host:/home/user$ sudo chmod o+x /usr/local/bin/somefile

user@host:/home/user$ ls -l /usr/local/bin/somefile

-rw-r--r-x 1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile

user@host:/home/user$

## **Recursive Permission Changes**

To change the permissions of multiple files and directories with one command. Please note the warning in the chmod with sudo section and the Warning with Recursive chmod section.

### **Recursive chmod with -R and sudo**

To change all the permissions of each file and folder under a specified directory at once, use sudo chmod with -R

user@host:/home/user$ sudo chmod 777 -R /path/to/someDirectory

user@host:/home/user$ ls -l

total 3

-rwxrwxrwx 1 user user 0 Nov 19 20:13 file1

drwxrwxrwx 2 user user 4096 Nov 19 20:13 folder

-rwxrwxrwx 1 user user 0 Nov 19 20:13 file2

### **Recursive chmod using find, pipemill, and sudo**

To assign reasonably secure permissions to files and folders/directories, it's common to give files a permission of 644, and directories a 755 permission, since chmod -R assigns to both. Use sudo, the find command, and a pipemill to chmod as in the following examples.

To change permission of only files under a specified directory.

user@host:/home/user$ sudo find /path/to/someDirectory -type f -print0 | xargs -0 sudo chmod 644

user@host:/home/user$ ls -l

total 3

-rw-r--r-- 1 user user 0 Nov 19 20:13 file1

drwxrwxrwx 2 user user 4096 Nov 19 20:13 folder

-rw-r--r-- 1 user user 0 Nov 19 20:13 file2

To change permission of only directories under a specified directory (including that directory):

user@host:/home/user$ sudo find /path/to/someDirectory -type d -print0 | xargs -0 sudo chmod 755

user@host:/home/user$ ls -l

total 3

-rw-r--r-- 1 user user 0 Nov 19 20:13 file1

drwxr-xr-x 2 user user 4096 Nov 19 20:13 folder

-rw-r--r-- 1 user user 0 Nov 19 20:13 file2

## **Warning with Recursive chmod**

WARNING: Although it's been said, it's worth mentioning in context of a gotcha typo. Please note, Recursively deleting or chown-ing files are extremely dangerous. You will not be the first, nor the last, person to add one too many spaces into the command. This example will hose your system:

user@host:/home/user$ sudo chmod -R / home/john/Desktop/tempfiles

Note the space between the first / and home.

You have been warned.

## **Changing the File Owner and Group**

A file's owner can be changed using the chown command. For example, to change the foobar file's owner to tux:

user@host:/home/user$ sudo chown tux foobar

To change the foobar file's group to penguins, you could use **either** chgrp or chown with special syntax:

user@host:/home/user$ sudo chgrp penguins foobar

user@host:/home/user$ sudo chown :penguins foobar

Finally, to change the foobar file's owner to tux and the group to penguins with a single command, the syntax would be:

user@host:/home/user$ sudo chown tux:penguins foobar

Info <!> Note that, by default, you must use sudo to change a file's owner or group.

## **Volume Permissions with umask**

This section has been moved to: [Fstab#Options](https://help.ubuntu.com/community/Fstab#Options)

## **ACL (Access Control List)**

Posix ACLs are a way of achieving a finer granularity of permissions than is possible with the standard Unix file permissions. See the full page on ACLs [FilePermissionsACLs](https://help.ubuntu.com/community/FilePermissionsACLs)

### **Setting up ACL**

1. Install the acl package:

sudo apt-get install acl

1. Edit /etc/fstab and add option acl to partition(s) on which you want to enable ACL. For example:

...

UUID=d027a8eb-e234-1c9f-aef1-43a7dd9a2345 /home ext4 defaults,acl 0 2

...

1. Remount partition(s) on which you want to enable ACL. For example:

sudo mount -o remount /home

1. Verify acl is enabled on the partition(s):

mount | grep acl

The commands, setfacl and getfacl, set and read ACLs on files and directories.

### **Example Usage**

This is a simple example for use with a Samba share to ensure that any files or sub-directories created could also be modified by any Samba user.

1. Create a directory with full permission:

mkdir shared\_dir

chmod 777 shared\_dir

1. Set the default ACL with '-d' and modify with '-m' the permissions for samba nobody user nogroup group which will apply to all newly created file/directories.

setfacl -d -m u:nobody:rwx,g:nogroup:rwx,o::r-x shared\_dir

### **GUI ACL Editor**

The [Eicielhttp://apt.ubuntu.com/p/eiciel](http://rofi.roger-ferrer.org/eiciel/) package allows GUI access to ACLs through the Nautilus file manager.

### **Useful ACL Resources**

* <http://brunogirin.blogspot.com/2010/03/shared-folders-in-ubuntu-with-setgid.html>
* <http://wiki.kaspersandberg.com/doku.php?id=howtos:acl>
* [man acl](http://manpages.ubuntu.com/cgi-bin/search.py?q=acl)
* [man setfacl](http://manpages.ubuntu.com/cgi-bin/search.py?q=setfacl)
* [man getfacl](http://manpages.ubuntu.com/cgi-bin/search.py?q=getfacl)

## **File removal**

To remove a file you cannot delete use

sudo rm -rf filename

where filename is the name and path of the file to delete.

**Nota bene:** Be very careful when using the command rm with the -rf option since -r makes the file removal recursive (meaning it will remove files inside of folders) and -f will force the removal even for files which aren't writable. To play it safe, please consider typing in the absolute path to the file

sudo rm -rf /path/to/file/filename

to prevent any mishaps that can/will occur. It takes longer to type but you can't put a price on peace of mind. See the *rm* man page for details.

## **Sticky Bit**

The sticky bit applies only to directories, and is typically used on publicly-writeable directories. Within a directory upon which the sticky bit is applied, users are prevented from deleting or renaming any files that they do not personally own.

To add or remove the sticky bit, use chmod with the "t" flag:

chmod +t <directory>

chmod -t <directory>

The status of the sticky bit is shown in the other execute field, when viewing the long output of ls. "t" or "T" in the other execute field indicates the sticky bit is set, anything else indicates it is not.

Making a public directory:

user@host:/home/user$ mkdir folder

user@host:/home/user$ chmod 777 folder

user@host:/home/user$ ls -l

total 3

drwxrwxrwx 2 user user 4096 Nov 19 20:13 folder

Adding the sticky bit (note the "t" in the other execute field):

user@host:/home/user$ chmod +t folder

user@host:/home/user$ ls -l

total 3

drwxrwxrwt 2 user user 4096 Nov 19 20:13 folder

## **See also**

* [man chmod](http://manpages.ubuntu.com/cgi-bin/search.py?q=chmod)
* [man chown](http://manpages.ubuntu.com/cgi-bin/search.py?q=chown)
* [man chgrp](http://manpages.ubuntu.com/cgi-bin/search.py?q=chgrp)
* [FindingFiles](https://help.ubuntu.com/community/FindingFiles)
* [User Private Groups](http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/ref-guide/s1-users-groups-private-groups.html)

## **ToDo**

* umask (add file and directory umask section, with specific focus on security)
* The User Private Group scheme. In other words, this page does the nuts and bolts ok, but we need to describe *what* the permissions should be. The default Ubuntu set up is *not* agnostic: Every user has their default private group. Directories for collaboration need to have special group and permission set for correct functioning.
* \* *Suggestion: I often use find instead of chmod -R, because it's easier to differentiate between files and directories that way. Yes, I know about the 'X' permission, but I don't trust it.*
* The sticky bit. It's needed for "other" in shared directories like /tmp. It's needed for "group" in shared directories where write permission is given to a group, like /var/www

# **Root Sudo**

In Linux (and Unix in general), there is a SuperUser named **root**. The Windows equivalent of root is the Administrators group. The SuperUser can do anything and everything, and thus doing daily work as the SuperUser can be dangerous. You could type a command incorrectly and destroy the system. Ideally, you run as a user that has only the privileges needed for the task at hand. In some cases, this is necessarily root, but most of the time it is a regular user.

**By default, the root account password is locked in Ubuntu.** This means that you cannot login as root directly or use the su command to become the root user. However, since the root account physically exists it is still possible to run programs with root-level privileges. This is where sudo comes in - it allows authorized users (normally "Administrative" users; for further information please refer to [AddUsersHowto](https://help.ubuntu.com/community/AddUsersHowto)) to run certain programs as root without having to know the root password.

This means that in the [terminal](https://help.ubuntu.com/community/UsingTheTerminal) you should use sudo for commands that require root privileges; simply prepend sudo to all the commands you need to run as root. For more extensive usage examples, please see below. Similarly, when you run GUI programs that require root privileges (e.g. the network configuration applet), use graphical sudo and you will also be prompted for a password (more below). Just remember, when sudo asks for a password, it needs **YOUR USER password**, and not the root account password.

**Please keep in mind, a substantial number of Ubuntu users are new to Linux.** There is a learning curve associated with any OS and many new users try to take shortcuts by enabling the root account, logging in as root, and changing ownership of system files.

Example: [Broken system via (ab)use of root by a new user](http://ubuntuforums.org/showthread.php?t=1877557)

Please note: At the time of the post, this was the users' first post on the Ubuntu forums. While some might call this a "learning experience", learning by breaking your system is frustrating and can result in data loss.

When giving advice on the Ubuntu Forums and IRC, please take the time to teach "the basics" such as ownership, permissions, and how to use sudo / gksu / kdesudo in such a way that new users do not break systems.

# **Advantages and Disadvantages**

## **Benefits of using sudo**

There are a number of benefits to Ubuntu leaving **root** logins disabled by default, including:

* The installer has fewer questions to ask.
* Users don't have to remember an extra password for occasional use (i.e. the root password). If they did, they'd be likely to forget it (or record it unsafely, allowing anyone to easily crack into their system).
* It avoids the "I can do *anything*" interactive login by default. You will be prompted for a password before major changes can happen, which should make you think about the consequences of what you are doing.
* sudo adds a log entry of the command(s) run (in /var/log/auth.log). If you mess up, you can go back and see what commands were run.
* On a server, every cracker trying to *brute-force* their way in will know it has an account named **root** and will try that first. What they don't know is what the usernames of your other users are. Since the root account password is locked, this attack becomes essentially meaningless, since there is no password to crack or guess in the first place.
* Allows easy transfer for admin rights by adding and removing users from groups. When you use a single root password, the only way to de-authorize users is to change the root password.
* sudo can be setup with a much more fine-grained security policy.
* The root account password does not need to be shared with everybody who needs to perform some type of administrative task(s) on the system (see the previous bullet).
* The authentication automatically expires after a short time (which can be set to as little as desired or 0); so if you walk away from the terminal after running commands as root using sudo, you will not be leaving a root terminal open indefinitely.

## **Downsides of using sudo**

Although for desktops the benefits of using sudo are great, there are possible issues which need to be noted:

* Redirecting the output of commands run with sudo requires a different approach. For instance consider sudo ls > /root/somefile will not work since it is the shell that tries to write to that file. You can use ls | sudo tee -a /root/somefile to append, or ls | sudo tee /root/somefile to overwrite contents. You could also pass the whole command to a shell process run under sudo to have the file written to with root permissions, such as sudo sh -c "ls > /root/somefile".
* In a lot of office environments the ONLY local user on a system is root. All other users are imported using NSS techniques such as nss-ldap. To setup a workstation, or fix it, in the case of a network failure where nss-ldap is broken, root is required. This tends to leave the system unusable unless cracked. An extra local user, or an enabled root password is needed here. The local user account should have its $HOME on a local disk, \_not\_ on NFS (or any other networked filesystem), and a .profile/.bashrc that doesn't reference any files on NFS mounts. This is usually the case for root, but if adding a non-root rescue account, you will have to take these precautions manually. However the advantage of using a local user with sudo is that commands can be easily tracked, as mentioned in the benefits above.

# **Usage**

* When using sudo, your password is stored by default for 15 minutes. After that time, you will need to enter your password again.
* Your password will **not** be shown on the screen as you type it, not even as a row of stars (\*\*\*\*\*\*). It is being entered with each keystroke!

## **sudo**

To use sudo on the command line, preface the command with sudo, as below: *Example #1*

sudo chown bob:bob /home/bob/\*

*Example #2*

sudo /etc/init.d/networking restart

To repeat the last command entered, except with sudo prepended to it, run:

sudo !!

## **Graphical sudo**

You should **never** use normal sudo to start graphical applications as root. Using sudo with graphical apps has the potential to corrupt your environment by allowing root to take ownership of and/or change permissions on critical files that you must own. The forums frequently see panicked requests for help from users who can no longer log in after running graphical applications under sudo.

Please note that many websites and old threads advise the use of gksu. However, such search results are obsolete. gksudo has not been updated for years and is not even available in Bionic (18.04) and higher. gksu has been replaced by pkexec, but even pkexec is being deprecated by the mainline Ubuntu developers. They have taken the position that file manipulation and editing under root should be restricted to the command line.

We can only surmise what the motives were behind this decision: perhaps there are just too many users who run into problems running graphical apps as root. In any case, running graphical apps as root now requires workarounds and additional steps.

### **Flavour-specific workarounds**

There are a number of flavour-specific options for running graphical applications as root:

* You can use pkexec on those flavours that support this option. As of 18.04, only Xubuntu supports this option by default, as shown in the following examples:

pkexec thunar

* + pkexec mousepad /etc/fstab
* By default, Kubuntu allows easy access to a root file manager: **KDE Launcher → Computer → Root-Dolphin**
  + From there: **(→ edit file)** will open up a root instance of Kate.
* Mainline Ubuntu and Gnome use Nautilus as their file manager. Any flavour running Nautilus will allow you to install the package nautilus-admin which will add two python extensions to Nautilus. These extensions add options that allow root access: **Open as Administrator** and **Edit as Administrator**
  + It is also possible to install the missing Policykit files for both Nautilus and Gedit. See [this site](http://www.webupd8.org/2015/03/how-to-run-gedit-and-nautilus-as-root.html) for instructions and links. A knowledgeable user could build further Policykit files for alternate file managers and editors by using the linked files as templates. They are simple XML files that can be edited with a standard text editor.

### **General workarounds**

The following methods will work on all flavours:

* Use the command line. Simple text editors like nano are quite easy to learn. If you prefer a quasi-graphical file manager, install Midnight Commander. Both of these apps run under sudo with no problems. Examples:

sudo mc

* + sudo nano /etc/fstab
* Notwithstanding the earlier warning, it is possible to use sudo with graphical apps provided you add the -H flag. This flag is critical: it properly sets root to its own environment instead of improperly inheriting the user's environment. Use of the -H flag is mandatory. Failing to use this flag may corrupt critical system files and prevent you from logging in.
  + With sudo -H almost any graphical app can be launched under root within any 'buntu flavour. This includes each flavour's default graphical editor and file manager.  
     An appreciable danger with sudo -H is that the -H flag is easy to forget. And all it takes is one omission for the damage to be done.

# **Users**

## **Allowing other users to run sudo**

To add a new user to sudo, open the **Settings** window, then **Details → Users** menu. First click **Unlock**, then you can select a user and hit **Administrator**.

Warning /!\ In the terminal (for Precise Pangolin, 12.04), this would be:

sudo adduser <username> sudo

where you replace <username> with the name of the user (without the <>).

In previous version of Ubuntu

sudo adduser <username> admin

would have been appropriate, but the admin group has been deprecated and no longer exists in Ubuntu 12.04.

## **Logging in as another user**

**Please don't use this to become root,** see further down in the page for more information about that.

sudo -i -u <username>

For example to become the user amanda for tape management purposes.

sudo -i -u amanda

The password being asked for is your own, not amanda's.

## **root account**

### **Enabling the root account**

| **IconsPage/warning.png** | **Enabling the root account is rarely necessary. Almost everything you need to do as administrator of an Ubuntu system can be done via sudo or gksudo. If you really need a persistent root login, the best alternative is to simulate a root login shell using the following command...** | IconsPage/warning.png |
| --- | --- | --- |

sudo -i

To enable the root account (i.e. set a password) use:

sudo passwd root

**Use at your own risk!**

| **IconsPage/dont.png** | **Logging in to X as root may cause very serious trouble.** If you believe you need a root account to perform a certain action, **please consult the official support channels first**, to make sure there is not a better alternative. | IconsPage/dont.png |
| --- | --- | --- |

### **Re-disabling your root account**

| **IconsPage/info.png** | If for some reason you have enabled your root account and wish to disable it again, use the following command in terminal... | IconsPage/info.png |
| --- | --- | --- |

sudo passwd -dl root

# **Other Information**

## **Misconceptions**

* *Isn't sudo less secure than su?*
  + The basic security model is the same, and therefore these two systems share their primary weaknesses. Any user who uses su **or** sudo must be considered to be a privileged user. If that user's account is compromised by an attacker, the attacker can also gain root privileges the next time the user does so. The user account is the weak link in this chain, and so must be protected with the same care as root.  
     On a more esoteric level, sudo provides some features which encourage different work habits, which can positively impact the security of the system. sudo is commonly used to execute only a single command, while su is generally used to open a shell and execute multiple commands. The sudo approach reduces the likelihood of a root shell being left open indefinitely, and encourages the user to minimize their use of root privileges.
* *I won't be able to enter single-user mode!*
  + The sulogin program in Ubuntu is patched to handle the default case of a locked root password.
* *I can get a root shell from the console without entering a password!*
  + You have to enter your password.  
     Console users have access to the boot loader, and can gain administrative privileges in various ways during the boot process. For example, by specifying an alternate init(8) program. Linux systems are not typically configured to be secure at the console, and additional steps (for example, setting a root password, a boot loader password and a BIOS password) are necessary in order to make them so. Note that console users usually have physical access to the machine and so can manipulate it in other ways as well.

## **Special notes on sudo and shells**

**None of the methods below are suggested or supported by the designers of Ubuntu.**

Please do not suggest this to others unless you personally are available 24/7 to support the user if they have issues as a result of running a shell as root.

To start a *root shell* (i.e. a command window where you can run root commands), starting root's environment and login scripts, use:

sudo -i (similar to sudo su - , gives you roots environment configuration)

To start a *root shell*, but keep the current shell's environment, use:

sudo -s (similar to sudo su)

For a brief overview of some of the differences between su, su -, and sudo -{i,s} see : [Ubuntu Forums Post with nice table](http://ubuntuforums.org/showpost.php?p=6188826&postcount=4) .

Summary of the differences found -

corrupted by user's

HOME=/root uses root's PATH env vars

sudo -i Y Y[2] N

sudo -s N Y[2] Y

sudo bash N Y[2] Y

sudo su Y N[1] Y

[1] PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

probably set by /etc/environment

[2] PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/X11R6/bin

For a detailed description of the differences see man su and man sudo .

# **Remove Password Prompt For sudo**

| **IconsPage/warning.png** | **If you disable the sudo password for your account, you will seriously compromise the security of your computer. Anyone sitting at your unattended, logged in account will have complete root access, and remote exploits become much easier for malicious crackers.** | IconsPage/warning.png |
| --- | --- | --- |

* **This method is NOT suggested nor supported by the designers of Ubuntu.**
* Please do not suggest this to others unless you personally are available 24/7 to support the user if they have issues as a result of running a shell as root.

These instructions are to remove the prompt for a password when using the **sudo** command. The **sudo** command will still need to be used for root access though.

**Edit the sudoers file**

Open a Terminal window. Type in **sudo visudo**. Add the following line to the END of the file (if not at the end it can be nullified by later entries):

<username> ALL=NOPASSWD: ALL

Replace <username> with your user name (without the <>). This is assuming that Ubuntu has created a group with the same name as your user name, which is typical. You can alternately use the group *users* or any other such group you are in. Just make sure you are in that group. This can be checked by going to **System->Administration->Users and Groups**

Example:

michael ALL=NOPASSWD: ALL

Type in **^x** to exit. This should prompt for an option to save the file, type in **Y** to save.

Log out, and then log back in. This should now allow you to run the sudo command without being prompted for a password.

Or to do this for the system wide group **sudo**

root$ echo "%sudo ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

Log out, and then back in.

# **Reset sudo timeout**

You can make sure sudo asks for password next time by running:

sudo -k

The default sudo timeout length can be changed by following this article: [RootSudoTimeout](https://help.ubuntu.com/community/RootSudoTimeout).

# **OpenSSH server**

OpenSSH is a powerful collection of tools for remotely controlling networked computers and transferring data between them. Here we’ll describe some of the configuration settings possible with the OpenSSH server application and how to change them on your Ubuntu system.

OpenSSH is a freely available version of the Secure Shell (SSH) protocol family of tools. Traditional tools, such as telnet or rcp, are insecure and transmit the user’s password in cleartext when used. OpenSSH provides a server daemon and client tools to facilitate secure, encrypted, remote control and file transfer operations, effectively replacing the legacy tools.

The OpenSSH server component, sshd, listens continuously for client connections from any of the client tools. When a connection request occurs, sshd sets up the correct connection depending on the type of client tool connecting. For example, if the remote computer is connecting with the SSH client application, the OpenSSH server sets up a remote control session after authentication. If a remote user connects to an OpenSSH server with scp, the OpenSSH server daemon initiates a secure copy of files between the server and client after authentication.

OpenSSH can use many authentication methods, including plain password, public key, and Kerberos tickets.

## **Install OpenSSH**

To install the OpenSSH client applications on your Ubuntu system, use this command at a terminal prompt:

sudo apt install openssh-client

To install the OpenSSH server application, and related support files, use this command at a terminal prompt:

sudo apt install openssh-server

## **Configure OpenSSH**

To configure the default behavior of the OpenSSH server application, sshd, edit the file /etc/ssh/sshd\_config. For information about the configuration directives used in this file, refer [to the online manpage](https://manpages.ubuntu.com/manpages/man5/sshd_config.5.html) or run man sshd\_config at a terminal prompt.

There are many directives in the sshd configuration file, which control things like communication settings and authentication modes. The following are examples of configuration directives that can be changed by editing the /etc/ssh/sshd\_config file.

Tip

Before editing the configuration file, you should make a copy of the original /etc/ssh/sshd\_config file and protect it from writing so you will have the original settings as a reference and to reuse as necessary. You can do this with the following commands:

sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.original

sudo chmod a-w /etc/ssh/sshd\_config.original

Since losing an SSH server might mean losing your way to reach a server, check the configuration after changing it and before restarting the server:

sudo sshd -t -f /etc/ssh/sshd\_config

### **Example configuration directive**

Let’s take a look at an example of a configuration directive change. To make your OpenSSH server display the contents of the /etc/issue.net file as a pre-login banner, you can add or modify this line in the /etc/ssh/sshd\_config file:

Banner /etc/issue.net

After making changes to the /etc/ssh/sshd\_config file, save the file. Then, restart the sshd server application to effect the changes using the following command:

sudo systemctl restart ssh.service

Warning

Many other configuration directives for sshd are available to change the server application’s behavior to fit your needs. Be advised, however, if your only method of access to a server is SSH, and you make a mistake when configuring sshd via the /etc/ssh/sshd\_config file, you may find you are locked out of the server upon restarting it. Additionally, if an incorrect configuration directive is supplied, the sshd server may refuse to start, so be particularly careful when editing this file on a remote server.

## **SSH keys**

SSH allows authentication between two hosts without the need of a password. SSH key authentication uses a **private key** and a **public key**.

To generate the keys, run the following command:

ssh-keygen -t rsa

This will generate the keys using the **RSA Algorithm**. At the time of this writing, the generated keys will have 3072 bits. You can modify the number of bits by using the -b option. For example, to generate keys with 4096 bits, you can use:

ssh-keygen -t rsa -b 4096

During the process you will be prompted for a password. Simply hit Enter when prompted to create the key.

By default, the public key is saved in the file ~/.ssh/id\_rsa.pub, while ~/.ssh/id\_rsa is the private key. Now copy the id\_rsa.pub file to the remote host and append it to ~/.ssh/authorized\_keys by running:

ssh-copy-id username@remotehost

Finally, double check the permissions on the authorized\_keys file – only the authenticated user should have read and write permissions. If the permissions are not correct then change them by:

chmod 600 .ssh/authorized\_keys

You should now be able to SSH to the host without being prompted for a password.

## **Import keys from public keyservers**

These days many users have already SSH keys registered with services like Launchpad or GitHub. Those can be imported with:

ssh-import-id <username-on-remote-service>

The prefix lp: is implied and means fetching from Launchpad. The alternative gh: will make the tool fetch from GitHub instead.

## 

# **Public and Private Keys**

Public key authentication is more secure than password authentication. This is particularly important if the computer is visible on the internet. If you don't think it's important, try [logging](https://help.ubuntu.com/community/SSH/OpenSSH/Configuring#Logging) the login attempts you get for the next week. My computer - a perfectly ordinary desktop PC - had over 4,000 attempts to guess my password and almost 2,500 break-in attempts in the last week alone.

With public key authentication, the authenticating entity has a public key and a private key. Each key is a large number with special mathematical properties. The private key is kept on the computer you log in from, while the public key is stored on the **.ssh/authorized\_keys** file on all the computers you want to log in to. When you log in to a computer, the SSH server uses the public key to "lock" messages in a way that can only be "unlocked" by your private key - this means that even the most resourceful attacker can't snoop on, or interfere with, your session. As an extra security measure, most SSH programs store the private key in a passphrase-protected format, so that if your computer is stolen or broken in to, you should have enough time to disable your old public key before they break the passphrase and start using your key. Wikipedia has a [more detailed explanation](http://en.wikipedia.org/wiki/Public-key_cryptography) of how keys work.

Public key authentication is a much better solution than passwords for most people. In fact, if you don't mind leaving a private key unprotected on your hard disk, you can even use keys to do secure automatic log-ins - as part of a network backup, for example. Different SSH programs generate public keys in different ways, but they all generate public keys in a similar format:

<ssh-rsa or ssh-dss> <really long string of nonsense> <username>@<host>

# **Key-Based SSH Logins**

Key-based authentication is the most secure of several modes of authentication usable with OpenSSH, such as plain password and Kerberos tickets. Key-based authentication has several advantages over password authentication, for example the key values are significantly more difficult to brute-force, or guess than plain passwords, provided an ample key length. Other authentication methods are only used in very specific situations.

SSH can use either "RSA" (Rivest-Shamir-Adleman) or "DSA" ("Digital Signature Algorithm") keys. Both of these were considered state-of-the-art algorithms when SSH was invented, but DSA has come to be seen as less secure in recent years. RSA is the only recommended choice for new keys, so this guide uses "RSA key" and "SSH key" interchangeably.

Key-based authentication uses two keys, one "public" key that anyone is allowed to see, and another "private" key that only the owner is allowed to see. To securely communicate using key-based authentication, one needs to create a key pair, securely store the private key on the computer one wants to log in from, and store the public key on the computer one wants to log in to.

Using key based logins with ssh is generally considered more secure than using plain password logins. This section of the guide will explain the process of generating a set of public/private RSA keys, and using them for logging into your Ubuntu computer(s) via OpenSSH.

# **Generating RSA Keys**

The first step involves creating a set of RSA keys for use in authentication.

This should be done on the client.

To create your public and private SSH keys on the command-line:

mkdir ~/.ssh

chmod 700 ~/.ssh

ssh-keygen -t rsa

You will be prompted for a location to save the keys, and a passphrase for the keys. This passphrase will protect your private key while it's stored on the hard drive:

Generating public/private rsa key pair.

Enter file in which to save the key (/home/b/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/b/.ssh/id\_rsa.

Your public key has been saved in /home/b/.ssh/id\_rsa.pub.

Your public key is now available as .ssh/id\_rsa.pub in your home folder.

Congratulations! You now have a set of keys. Now it's time to make your systems allow you to login with them

## **Choosing a good passphrase**

You need to change all your locks if your RSA key is stolen. Otherwise the thief could impersonate you wherever you authenticate with that key.

An SSH key passphrase is a secondary form of security that gives you a little time when your keys are stolen. If your RSA key has a [strong passphrase](https://help.ubuntu.com/community/StrongPasswords), it might take your attacker a few hours to guess by brute force. That extra time should be enough to log in to any computers you have an account on, delete your old key from the .ssh/authorized\_keys file, and add a new key.

Your SSH key passphrase is *only* used to protect your private key from thieves. It's never transmitted over the Internet, and the strength of your key has nothing to do with the strength of your passphrase.

The decision to protect your key with a passphrase involves convenience x security. Note that if you protect your key with a passphrase, then when you type the passphrase to unlock it, your local computer will generally leave the key unlocked for a time. So if you use the key multiple times without logging out of your local account in the meantime, you will probably only have to type the passphrase once.

If you do adopt a passphrase, pick a [strong](https://help.ubuntu.com/community/StrongPasswords) one and store it securely in a password manager. You may also write it down on a piece of paper and keep it in a secure place. If you choose not to protect the key with a passphrase, then just press the return when ssh-keygen asks.

## **Key Encryption Level**

Note: The default is a 2048 bit key. You can increase this to 4096 bits with the -b flag (Increasing the bits makes it harder to crack the key by brute force methods).

ssh-keygen -t rsa -b 4096

## **Password Authentication**

The main problem with public key authentication is that you need a secure way of getting the public key onto a computer before you can log in with it. If you will only ever use an SSH key to log in to your own computer from a few other computers (such as logging in to your PC from your laptop), you should copy your SSH keys over on a memory stick, and [disable password authentication](https://help.ubuntu.com/community/SSH/OpenSSH/Configuring#disable-password-authentication) altogether. If you would like to log in from other computers from time to time (such as a friend's PC), make sure you have a [strong password](https://help.ubuntu.com/community/StrongPasswords).

# **Transfer Client Key to Host**

The key you need to transfer to the host is the public one. If you can log in to a computer over SSH using a password, you can transfer your RSA key by doing the following from your own computer:

ssh-copy-id <username>@<host>

Where <username> and <host> should be replaced by your username and the name of the computer you're transferring your key to.

(i) Due to [this bug](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=99785), you cannot specify a port other than the standard port 22. You can work around this by issuing the command like this: ssh-copy-id "<username>@<host> -p <port\_nr>". If you are using the standard port 22, you can ignore this tip.

Another alternative is to copy the public key file to the server and concatenate it onto the authorized\_keys file manually. It is wise to back that up first:

cp authorized\_keys authorized\_keys\_Backup

cat id\_rsa.pub >> authorized\_keys

You can make sure this worked by doing:

ssh <username>@<host>

You should be prompted for the passphrase for your key:

| Enter passphrase for key '/home/<user>/.ssh/id\_rsa': |
| --- |

Enter your passphrase, and provided *host* is configured to allow key-based logins, you should then be logged in as usual.

# **Troubleshooting**

### **Encrypted Home Directory**

If you have an encrypted home directory, SSH cannot access your authorized\_keys file because it is inside your encrypted home directory and won't be available until after you are authenticated. Therefore, SSH will default to password authentication.

To solve this, create a folder outside your home named /etc/ssh/<username> (replace "<username>" with your actual username). This directory should have 755 permissions and be owned by the user. Move the authorized\_keys file into it. The authorized\_keys file should have 644 permissions and be owned by the user.

Then edit your /etc/ssh/sshd\_config and add:

AuthorizedKeysFile /etc/ssh/%u/authorized\_keys

Finally, restart ssh with:

sudo service ssh restart

The next time you connect with SSH you should not have to enter your password.

### **username@host's password:**

If you are not prompted for the passphrase, and instead get just the

| username@host's password: |
| --- |

prompt as usual with password logins, then read on. There are a few things which could prevent this from working as easily as demonstrated above. On default Ubuntu installs however, the above examples should work. If not, then check the following condition, as it is the most frequent cause:

On the host computer, ensure that the /etc/ssh/sshd\_config contains the following lines, and that they are *uncommented*;

PubkeyAuthentication yes

RSAAuthentication yes

If not, add them, or uncomment them, restart OpenSSH, and try logging in again. If you get the *passphrase* prompt now, then congratulations, you're logging in with a key!

### **Permission denied (publickey)**

If you're sure you've correctly configured sshd\_config, copied your ID, and have your private key in the .ssh directory, and still getting this error:

| Permission denied (publickey). |
| --- |

Chances are, your /home/<user> or ~/.ssh/authorized\_keys permissions are too open by OpenSSH standards. You can get rid of this problem by issuing the following commands:

chmod go-w ~/

chmod 700 ~/.ssh

chmod 600 ~/.ssh/authorized\_keys

### **Error: Agent admitted failure to sign using the key.**

This error occurs when the ssh-agent on the client is not yet managing the key. Issue the following commands to fix:

ssh-add

This command should be entered after you have copied your public key to the host computer.

### **Debugging and sorting out further problems**

The permissions of files and folders is crucial to this working. You can get debugging information from both the client and server.

if you think you have set it up correctly , yet still get asked for the password, try starting the server with debugging output to the terminal.

sudo /usr/sbin/sshd -d

To connect and send information to the client terminal

ssh -v ( or -vv) username@host's

# **Where to From Here?**

No matter how your public key was generated, you can add it to your Ubuntu system by opening the file .ssh/authorized\_keys in your favourite text editor and adding the key to the bottom of the file. You can also limit the SSH features that the key can use, such as disallowing port-forwarding or only allowing a specific command to be run. This is done by adding "options" before the SSH key, on the same line in the authorized\_keys file. For example, if you maintain a CVS repository, you could add a line like this:

command="/usr/bin/cvs server",no-agent-forwarding,no-port-forwarding,no-X11-forwarding,no-user-rc ssh-dss <string of nonsense>...

When the user with the specified key logged in, the server would automatically run /usr/bin/cvs server, ignoring any requests from the client to run another command such as a shell. For more information, see [the sshd man page](http://www.openbsd.org/cgi-bin/man.cgi?query=sshd&sektion=8#SSHRC). /755

Tema seguinte: Add user e Usermod

# **Why use the terminal?**

*"Under Linux there are GUIs (graphical user interfaces), where you can point and click and drag, and hopefully get work done without first reading lots of documentation. The traditional Unix environment is a CLI (command line interface), where you type commands to tell the computer what to do. That is faster and more powerful, but requires finding out what the commands are."  
 -- from* ***man intro(1)***

This page gives an introduction to using the command-line interface terminal, from now on abbreviated to the **terminal**. There are many varieties of Linux, but almost all of them use similar commands that can be entered from the terminal.

There are also many graphical user interfaces (GUIs), but each of them works differently and there is little standardization between them. Experienced users who work with many different Linux distributions therefore find it easier to learn commands that can be used in all varieties of Ubuntu and, indeed, in other Linux distributions as well.

For the novice, commands can appear daunting:

sudo gobbledegook blah\_blah -w -t -h --long-switch aWkward/ComBinationOf/mixedCase/underscores\_strokes/and.dots

However, it is important to note that even experienced users often cut and paste commands (from a guide or manual) into the terminal; they do not memorize them.

It is important, of course, to know how to use the terminal - and anyone who can manage typing, backspacing, and cutting and pasting will be able to use the terminal (it is not more difficult than that).

# **Starting a terminal**

## **In Unity**

Unity is the default desktop environment used as of 11.04. Where systems are not ready for Unity they revert to GNOME which is also used in previous releases such as Ubuntu 10.04 LTS (Lucid), see next sub-section.

The easiest way to open the terminal is to use the 'search' function on the dash. Or you can click on the 'More Apps' button, click on the 'See more results' by the installed section, and find it in that list of applications. A third way, available after you click on the 'More Apps' button, is to go to the search bar, and see that the far right end of it says 'All Applications'. You then click on that, and you'll see the full list. Then you can go to Accessories -> Terminal after that. So, the methods in Unity are:

**Dash** -> **Search for Terminal**

**Dash** -> **More Apps** -> **'See More Results'** -> **Terminal**

**Dash** -> **More Apps** -> **Accessories** -> **Terminal**

**Keyboard Shortcut:** Ctrl + Alt + T

## **In GNOME**

GNOME is the classic desktop environment for Ubuntu 11.04 (Natty) and is the default desktop environment in earlier releases, such as Ubuntu 10.04 LTS (Lucid).

**Applications menu** -> **Accessories** -> **Terminal**.

**Keyboard Shortcut:** Ctrl + Alt + T

## **In Xfce (Xubuntu)**

**Applications menu** -> **System** -> **Terminal**.

**Keyboard Shortcut:** Super + T

**Keyboard Shortcut:** Ctrl + Alt + T

## **In KDE (Kubuntu)**

**KMenu** -> **System** -> **Terminal Program (Konsole)**.

## **In LXDE (Lubuntu)**

**Menu** -> **Accessories** -> **LXTerminal**.

**Keyboard Shortcut:** Ctrl + Alt + T

# **Commands**

## **sudo: Executing Commands with Administrative Privileges**

The sudo command executes a command with administrative privileges (root-user administrative level), which is necessary, for example, when working with directories or files not owned by your user account. When using sudo you will be prompted for your password. Only users with administrative privileges are allowed to use sudo.

Be careful when executing commands with administrative privileges - you might damage your system! You should **never** use normal sudo to start graphical applications with administrative privileges. Please see [RootSudo](https://help.ubuntu.com/community/RootSudo) for more information on using sudo correctly.

## **File & Directory Commands**

* The tilde (~) symbol stands for your home directory. If you are *user*, then the tilde (~) stands for /home/*user*
* **pwd**: The **pwd** command will allow you to know in which directory you're located (**pwd** stands for "print working directory"). Example: **"pwd"** in the Desktop directory will show "~/Desktop". Note that the GNOME Terminal also displays this information in the title bar of its window. A useful gnemonic is "present working directory."
* **ls**: The **ls** command will show you ('list') the files in your current directory. Used with certain options, you can see sizes of files, when files were made, and permissions of files. Example: **"ls ~"** will show you the files that are in your home directory.
* **cd**: The **cd** command will allow you to change directories. When you open a terminal you will be in your home directory. To move around the file system you will use **cd**. Examples:
  + To navigate into the root directory, use **"cd /"**
  + To navigate to your home directory, use **"cd"** or **"cd ~"**
  + To navigate up one directory level, use **"cd .."**
  + To navigate to the previous directory (or back), use **"cd -"**
  + To navigate through multiple levels of directory at once, specify the full directory path that you want to go to. For example, use, **"cd /var/www"** to go directly to the /www subdirectory of /var/. As another example, **"cd ~/Desktop"** will move you to the Desktop subdirectory inside your home directory.
* **cp**: The **cp** command will make a copy of a file for you. Example: **"cp file foo"** will make an exact copy of "file" and name it "foo", but the file "file" will still be there. If you are copying a directory, you must use **"cp -r directory foo"** (copy recursively). (To understand what "recursively" means, think of it this way: to copy the directory and all its files and subdirectories and all their files and subdirectories of the subdirectories and all their files, and on and on, "recursively")
* **mv**: The **mv** command will move a file to a different location or will rename a file. Examples are as follows: **"mv file foo"** will rename the file "file" to "foo". **"mv foo ~/Desktop"** will move the file "foo" to your Desktop directory, but it will not rename it. You must specify a new file name to rename a file.
  + To save on typing, you can substitute '~' in place of the home directory.
  + Note that if you are using **mv** with **sudo** you can use the ~ shortcut, because the terminal expands the ~ to your home directory. However, when you open a root shell with **sudo -i** or **sudo -s**, ~ will refer to the root account's home directory, not your own.
* **rm**: Use this command to remove or delete a file in your directory.
* **rmdir**: The **rmdir** command will delete an *empty* directory. To delete a directory and all of its contents recursively, use **rm -r** instead.
* **mkdir**: The **mkdir** command will allow you to create directories. Example: **"mkdir music"** will create a directory called "music".

Here is an example of when it would be necessary to execute a command with administrative privileges. Let's suppose that another user has accidentally moved one of your documents from your **Documents** directory to the root directory. Normally, to move the document back, you would type **mv /mydoc.odt ~/Documents/mydoc.odt**, but by default you are not allowed to modify files outside your home directory. To get around this, you would type **sudo mv /mydoc.odt ~/Documents/mydoc.odt**. This will successfully move the document back to its correct location, provided that you have administrative privileges.

## **Running a File Within a Directory**

So you've decided to run a file using the command-line? Well... there's a command for that too!

**./***filename.extension*

After navigating to the file's directory, this command will enable any Ubuntu user to run files compiled via GCC or any other programming language. Although the example above indicates a file name extension, please notice that, differently from some other operating systems, Ubuntu (and other Linux-based systems) do not care about file extensions (they can be anything, or nothing). **Keep in mind that the 'extension' will vary depending upon the language the source code is written in. Also, it is not possible, for compiled languages (like C and C++) to run the source code directly -- the file must be** [**compiled**](https://help.ubuntu.com/community/CompilingEasyHowTo) **first, which means it will be translated from a human-readable programming language to something the computer can understand.** Some possible extensions: ".c" for C source, ".cpp" for C++, ".rb" for Ruby, ".py" for Python, etc. Also, remember that (in the case of interpreted languages like Ruby & Python) ***you must have a version of that language installed on Ubuntu before trying to run files written with it.***

Finally, the file will only be executed if the file permissions are correct -- please see the [FilePermissions](https://help.ubuntu.com/community/FilePermissions) help page for details.

## **System Information Commands**

**df**: The **df** command displays filesystem disk space usage for all mounted partitions. "**df -h**" is probably the most useful - it uses megabytes (M) and gigabytes (G) instead of blocks to report. (**-h** means "human-readable")

**du**: The **du** command displays the disk usage for a directory. It can either display the space used for all subdirectories or the total for the directory you run it on. Example:

user@users-desktop:~$ du /media/floppy

1032 /media/floppy/files

1036 /media/floppy/

user@users-desktop:~$ du -sh /media/floppy

1.1M /media/floppy/

In the above example **-s** means "Summary" and **-h** means "Human Readable".

**free**: The **free** command displays the amount of free and used memory in the system. **"free -m"** will give the information using megabytes, which is probably most useful for current computers.

**top**: The **top** ('table of processes') command displays information on your Linux system, running processes and system resources, including CPU, RAM & swap usage and total number of tasks being run. To exit **top**, press **"q"**.

**uname -a**: The **uname** command with the **-a** option prints all system information, including machine name, kernel name & version, and a few other details. Most useful for checking which kernel you're using.

**lsb\_release -a**: The **lsb\_release** command with the **-a** option prints version information for the Linux release you're running, for example:

user@computer:~$ lsb\_release -a

No LSB modules are available.

Distributor ID: Ubuntu

Description: Ubuntu 11.10

Release: 11.10

Codename: oneiric

**ip addr** reports on your system's network interfaces.

## **Adding A New User**

The **"adduser newuser"** command will create a new general user called "newuser" on your system, and to assign a password for the newuser account use **"passwd newuser"**.

# **Options**

The default behaviour for a command may usually be modified by adding a **--*option*** to the command. The **ls** command for example has an **-s** option so that **"ls -s"** will include file sizes in the listing. There is also a **-h** option to get those sizes in a "human readable" format.

Options can be grouped in clusters so **"ls -sh"** is exactly the same command as **"ls -s -h"**. Most options have a long version, prefixed with two dashes instead of one, so even **"ls --size --human-readable"** is the same command.

# **"Man" and getting help**

**Warning /!\** **man *command***, **info *command*** and ***command* --help** are the most important tools at the command line.

Nearly every command and application in Linux will have a man (manual) file, so finding them is as simple as typing **"man "command""** to bring up a longer manual entry for the specified command. For example, **"man mv"** will bring up the **mv** (move) manual.

Move up and down the man file with the arrow keys, and quit back to the command prompt with **"q"**.

**"man man"** will bring up the manual entry for the **man** command, which is a good place to start!

**"man intro"** is especially useful - it displays the "Introduction to user commands" which is a well-written, fairly brief introduction to the Linux command line.

There are also **info** pages, which are generally more in-depth than **man** pages. Try **"info info"** for the introduction to info pages.

Some software developers prefer **info** to **man** (for instance, GNU developers), so if you find a very widely used command or app that doesn't have a **man** page, it's worth checking for an **info** page.

Virtually all commands understand the **-h** (or **--help**) option which will produce a short usage description of the command and it's options, then exit back to the command prompt. Try **"man -h"** or **"man --help"** to see this in action.

*Caveat: It's possible (but rare) that a program doesn't understand the -h option to mean help. For this reason, check for a* ***man*** *or* ***info*** *page first, and try the long option --help before -h.*

## **Searching the manual pages**

If you aren't sure which command or application you need to use, you can try searching the manual pages. Each manual page has a name and a short description.

To search the names for <string> enter:

whatis -r <string>

For example, whatis -r cpy will list manual pages whose names contain **cpy**. The output from whatis -r cpy will in part depend on your system - but might be as follows:

memccpy (3) - copy memory area

memcpy (3) - copy memory area

mempcpy (3) - copy memory area

[some lines removed]

wcsncpy (3) - copy a fixed-size string of wide characters

wmemcpy (3) - copy an array of wide-characters

wmempcpy (3) - copy memory area

To search the names or descriptions for <string> enter:

apropos -r <string>

For example, apropos -r "copy files" will list manual pages whose names or descriptions contain **copy files**. The output from apropos -r "copy files" will in part depend on your system - but might be as follows:

cp (1) - copy files and directories

cpio (1) - copy files to and from archives

gvfs-copy (1) - Copy files

gvfs-move (1) - Copy files

install (1) - copy files and set attributes

# **Other Useful Things**

## **Prettier Manual Pages**

Users who have *Konqueror* installed will be pleased to find they can read and search man pages in a web browser context, prettified with their chosen desktop fonts and a little colour, by visiting **man:/*command*** in Konqueror's address bar. Some people might find this lightens the load if there's lots of documentation to read/search.

## **Pasting in commands**

Often, you will be referred to instructions that require commands to be pasted into the terminal. You might be wondering why the text you've copied from a web page using **Ctrl + C** won't paste in with **ctrl+V**. Surely you don't have to type in all those nasty commands and filenames? Relax. **ctrl+shift+V** pastes into a GNOME terminal; you can also do middle button click on your mouse (both buttons simultaneously on a two-button mouse) or right click and select *Paste* from the menu. However, if you want to avoid the mouse and yet paste it, use "Shift + Insert", to paste the command. If you have to copy it from another terminal / webpage, you can use "Ctrl + Insert" to copy.

## **Save on typing**

| **Up Arrow** or **Ctrl + P** | Scrolls through the commands you've entered previously. | |
| --- | --- | --- |
| **Down Arrow** or **Ctrl + N** | Takes you back to a more recent command. | |
| **Enter** | When you have the command you want. | |
| **tab** | A very useful feature. It autocompletes any commands or filenames, if there's only one option, or else gives you a list of options. | |
| **Ctrl + R** | Searches for commands you've already typed. When you have entered a very long, complex command and need to repeat it, using this key combination and then typing a portion of the command will search through your command history. When you find it, simply press **Enter**. | |
| **History** | The **history** command shows a very long list of commands that you have typed. Each command is displayed next to a number. You can type **!x** to execute a previously typed command from the list (replace the X with a number). If you **history** output is too long, then use **history | less** for a scrollable list. | |

* *Example*: you ran **history** and found you want to use command 1967. Simply enter

!1967

## **Change the text**

The mouse won't work. Use the left/right arrow keys to move around the line.

When the cursor is where you want it in the line, typing *inserts* text - ie it doesn't overtype what's already there.

| **Ctrl + A** or **Home** | Moves the cursor to the *start* of a line. | |
| --- | --- | --- |
| **Ctrl+ E** or **End** | Moves the cursor to the ***e****nd* of a line. | |
| **Esc + B** | Moves to the **b**eginning of the previous or current word. | |
| **Ctrl + K** | Deletes from the current cursor position to the end of the line. | |
| **Ctrl + U** | Deletes from the start of the line to the current cursor position. | |
| **Ctrl + W** | Deletes the **w**ord before the cursor. | |
| **Alt + B** | Goes **b**ack one word at a time. | |
| **Alt + F** | Moves **f**orward one word at a time. | |
| **Alt + C** | **C**apitalizes letter where cursor is and moves to end of word. | |

# **More ways to run a terminal**

You can set your own keyboard shortcut to run a terminal. See [KeyboardShortcuts](https://help.ubuntu.com/community/KeyboardShortcuts) for details of keyboard shortcuts.

You can run more than terminal - in tabs or separate windows.

You can also install guake (GNOME), tilda (XFCE / LXDE/Mate) or yakuake (KDE) and have a terminal which appears and hides on shortcut key. This can be particularly useful if you use terminal a lot. Drop down terminals can make things a lot easier if you are trying to run a desktop enviroment with a non defualt window manager and something goes wrong drop down terminals can run the orginal window manager --replace to restore a previous option to make things much less painful.

# **An extremely handy tool :: Incremental history searching**

In terminal enter:

gedit ~/.inputrc

Then copy paste and save:

"\e[A": history-search-backward

"\e[B": history-search-forward

"\e[C": forward-char

"\e[D": backward-char

From now on, **and many agree this is the most useful terminal tool**, it saves you a lot of writing/memorizing...

All you need to do to find a previous command is to enter say the first two or three letters and upward arrow will take you there quickly:

Say I want:

for f in \*.mid ; do timidity "$f"; done

All I need to do is enter:

fo

And hit upward arrow command will soon appear.

# **How to create upsidedown and/or reverse text with your terminal**

If you wish or need to ever flip text upside down [vertical flip] "uʍop ǝpısdn ʇxǝʇ dıʃɟ" or/and create reverse text here is a terminal way to achieve this.

Copy/paste and save the following as flip.pl in your home folder (thanks to Lars Noodén for script).

#!/usr/bin/perl

use strict;

use warnings;

use utf8;

binmode(STDOUT, ":utf8");

my %flipTable = (

"a" => "\x{0250}",

"b" => "q",

"c" => "\x{0254}",

"d" => "p",

"e" => "\x{01DD}",

"f" => "\x{025F}",

"g" => "\x{0183}",

"h" => "\x{0265}",

"i" => "\x{0131}",

"j" => "\x{027E}",

"k" => "\x{029E}",

"l" => "|",

"m" => "\x{026F}",

"n" => "u",

"o" => "o",

"p" => "d",

"q" => "b",

"r" => "\x{0279}",

"s" => "s",

"t" => "\x{0287}",

"u" => "n",

"v" => "\x{028C}",

"w" => "\x{028D}",

"x" => "x",

"y" => "\x{028E}",

"z" => "z",

"A" => "\x{0250}",

"B" => "q",

"C" => "\x{0254}",

"D" => "p",

"E" => "\x{01DD}",

"F" => "\x{025F}",

"G" => "\x{0183}",

"H" => "\x{0265}",

"I" => "\x{0131}",

"J" => "\x{027E}",

"K" => "\x{029E}",

"L" => "|",

"M" => "\x{026F}",

"N" => "u",

"O" => "o",

"P" => "d",

"Q" => "b",

"R" => "\x{0279}",

"S" => "s",

"T" => "\x{0287}",

"U" => "n",

"V" => "\x{028C}",

"W" => "\x{028D}",

"X" => "x",

"Y" => "\x{028E}",

"Z" => "z",

"." => "\x{02D9}",

"[" => "]",

"'" => ",",

"," => "'",

"(" => ")",

"{" => "}",

"?" => "\x{00BF}",

"!" => "\x{00A1}",

"\"" => ",",

"<" => ">",

"\_" => "\x{203E}",

";" => "\x{061B}",

"\x{203F}" => "\x{2040}",

"\x{2045}" => "\x{2046}",

"\x{2234}" => "\x{2235}",

"\r" => "\n",

" " => " "

);

while ( <> ) {

my $string = reverse( $\_ );

while ($string =~ /(.)/g) {

print $flipTable{$1};

}

print qq(\n);

}

Then to set it up:

sudo mv flip.pl /bin/

cd /bin/

sudo chown yourusername flip.pl && sudo chmod +x flip.pl

Then open terminal and enter:

flip.pl

else

perl /bin/flip.pl

Write what you want and hit return

Copy and paste wherever you want text document or Internet forum, etc...

* ɹǝʇuǝ puɐ ʇuɐʍ noʎ ʇɐɥʍ ǝʇıɹʍ ˙˙˙ɔʇǝ ɯnɹoɟ ʇǝuɹǝʇuı ɹo ʇuǝɯnɔop ʇxǝʇ ʇuɐʍ noʎ ɹǝʌǝɹǝɥʍ ǝʇsɐd puɐ ʎdoɔ

==================

If you want to reverse back to front, write your text in a text editor and save as mytext to the home folder.

Then enter:

rev mytext

Copy and paste the result, tluser eht etsap dna ypoc.

And of course you can combine both for truly cryptic results, ɔodʎ ɐup dɐsʇǝ ʇɥǝ ɹǝsnʃʇ

# **What Is grep?**

**Grep** is a command-line tool that allows you to find a string in a file or stream. It can be used with a regular expression to be more flexible at finding strings.

This page gives an introduction to grep. For more information enter:

man grep

in a terminal.

# **How To Use grep**

In the simplest case grep can be invoked as follows:

grep 'STRING' filename

The above command searches the file for STRING and lists the lines that contain a match.

This is OK but it does not show the true power of grep. The above command only looks at one file. A cool example of using grep with multiple files would be to find all lines in all files in a given directory that contain the name of a person. This can be easily accomplished as follows:

grep 'Nicolas Kassis' \*

The above command searches all files in the current directory for the name and lists all lines that contain a match.

Notice the use of quotes in the above command. Quotes are not usually essential, but in this example they are essential because the name contains a space. Double quotes could also have been used in this example.

# **Regular Expressions**

grep can search for complicated patterns to find what you need. Here is a list of some of the special characters used to create a regular expression:

| ^ | Denotes the beginning of a line |
| --- | --- |
| $ | Denotes the end of a line |
| . | Matches any single character |
| \* | The preceding item in the regular expression will be matched zero or more times |
| [] | A bracket expression is a list of characters enclosed by [ and ]. It matches any single character in that list; if the first character of the list is the caret ^ then it matches any character not in the list |
| \< | Denotes the beginning of a word |
| \> | Denotes the end of a word |

Here is an example of a regular expression search:

grep "\<[A-Za-z].\*\>" file

This regular expression matches any "word" that begins with a letter (upper or lower case). For example, "words" that begin with a digit would not match. The grep command lists the lines that contain a match.

## **Finding Files**

### **Places --> Search for Files**

If you need to search for a file on the computer, there is an easy and built-in way to do it. Launch the Search for Files program from the Places menu. By default, this will only search your files (your home directory), but you can tell it to search in other folders with the *Look in folder* dropdown. This will only search on file names. If you want to search inside file, see the next section on Beagle.

### **Thunar (File Manager for Xubuntu)**

In Xubuntu 7.10, Thunar is not initially set up with a file-searcher. There are a number of file-searchers that can be set up, but I will explain how to set up the gnome-search-tool.

1. Install 'gnome-utils' using Synaptic Package Manager

2. In Thunar, choose 'Edit'/'Configure Custom Actions' and add a new custom action with the name 'Search for Files' and the command 'gnome-search-tool --path=%f'. Under the 'Appearance Conditions' tab, tick 'Directories' - this is important.

You will now be able to tick on any directory in the **right** pane, and choose 'Search for Files'. This file-searcher has many different options for doing advanced searches.

### **Tracker**

Tracker is an indexing system. It will keep track of all your files and their contents and allow you to quickly find files on your computer. For more information see [Tracker](https://help.ubuntu.com/community/Tracker).

### **Beagle**

Beagle is also indexing system, designed to help you search inside your files. It is not installed by default. For more information on installation, see [Beagle](https://help.ubuntu.com/community/Beagle).

### **Recoll**

Beagle is also indexing system. It is a full text search package based on a very strong backend (Xapian), for which it provides an easy to use and feature-rich interface. It is not installed by default. For more information on installation, see [Recoll](https://help.ubuntu.com/community/Recoll).

## **Command Line**

There are a number of command line tools to help you find a file on system.

### **apt-file**

The first method is for finding files that are (or should be) provided by Ubuntu's packages. If you're not familiar with the file being complained about, odds are good that it falls into this category. The apt packaging system is aware of almost all the files it provides and can be queried to learn what package provides a given file and where that file is (or should be). To take advantage of this feature, install the package apt-file and then run apt-file update in a terminal.

Now, say you've been trying to compile a new kernel module, and its build system complains at you that there's "No such file or directory - version.h" or some such. You could "apt-file search version.h", but that would be a pretty broad search and would return such files as subversion.html. You can guess that this file has something to do with the kernel, and I happen to know that in a fully prepped kernel tree, it should be found in a directory called "linux", so I say "apt-file search linux/version.h" and find just what I was looking for: which package I need to install to have the file I need. See the apt-file man page for some handy options like case-insensitive searching and regular expression matching.

### **dpkg -L**

Maybe you suspect that the file in question is supposed to be provided by the same package you're working with.

dpkg -L <packagename>

will show you a list of files provided by that package. For example, you've just installed kxdocker\_0.32-1\_i386.deb and your first guess, "kxdocker", doesn't run the program.

$ kxdocker

-bash: kxdocker: command not found

Well it's in there somewhere:

$ dpkg -L kxdocker | grep bin

/usr/local/kde/bin

/usr/local/kde/bin/kxdocker

Ah, it's there, but /usr/local/kde/bin isn't in your $PATH. Now you know that you can add it to your $PATH or run the command with the full path.

### **dpkg -S**

Sometimes you might want to find out which package provides a certain file.

dpkg -S /full/path/to/file

will show you the package. For example,

dpkg -S /usr/bin/gnome-keybindings-properties

gnome-control-center: /usr/bin/gnome-keybinding-properties

tells you that gnome-keybinding-properties is provided by the package gnome-control-center.

### **find**

The Unix command "find" is quite powerful, and if you know how to use it you can find pretty well anything. The basic syntax is "find <path> <options>". Options include criteria for your search, actions to take on files found, etc. I'll give you a couple of examples then point you to the man page for detailed usage and more interesting examples.

You want to find every file in ~/mydir and all its subdirectories, recursively, with a file extension of .htm (or .HTM or .Htm...) and delete it. I've seen a lot of attempts like rm -rf ~/mydir/\*.htm which really don't come close. The correct solution is

find ~/mydir -iname '\*.htm' -exec rm {} \;

"-iname" says that you want to do a case-insensitive search on the filename. '\*.htm' is in single quotes to prevent bash from expanding the \*, which will produce unexpected results. The rest of the command says to remove any file matching the query. The "{}" will be replaced by the filename (with path) returned by the search, and "\;" will separate one rm command from the next. Nearly every -exec option should be terminated with a "\;".

Now you want to fix permissions. For some reason, there seem to be directories in your home directory that you don't have permission to enter. You know that the operative bit for directories is the execute bit. You know that "chmod -R +x ~" will add the execute bit to every file and directory in ~ (or $HOME), but you only want to operate on directories - BritneySpearsOopsIDidItAgain.avi doesn't need to be executable. This is solved with:

find ~ -type d -exec chmod +x {} \;

where "-type d" of course means directories.

Finally, you want to make a playlist out of all the mp3 and ogg files in your home directory.

find ~ -type f \( -iname '\*.mp3' -o -iname '\*.ogg' \) > mynewplaylist.m3u

We group the -iname parameters in parentheses and separate them with -o (the "OR" operator) to say that any match must be a file, AND it must be named .mp3 OR .ogg (case-insensitive) to be returned. We redirect the output to a new file called mynewplaylist.m3u, and presto! We have a playlist.

### **locate**

Oh, where did I put that file? I've got directories and partitions all over the computer where I put files, and they're not quite well-organized enough for me to figure out where I put resume.doc last year when I was job-hunting. I don't want to use "find" because it'll take forever to search my entire computer. I can't use apt-file because resume.doc is not provided by any Ubuntu package. Thankfully my computer indexes all my files every night while I sleep and I can search just the index, which will take only a few seconds, even on as an expansive a filesystem as mine. Of course the software needs to be installed ("sudo apt-get install slocate"). If I look at root's cron jobs ("sudo crontab -e") I see that the slocate package has been faithfully updating my index

02 4 \* \* \* /usr/bin/updatedb -e /mnt/data,/mnt/files

every night at 4:02am, excluding /mnt/data and /mnt/files, which are remote Samba mounts that I prefer not to index. So as long as my resume is on one of my local mounted filesystems, I should be able to find it.

locate resume.doc

It's that simple, unless I'm not quite sure about the spelling, in which case I might use -i for a case-insensitive search or -r to use a POSIX regular expression.

### **which and whereis**

Two more commands that occasionally come in handy are "which" and "whereis". "which" will search your $PATH ("echo $PATH") for a given command and return the first match - the one that will be run if you specify the command without a path. "whereis" will return any and all binaries, sources and man pages associated with the argument you give it. "which" can come in handy for example if you've installed the same software via apt-get and again from source. Very likely you'll have the same command in two different places, and "which" can help you figure out why the version you thought you updated to isn't running. If you have "xchat" in both /usr/bin and /usr/local/bin, "which" will tell you what will happen when you just run "xchat". "whereis" will show you both, plus any man pages and (depending on the circumstances) the source tree from which you compiled it.

Gnome, KDE etc. have numerous utilities for finding files, and their usage is left as an exercise for the reader. Learn how to use these commands effectively and I predict you'll forget those graphical utilities ever existed.

# **Find**

The GNU find command searches files within a directory and its subdirectories according to several criteria such as name, size and time of last read/write. By default find prints the name of the located files but it can also perform commands on these files.

The GNU find command is part of the [GNU findutils](http://www.gnu.org/software/findutils/) and is installed on every Ubuntu system. findutils is actually made up of 4 utilities:

1. **find** - search for files in a directory hierarchy, whether its a database or not
2. **locate** - list files in databases that match a pattern, i.e. find inside updatedb's list
3. **updatedb** - update a file name database, i.e. collection of **db's only**, such as sqlite
4. **xargs** - build and execute command lines from standard input - usually you do this directly w/o xargs

This wiki page will be only be dealing with find while also briefly mentioning xargs. Hopefully [locate](https://help.ubuntu.com/community/locate) and [updatedb](https://help.ubuntu.com/community/updatedb) will be covered on their own page in the near future. "find", like "locate", can find database-files as well, but "locate" is more specialized for this task. You would run updatedb before using locate, which relies on the data produced by "updateDB".

# **The Basics**

**find ~ -name readme.txt**

will find this file below the home directory. find works **incredibly fast on the second run!** You can search the whole / root-dir-tree in a mere approx. 3 seconds (on second run, when cache is effective) on a 500 GB ext4-fs hard disk.

The syntax for using find is:

find [-H] [-L] [-P] [path...] [expression]

The 3 options [-H] [-L] [-P] are not commonly seen but should at least be noted if only to realise that the -P option will be the *default* unless another option is specified:

* -H : Do not follow symbolic links, except while processing the command line arguments.
* -L : Follow symbolic links.
* -P : Never follow symbolic links: the default option.

The option [path...] refers to the particular location that you wish to search, whether it be your $HOME directory, a particular directory such as /usr, your present working directory which can simply be expressed as '.' or your entire computer which can be expressed as '/'.

The option [expression] refers to one or a series of options which effect the overall option of the find command. These options can involve a search by name, by size, by access time or can also involve actions taken upon these files.

## **Locating Files by Name**

The most common use of find is in the search for a specific file by use of its name. The following command searches the home directory and all of its subdirectories looking for the file mysong.ogg:

find $HOME -name 'mysong.ogg'

It is important to get into the habit of quoting patterns in your search as seen above or your search results can be a little unpredictable. Such a search can be much more sophisticated though. For example if you wished to search for *all* of the ogg files in your home directory, some of which you think might be named 'OGG' rather than 'ogg', you would run:

find $HOME -iname '\*.ogg'

Here the option '-iname' performs a case-insensitive search while the wildcard character '\*' matches any character, or number of characters, or zero characters. To perform the same search on your *entire drive* you would run:

sudo find / -iname '\*.ogg'

This could be a slow search depending on the number of directories, sub-directories and files on your system. This highlights an important difference in the way that find operates in that it examines the system *directly* each time unlike programs like locate or slocate which actually examine a regularly updated *database* of filnames and locations.

## **Locating Files by Size**

Another possible search is to search for files by *size*. To demonstrate this we can again search the home directory for Ogg Vorbis files but this time looking for those that are 100 megabytes or larger:

find $HOME -iname '\*.ogg' -size +100M

There are several options with -size, I have used 'M' for 'megabytes' here but 'k' for 'kilobytes' can be used or 'G' for 'Gigabytes'. This search can then be altered to look for *files* only that are *less* than 100 megabytes:

find $HOME -iname '\*.ogg' -type f -size -100M

Are you starting to see the power of find, and the thought involved in constructing a focused search? If you are interested there is more discussion of these *combined* searches in the Advanced Usage section below.

## **Locating Files by Access Time**

It is also possible to locate files based on their access time or the time that they were last used, or viewed by the system. For example to show all files that have not been accessed in the $HOME directory for 30 days or more:

find $HOME -atime +30

This type of search is normally more useful when combined with *other* find searches. For example one could search for all ogg files in the $HOME directory that have an access time of greater than 30 days:

find $HOME -iname '\*.ogg' -atime +30

The syntax works from left to right and by default find joins the 2 expressions with an implied "and". This is dealt with in more depth in the section below entitled "Combining Searches".

# **Advanced Usage**

The sections above detail the most common usage of find and this would be enough for most searches. However there are many more possibilities in the usage of find for quite advanced searches and this sections discusses a few of these possibilities.

## **Combining Searches**

It is possible to combine searches when using find with the use of what is known in the find man pages as *operators*. These are

| -and | -or | -not |
| --- | --- | --- |

as in

**find ~ -name 'xx\*' -and -not -name 'xxx'**

unless you prefer the cryptic syntax below (-o instead of -or)

The classic example is the use of a logical AND syntax:

find $HOME -iname '\*.ogg' -size +20M

This find search performs initially a case insensitive search for all the ogg files in your $HOME directory and for every true results it then searches for those with a size of 20 megabytes and over. This contains and implied operator which could be written joined with an *-a*. This search can be altered slightly by use of an exclamation point to signify negation of the result:

find $HOME -iname '\*.ogg' ! -size +20M

This performs the same search as before but will look for ogg files that are *not* greater than 20 megabytes. It is possible also to use a logical OR in your find search:

find $HOME -iname '\*.ogg' -o -iname '\*.mp3'

This will perform a case insensitive search in the $HOME directories and find all files that are either ogg OR mp3 files. There is great scope here for creating very complex and finely honed searches and I would encourage a through reading of the find man pages searching for the topic OPERATORS.

## **Acting On The files**

One advanced but highly useful aspect of find usage is the ability to perform a user-specified action on the result of a find search. For example the following search looks for all ogg vorbis files in the $HOME directory and then uses -exec to pass the result to the du program to give the size of each file:

find $HOME -name '\*.ogg' -type f -exec du -h '{}' \;

This syntax is often used to delete files by using *-exec rm -rf* but this must be used with great caution, if at all, as recovery of any deleted files can be quite difficult.

## **Using xargs**

**xargs <<< / ls**

same as: ls /

xargs feeds here-string **/** as parameter ("argument") to the ls command

When using a really complex search it is often a good idea to use another member of the findutils package: xargs. Without its use the message *Argument list too long* could be seen signalling that the kernel limit on the combined length of a commandline and its environment variables has been exceeded. xargs works by feeding the results of the search to the subsequent command in batches calculated on the system capabilities (based on ARG\_MAX). An example:

find /tmp -iname '\*.mp3' -print0 | xargs -0 rm

This example searches the /tmp folder for all mp3 files and then deletes them. You will note the use of both *-print0* and *xargs -0* which is a deliberate strategy to avoid problems with spaces and/or newlines in the filenames. Modern kernels do not have the ARG\_MAX limitation but to keep your searches portable it is an excellent idea to use xargs in complex searches with subsequent commands.

# **Recover Lost DiskSpace**

This guide will help users who are having issues with a lack of free disk space. The user may have received an error message such as "There is not enough room on the disk to save ..." or perhaps a message about "insufficient disk space". This page presents ways to discover what is using large portions of your disk space and how to regain some free space.

The primary focus is on restoring space on the system partition ( **/** ) but the procedures can easily be modified for other partitions as well.

# **Common Reasons for "Lost" Free Space**

* Backup files were mistakenly saved to the wrong location.
* Deleted files in the trash bin are still on the system taking up disk space.
* The user unknowingly created a large file in the wrong location.
* Downloads have accumulated in /var/cache/apt/archives.
* Various log files have increased in size and/or number.
* A cloned partition doesn't show the new partition's correct size.
* An NTFS partition shows the incorrect size.
* The contents of a lost+found folder has grown.
* An advertised 500 GB drive doesn't format to 500 GB.
* Nothing has changed, the user just needs a bit more space.
* Finally, a partition, such as / or /boot, is simply too small.

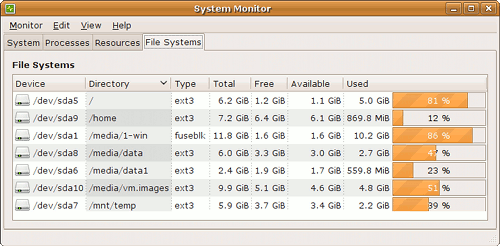
# **Tips Before You Start**

* **Mounting**. Many of the commands and applications used in this guide present information only on mounted partitions. If possible, mount only those partitions you wish to investigate. At best additional mounted devices provide extra results - at worst a mounted device might hide what is actually on the system partition.
  + To view only the system folders, first close all open applications and then run the following command. It will attempt to unmount all partitions listed in /etc/mtab. You will get messages stating "device is busy" for any system partition or partition currently being used by the system. These messages are normal and can be ignored.  
     sudo umount -a
  + To investigate other devices/partitions, you can mount all those listed in /etc/fstab with the following command. Once you have done that, you can manually mount any other devices which are not listed therein.  
     sudo mount -a
* **Permissions**. Users of Ubuntu are allowed access to files as set by permissions. These file permissions may prevent a user from seeing files created or stored in system folders and those owned by other users. If using a graphical file browser don't forget to enable hidden file viewing (usually CTRL+H). Run the commands/applications as 'root'. For command line operations, precede the command with sudo. For graphical applications such as nautilus and baobab/Disk Usage Analyzer start the command to launch the application with gksudo.
* **Deletions**. When deleting folders/files from within a file browser such as nautilus remember that the deleted folders/files are moved to the Trash bin. Until you empty the trash, these files will continue to use disk space. Use one or more of these methods to permanently remove these files:
  + In a file browser, use SHIFT-DELETE to bypass the Trash bin.
  + In terminal, use the rm command.
  + Empty the Trash bins - the user's and root's. Starting with Ubuntu 8.04 (Hardy) these are located at ~/.local/share/Trash and /root/.local/share/Trash, respectively. The user's Trash can also be emptied by right clicking on the Trash icon and selecting Empty Trash.  
     **Warning:** SHIFT-DELETE and rm cannot be easily reversed. Make sure you are deleting the correct item(s) when using these methods.
* **Formatting**. A new drive rarely will format to the advertised size. Manufacturers normally promote unformatted capacities. Expect to lose approximately 5-7% of the advertised capacity. Additionally, by default Ubuntu reserves 5% for system use. This can be altered with the tune2fs command and is discussed later.

# **Checking Your Partitions Graphically (GUI)**

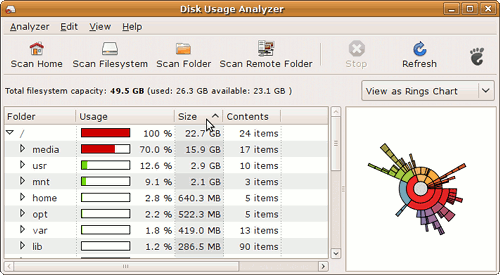
## **System Monitor**

* Open with: System > Administration > System Monitor: File Systems tab. (In terminal: gksudo gnome-system-monitor &)
* The information presented includes only currently mounted devices. Trash is considered used space.
* The System tab lists the available disk space on the system ( / ) partition.



## **Disk Usage Analyzer**

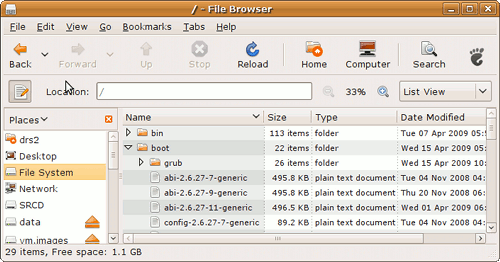
* Open with: Applications > Accessories > Disk Usage Analyzer. (In terminal: gksudo baobab &). Also known as: **baobab**
* Disk Usage Analyzer reports information only on mounted devices.



* DUA provides valuable information but often brings up questions about its use. Here are some things to keep in mind:
  + Once a scan is complete, the top entry, whether it is the system or a single folder, will always show 100%. The sub-folder percentages add up to 100%. 100% does not necessarily mean there is no space left on the partition.
  + *Total file system capacity* includes the space on all mounted devices. If you have an external drive mounted, it's contents are included in the totals.
  + *Scan Home* scans the user's home folder, or root's if baobab is opened with gksudo. "Scan FileSystem" scans the system whether or not gksudo is used. If DUA is opened without gksudo baobab & not all folders/files will be visible/reported.
  + In the top section below the menus, DUA will list the *Total file system capacity*. This total is the sum of all reported devices. If DUA sees a **/** partition of 15 GB, a data partition of 30 GB, and an external drive mounted on /media/backup of 30 GB, it will report a total of 75 GB. The available space reported is the total space available, even if one partition is completely full. This value is the total of partitions selected in Preferences. You can select or deselect a partition/device by deselecting it in the Edit > Preferences section.
  + The *Size* column entries reflect actual disk space usage (allocated space), not the apparent folder size. You can change this option via the View menu.
  + If the disk usage is approximately double the size you expected, baobab may be including the .gvfs (a virtual file system). You may be able to deselect it via the Edit > Preferences menu.
  + Running DUA as root may show different results in the folders section. The reason is that opening DUA as a normal user through the main menu will not allow you to view certain folders, such as root's deleted Trash. For the most accurate results start this application with gksudo baobab &.
  + To narrow the search, you can expand a folder to see what amount of disk space each of the sub-folders is using.

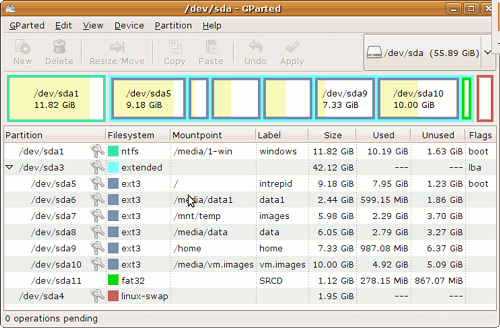
## **Nautilus**

* Open with: Places > Home Folder. (In terminal: gksudo nautilus &)
* The familiar file browser is good for reviewing folder contents and deleting specific folders/files.
* Nautilus will show mount points (although not the contents of unmounted devices) even when a device is not mounted on the mount point.
* Unless nautilus is opened with administrative privileges (gksudo nautilus &) certain folders/files such a root's Trash will not be visible. The folder may show no contents even though sub-folders or files exist.
* Right click a folder and select Properties to display the size (for files) and free space remaining on the parent partition.



## **Gparted**

* Open with: System > Administration > Partition Editor. (In terminal: gksudo gparted &)
  + gparted is a simple way of seeing how much of a partition the system thinks is used.
  + Partition > Information provides information on usage and free space percentage.



# **Checking Your Partitions via Command Line**

Access the terminal via Applications > Accessories > Terminal or use CTRL-ALT-T, or use ALT-F2 and then enter the command in the top input area.

A brief note on df and du: It is beyond the scope of this guide to go into detail about how df and du report disk usage. They look at disk space differently and the results will not be the same. Generally the results of df will show more disk usage than those of du since it considers block size rather than just the specific file size. Additionally, df's results will include information about open files - those which are not currently written to disk but do exist in memory.

Terminal command results may include /proc/ entries. These are virtual file entries, are not the cause of lost disk space, and should be disregarded.

## **df (Disk Find)**

Report file system disk space usage - check /dev/sd''XX''.

Since the df command provides details on mounted partitions you should run the df command after you have decided which devices you wish to investigate and have run the appropriate mount/umount command. By closing all applications and running sudo umount -a first, df commands are more likely to report only system partition information.

* Deleted files in Trash are included in used space until completely removed from the system.
  + df -Th  
    # Optionally: df -Th | grep -v "fs" | sort
* Sample result:

/dev/sda1 ext3 15G 7.7G 6.3G 55% /

/dev/sda2 ext3 15G 2.1G 12G 15% /media/data

/dev/sdb2 ext3 48G 35G 11G 76% /media/backup

* + Filesystem Type Size Used Avail Use% Mounted on
* Mounted NTFS partitions will be displayed as type: fuseblk.

## **du (Disk Usage)**

Estimate file space usage - check your folders.

This command will show you how much space is being used by the **mounted** file systems on a given folder. *If a device is mounted to a folder which already contains data, the underlying data size will not be included in the results*. This means that if /media/data contains 40 GB of data, but a 20 GB partition is then mounted to /media/data, the results will show only the 20 GB of the last-mounted device (because the 40 GB are now inaccessible).

* The --max-depth switch allows you to set how many sub-levels you wish to view. Once you have run the command starting at the top level ( / ) you can then investigate specific folders by replacing / with another starting point (for example, /var/log). This command reads the the disk contents when executed and thus will take a while to complete if the entire disk is to be searched. Using grep can help display only folders meeting specific criteria. Use sudo to gain access to system folders.
  + sudo du -h --max-depth=1 /  
    # *Optionally*: sudo du -h --si --max-depth=1 / | grep '[0-9]G\>'  
     The grep command reduces returns to folders 1 GB or larger (if du is run without the --si option, grep would then reduce returns to folders 1 GiB or larger).
* Folder usage is recursive - that means that the command will report the total usage of the folder and its sub-folders. Used with the right switches and run as root, it is a good tool to quickly locate files/folders which use large amounts of disk space. Once the large folders are located, investigating them with other commands or a file browser probably will provide the best results for a new user. Remember that the du command reports information on mounted partitions.
* Here is a sample of the sudo du / -h --si --max-depth=1 | grep '[0-9]G\>' command, which searches for folder/subfolders which use at least 1 GB of space:

du: cannot access `/proc/6793/fdinfo/3': No such file or directory

1.8G /media

2.8G /usr

3.1G /home

* + 8.9G /
* Note: --max-depth limits the results to one level but includes the total disk usage of the folder and all subfolders. Once you identify a specific folder to investigate, you can substitute its address for / (e.g. /usr ) and/or increase the --max-depth=1 to a larger number.

## **find (Find)**

Find a file or folder

* The find command, used with the appropriate options, is an excellent tool to locate large files.

sudo find / -name '\*' -size +1G

* + # ''Optionally'': sudo find / -name '\*' -size +500M
* An initial Ubuntu installation will not normally have files larger than 500 MB on the system partition.
* Starting with / will help ensure a thorough search of your entire system. You can specify a different starting point to speed up the search if you know which folder you want to search. Example: sudo find /var/log -name '\*' -size +1G
* The -name '\*' option is included to show the format should the user wish to replace the universal search with a specific file name.
* The difference between this command and the du command discussed previously is that this command will search only for individual files and not for folders. This would be a good choice to search for a backup file you think may have ended up in the wrong place.

# **Problems & Solutions**

## **Partition Is Too Small**

How to Find It:

* If you plan on going throught the list from start to finish a logical starting point is the actual size of your partition. A quick review of the Ubuntu System Requirements shows that for a standard Ubuntu 14.04.2 LTS desktop installation the bare minimum is 4.5 GB. A recommended minimum would be about 8 GB. Of course, you can get by with less or need much more, depending on how many apps you install. Note: If you have a separate /boot partition, the minimum size depends on how many kernels you have installed, but users who created a separate /boot partition of 50 MB often run out of room!

Backup any important files before performing any partitioning operations. If the operations involve NTFS partitions, defragment them at least once as well.

How to Fix It:

* **Take Space from Another Partition**. If you drive has another partition with extra space, reduce the size of one or more of them and expand the partition short of space. The partition must not be mounted. For system partitions unmount / and /swap. [Repartitioning](https://help.ubuntu.com/community/HowtoPartition) in Ubuntu is often accomplished using gparted, Ubuntu's default partition editor (System > Administration > Partition Editor). Since the system folder must be unmounted during resizing, gparted should be run from the LiveCD or a third party CD such as the GParted LiveCD or a SystemRescueCD.
* **Move /home**. Move parts of your system, such as /home, to another location. Here is a link on [how to move your `/home` folder](https://help.ubuntu.com/community/Partitioning/Home/Moving), as well as another on [how to move partitions in general](https://help.ubuntu.com/community/HowtoPartition/MovingPartition).
* **Move data**. Data files, music collections, video files, etc. can take up a lot of space. Creating a separate data partition on the same or another drive can provide more space for critical system files.
* Try **Xubuntu**. If you just have a small hard drive or don't have the space to dedicate to a normal Ubuntu installation try [Xubuntu](http://xubuntu.org/). It is an official Ubuntu derivative which uses less resources. The minimum space required is only 2 GB; the recommended minimum is 20 GB.

## **Kernel installation failure**

Some users created very small /boot partitions. If older kernels are not removed the partition may become full and this or a similar message may result: gzip: stdout: No space left on device update-initramfs: failed for /boot/initrd.img If the user has a separate /boot partition and/or received the previous warning, check the available space and the kernel in use:

df -Th | grep 'boot' # Only if separate /boot partition exists.

* uname -r
* To regain space in the /boot partition one of the easiest options is to remove older kernels with the Synaptic Package Manager. Open Synaptic and search for linux-image. Synaptic will display all installed kernels. Unused kernels can be safely removed; many experienced users keep at least one older kernel which they have previously used. The same kernel's linux-headers-... may also be uninstalled. Note that Ubuntu will not allow you to remove the current kernel while it is in use.

## **Back Ups Gone Wrong**

On of the most common causes of disappearing disk space is a back up saved to the wrong location. Improperly designating the back up location is a frequent cause. It can also be the result of a file saved to a mount point on which no device, such as a back up drive, is mounted.

* Example: A script calls for Simple Backup (sbackup) to back up a partition and automatically save the result to an external drive mounted on /media/backup. At the given time, the device is neither turned on nor mounted. Since the mount point exists, the file is created and saved in /media/backup, using up space on the system partition instead of on the planned external drive.

How to Find It:

For problems with your system partition, unmount all but your system folders using the sudo umount -a command discussed earlier.

* Run the following command to search for files larger than 1 GiB (find has no option for power-of-ten sizes). Disregard findings which include /proc/.
  + sudo find / -name '\*' -size +1G
* Any file found larger than 1 GiB is a likely suspect for further investigation, especially any file found on one of the user's mount points. Mount points will normally be empty unless a device is mounted to it.
* If the backup contains many smaller folders, such as a music collection, rather than look for one large file the user may wish to look at the combined folder size. In this case, run the following command. It will look for folders using more than 1 GB of space if the grep option is included.
  + sudo du -h --si --max-depth=1 / | grep '[0-9]G\>'
* Once large folders are located, you can refine the search by identifying a specific folder as the starting point and/or increase the maxdepth level. The folders /usr, /root, and /home will normally be included in the results. If all your mount points are in the /media folder, your search might be: sudo du -h --max-depth=2 /media.

How to Fix It:

* **Move the file to another partition or delete it**. If you delete it, you must use SHIFT-DEL in a file browser, the rm command via terminal, or empty your Trash (or root's for root-owned Trash) before the space is actually recovered and made available for use. Simply highlighting the item and hitting the DELETE key will move it to Trash, where it will continue to take up space. Warning: These delete methods cannot be reversed! Make sure you are deleting the correct item.
* **Amend the script**. If the problem was the result of an automatic back up, amend the script or take steps to ensure the back up device is mounted prior to executing the script.
  + Tip: To help prevent this when using sbackup, the user can select the "Abort backup if destination directory does not exist" in the "Destination" tab during setup.

## **Trash Folders Not Empty**

When a file or folder is deleted via a file browser in most cases it is not removed from the system. Instead it is placed in Trash. Until Trash is emptied, these files are recoverable and continue to take up space on the partition. There are several places on the system that deleted files are stored. It varies by which version of Ubuntu is running, the origin of the deleted folders/files, and who deleted them.

Example: A user downloads a collection of large .iso files or installation packages. Deciding not to use the files, the user deletes them using administrative privileges (as root). The user's Trash shows empty, the user can't find the files with his/her browser, but the free space isn't restored. In a different scenario, another user on the same machine deletes files but never empties Trash.

How to Find It:

* Run the following command to locate all Trash folders in the system. This will find the Trash folders of all users as well as root. The second part of the command will display the size of the located folders.
  + sudo find / -type d -name '\*Trash\*' | sudo xargs du -h | sort

How to Fix It:

* **Empty the Trash completely**. Items in Trash must be deleted in a special manner - otherwise they will be moved to the ...Trash!
  + Open a file browser with administrative privileges (gksudo nautilus &). Navigate to the Trash folder, highlight it and press SHIFT-DELETE. Using this key combination ensures the Trash folder is permanently deleted, The Trash folder contains two sub-folders, info and files. Deleting the parent Trash folder will delete both these folders. All three will be restored the next time something is deleted. Warning: These delete methods cannot be reversed! Make sure you are deleting the correct item.

## **Unexpectedly Large File**

A single 'rogue' file may be consuming large amounts of space. Use the following command to locate abnormally large files.

How to Find It:

As mentioned earlier the user can use the find command to search for large files.

* sudo find / -size +1G

How to Fix It:

Decide what the files are, if they are in the proper location, and whether or not you wish to retain them. If you delete a large file, either bypass the Trash bin with the rm command, SHIFT-DEL in nautilus, or empty the appropriate Trash bin after deleting it.

## **Too Many Collected Packages**

Packages (.deb files) downloaded for installation by a user via the system (apt, synaptic, dpkg) are stored in /var/cache/apt/archives. Over time, this folder can become quite large. Under normal circumstances it is not necessary to keep packages locally; they can be retrieved from a server if re-installation is desired.

How to Find It: To check the amount of space being used for package storage, run:

* du -h /var/cache/apt/

How to Fix It:

There are several system commands you can use to reduce or eliminate the number of locally stored packages.

* **Remove all the packages** from /var/cache/apt/archives and /var/cache/apt/archives/partial folders:
* sudo apt-get clean
* **Remove all expired packages** from /var/cache/apt/archives and /var/cache/apt/archives/partial that are no longer available for download:
* sudo apt-get autoclean  
  "Not available for download" does not mean the user should save them - normally these files have been replaced or are no longer needed.
* **Remove unneeded dependencies** which are no longer needed:
  + sudo apt-get autoremove
  + An application called [APTonCD](https://help.ubuntu.com/community/APTonCD) can save these packages to a CD/DVD if you want to preserve your .deb files prior to removing them.

## **Check Your Logs**

Log files are usually stored in /var/log and its subfolders. An excessive number of small log files or large individual logs can eat up disk space.

How to Find It:

You can run the following to see the size of each /var/log/ subfolder:

* sudo du -h /var/log

How to Fix It:

* **Change the manner** log files are generated and stored.
* **Move or delete unnecessary log files**, especially older files with more current entries. Archived \*.gz log files are safe to delete.
* **Remove the contents** of a particularly large log file if the user doesn't want to delete it. Remove the file's contents while keeping the file structure intact with the following command. Change the filename to the name of the actual file:
  + sudo cp /dev/null /var/log/log\_filename.log

## **Cloned Partitions**

When partitions are cloned to a partition of a different size (such as restoring a Partimage file to a larger partition), the partition table may need to be updated. How to Find It:

The size of a partition after cloning may reflect the original cloned partition size rather than the size of the partition into which it was restored. This is discussed in the Partimage user's manual. gparted may show the entire partition as used even when you know it isn't. How to Fix It:

Note: The following procedure is for ext2 and ext3 formatted partitions only. Change sd'''XX''' to the correct device designation (example: sd'''b3''')

# Unmount the new partition Example: sudo umount /dev/sdb3

sudo umount /dev/sdXX

# Optionally, perform a check of the partition

sudo e2fsck /dev/sdXX

# Redraw the partition table to reflect the correct partition size

sudo resize2fs -p /dev/sdXX

# Check the results

* df -h

## **NTFS Partition Is Not Correctly Sized**

No matter what the cause, it is possible an NTFS partition table fails to indicate the correct size of the partition.

How to Find It:

The size of a partition isn't what it should be. You can check the disk size by running df -Th -BGB /dev/sdXX. The command will indicate the file type (NTFS), size in GB (in GiB if -BGB is omitted or if -BG is used), used/remaining space and percentage free.

* How to Fix It:

Note: You will be warned in the final stages of the operation to backup data and ensure a reliable power source. The warning may appear a bit intimidating but merely reflects precautions which should be taken any time a partition table is altered.

* Replace /dev/sdXX with the correct partition designation. Example: Replace /dev/sdXX with /dev/sda1 if the device is /dev/sda1 When you remount the NTFS partition after resizing make sure the mount point exists. Example: /mnt/windows

sudo apt-get update

sudo apt-get install ntfsprogs # ntfsprogs is included in the 'main' repository.

sudo umount /dev/sdXX

sudo ntfsresize /dev/sdXX

sudo mount /dev/sd/XX /path/mount\_ point # Or 'sudo mount -a' if listed in fstab.

* df -Th # Check the results

## **`lost+found` Folders**

Each ext2/3/4 partition will contain a lost+found folder. This folder contains corrupted files discovered by the system during an fsck check.

How to Find It:

You can run the following to locate each lost+found folder and, optionally, see the size of each folder:

* sudo find / -name `lost+found` | sudo xargs du -h

How to Fix It:

* **Delete them**. Open a file browser with administrative powers (gksudo nautilus &) and inspect the contents of any large lost+found folder. If there is nothing recoverable, you may delete the folder. This folder will be recreated after the next fsck check.

If you delete it, you must use SHIFT-DEL in a file browser, the rm command via terminal, or empty your Trash (or root's for root-owned folders) before the space is actually recovered and made available for use. Simply highlighting the item and hitting the DELETE key will move it to Trash, where it will continue to take up space. Warning: These delete methods cannot be reversed! Make sure you are deleting the correct item.

## **Gain Just a Bit of Space**

You can gain a bit of space by following these steps. Don't expect large free space gains.

How to Fix It:

* **Computer Janitor**. Recent versions of Ubuntu come with an app which can locate unnecessary libraries and other files. Access it via System > Administration > Computer Janitor. There is another app located in Synaptic called cruft that can also locate and remove unnecessary files.
* **tune2fs**. Reduce the space reserved for system use. By default, Ubuntu reserves 5% of linux partitions for use by the operating system. Some commands and applications do not accurately reflect reserved system space (gparted/baobab). Nautilus and the df command accurately display available usable space by accounting for reserved system space.
  + The amount of partition space reserved for this can be altered with tune2fs. Replace *X* with the percentage of disk space you wish to reserve and *XX* with the device designation.  
     sudo tune2fs -m X /dev/sdXX  
    Example: sudo tune2fs -m 4 /dev/sda1 Note: There is a reason Ubuntu reserves this space. Carefully consider if you want to change this setting before doing so.
* **deborphan / gtkorphan**. Install deborphan, a command-line application, and gtkorphan, a gui-based application. These programs help find and eliminate orphaned libraries which are no longer needed.
  + sudo apt-get install deborphan gtkorphan
  + **deborphan**. This app can identify orphaned packages. To run it:
    - sudo deborphan
  + **gtkorphan**. System > Administration > Remove orphaned packages. This application will allow the user to identify and delete selected orphan libraries.
* **localepurge**. A number of language packages may be installed on the system. localepurge can be installed and run to remove and prevent future installation of extra language packages that the user does not use. *Read the man page for warnings about its use*.

## **Install Apps When the Disk is Full**

Well, *almost full*. Use this technique when you don't have the space to download apps and then install them from cache.

* Download the packages without installing them.
  + sudo apt-get upgrade -d
* Move the downloaded packages to an external source or another partition.
* Remove any remaining cached packages.
  + sudo apt-get clean
* Install the packages from the external source.

sudo dpkg -i /media/disk/\*.deb

* + # Replace /media/disk with the path to the stored packages.

## **Machine-readable debian/copyright file**

## **1. Introduction**

This document describes a standard, machine-interpretable format for the debian/copyright file. This file is one of the most important files in Debian packaging, but, prior to this specification, no standard format was defined for it and its contents varied tremendously across packages. This made it difficult to automatically extract licensing information.

Use of this specification is optional.

Nothing in this proposal supersedes or modifies any of the requirements specified in Debian Policy regarding the appropriate detail or granularity to use when documenting copyright and license status in debian/copyright.

## **2. Rationale**

The diversity of free software licenses means that Debian needs to care not only about the freeness of a given work, but also its license's compatibility with the other parts of Debian it uses.

The arrival of the GPL version 3, its incompatibility with version 2, and our inability to spot the software where the incompatibility might be problematic is one prominent occurrence of this limitation.

There are earlier precedents, also. One is the GPL/OpenSSL incompatibility. Apart from grepping debian/copyright, which is prone to numerous false positives (packaging under the GPL but software under another license) or negatives (GPL software but with an “OpenSSL special exception” dual licensing form), there is no reliable way to know which software in Debian might be problematic.

And there is more to come. There are issues with shipping GPLv2-only software with a CDDL operating system such as Nexenta. The GPL version 3 solves this issue, but not all GPL software can switch to it and we have no way to know how much of Debian should be stripped from such a system.

Even where licenses are DFSG-free and mutually compatible, users may wish for a way to identify software under certain licenses (if, for example, they have special reasons to avoid certain licenses).

## **3. Acknowledgements**

Many people have worked on this specification over the years. The following alphabetical list is incomplete; please suggest missing people: Russ Allbery, Ben Finney, Sam Hocevar, Steve Langasek, Charles Plessy, Noah Slater, Jonas Smedegaard, Lars Wirzenius.

## **4. File syntax**

The debian/copyright file must be machine-interpretable, yet human-readable, while communicating all mandated upstream information, copyright notices and licensing details.

The syntax of the file is the same as for other Debian control files, as specified in the Debian Policy Manual. See its [section 5.1](https://www.debian.org/doc/debian-policy/ch-controlfields#s-controlsyntax) for details. Extra fields can be added to any stanza. No prefixing is necessary or desired, but please avoid names similar to standard ones so that mistakes are easier to catch. Future versions of the debian/copyright specification will attempt to avoid conflicting specifications for widely used extra fields.

The file consists of two or more stanzas. At minimum, the file must include one [header stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#header-stanza) and one [Files stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#files-stanza).

There are four types of fields. The definition for each field in this document indicates which type of value it takes.

### **4.1. Single-line values**

The entire value of a single-line field must be on a single line. For example, the Format field has a single-line value specifying the version of the machine-readable format that is used.

### **4.2. Whitespace-separated lists**

Field values defined as whitespace-separated lists may be on one line or many. Values in the list are separated by one or more whitespace characters (space, tab, or newline). For example, the Files field contains a whitespace-separated list of filename patterns.

### **4.3. Line-based lists**

Line-based lists have one value per line. For example, the Upstream-Contact field contains a line-based list of contact addresses.

### **4.4. Formatted text**

Formatted text fields use the same rules as the long description in a package's Description field in Debian control files.

In some but not all cases, the first line may have special meaning as a synopsis, similar to how the Description field uses the first line for the short description. See Debian Policy's section 5.6.13, [“Description”](https://www.debian.org/doc/debian-policy/ch-controlfields#s-f-Description), for details. For example, Disclaimer is a formatted text field that has no special first line, and License is a formatted text field where the first line indicates the short name or names of the licenses.

## **5. Stanzas**

There are three kinds of stanzas. The first stanza in the file is called the [header stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#header-stanza). Every other stanza is either a [Files stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#files-stanza) or a [stand-alone License stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#stand-alone-license-stanza). This is similar to source and binary package stanzas in debian/control files.

### **5.1. Header stanza (once)**

The following fields may be present in a header stanza.

* [Format](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#format-field): required.
* [Upstream-Name](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#upstream-name-field): optional.
* [Upstream-Contact](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#upstream-contact-field): optional.
* [Source](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#source-field): optional.
* [Disclaimer](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#disclaimer-field): optional.
* [Comment](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#comment-field): optional.
* [License](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#license-field): optional.
* [Copyright](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#copyright-field): optional.

The Copyright and License fields in the *header stanza* may complement but do not replace the fields in the *Files stanzas*. If present, they summarise the copyright notices or redistribution terms for the package as a whole.

For example, when a work has a grant of license under both a permissive and a copyleft license, License can be used to clarify the license terms for the combination. Copyright and License together can also be used to document a *compilation copyright* and license.

It is valid to use License in the header stanza without an accompanying Copyright field, but Copyright alone is not sufficient.

#### **5.1.1. Example header stanza**

Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

Source: https://www.example.com/software/project

Upstream-Name: SOFTware

Upstream-Contact: John Doe <john.doe@example.com>

### **5.2. Files stanza (repeatable)**

The declaration of copyright and license for files may consist of one or more stanzas. In the simplest case, a single stanza with Files: \* can be used to state the license and copyright for the whole package. Only the license and copyright information required by the Debian archive is required to be listed here.

The following fields may be present in a Files stanza.

* [Files](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#files-field): required.
* [Copyright](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#copyright-field): required.
* [License](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#license-field): required.
* [Comment](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#comment-field): optional.

#### **5.2.1. Example files stanzas**

Files:

\*

Copyright: 1975-2010 Ulla Upstream

License: GPL-2+

Files:

debian/\*

Copyright: 2010 Daniela Debianizer

License: GPL-2+

Files:

debian/patches/fancy-feature

Copyright: 2010 Daniela Debianizer

License: GPL-3+

Files:

\*/\*.1

Copyright: 2010 Manuela Manpager

License: GPL-2+

In this example, copyright in all files is held by the upstream, and that copyright holder grants license under the GPL, version 2 or later. There are three exceptions. All the Debian packaging files have copyright held by the packager, and further one specific file providing a new feature has a different grant of license. Finally, there are some manual pages added to the package, with copyright held by a third person.

Since the license of the manual pages is the same as most other files in the package, the final stanza above could instead be combined with the first stanza, listing both copyright statements in one Copyright field. Whether to combine stanzas with the same grant of license is left to the discretion of the author of the debian/copyright file.

### **5.3. Stand-alone License Stanza (optional, repeatable)**

Stand-alone License stanzas can be used to provide the full license text for a given license once, instead of repeating it in each Files stanza that refers to it.

The synopsis (on the first line) of the License field must be a single license short name or a short name followed by a license exception.

The following fields may be present in a stand-alone License stanza.

* [License](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#license-field): required.
* [Comment](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#comment-field): optional.

**Example 1. tri-licensed files**

Files: src/js/editline/\*

Copyright: 1993, John Doe

1993, Joe Average

License: MPL-1.1 or GPL-2 or LGPL-2.1

License: MPL-1.1

[LICENSE TEXT]

License: GPL-2

[LICENSE TEXT]

License: LGPL-2.1

[LICENSE TEXT]

**Example 2. recurrent license**

Files:

src/js/editline/\*

Copyright: 1993, John Doe

1993, Joe Average

License: MPL-1.1

Files:

src/js/fdlibm/\*

Copyright: 1993, J-Random Corporation

License: MPL-1.1

License: MPL-1.1

[LICENSE TEXT]

## **6. Fields**

The following fields are defined for use in debian/copyright.

### **6.1. Format**

Single-line: URI of the format specification. The field that should be used for the current version of this document is:

Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

The original version of this specification used the non-https version of this URL as its URI, namely:

Format: http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

Both versions are valid and refer to the same specification, and parsers should interpret both as referencing the same format. The https URI is preferred.

### **6.2. Upstream-Name**

Single-line: the name upstream uses for the software

### **6.3. Upstream-Contact**

Line-based list: the preferred address(es) to reach the upstream project. May be free-form text, but by convention will usually be written as a list of RFC5322 addresses or URIs.

### **6.4. Source**

Formatted text, no synopsis: an explanation of where the upstream source came from. Typically this would be a URL, but it might be a free-form explanation. The Debian Policy section [12.5](https://www.debian.org/doc/debian-policy/ch-docs#s-copyrightfile) requires this information unless there are no upstream sources, which is mainly the case for native Debian packages. If the upstream source has been modified to remove non-free parts, that should be explained in this field.

### **6.5. Disclaimer**

Formatted text, no synopsis: this field is used for non-free or contrib packages to state that they are not part of Debian and to explain why (see Debian Policy section [12.5](https://www.debian.org/doc/debian-policy/ch-docs#s-copyrightfile)).

### **6.6. Comment**

Formatted text, no synopsis: this field can provide additional information. For example, it might quote an e-mail from upstream justifying why the license is acceptable to the main archive, or an explanation of how this version of the package has been forked from a version known to be DFSG-free, even though the current upstream version is not.

### **6.7. License**

Formatted text, with synopsis.

In the [header stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#header-stanza), this field gives the license information for the package as a whole, which may be different or simplified from a combination of all the per-file license information. In a [Files stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#files-stanza), this field gives the licensing terms for the files listed in the Files field for this stanza. In a [stand-alone License stanza](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#stand-alone-license-stanza), it gives the licensing terms for those stanzas which reference it.

First line (synopsis): an abbreviated name for the license, or expression giving alternatives (see the [Short name](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#license-short-name) section for a list of standard abbreviations). If there are licenses present in the package without a standard short name, an arbitrary short name may be assigned for these licenses. These arbitrary names are only guaranteed to be unique within a single copyright file.

If there are no remaining lines, then all of the short names or short names followed by license exceptions in the synopsis must be described in [stand-alone License stanzas](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#stand-alone-license-stanza). Otherwise, this field should either include the full text of the license(s) or include a pointer to the license file under /usr/share/common-licenses. This field should include all text needed in order to fulfill both Debian Policy's requirement for including a copy of the software's distribution license ([12.5](https://www.debian.org/doc/debian-policy/ch-docs#s-copyrightfile)), and any license requirements to include warranty disclaimers or other notices with the binary package.

### **6.8. Copyright**

Formatted text, no synopsis: one or more free-form copyright statements. Any formatting is permitted; see the examples below for some ideas for how to structure the field to make it easier to read. In the header stanza, this field gives the copyright information for the package as a whole, which may be different or simplified from a combination of all the per-file copyright information. In the Files stanzas, it gives the copyright information that applies to the files matched by the Files pattern. If a work has no copyright holder (i.e., it is in the public domain), that information should be recorded here.

The Copyright field collects all relevant copyright notices for the files of this stanza. Not all copyright notices may apply to every individual file, and years of publication for one copyright holder may be gathered together. For example, if file A has:

Copyright 2008 John Smith

Copyright 2009 Angela Watts

and file B has:

Copyright 2010 Angela Watts

a single stanza may still be used for both files. The Copyright field for that stanza would contain:

Copyright 2008 John Smith

Copyright 2009, 2010 Angela Watts

The Copyright field may contain the original copyright statement copied exactly (including the word “Copyright”), or it may shorten the text or merge it with other copyright statements as described above, as long as it does not sacrifice information. Examples in this specification use both forms.

### **6.9. Files**

Whitespace-separated list: list of patterns indicating files covered by the license and copyright specified in this stanza.

Filename patterns in the Files field are specified using a simplified shell glob syntax. Patterns are separated by whitespace.

* Only the wildcards \* and ? apply; the former matches any number of characters (including none), the latter a single character. Both match slashes (/) and leading dots, unlike shell globs. The pattern \*.in therefore matches any file whose name ends in .in anywhere in the source tree, not just at the top level.
* Patterns match pathnames that start at the root of the source tree. Thus, “Makefile.in” matches only the file at the root of the tree, but “\*/Makefile.in” matches at any depth.
* The backslash (\) is used to remove the magic from the next character; see table below.

| **Escape sequence** | **Matches** |
| --- | --- |
| \\* | star (asterisk) |
| \? | question mark |
| \\ | backslash |

Any other character following a backslash is an error.

This is the same pattern syntax as fnmatch(3) without the FNM\_PATHNAME flag, or the argument to the -path test of the GNU **find** command, except that [] wildcards are not recognized.

Multiple Files stanzas are allowed. The last stanza that matches a particular file applies to it. More general stanzas should therefore be given first, followed by more specific overrides.

Exclusions are only supported by adding Files stanzas to override the previous match.

This syntax does not distinguish file names from directory names; a trailing slash in a pattern will never match any actual path. A whole directory tree may be selected with a pattern like "foo/\*".

The space character, used to separate patterns, cannot be escaped with a backslash. A path like "foo bar" may be selected with a pattern like "foo?bar".

## **7. License specification**

### **7.1. Short name**

Much of the value of a machine-parseable copyright file lies in being able to correlate the licenses of multiple pieces of software. To that end, this spec defines standard short names for a number of commonly used licenses, which can be used in the synopsis (first line) of a License field.

These short names have the specified meanings across all uses of this file format, and *must not* be used to refer to any other licenses. Parsers may thus rely on these short names referring to the same licenses wherever they occur, without needing to parse or compare the full license text.

From time to time, licenses may be added to or removed from the list of standard short names. Such changes in the list of short names will always be accompanied by changes to the version of this standard and to the recommended Format value. Implementers who are parsing copyright files should take care not to assume anything about the meaning of license short names for unknown Format versions.

Use of a standard short name does not override the Debian Policy requirement to include the full license text in debian/copyright, nor any requirements in the license of the work regarding reproduction of legal notices. This information must still be included in the License field, either in a stand-alone License stanza or in the relevant files stanza.

For licenses that have multiple versions in use, the short name is formed from the general short name of the license family, followed by a dash and the version number. If the version number is omitted, the lowest version number is implied. When the license grant permits using the terms of any later version of that license, add a plus sign to the end of the short name. For example, the short name GPL refers to the GPL version 1 and is equivalent to GPL-1, although the latter is clearer and therefore preferred. If the package may be distributed under the GPL version 1 or any later version, use a short name of GPL-1+.

For [SPDX](https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/#spdx) compatibility, versions with trailing *dot-zeroes* are considered to be equivalent to versions without (e.g., “2.0.0” is considered equal to “2.0” and “2”).

Currently, the full text of the licenses is only available in the [SPDX Open Source License Registry](https://spdx.org/licenses).

| **Keyword** | **Meaning** |
| --- | --- |
| public-domain | No license required for any purpose; the work is not subject to copyright in any jurisdiction. |
| Apache | Apache license [1.0](https://spdx.org/licenses/Apache-1.0), [2.0](https://spdx.org/licenses/Apache-2.0). |
| Artistic | Artistic license [1.0](https://spdx.org/licenses/Artistic-1.0), [2.0](https://spdx.org/licenses/Artistic-2.0). |
| BSD-2-clause | Berkeley software distribution license, [2-clause version](https://spdx.org/licenses/BSD-2-Clause). |
| BSD-3-clause | Berkeley software distribution license, [3-clause version](https://spdx.org/licenses/BSD-3-Clause). |
| BSD-4-clause | Berkeley software distribution license, [4-clause version](https://spdx.org/licenses/BSD-4-Clause). |
| ISC | [Internet Software Consortium](https://spdx.org/licenses/ISC), sometimes also known as the OpenBSD License. |
| CC-BY | Creative Commons Attribution license [1.0](https://spdx.org/licenses/CC-BY-1.0), [2.0](https://spdx.org/licenses/CC-BY-2.0), [2.5](https://spdx.org/licenses/CC-BY-2.5), [3.0](https://spdx.org/licenses/CC-BY-3.0). |
| CC-BY-SA | Creative Commons Attribution Share Alike license [1.0](https://spdx.org/licenses/CC-BY-SA-1.0), [2.0](https://spdx.org/licenses/CC-BY-SA-2.0), [2.5](https://spdx.org/licenses/CC-BY-SA-2.5), [3.0](https://spdx.org/licenses/CC-BY-SA-3.0). |
| CC-BY-ND | Creative Commons Attribution No Derivatives license [1.0](https://spdx.org/licenses/CC-BY-ND-1.0), [2.0](https://spdx.org/licenses/CC-BY-ND-2.0), [2.5](https://spdx.org/licenses/CC-BY-ND-2.5), [3.0](https://spdx.org/licenses/CC-BY-ND-3.0). |
| CC-BY-NC | Creative Commons Attribution Non-Commercial license [1.0](https://spdx.org/licenses/CC-BY-NC-1.0), [2.0](https://spdx.org/licenses/CC-BY-NC-2.0), [2.5](https://spdx.org/licenses/CC-BY-NC-2.5), [3.0](https://spdx.org/licenses/CC-BY-NC-3.0). |
| CC-BY-NC-SA | Creative Commons Attribution Non-Commercial Share Alike license [1.0](https://spdx.org/licenses/CC-BY-NC-SA-1.0), [2.0](https://spdx.org/licenses/CC-BY-NC-SA-2.0), [2.5](https://spdx.org/licenses/CC-BY-NC-SA-2.5), [3.0](https://spdx.org/licenses/CC-BY-NC-SA-3.0). |
| CC-BY-NC-ND | Creative Commons Attribution Non-Commercial No Derivatives license [1.0](https://spdx.org/licenses/CC-BY-NC-ND-1.0), [2.0](https://spdx.org/licenses/CC-BY-NC-ND-2.0), [2.5](https://spdx.org/licenses/CC-BY-NC-ND-2.5), [3.0](https://spdx.org/licenses/CC-BY-NC-ND-3.0). |
| CC0 | Creative Commons Zero [1.0 Universal](https://spdx.org/licenses/CC0-1.0). Omit “Universal” from the license version when forming the short name. |
| CDDL | Common Development and Distribution License [1.0](https://spdx.org/licenses/CDDL-1.0). |
| CPL | [Common Public License](https://spdx.org/licenses/CPL-1.0). |
| EFL | The Eiffel Forum License [1.0](https://spdx.org/licenses/EFL-1.0), [2.0](https://spdx.org/licenses/EFL-2.0). |
| Expat | The [Expat](http://www.jclark.com/xml/copying.txt) license. |
| GPL | GNU General Public License [1.0](https://spdx.org/licenses/GPL-1.0), [2.0](https://spdx.org/licenses/GPL-2.0), [3.0](https://spdx.org/licenses/GPL-3.0). |
| LGPL | GNU Lesser General Public License [2.1](https://spdx.org/licenses/LGPL-2.1), [3.0](https://spdx.org/licenses/LGPL-3.0), or GNU Library General Public License [2.0](https://spdx.org/licenses/LGPL-2.0). |
| GFDL | GNU Free Documentation License 1.0, [1.1](https://spdx.org/licenses/GFDL-1.1), [1.2](https://spdx.org/licenses/GFDL-1.2), or [1.3](https://spdx.org/licenses/GFDL-1.3). Use GFDL-NIV instead if there are no Front-Cover or Back-Cover Texts or Invariant Sections. |
| GFDL-NIV | GNU Free Documentation License, with no Front-Cover or Back-Cover Texts or Invariant Sections. Use the same version numbers as GFDL. |
| LPPL | [LaTeX Project Public License](https://www.latex-project.org/lppl/) [1.0](https://spdx.org/licenses/LPPL-1.0), [1.1](https://spdx.org/licenses/LPPL-1.1), [1.2](https://spdx.org/licenses/LPPL-1.2), [1.3c](https://spdx.org/licenses/LPPL-1.3c). |
| MPL | Mozilla Public License [1.1](https://spdx.org/licenses/MPL-1.1). |
| Perl | [Perl](https://dev.perl.org/licenses/) license (use “GPL-1+ or Artistic-1” instead). |
| Python | Python license [2.0](https://spdx.org/licenses/Python-2.0). |
| QPL | Q Public License [1.0](https://spdx.org/licenses/QPL-1.0). |
| W3C | [W3C Software License](https://spdx.org/licenses/W3C) For more information, consult the [W3C Intellectual Rights FAQ](https://www.w3.org/Consortium/Legal/IPR-FAQ-20000620). |
| Zlib | [zlib/libpng license](https://spdx.org/licenses/Zlib). |
| Zope | Zope Public License 1.0, [1.1](https://spdx.org/licenses/ZPL-1.1), [2.0](https://spdx.org/licenses/ZPL-2.0), [2.1](https://spdx.org/licenses/ZPL-2.1). |

There are [many versions of the MIT license](https://en.wikipedia.org/wiki/MIT_License#Various_versions). Please use Expat instead, when it matches.

An exception or clarification to a license is signalled in plain text, by appending with *keywords* exception to the short name. This document provides a list of keywords that must be used when referring to the most frequent exceptions. When exceptions other than these are in effect that modify a common license by granting additional permissions, you may use an arbitrary keyword not taken from the below list of keywords. When a license differs from a common license because of added restrictions rather than because of added permissions, a distinct short name should be used instead of with *keywords* exception.

Only one exception may be specified for each license within a given license specification. If more than one exception applies to a single license, an arbitrary short name indicating that combination of multiple exceptions must be used instead.

The GPL Font exception refers to the text added to the license notice of each file as specified at [How does the GPL apply to fonts](https://www.gnu.org/licenses/gpl-faq#FontException). The precise text corresponding to this exception is:

As a special exception, if you create a document which uses this font,

and embed this font or unaltered portions of this font into the

document, this font does not by itself cause the resulting document to

be covered by the GNU General Public License. This exception does not

however invalidate any other reasons why the document might be covered

by the GNU General Public License. If you modify this font, you may

extend this exception to your version of the font, but you are not

obligated to do so. If you do not wish to do so, delete this exception

statement from your version.

The GPL OpenSSL exception gives permission to link GPL-licensed code with the OpenSSL library, which contains GPL-incompatible clauses. For more information, see [The OpenSSL License and The GPL](https://www.gnome.org/~markmc/openssl-and-the-gpl) by Mark McLoughlin and the message [middleman software license conflicts with OpenSSL](https://lists.debian.org/debian-legal/2004/05/msg00595.html) by Mark McLoughlin on the *debian-legal* mailing list. The text corresponding to this exception is:

In addition, as a special exception, the copyright holders give

permission to link the code of portions of this program with the

OpenSSL library under certain conditions as described in each

individual source file, and distribute linked combinations including

the two.

You must obey the GNU General Public License in all respects for all

of the code used other than OpenSSL. If you modify file(s) with this

exception, you may extend this exception to your version of the

file(s), but you are not obligated to do so. If you do not wish to do

so, delete this exception statement from your version. If you delete

this exception statement from all source files in the program, then

also delete it here.

#### **7.1.1. Public domain**

The License short name public-domain does not refer to a set of license terms. There are some works which are not subject to copyright in any jurisdiction and therefore no license is required for any purpose covered by copyright law. This short name is an explicit declaration that the associated files are “in the public domain”.

Widespread misunderstanding about copyright in general, and the public domain in particular, results in the common assertion that a work is in the public domain when this is partly or wholly untrue for that work. The [Wikipedia article on public domain](https://en.wikipedia.org/wiki/Public_domain) is a useful reference for this subject.

When the License field in a stanza has the short name public-domain, the remaining lines of the field *must* explain exactly what exemption the corresponding files for that stanza have from default copyright restrictions.

### **7.2. Syntax**

License names are case-insensitive, and may not contain spaces.

In case of multi-licensing, the license short names are separated by or when the user can chose between different licenses, and by and when use of the work must simultaneously comply with the terms of multiple licenses.

For instance, this is a simple, “GPL version 2 or later” field:

License: GPL-2+

This is a dual-licensed GPL/Artistic work such as Perl:

License: GPL-1+ or Artistic

This is for a file that has both GPL and classic BSD code in it:

License: GPL-2+ and BSD-3-clause

For the most complex cases, a comma is used to disambiguate the priority of ors and ands. The conjunction “and” has priority over “or” unless preceded by a comma. For instance:

A or B and C means A or (B and C).

A or B, and C means (A or B) and C.

This is for a file that has Perl code and classic BSD code in it:

License: GPL-2+ or Artistic-2.0, and BSD-3-clause

A GPL-2+ work with the OpenSSL exception is in effect a dual-licensed work that can be redistributed either under the GPL-2+, or under the GPL-2+ with the OpenSSL exception. It is thus expressed as GPL-2+ with OpenSSL exception. A possible License field for such a license is:

License: GPL-2+ with OpenSSL exception

This program is free software; you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation; either version 2 of the License, or

(at your option) any later version.

.

In addition, as a special exception, the author of this program gives

permission to link the code of its release with the OpenSSL project's

"OpenSSL" library (or with modified versions of it that use the same

license as the "OpenSSL" library), and distribute the linked executables.

You must obey the GNU General Public License in all respects for all of

the code used other than "OpenSSL". If you modify this file, you may

extend this exception to your version of the file, but you are not

obligated to do so. If you do not wish to do so, delete this exception

statement from your version.

.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

.

You should have received a copy of the GNU General Public License

along with this package; if not, see <https://www.gnu.org/licenses/>.

Comment:

On Debian systems, the full text of the GNU General Public License

version 2 can be found in the file '/usr/share/common-licenses/GPL-2'.

### **7.3. SPDX**

[SPDX](https://spdx.org/) is an attempt to standardize a format for communicating the components, licenses and copyrights associated with a software package. It and the machine-readable debian/copyright format attempt to be somewhat compatible. However, the two formats have different aims, and so the formats are different. The [DEP5 wiki page](https://wiki.debian.org/Proposals/CopyrightFormat) will be used to track the differences.

## **8. Examples**

**Example 3. Simple**

A possible debian/copyright file for an program “X Solitaire” distributed in the Debian source package xsol (this is not a complete or correct copyright file for the actual xsol package):

Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/

Source: ftp://ftp.example.com/pub/games

Upstream-Name: X Solitaire

Files:

\*

Copyright: 1998 John Doe <jdoe@example.com>

1998 Jane Smith <jsmith@example.net>

License: GPL-2+

This program is free software; you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation; either version 2 of the License, or

(at your option) any later version.

.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

.

You should have received a copy of the GNU General Public License

along with this package; if not, see &lt;https://www.gnu.org/licenses/&gt;.

Comment:

On Debian systems, the full text of the GNU General Public License

version 2 can be found in the file '/usr/share/common-licenses/GPL-2'.

# **Viewing and monitoring log files**

## **2. Log files locations**

There are many different log files that all serve different purposes. When trying to find a log about something, you should start by identifying the most relevant file. Below is a list of common log file locations.

### [**System logs**](https://ubuntu.com/tutorials/viewing-and-monitoring-log-files#system-logs)

System logs deal with exactly that - the Ubuntu system - as opposed to extra applications added by the user. These logs may contain information about authorizations, system daemons and system messages.

#### **Authorization log**

Location: /var/log/auth.log

Keeps track of authorization systems, such as password prompts, the sudo command and remote logins.

#### **Daemon Log**

Location: /var/log/daemon.log

Daemons are programs that run in the background, usually without user interaction. For example, display server, SSH sessions, printing services, bluetooth, and more.

#### **Debug log**

Location: /var/log/debug

Provides debugging information from the Ubuntu system and applications.

#### **Kernel log**

Location: /var/log/kern.log

Logs from the Linux kernel.

#### **System log**

Location: /var/log/syslog

Contains more information about your system. If you can’t find anything in the other logs, it’s probably here.

### [**Application logs**](https://ubuntu.com/tutorials/viewing-and-monitoring-log-files#application-logs)

Some applications also create logs in /var/log. Below are some examples.

#### **Apache logs**

Location: /var/log/apache2/ (subdirectory)

Apache creates several log files in the /var/log/apache2/ subdirectory. The access.log file records all requests made to the server to access files. error.log records all errors thrown by the server.

#### **X11 server logs**

Location: /var/log/Xorg.0.log

The X11 server creates a seperate log file for each of your displays. Display numbers start at zero, so your first display (display 0) will log to Xorg.0.log. The next display (display 1) would log to Xorg.1.log, and so on.

### [**Non-human-readable logs**](https://ubuntu.com/tutorials/viewing-and-monitoring-log-files#non-human-readable-logs)

Not all log files are designed to be read by humans. Some were made to be parsed by applications. Below are some of examples.

#### **Login failures log**

Location: /var/log/faillog

Contains info about login failures. You can view it with the faillog command.

#### **Last logins log**

Location: /var/log/lastlog

Contains info about last logins. You can view it with the lastlog command.

#### **Login records log**

Location: /var/log/wtmp

Contains login info used by other utilities to find out who’s logged in. To view currently logged in users, use the who command.

# **Installing A New HardDrive**

While it's not every day that you need to add a new hard drive to your computer, the task does not have to be complicated. Use this guide to help you install a new hard drive with an existing Ubuntu system, and partition it for use. Before beginning, you need to consider for what you will be using the hard drive.

* Will the drive be used only with Ubuntu?
* Will the drive need to be accessible from both Ubuntu and Windows?
* How do you want to divide the free space? As a single partition, or as several?
* Do you want any of the partitions to be larger than 2 TB?

This guide goes over procedures for a single partition drive install only. Multiple partition drive installations are not very hard, and you may very well figure it out by using this guide; however, make sure you add an entry in /etc/fstab for each partition, not just the drive.

**A Note about File Systems:**

Drives that are going to be used only under Ubuntu should be formatted using the ext3/ext4 file system (depending on which version of Ubuntu you use and whether you need Linux backwards compatibility). For sharing between Ubuntu and Windows, FAT32 is often the recommended file system, although NTFS works quite well too. If you are new to file systems and partitioning, please do some preliminary research on the two before you attempt this procedure.

# **Determine Drive Information**

We assume that the hard drive is physically installed and detected by the BIOS.

To determine the path that your system has assigned to the new hard drive, open a terminal and run:

sudo lshw -C disk

IconsPage/example.png This should produce output similar to this sample:

\*-disk

description: ATA Disk

product: IC25N040ATCS04-0

vendor: Hitachi

physical id: 0

bus info: ide@0.0

logical name: /dev/sdb

version: CA4OA71A

serial: CSH405DCLSHK6B

size: 37GB

capacity: 37GB

Be sure to note the "logical name" entry, as it will be used several times throughout this guide.

# **Partition The Disk**

If you have already formatted the drive and it contains data, skip this step and move on to "Mount Point." If the drive is still blank and unformatted, then you have two options: formatting the drive using the command line, or installing GParted for a graphical approach. Decide whether you want the drive to contain one single partition, or if you want to divide the space up between two or more partitions.

## **Partitioning Using GParted**

If System > Administration > GNOME Partition Editor (or 'Partition Editor') is not available, install "GParted" using "sudo apt-get install gparted" from the command line, "Add/Remove Software" (or "Add/Remove...") from the Applications menu, or "Synaptic Package Manager" from the System > Administration menu. Open GParted and let's get started.

gksudo gparted

IconsPage/note.png Always use gksu or gksudo for graphical applications like **gparted** and sudo for command line applications, like **apt-get**.

In the top-right corner of the window, choose your new hard drive from the drop-down list, referring back to the "logical name" from earlier. The window should refresh and show you a representation of the new drive. Assuming that the drive has yet to have been used, a white bar will run across the window. Use these steps to partition the drive with a single partition...

1) Right-click on the white bar and choose "New."

2) For "New Size" the number should be the maximum allowable, to fill the entire disk.

3) Choose "Primary Partition"

4) Now decide on a filesystem. Use "ext3" if the drive will only be used with Ubuntu. For file-sharing between Ubuntu and Windows, you should use "fat32." If you are unsure, search around the wiki and forums for advice.

5) Now click Add to compute the partition. The graphical display should update to show a new partition covering the entire disk.

6) To finish, click "Apply," or Edit > Apply. The disk will then be partitioned and formatted. You may now close GParted.

## **Command Line Partitioning**

There are two commands that can be used in the command line to partition a new drive: fdisk and parted. fdisk is an older program, and its main downside is that it can only create MBR partitions. parted allows you to create MBR or GPT partitions.

### **GPT vs MBR**

MBR (Master Boot Record) has two main limitations: you cannot have a partition larger than 2 TB and you cannot have more than 4 primary partitions. GPT (GUID Partition Table) can do both of these things, but it is part of the EFI standard. This means your kernel must support EFI. The latest version of the kernel supports EFI, and almost all the latest distros do too.

See this tutorial for information about EFI support and parted usage:<https://www.cyberciti.biz/tips/fdisk-unable-to-create-partition-greater-2tb.html>

### **parted**

Refer back to the logical name you noted from earlier. For illustration, I'll use /dev/sdb, and assume that you want a single partition on the disk, occupying all the free space.

1) Start parted as follows:

sudo parted /dev/sdb

2) Create a new GPT disklabel (aka partition table):

(parted) mklabel gpt

3) Set the default unit to TB:

(parted) unit TB

4) Create one partition occupying all the space on the drive. For a 4TB drive:

(parted) mkpart

Partition name? []? primary

File system type? [ext2]? ext4

Start? 0

End? 4

Alternatively, you can set the partition size as a percentage of the disk. To create a partition occupying all the space on the drive:

(parted) mkpart

Partition name? []? primary

File system type? [ext2]? ext4

Start? 0%

End? 100%

5) Check that the results are correct:

(parted) print

There should be one partition occupying the entire drive.

6) Save and quit "parted":

(parted) quit

### **fdisk**

Refer back to the logical name you noted from earlier. For illustration, I'll use /dev/sdb, and assume that you want a single partition on the disk, occupying all the free space.

If the number of cylinders in the disk is larger than 1024 (and large hard drives always have more), it could, in certain setups, cause problems with:

1. software that runs at boot time (e.g., old versions of LILO)
2. booting and partitioning software from other OSs (e.g., DOS FDISK, OS/2 FDISK)

Otherwise, this will not negatively affect you.

1) Initiate fdisk with the following command:

* sudo fdisk /dev/sdb

2) Fdisk will display the following menu:

Command (m for help): m <enter>

Command action

a toggle a bootable flag

b edit bsd disklabel

c toggle the dos compatibility flag

d delete a partition

l list known partition types

m print this menu

n add a new partition

o create a new empty DOS partition table

p print the partition table

q quit without saving changes

s create a new empty Sun disklabel

t change a partition's system id

u change display/entry units

v verify the partition table

w write table to disk and exit

x extra functionality (experts only)

* Command (m for help):

3) We want to add a new partition. Type "n" and press enter.

Command action

e extended

p primary partition (1-4)

4) We want a primary partition. Enter "p" and enter.

Partition number (1-4):

5) Since this will be the only partition on the drive, number 1. Enter "1" and enter.

Command (m for help):

If it asks about the first cylinder, just type "1" and enter. (We are making 1 partition to use the whole disk, so it should start at the beginning.)

6) Now that the partition is entered, choose option "w" to write the partition table to the disk. Type "w" and enter.

The partition table has been altered!

7) If all went well, you now have a properly partitioned hard drive that's ready to be formatted. Since this is the first partition, Linux will recognize it as /dev/sdb1, while the **disk** that the partition is on is still /dev/sdb.

## **Command Line Formatting**

To format the new partition as ext4 file system (best for use under Ubuntu):

* sudo mkfs -t ext4 /dev/sdb1

To format the new partition as fat32 file system (best for use under Ubuntu & Windows):

* sudo mkfs -t fat32 /dev/sdb1

As always, substitute "/dev/sdb1" with your own partition's path.

# **Modify Reserved Space (Optional)**

When formatting the drive as ext2/ext3, 5% of the drive's total space is reserved for the super-user (root) so that the operating system can still write to the disk even if it is full. However, for disks that only contain data, this is not necessary.

NOTE: You may run this command on a fat32 file system, but it will do nothing; therefore, I highly recommend not running it.

You can adjust the percentage of reserved space with the "tune2fs" command, like this:

sudo tune2fs -m 1 /dev/sdb1

This example reserves 1% of space - change this number if you wish.

* (i) Using this command does not change any existing data on the drive. You can use it on a drive which already contains data.

# **Create A Mount Point**

Now that the drive is partitioned and formatted, you need to choose a mount point. This will be the location from which you will access the drive in the future. I would recommend using a mount point with "/media", as it is the default used by Ubuntu. For this example, we'll use the path "/media/mynewdrive"

* sudo mkdir /media/mynewdrive

Now we are ready to mount the drive to the mount point.

# **Mount The Drive**

You can choose to have the drive mounted automatically each time you boot the computer, or manually only when you need to use it.

## **Automatic Mount At Boot**

Note: Ubuntu now recommends to use UUID instead, see the instructions here:<https://help.ubuntu.com/community/UsingUUID>

You'll need to edit /etc/fstab:

* gksu gedit /etc/fstab

or in terminal:

* sudo nano -Bw /etc/fstab

Note:<https://help.ubuntu.com/community/Fstab#Editing_fstab>

Add this line to the end (for ext3 file system):

* /dev/sdb1 /media/mynewdrive ext3 defaults 0 2

Add this line to the end (for fat32 file system):

* /dev/sdb1 /media/mynewdrive vfat defaults 0 2  
  The defaults part may allow you to read, but not write. To write other partition and FAT specific options must be used. If gnome nautilus is being used, use the right-click, mount method, from computer folder. Then launch the mount command from terminal, no options. The last entry should be the FAT drive and and look something like:  
   /dev/sda5 on /media/mynewdrive type vfat (rw,nosuid,nodev,uhelper=hal,shortname=mixed,uid=1000,utf8,umask=077,flush)  
  All of the parts between the parenthesis are the mount options and should replace "defaults" in the fstab file. The "2" at the end instructs your system to run a quick file system check on the hard drive at every boot. Changing it to "0" will skip this. Run 'man fstab' for more info here.

You can now run "sudo mount -a" (or reboot the computer) to have the changes take effect.

If you want to allow a normal user to create files on this drive, you can either give this user ownership of the top directory of the drive filesystem: (replace **USERNAME** with the username)

* sudo chown -R USERNAME:USERNAME /media/mynewdrive

or in a more flexible way, practical if you have several users, allow for instance the users in the plugdev group (usually those who are meant to be able to mount removable disks, desktop users) to create files and sub-directories on the disk:

sudo chgrp plugdev /media/mynewdrive

sudo chmod g+w /media/mynewdrive

* sudo chmod +t /media/mynewdrive

The last "chmod +t" adds the sticky bit, so that people can only delete their own files and sub-directories in a directory, even if they have write permissions to it (see man chmod).

## **Manually Mount**

Alternatively, you may want to manually mount the drive every time you need it.

For manual mounting, use the following command:

sudo mount /dev/sdb1 /media/mynewdrive

When you are finished with the drive, you can unmount it using:

sudo umount /media/mynewdrive

# **Mounting Partitions Automatically**

There are broadly two aproaches -

* [Per-user mounting](https://help.ubuntu.com/community/AutomaticallyMountPartitions#Per-User_Mounts) (usually under /media)
* [Systemwide mounting](https://help.ubuntu.com/community/AutomaticallyMountPartitions#Systemwide_Mounts) (anywhere, often under /mnt)

Per-user mounting does not require root access, it's just automating the desktop interface. Systemwide mounts (/etc/fstab) can allow access from before login, and are therefore much more suitable for access through a network, or by system services.

Commands should be entered on a terminal (Type **terminal** in the program launcher of recent unity based Ubuntu releases, or select Applications -> Accessories -> Terminal from the menus on older releases).

# **Per-User Mounts**

## **udisks**

This is the modern replacement for **gnome-mount**. It's not gnome specific.

When you mount a disc normally with the file browser (nautilus etc) it mounts disks by interacting with udisks behind the scenes.

You can do the same thing on the command line with the [udisks](http://manpages.ubuntu.com/manpages/precise/en/man1/udisks.1.html) tool. For example:

/usr/bin/udisks --mount /dev/sdb1

The bit after **--mount** is the device name of the partition you want to mount. (**/dev/something**). The command will mount */dev/sdb1* in */media/<uuid>* where *<uuid>* is the identifier of the particular partition. Read below to find the uuid of your partition.

### **Finding the device name of your Partition**

* Open your partition in nautilus (this makes sure it's mounted)
* type **mount** in a terminal. You should see a line with your disk name on it like:

/dev/sdb1 on /media/My-Happy-Disk type vfat ....

### **Finding the UUID of your partition**

A device name like /dev/sdb1 is based on where your physical drive is plugged in and the order the drives were made available to the computer, so if your computer changes the same command could mount a different partition. It's possible for this to happen just from a software upgrade.

The solution is to use a UUID. A UUID is a globally unique name for the partition. A UUID will remain the same if you put an internal disk into an external USB caddy, or change the name of the partition.

* Type **ls -al /dev/disk/by-uuid/** you will see an entry that matches the name you saw before:

lrwxrwxrwx 1 root root 10 2012-02-15 10:23 1313-F422 -> ../../sdb1

The UUID in the example is *1313-F422*

You can now determine the command you need for mounting the device by UUID. For our example it would be

/usr/bin/udisks --mount /dev/disk/by-uuid/1313-F422

Of course you need to replace *1313-F422* with the UUID of the device you want to mount.

Again, this will mount your partition in */media/<uuid>* which is not consistent with how nautilus mounts partitions. The partitions mounted by nautilus can be found in */media/<user>/<uuid>* with *<user>* being the current logged-in user.

To keep the folder structure consistent an alternative command can be used that takes care of the correct mountpoint automatically:

udisksctl mount --block-device /dev/disk/by-uuid/<uuid>

Of course, *<uuid>* needs to be adjusted to the correct UUID of your partition, e.g., 1313-F422 in this example.

### **Adding to startup**

* From the Ubuntu dash (click logo in top left) find **startup applications** or press Alt+F2 and type **gnome-session-properties**
* Push the **Add** button.
* Choose a name, paste in your command and push the **Add** button

# **Systemwide Mounts**

Three methods will be discussed:

1. The first method is manually editing Ubuntu's filesystem table. This sounds more complex than it really is.
2. The second method, for versions 6.06 and later, is described at [MountingWindowsPartitions](https://help.ubuntu.com/community/MountingWindowsPartitions).
3. The third - simple - method is to install the **pysdm** package (in Gutsy) and then use **System-Administration-Storage Device Manager** without any manual editing of the fstab file, and disregard most of the instructions that follow. (NOTE: psydm removed from repositories in 12.10 and presumably beyond - see<http://ubuntuforums.org/showthread.php?p=12338212>. Use of arios automount or mount manager not recommended)

## **Manual Setup Help**

To mount hard disk partitions, you should have a basic understanding of the information below.

Once installed you can browse to System > Network and File Systems >

Basic understanding is still required...

## **Viewing the system's physical information**

To read the layout of the physical disks in the system, the 'fdisk' command is used. Before panicking, realize that fdisk will be used with only non-destructive options; specifically, it will be used with 'l' (lower-case 'L', not '1'), which lists the partition table of the specified disk.

sudo fdisk -l

As an example, here is what the output would be if one had a laptop with one internal drive, and usb drive plugged in:

WARNING: GPT (GUID Partition Table) detected on '/dev/sda'! The util fdisk doesn't support GPT. Use GNU Parted.

Disk /dev/sda: 180.0 GB, 180045766656 bytes

255 heads, 63 sectors/track, 21889 cylinders, total 351651888 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x00000000

Device Boot Start End Blocks Id System

/dev/sda1 1 351651887 175825943+ ee GPT

Disk /dev/sdb: 8019 MB, 8019509248 bytes

255 heads, 63 sectors/track, 974 cylinders, total 15663104 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0xc3072e18

Device Boot Start End Blocks Id System

/dev/sdb1 \* 2192 15663103 7830456 b W95 FAT32

If the system has multiple hard disk drives, multiple lists will be displayed.

## **Deciding which partitions to mount**

Most systems only have */dev/hda*, which is the hard disk drive, and */dev/hdc*, which is the CD-ROM, or optical, drive. If more were listed when the command above was run, they can be identified as follows: *hda* is the first drive on the first IDE channel (0:0), *hdb* is the second drive (0:1), *hdc* is the third drive (1:0), and *hdd* is the fourth (1:1). SCSI and S-ATA disks have names like *sda* and *sdb*.

Look through the list generated above to identify the partition(s) to be mounted. The following table lists some common 'System' types, which may help this process.

| System name | English name | Linux type |
| --- | --- | --- |
| W95 FAT32 | Microsoft FAT32 | vfat |
| W95 FAT32 (LBA) | Microsoft FAT32 | vfat |
| W95 FAT16 (LBA) | Microsoft FAT16 | vfat |
| W95 Ext'd (LBA) | Microsoft extended partition | Not used |
| NTFS volume set | Microsoft NTFS | ntfs |
| NTFS volume set | Microsoft NTFS with read-write access | ntfs-3g |
| Apple\_HFS | Apple HFS | hfsplus |

A list of the form '/dev/hda1: /media/windows/ (vfat)', where '/dev/hda1' is the device, '/media/windows' is the arbitrary location where the partition will appear when mounted, and 'vfat' is the Linux type, should be created on paper, containing all partitions to be added.

## **Preparing the system**

Look through the list which was just created. For every location ('/media/windows'), run the following command.

ls /media/windows

If a response like *ls: /media/windows: No such file of directory* is returned, the location is open. If a list of files or nothing is returned, the location exists already, and the planned location will need to be altered.

Once every location has been confirmed as free, run the following command for each entry, replacing '/media/windows' with the chosen location.

sudo mkdir /media/windows

## **Editing Ubuntu's filesystem table**

**It is possible to break Ubuntu if some of the earlier lines in the file opened during this step are modified, so be sure to read this section carefully.**

Ubuntu's filesystem table is located at '/etc/fstab'. Open this file for editing by running the following command for Ubuntu

gksu gedit /etc/fstab

or this command for Kubuntu

kdesu kate /etc/fstab

or command line

nano -w /etc/fstab

The file opened contains lines of the form *<device> <location> <Linux type> <options> <dump> <pass>*. Every element in this line is separated by whitespace (spaces and tabs):

# /etc/fstab: static file system information.

#

# Use 'blkid' to print the universally unique identifier for a

# device; this may be used with UUID= as a more robust way to name devices

# that works even if disks are added and removed. See fstab(5).

#

# <file system> <mount point> <type> <options> <dump> <pass>

# / was on /dev/sda2 during installation

UUID=a2db89ed-d599-4138-8838-0b950b6c3fbb / ext4 errors=remount-ro 0 1

# /boot/efi was on /dev/sda1 during installation

UUID=AEF0-9F26 /boot/efi vfat defaults 0 1

# swap was on /dev/sda3 during installation

UUID=df17fdb9-57a4-4302-856e-3cd656848355 none swap sw 0 0

### **<file system>**

The first field, (fs\_spec), describes the block special device or remote filesystem to be mounted.

For ordinary mounts it will hold (a link to) a block special device node (as created by mknod(8)) for the device to be mounted, like /dev/cdrom or /dev/sdb7. For NFS mounts one will have <host>:<dir>, e.g., knuth.aeb.nl:/. For procfs, use proc.

Instead of giving the device explicitly, one may indicate the (ext2 or xfs) filesystem that is to be mounted by its UUID or volume label (cf. e2label(8) or xfs\_admin(8)), writing LABEL=<label> or UUID=<uuid>, e.g., LABEL=Boot or UUID=3e6be9de-8139-11d1-9106-a43f08d823a6. This will make the system more robust: adding or removing a SCSI disk changes the disk device name but not the filesystem volume label. The UUID of a drive can be determined by typing ls -l /dev/disk/by-uuid/ in a shell.

### **<mount point>**

The second field, (fs\_file), describes the mount point for the filesystem. For swap partitions, this field should be specified as none. If the name of the mount point contains spaces these can be escaped as \040.

### **<type>**

The third field, (fs\_vfstype), describes the type of the filesystem. Linux supports lots of filesystem types, such as adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, and possibly others. For more details, see mount(8). For the filesystems currently supported by the running kernel, see /proc/filesystems. An entry swap denotes a file or partition to be used for swapping, cf. swapon(8). An entry ignore causes the line to be ignored. This is useful to show disk partitions which are currently unused.

### **<options>**

The fourth field, (fs\_mntops), describes the mount options associated with the filesystem.

It is formatted as a comma separated list of options. It contains at least the type of mount plus any additional options appropriate to the filesystem type. For documentation on the available options for non-nfs file systems, see mount(8). For documention on all nfs-specific options have a look at nfs(5). Common for all types of file system are the options noauto (do not mount when "mount -a" is given, e.g., at boot time), user (allow a user to mount), and owner (allow device owner to mount), and comment (e.g., for use by fstab-maintaining programs). The owner and comment options are Linux-specific. For more details, see mount(8).

### **<dump>**

The fifth field, (fs\_freq), is used for these filesystems by the dump(8) command to determine which filesystems need to be dumped. If the fifth field is not present, a value of zero is returned and dump will assume that the filesystem does not need to be dumped.

### **<pass>**

The sixth field, (fs\_passno), is used by the fsck(8) program to determine the order in which filesystem checks are done at reboot time. The root filesystem should be specified with a fs\_passno of 1, and other filesystems should have a fs\_passno of 2. Filesystems within a drive will be checked sequentially, but filesystems on different drives will be checked at the same time to utilize parallelism available in the hardware. If the sixth field is not present or zero, a value of zero is returned and fsck will assume that the filesystem does not need to be checked.

### **<options> example**

To learn more about options, type 'man mount'.

| Description | Accessible by everyone | Accessible by a subset of users\*\* |
| --- | --- | --- |
| FAT(16/32) partition | user,auto,fmask=0111,dmask=0000 | user,auto,fmask=0177,dmask=0077,uid=1000 |
| NTFS partition\* | rw,auto,user,fmask=0111,dmask=0000 | rw,user,auto,fmask=0177,dmask=0077,uid=1000 |
| Apple Partition | user,auto,file\_umask=0111,dir\_umask=0000 | user,auto,file\_umask=0177,dir\_umask=0077,uid=1000 |

\*If you want write access to your file system, you should set the filesystem type to 'ntfs-3g' instead of 'ntfs'. You may need to install the package 'ntfs-3g' for this to work, so make sure it is installed **before** you use ntfs-3g.

\*\*uid=1000 restricts access to the user created while installing Ubuntu. 1001 is the user created after that, and so forth. gid=# may be used with or in place of uid to grant access to a group. However, group and user enumeration is beyond the scope of this article.

### **<type> example**

Note for international users: if your filesystem contains funny symbols, you may need to add an option for utf-8 support.

| Filesystem type | Option to enable utf-8 support |
| --- | --- |
| ntfs | nls=utf8 |
| vfat | utf8 |
| smbfs | iocharset=utf8,codepage=unicode,unicode |

### **fstab example**

So, to grant all users access to '/dev/hda1', which will be located at '/media/windows', and is of type 'vfat', the line added would be.

/dev/hda1 /media/windows vfat user,fmask=0111,dmask=0000 0 0

This entry is case-sensitive. In general, lower-case letters are used to avoid confusion.

'dump' and 'pass' are only of use to native filesystems. They can be set to '0' for all additional partitions.

For every item in the list of partitions to be mounted, add one line of the form above to the end of the fstab file.

## **Mounting Fakeraid**

Its assumed you have formated you raid set using dmraid command with instructions found at [FakeRaidHowto](https://help.ubuntu.com/community/FakeRaidHowto).

You need to know the name of your raid set

sudo dmraid -ay

RAID set "sil\_aiaedhaeafaa" already active

RAID set "sil\_aiaedhaeafaa1" already active

Edit fstab as per instructions above :

nano -w /etc/fstab

An example line to add

/dev/mapper/sil\_aiaedhaeafaa1 /media/raid reiserfs user,nosuid,exec,nodev 0 0

Make sure you create the directory /media/raid

mkdir /media/raid

Reboot

## **Mounting and checking the partitions**

In the terminal, type the following command:

sudo mount -a

To verify that the partitions were mounted properly, open Gnome's file browser and direct it to the locations at which the partitions were mounted. Click the 'File System' button to access '/', and navigate from there. If the partition being examined contains files, the modifications were successful, and the partitions will be automatically mounted every time the system is restarted. If no files are found, please see [XChatHowto](https://help.ubuntu.com/community/XChatHowto) and join #ubuntu on irc.freenode.net.

## **Using pysdm in Precise**

Pysdm is a program to automatically setup partitions every time Ubuntu starts.

### **Installation**

sudo apt-get install pysdm

### **Usage**

Select each partition you want to change in the list. Note the type. Often it is ext3 (Linus) or NTFS (Windows). Use assistant and press OK. By default the partition is mounted at boottime. You can also mount the partition now. Press Apply. Done.

# **Hints, Tips, and Technical Information**

## **Adding a bookmark**

Bookmarks may be added to help speed access to commonly used files and locations, such as mounted partitions.

While browsing the filesystem using Gnome's file browser, click 'Bookmarks', then 'Add Bookmark'. A new entry will be placed in the bar on the left, and the location will appear under the 'Places menu'.

Alternatively, bookmarks may be added while opening or saving a file by clicking the 'Add' button.

To remove a bookmark, click either 'Edit Bookmarks', or the 'Remove' button, either of which is the counterpart of the two addition methods stated above.

## **More technical tips**

### **Symlinking for greater convenience**

If navigating to a partition's mount point seems inconvenient, even with the links on the left of Gnome's file browser, a link can be placed on the desktop, or anywhere else, for that matter.

Try the following command:

ln -s /media/windows ~/Desktop/

A link to the directory '/media/windows' will be placed on the desktop. Files may be dragged into it, it may be opened, it can be renamed and moved, and if it proves to be annoying, it can be deleted like any other file without risking damage to its contents.

This process is called symlinking because the link created is symbolic. It merely points to the location being referenced.

### **Unmounting a partition to prevent unwanted access**

While it's easy and effective to set permissions on partitions, there may be times when setting permissions won't be enough. Sometimes, people who can't be trusted with important information, such as a term paper, may need access to a system, and modifying the filesystem table may be impractical. When things like this happen, it's easy to archive data to preserve permissions and stick it on a mounted partition or device, then unmount the device, preventing "accidental" access. (The potentially jokingly malicious user would need to know the filesystem table, and while that information isn't well hidden, it's hard to access without drawing attention).

Try the following command to unmount a partition:

sudo umount /media/windows

The partition mounted at '/media/windows' will be unmounted, and attempts to access it will yeild only an empty directory. To bring it back later, either reboot, or simply run the following command.

sudo mount /media/windows

## **Technical background information**

### **How Linux manages partitions**

Linux uses a virtual filesystem (VFS) to maintain a single tree of files, all spread from '/'. When a partition is mounted, it is added to the tree at its mount point.

When a FAT32 partition is mounted at '/media/windows', all access to '/media/windows' and everything below it is transparently handled by the Linux kernel using the 'vfat' module. Applications need not know they're dealing with anything else. However, mounting a partition at a location inside of another mounted partition is unpredictable, unstable, and generally a bad idea.

Every partiton maintains its own free space and internal filesystem, so they can be mounted on other systems and behave properly. (If you have two operating systems or a removable hard disk drive, feel free to test this -- it's hard to break anything, and sharing /home (provided no usernames overlap) and swap between two Linux installations is very convenient)

To see what's going on behind the scenes, try the 'df' and 'mount' commands.

## Próximo tópico: FSTAB

Rename USB Drive

This guide is primarily for **external drives** such as USB hard drives, USB flash drives, and flash memory cards. You can label internal disks, but to change their mount points, use [MoveMountpointHowto](https://help.ubuntu.com/community/MoveMountpointHowto) which uses the file called [Fstab](https://help.ubuntu.com/community/Fstab). This guide covers editing partition labels (disk names) for FAT16/FAT32, NTFS, ext2/ext3, JFS, ReiserFS, and XFS filesystems.

IconsPage/tip.pngBy default, external drives automatically mounted at /media/disk then /media/disk-1 and so on. This is not very helpful when trying to find the drive you are looking for, especially if you have multiple devices plugged in. Labeled devices that are automatically mounted will be mounted in the */media* directory using their label as the mount point, /media/<label>. ex: /media/my\_external .

IconsPage/warning.png When choosing labels, be sure that the new mount point /media/<label> does not already exist since the directory will be created when the disk is mounted.

# **Using the Partition Editor**

The file manager (Nautilus) currently [does not support renaming disk partitions](https://bugs.launchpad.net/ubuntu/+source/nautilus/+bug/68924/), but Gnome's Partition Editor (GParted) does. To change a partition's label, follow these directions. (Be careful using Partition Editor, as it's capable of making your computer completely unusable if you do the wrong thing.)

1. Open the **System** > **Administration** menu and see if there's an entry for **GParted** (previously **Partition Editor**).
2. If there is, launch it. If there isn't, install the "gparted" package and it should now appear in the menu. Enter your password when prompted.
3. Disk drives are divided up into partitions. To find the partition you want to re-label, you first have to find the disk drive that contains it, using the drop-down menu in the upper right. It will show a device name like /dev/sdb and the drive's total size in parentheses. After selecting a drive, you will see a list of all partitions on that drive.
4. If the partition is mounted (has a key icon next to it), right-click on the partition and select **Unmount**.
5. With the key icon gone, right-click on the partition and select **Label**. If you can't select it, install the ntfsprogs package.
6. Enter the new partition name and press **Ok**.
7. The label change is now pending, but has not been completed. Press the **Apply** button near the top of the window. After confirming, it should say "All operations successfully completed". The drive now has a new label.

# **Using the Command line**

There are at least 6 separate command line tools used to label a partition - the program used depends on the partition's [filesystem](https://help.ubuntu.com/community/LinuxFilesystemsExplained) type:

* For **FAT16** and **FAT32** partitions, use **mlabel** from the *mtools* package.
* For **NTFS** partitions, use **ntfslabel** from the *ntfs-3g* package.
* For **ext2**, **ext3**, or **ext4** partitions, use **e2label**.
* For **JFS** partitions, use **jfs\_tune**.
* For **ReiserFS** (v3) partitions, use **reiserfstune**.
* For **XFS** partitions, use **xfs\_admin**

## **Identify your Partition**

**IconsPage/terminal.png** For help with the terminal, see [UsingTheTerminal](https://help.ubuntu.com/community/UsingTheTerminal).

Plug in your USB device and list your partitions with:

sudo fdisk -l

You can also list your mounted devices and their descriptions with:

mount

IconsPage/example.png For the rest of this tutorial we will use the following:

* <device> = your device /dev/sdxy, ex: */dev/sdb1*
* <label> = your desired (new) label, ex: *my\_external*

## **Install the Labeling Program**

Based on the package names listed above for each filesystem type, install the correct package for your partition:

sudo apt-get install <package>

Here are all the different ones:

sudo apt-get install mtools

sudo apt-get install ntfsprogs

sudo apt-get install e2fsprogs

sudo apt-get install jfsutils

sudo apt-get install reiserfsprogs

* sudo apt-get install xfsprogs

or install the appropriate package from [Synaptic](https://help.ubuntu.com/community/SynapticHowto).

## **Unmount the Partition**

Partitions generally need to be unmounted before you can fiddle with them, so unmount the partition of the device you want to change the label for:

sudo umount <device>

ex:

* sudo umount /dev/sdb1

If it was automounted, you can also unmount the drive by right clicking the desktop icon and clicking **Unmount** (or **Eject** in some cases).

## **Changing the Label**

After you complete the appropriate porition for your filesystem, jump to the next section to verify the change.

# **Filesystems**

## **FAT16 and FAT32**

These filesystems are most often found on USB thumb drives, flash cards (like for a camera or cell phone), and older external USB hard drives.

### **Check the current label**

sudo mlabel -i <device> -s ::

ex:

* sudo mlabel -i /dev/sdb1 -s ::

Note that we're using the special "::" drive which allows us to specify the device descriptor on the command line; otherwise we'd have to edit ~/.mtoolsrc to assign a drive letter (see Option 2 under "Change the label").

### **Change the label**

#### **Option 1**

After unmounting and checking the current label (above), use

sudo mlabel -i <device> ::<label>

ex:

* sudo mlabel -i /dev/sdb1 ::my\_external

Ignore the "Volume label is XYZ" output as this is the *old* label. Jump to the Verify the Change section below.

#### **Option 2**

For Ubuntu 8.10 and up, edit mtools.conf as sudo

sudo nano /etc/mtools.conf

add something like for each drive:

drive p: file="/dev/sdb1"

* drive q: file="/dev/sdb2"

etc.

Then use

sudo mlabel p:new\_label

ex:

* sudo mlabel p:30GB\_FAT32

(note the underscore \_ should be used, as spaces are not allowed)

<http://ubuntuforums.org/showpost.php?p=6356016&postcount=9>

#### **Error message**

If you get a message like this:

Total number of sectors (7831520) not a multiple of sectors per track (63)!

You can easily ignore the check by running this command:

echo mtools\_skip\_check=1 >> ~/.mtoolsrc

## **NTFS**

This filesystem is most often found on external USB and firewire hard drives or other Windows formatted disks.

### **Check the current label**

sudo ntfslabel <device>

ex:

* sudo ntfslabel /dev/sdb1

### **Change the label**

Note: 128 characters maximum.

sudo ntfslabel <device> <label>

ex:

* sudo ntfslabel /dev/sdb1 my\_external

Ubuntu caches the drive's label so to see the full affects of the change it is not enough just to umount and mount it again. You have to umount, remove, put back, mount again.

## **ext2, ext3, and ext4**

These filesystems are most often found on linux formatted drives.

### **Check the current label**

sudo e2label <device>

ex:

* sudo e2label /dev/sdb1

### **Change the label**

Note: 16 characters maximum.

sudo e2label <device> <label>

ex:

* sudo e2label /dev/sdb1 my\_external

## **JFS**

These filesystems are most often found on IBM and some linux formatted disks.

### **Check the current label**

sudo jfs\_tune -l <device>

ex:

* sudo jfs\_tune /dev/sdb1

### **Change the label**

Note: 16 characters maximum.

sudo jfs\_tune -L <label> <device>

ex:

* sudo jfs\_tune -L my\_external /dev/sdb1

## **ReiserFS (v3)**

This filesystem is most often found on linux formatted disks.

Note: this could work with ReiserFS 4 too, I have not tried.

### **Change the label**

Note: 16 characters maximum.

sudo reiserfstune -l <label> <device>

ex:

* sudo reiserfstune -l my\_external /dev/sdb1

## **XFS**

This filesystem is most often found on UNIX formatted disks.

### **Check the current label**

xfs\_admin -l <device>

ex:

* xfs\_admin -l /dev/sdb1

### **Change the label**

Note: 12 characters maximum.

sudo xfs\_admin -L <label> <device>

ex:

* xfs\_admin -l my\_external /dev/sdb1

# **Verify the Change**

Now for the easiest part: unplug the drive, wait a second, then plug it back in. It should appear on your desktop with the new label and have its new mount point.

Without unplugging and having the device remount, you can also just run:

sudo blkid

CRON

Cron is a system daemon used to execute desired tasks (in the background) at designated times.

A crontab file is a simple text file containing a list of commands meant to be run at specified times. It is edited using the crontab command. The commands in the crontab file (and their run times) are checked by the cron daemon, which executes them in the system background.

Each user (including root) has a crontab file. The cron daemon checks a user's crontab file regardless of whether the user is actually logged into the system or not.

To display the on-line help for crontab enter:

man crontab

or further information is available from the [OpenGroup](http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html) specifications.

On Gnome-based Ubuntu systems Gnome *Scheduled tasks* tool (from the *gnome-schedule* package) in *Applications* --> *System Tools* provides a graphical interface with prompting for using Cron. The project website is at<http://gnome-schedule.sourceforge.net/>; the software is installable from the Software Center or by typing

sudo apt-get install gnome-schedule

in a terminal.

# **Starting to Use Cron**

To use cron for tasks meant to run only for your user profile, add entries to your own user's crontab file. To edit the crontab file enter:

crontab -e

Edit the crontab using the format described in the next sections. Save your changes. (Exiting without saving will leave your crontab unchanged.) To display the on-line help describing the format of the crontab file enter:

man 5 crontab

Commands that normally run with administrative privileges (i.e. they are generally run using sudo) should be added to the root crontab. To edit the root crontab enter:

sudo crontab -e

# **Crontab Lines**

Each line has five time-and-date fields, followed by a command, followed by a newline character ('\n'). The fields are separated by spaces. The five time-and-date fields cannot contain spaces. The five time-and-date fields are as follows: minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday).

01 04 1 1 1 /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on January 1st plus every Monday in January.

An asterisk (\*) can be used so that every instance (every hour, every weekday, every month, etc.) of a time period is used.

01 04 \* \* \* /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on every day of every month.

Comma-separated values can be used to run more than one instance of a particular command within a time period. Dash-separated values can be used to run a command continuously.

01,31 04,05 1-15 1,6 \* /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 01 and 31 past the hours of 4:00am and 5:00am on the 1st through the 15th of every January and June.

The "/usr/bin/somedirectory/somecommand" text in the above examples indicates the task which will be run at the specified times. It is recommended that you use the full path to the desired commands as shown in the above examples. Enter *which somecommand* in the terminal to find the full path to *somecommand*. The crontab will begin running as soon as it is properly edited and saved.

You may want to run a script some number of times per time unit. For example if you want to run it every 10 minutes use the following crontab entry (runs on minutes divisible by 10: 0, 10, 20, 30, etc.)

\*/10 \* \* \* \* /usr/bin/somedirectory/somecommand

which is also equivalent to the more cumbersome

0,10,20,30,40,50 \* \* \* \* /usr/bin/somedirectory/somecommand

Cron also offers some special strings, which can be used in place of the five time-and-date fields:

| **string** | **meaning** |
| --- | --- |
| @reboot | Run once, at startup. |
| @yearly | Run once a year, "0 0 1 1 \*". |
| @annually | (same as @yearly) |
| @monthly | Run once a month, "0 0 1 \* \*". |
| @weekly | Run once a week, "0 0 \* \* 0". |
| @daily | Run once a day, "0 0 \* \* \*". |
| @midnight | (same as @daily) |
| @hourly | Run once an hour, "0 \* \* \* \*". |

@reboot /path/to/execuable1

The above example will execute /path/to/executable1 when the system starts.

For more information on special strings enter "man 5 crontab".

# **Crontab Options**

* The -l option causes the current crontab to be displayed on standard output.
* The -r option causes the current crontab to be removed.
* The -e option is used to edit the current crontab using the editor specified by the EDITOR environment variable.

After you exit from the editor, the modified crontab is checked for errors and, if there are no errors, it is installed automatically. The file is stored in */var/spool/cron/crontabs* but should only be edited using the crontab command.

# **Allowing/Denying User-Level Cron**

If the **/etc/cron.allow** file exists, then users must be listed in it in order to be allowed to run the **crontab** command. If the **/etc/cron.allow** file does not exist but the **/etc/cron.deny** file does, then users must not be listed in the **/etc/cron.deny** file in order to run **crontab**.

In the case where neither file exists, the default on current Ubuntu (and Debian, but not some other Linux and UNIX systems) is to allow all users to run jobs with **crontab**.

No cron.allow or cron.deny files exist in a standard Ubuntu install, so all users should have cron available by default, until one of those files is created. If a blank cron.deny file has been created, that will change to the standard behavior users of other operating systems might expect: cron only available to root or users in cron.allow.

Note, userids on your system which do not appear in /etc/shadow will NOT have operational crontabs, if you desire to enter a user in /etc/passwd, but NOT /etc/shadow that user's crontab will never run. Place an entry in /etc/shadow for the user with a \* for the password crypt, eg:

joeuser:\*:15169::::::

# **Further Considerations**

Crontab commands are generally stored in the crontab file belonging to your user account (and executed with your user's level of permissions). If you want to regularly run a command requiring administrative permissions, edit the root crontab file:

sudo crontab -e

Depending on the commands being run, you may need to expand the root users PATH variable by putting the following line at the top of the root crontab file:

PATH=/usr/sbin:/usr/bin:/sbin:/bin

**crontab -e** uses the EDITOR environment variable. To change the editor to your own choice, just set that variable. You may want to set EDITOR in your .bashrc because many commands use this variable. For example, in order to set the editor to be nano (a very easy editor to use) add this line to .bashrc:

export EDITOR=nano

It is sensible to test that your cron jobs work as intended. One method for doing this is to set up the job to run a couple of minutes in the future and then check the results before finalising the timing. You may also find it useful to put the commands into script files that log their success or failure, for example:

echo "Nightly Backup Successful: $(date)" >> /tmp/mybackup.log

If your machine is regularly switched off, you may also be interested in **at** and **anacron**, which provide other approaches to scheduled tasks. For example, **anacron** offers simple system-wide directories for running commands hourly, daily, weekly, and monthly. Scripts to be executed in said times can be placed in **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, and **/etc/cron.monthly/**. All scripts in each directory are run as root, and a specific order to running the scripts can be specified by prefixing the scripts' filenames with numbers (see the **man** page for **run‑parts** for more details). Although the directories contain periods in their names, run‑parts **will not** accept a file name containing a period and will fail silently when encountering them ([bug #38022](https://bugs.launchpad.net/ubuntu/+source/debianutils/+bug/38022)). Either rename the file or use a symlink (without a period) to it instead (see, for example, [*python + cron without login?*](http://ubuntuforums.org/showthread.php?t=1629301) and [*Problems with Hourly Cron Job*](http://ubuntuforums.org/showthread.php?t=1692253)).

# **Common Problems**

Edits to a user's crontab and the cron jobs run are all logged by default to **/var/log/syslog** and that's the first place to check if things are not running as you expect.

If a user was not allowed to execute jobs when their crontab was last edited, just adding them to the allow list won't do anything. The user needs to re-edit their crontab after being added to cron.allow before their jobs will run.

Note that user-specific crontabs (including the root crontab) do not specify the user name after the date/time fields. If you accidentally include the user name in a user-specific crontab, the system will try to run the user name as a command.

Cron jobs may not run with the environment, in particular the PATH, that you expect. Try using full paths to files and programs if they're not being located as you expect.

The "%" character is used as newline delimiter in cron commands. If you need to pass that character into a script, you need to escape it as "\%".

If you're having trouble running a GUI application using cron, see the GUI Applications section below.

# **Two Other Types of Crontab**

The crontab files discussed above are **user** crontabs. Each of the above crontabs is associated with a user, even the root crontab, which is associated with the root user. There are two other types of crontab, with syntax as follows:

minute(s) hour(s) day(s)\_of\_month month(s) day(s)\_of\_week user command

Note that the only difference from the syntax of the user crontabs is that the line specifies the user to run the job as.

The first type is as follows. As mentioned above **anacron** uses the **run‑parts** command and **/etc/cron.hourly**, **/etc/cron.weekly**, and **/etc/cron.monthly** directories. However **anacron** itself is invoked from the **/etc/crontab** file. This file could be used for other cron commands, but probably shouldn't be. Here's an example line from a fictitious **/etc/crontab**:

00 01 \* \* \* rusty /home/rusty/rusty-list-files.sh

This would run Rusty's command script as user **rusty** from his home directory. However, it is not usual to add commands to this file. While an experienced user should know about it, it is not recommended that you add anything to **/etc/crontab**. Apart from anything else, this could cause a problem if the **/etc/crontab** file is affected by updates! Rusty could lose his command.

The second type is to be found in the directory /etc/cron.d. This directory can contain crontab files. The directory is often used by packages, and the crontab files allow a user to be associated with the commands in them.

Example: Instead of adding a line to /etc/crontab, which Rusty knows is not a good idea, he might well add a file to the directory /etc/cron.d with the name **rusty**, containing his cron line above. This would not be affected by updates but is a **well known** location.

When would you use these alternate crontab locations? Well, on a single user machine or a shared machine such as a school or college server, a **user** crontab would be the way to go. But in a large IT department, where several people might look after a server, then the directory /etc/cron.d is probably the best place to install crontabs - it's a central point and saves searching for them!

You may not need to look at **/etc/crontab** or **/etc/cron.d**, let alone edit them by hand. But an experienced user should perhaps know about them and that the packages that he/she installs may use these locations for their crontabs.

# **GUI Applications**

It is possible to run gui applications via cronjobs. This can be done by telling cron which display to use.

00 06 \* \* \* env DISPLAY=:0 gui\_appname

The *env DISPLAY=:0* portion will tell cron to use the current display (desktop) for the program "gui\_appname".

And if you have multiple monitors, don't forget to specify on which one the program is to be run. For example, to run it on the first screen (default screen) use :

00 06 \* \* \* env DISPLAY=:0.0 gui\_appname

The *env DISPLAY=:0.0* portion will tell cron to use the first screen of the current display for the program "gui\_appname".

**Note:** GUI users may prefer to use gnome-schedule (aka "Scheduled tasks") to configure GUI cron jobs. In gnome-schedule, when editing a GUI task, you have to select "X application" in a dropdown next to the command field.

**Note:** In Karmic(9.10), you have to enable X ACL for localhost to connect to for GUI applications to work.

~$ xhost +local:

non-network local connections being added to access control list

~$ xhost

access control enabled, only authorized clients can connect

LOCAL:

...

# **Crontab Example**

Below is an example of how to setup a crontab to run updatedb, which updates the slocate database: Open a terminal, type "crontab -e" (without the double quotes) and press enter. Type the following line, substituting the full path of the application you wish to run for the one shown below, into the editor:

45 04 \* \* \* /usr/bin/updatedb

Save your changes and exit the editor.

Crontab will let you know if you made any mistakes. The crontab will be installed and begin running if there are no errors. That's it. You now have a cronjob setup to run updatedb, which updates the slocate database, every morning at 4:45.

Note: The double-ampersand (&&) can also be used in the "command" section to run multiple commands consecutively, but only if the previous command exits successfully. A string of commands joined by the double-ampersand will only get to the last command if all the previous commands are run successfully. If exit error-checking is not required, string commands together, separated with a semi-colon (;).

45 04 \* \* \* /usr/sbin/chkrootkit && /usr/bin/updatedb

The above example will run chkrootkit followed by updatedb at 4:45am daily - providing you have all listed apps installed. If chkrootkit fails, updatedb will NOT be run.

# **How Anacron is Set Up**

On Ubuntu 9.10 (and presumably, on later versions), anacron seems to be set up as follows:

There is a Upstart task, located in **/etc/init/anacron.conf**, which runs all the jobs in **/etc/anacrontab**. It is set to run on startup.

There is a cron.d file (**/etc/cron.d/anacron**) which causes the Upstart task to be started every day at 7:30 AM.

There is a file **/etc/apm/event.d/anacron**, which causes the Upstart task to be started when a laptop is plugged in to A/C power, or woken up.

In the system crontab (**/etc/crontab**), if anacron is not execuatable, run‑parts is used to run the files in cron.daily, cron.weekly, and cron.monthly at 6:25 AM, 6:47 AM and 6:52 AM, respectively.

In **/etc/anacrontab**, run‑parts is used to run cron.daily 5 minutes after anacron is started, and cron.weekly after 10 minutes (once a week), and cron.monthly after 15 (once a month).

Within the cron.daily, weekly, and monthly directories ( **/etc/cron.daily**, etc.) there is a **0anacron** file that sets the timestamps for anacron, so it will know they have been run, even if it didn't run them.

So it appears anacron is run on every startup, wake up, plug-in, and at 7:30 AM every day. Looking at the respective Changelogs and package databases, it looks like this setup is directly from Debian, and hasn't been changed since at least 2009.

# **Alternatives to Cron**

Some hosting companies don’t allow access to cron, but there are websites offering alternative ways of scheduling jobs (free or paid-for). Here are some examples:

* [SetCron](https://www.setcron.com)
* [SetCronJob](http://www.setcronjob.com)
* [OnlineCronJobs](http://www.onlinecronjobs.com)
* [EasyCron](https://www.easycron.com)