
A collection of useful commands I've picked up over time (*history* | *grep* ***X***). Understanding the purpose of each parameter is left to the reader - note typing '/' in *man* lets you search for specific terms in the manual page. Note that these commands have all been tested on an Ubuntu installation, and with the exception of package management they should work universally. Many commands require extra permissions and start with "sudo" - take extra caution when using these.

I update this page pretty frequently. Always lots to learn! :)

Index of commands:



Other stuff:



- ## aspell (command line spell checking)

  Spell check an input file:

  ```
  aspell -c sample.tex
  ```

Spell check an input file (do not create backup file):

```
aspell -dont-backup -c sample.tex
```

Show personal dictionary (note: the first one does not work on Ubuntu, alternate provided):

```
aspell dump personal
```

```
sudo updatedb
locate .aspell*
cat ~/.aspell.en.pws
```

Show aspell configuration:

```
aspell dump config
```

- **apt/dpkg (package management, software installation)**

  Add a package:

  ```
  sudo apt-get install package_name
  ```

  Remove a package:

  ```
  sudo apt-get remove package_name
  ```

  Add repository:

  ```
  sudo apt-add-repository ppa:sample_ppa/sample
  ```

  Search packages:

  ```
  sudo apt-get update
  apt-cache search search_term
  ```

  List installed packages:

  ```
  dpkg -l
  ```

  List files and directories of an installed package:

  ```
  dpkg -L package_name
  ```

- **cp/mv/rm (copy, move, delete files)**

Copy a file:

```
cp /example/file1 /example/file2
```

Copy a directory (dir1 into dir2):

```
cp -a /example/dir1 /example/dir2
```

Move a file:

```
mv /example/file1 /example/file2
```

Delete a file:

```
rm /example/file1
```

Delete a directory (be careful):

```
rm -rf /example/dir
```

- **dd (clone a hard drive, backup drive to file)**

List drives and partitions:

```
sudo fdisk -l
```

Clone a drive *(note: drive /dev/sdb must have size greater than or equal to /dev/sda)*:

```
sudo dd if=/dev/sda of=/dev/sdb
```

Clone drive to a file *(note: output file location should not be on any partition of /dev/sda)*:

```
sudo dd if=/dev/sda of=/home/sda_as_backup_file
```

Report progress (sends signal to dd every 10 seconds):

```
ps au | grep dd
sudo watch -n 10 "kill -USR1 processid"
```

List partition UUIDs:

```
sudo blkid
```

Modify partition UUID (if ext2/3/4, device is e.g. /dev/sda1):

```
tune2fs -U random /dev/sda1
```

- **df/du/find (drive usage)**

  Report usage for all mounted drives:

  ```
  df -h
  ```

  Show sorted usage of directories at root:

  ```
  sudo du -hsx /* | sort -n
  ```

  List files bigger than 100 MB:

  ```
  find -size +100M -ls
  ```

- **ffmpeg (video transcoding, desktop recording, etc.)**

  Record desktop (at 24 fps, 1920x1080 resolution, on screen 0, using mpeg4):

  ```
  ffmpeg -f x11grab -y -r 24 -s 1920x1080 -i :0+0,0 -vcodec mpeg4 -sameq
  sample_out.mp4
  ```

  Record desktop (400x400 portion of desktop at offset 200x200 on screen 1):

  ```
  ffmpeg -f x11grab -y -r 24 -s 400x400 -i :1+200,200 -vcodec mpeg4 -sameq
  sample_out.mp4
  ```

  Record webcam/microphone:

  ```
  ffmpeg -f alsa -ac 2 -i hw:0 -i /dev/video0 -acodec ac3 -ab 128k -vcodec
  mpeg4 -sameq -r 24 sample_out.mp4
  ```

  Transcode video (and resize):

  ```
  ffmpeg -i sample.mp4 -sameq -s hd720 sample_720.mp4
  ```

  Transcode audio only:

  ```
  ffmpeg -i sample.mp4 -vcodec copy -acodec aac -sameq sample_out.mp4
  ```

  Speed up video:

  ```
  ffmpeg -i sample.mp4 -vf "setpts=0.05*PTS" -r 25 -s hd720 -sameq
  sample_out.mp4
  ```

Rotate video 90 degrees clockwise:

```
ffmpeg -i sample.mp4 -vf "transpose=1" -sameq sample_out.mp4
```

Rotate video 90 degrees counter-clockwise:

```
ffmpeg -i sample.mp4 -vf "transpose=2" -sameq sample_out.mp4
```

Video overlay:

```
ffmpeg -i sample.mp4 -aspect 4:3 -vf "movie=overlay.png [logo]; [in][logo]
overlay=0:0:1" -sameq sample_out.mp4
```

Images to video:

```
ffmpeg -r 1 -i dgplogo%01d_1080.png -sameq sample_out.mp4
```

Merge videos (not supported by all formats):

```
ffmpeg -i "concat:sample1.avi|sample2.avi" -vcodec copy -acodec copy
sample_out.mp4
```

- **gcc (creating software programs)**

Compile object file(s):

```
gcc -c sample.c -o sample.o
```

Create static library (.a):

```
ar crs libsample.a sample.o
```

Create shared library (.so):

```
gcc -shared -o libsample.so sample.o
```

Linking a static library (creates executable program "test_static"):

```
gcc -static test.c -L. -lsample -o test_static
```

Linking a shared library (creates executable program "test_shared"):

```
gcc test.c -L. -lsample -o test_shared
```

Add shared library path:

```
(General) Append path to /etc/ld.so.conf
(Ubuntu) Append path to /etc/ld.so.conf.d/.conf
```

Update shared library paths:

```
sudo ldconfig
```

Add shared library path temporarily to run executable:

```
#!/bin/bash
LD_LIBRARY_PATH=/directory/for/libs/:$LD_LIBRARY_PATH
echo 'looking for shared libraries here first' $LD_LIBRARY_PATH
/path/to/executable
```

View shared libraries needed by executable, and paths to files linked at runtime:

```
ldd /path/to/executable
```

Output assembly code (extension is .s):

```
gcc -S -c sample.c
```

- **git (revision control system)**

Configure username and email (globally), and set merge tool:

```
git config --global user.name "My Name"
git config --global user.email "example@email.ca"
git config merge.tool kdiff3
```

Copy remote repository contents locally:

```
git clone https://github.com/Example/repo.git
```

Add and show remote repository (call it "upstream"):

```
git remote add upstream https://github.com/Example/repo.git
git remote -v
```

See history of repository ("upstream"):

```
git log upstream
```

Show details of last commit:

```
git show
```

Show remote branches (fetch updates local list):

```
git fetch
git branch -r
```

Show local branches:

```
git branch
```

Delete local branch:

```
git branch -D branch_name
```

Add files to be committed:

```
git add /path/to/files
```

Upload all local changes to master branch of remote repository "origin":

```
git commit -a -m'made some changes'
git push origin master
```

Merge pull request:

```
git pull upstream master
```

Stash local changes (to pull):

```
git stash
```

Merge another branch to active branch (note this auto-commits):

```
git merge origin/some_branch
```

Show merge conflicts:

```
git status
```

Resolve merge conflicts:

```
git mergetool
```

Revert all local changes to previous commit:

```
git reset --hard commit_hash_value
```

Create a new branch:

```
git branch new_branch
```

Switch to work on other branch:

```
git checkout new_branch
```

- **grep (search files for strings)**

  Search file for string:

  ```
  grep search_string filename
  ```

  Output line numbers where search string is found:

  ```
  grep -n search_string filename
  ```

  Display filenames containing search string:

  ```
  grep -l search_string filenames*
  ```

- **locate (search for files/directories)**

  Search:

  ```
  locate 'search_term1 search_term2'
  ```

  Search (case insensitive):

  ```
  locate -i search_term
  ```

  Update filesystem database:

  ```
  sudo updatedb
  ```

  Find locations of program executable/source/documentation:

  ```
  whereis program_name
  ```

  Locate a command:

  ```
  which program_name
  ```

- **man (software documentation)**

  View manual page:

  ```
  man program_name
  ```

  Search manual pages for keyword:

  ```
  man -k keyword
  apropos keyword
  ```

- **mount (enabling access to drives/filesystems)**

  List mounted drives:

  ```
  mount
  ```

  Mount a device (SATA drive 1, /media/drivedir must exist):

  ```
  sudo mkdir /media/drivedir
  sudo mount /dev/sda1 /media/drivedir
  ```

  Unmount device:

  ```
  sudo umount /media/drivedir
  ```

- **rsync (backups)**

  Back up a directory:

  ```
  rsync -av source_path dest_path
  ```

  Back up a directory and exclude files/directories:

  ```
  rsync -av --delete-excluded --ignore-errors --exclude=*.gvfs* --
  exclude='.local' source_path dest_path
  ```

  Creating a backup script:

  ```
  #!/bin/bash
  echo "Backing up..."
  # may want to mount drive for source/dest paths first here
  rsync -av source_path dest_path
  ```

  ```
  chmod 744 ./backup.sh
  ```

Set ownership of dest path (e.g. if new partition owned by 'root'):

```
chown james:james /media/james/Drive -R
```

- **ssh (log into another machine)**

  Connect to server using ssh:

  ```
  ssh username@server_address
  ```

  Create public/private key (RSA):

  ```
  ssh-keygen -t rsa
  ```

  List keys loaded:

  ```
  ssh-add -l
  ```

  Add a new key:

  ```
  ssh-add /path/to/private_key
  ```

  Print key fingerprint:

  ```
  ssh-keygen -lf /path/to/key
  ```

- **tar/gzip (archive, compress)**

  Create archive of directory and compress:

  ```
  tar -czvf file.tar.gz /path/to/compress/
  ```

  Extract archive:

  ```
  tar -xvf file.tar.gz
  ```

  Compress single file:

  ```
  gzip input_file
  ```

  Uncompress single file:

  ```
  gunzip input_file.gz
  ```

## On getting nvidia_experimental_310 to work:

I had issues getting the nvidia_experimental_310 package working. It turned out there was a "nvidia_experimental_310" entry in a file "/etc/modprobe.d/blacklist-local.conf" causing the trouble. How and why that got there I may never know. Commands I tried to track down this problem and my path to a solution:

Show loaded modules (with 'nvidia' in name):

```
lsmod | grep nvidia
```

Not loaded. Show errors in X.org log:

```
cat /var/log/Xorg.0.log | grep '(EE)'
```

*Error found, suggests checking kernel log:*

*[ 10.865] (EE) NVIDIA: Failed to load the NVIDIA kernel module. Please check your*
*[ 10.865] (EE) NVIDIA: system's kernel log for additional error messages.*
*[ 10.865] (EE) Failed to load module "nvidia" (module-specific error, 0)*

Show kernel log (and view in 'less'):

```
cat /var/log/kern.log | less
```

*Last entry shows it trying to load nvidia module? Not very helpful unfortunately.*

Show all nvidia kernel objects:

```
locate *.ko | grep nvidia
```

*...*
*/lib/modules/3.5.0-21-generic/updates/dkms/nvidia_experimental_310.ko*

*Existence of the kernel object for the current version of the kernel I'm running (uname -r) is encouraging as it means when installing the nvidia_experimental_310 package it compiled OK. Let's try to link it into the kernel again ourselves.*

Load a module (in this case, 'nvidia_experimental_310'):

```
sudo modprobe nvidia_experimental_310
```

*modprobe: WARNING: Not loading blacklisted module nvidia_experimental_310*

*Blacklisted? Huh?*

Search all blacklisted modules (whose name contains 'nvidia'):

```
cat /etc/modprobe.d/blacklist* | grep nvidia
```

*blacklist nvidiafb*
*blacklist nvidia_experimental_310*

*Aha! After removing blacklist-local.conf which contained the above entry, I tried again.*

Load a module, then restart display manager:

```
sudo modprobe nvidia_experimental_310
sudo service restart lightdm
```

*Working well!*

## copy/pasting into the terminal:

"Control-Shift-V" pastes text directly into the terminal, avoiding the need to right click and choose "Paste" each time.

## filesystem mounting options for a solid state drive:

You can edit "fstab" (the file system table) by doing the following:

```
gksudo gedit /etc/fstab
```

To identify which is the SSD:

```
blkid
```

Listing all partition information can be helpful to identify the drive:

```
sudo fdisk -l
```

Under "options" in the fstab, for the SSD, adding **discard** adds TRIM support, adding **noatime, nodiratime** disables extra writes to keep track of when files and directories were last accessed. For instance if the SSD line in fstab is:

```
UUID=03f7d0a1-2fcb-4878-afef-8487799df4a4 / ext4 errors=remount-ro 0 1
```

Change it to the following, using commas to separate each option:

```
UUID=03f7d0a1-2fcb-4878-afef-8487799df4a4 / ext4
discard,noatime,nodiratime,errors=remount-ro 0 1
```

Do not forget to save and restart.

## using udisks and gnome-session-properties to automount partitions on login:

The easiest way to set up automatic mounting is using the *gnome-disks* tool, and choosing "Edit mount options...". The next easiest way is by running *gnome-session-properties* and add

the following command:

```
/usr/bin/udisks --mount /dev/sdb1
```

where "/dev/sdb1" the partition you wish to mount. Using a UUID is preferred in case drives get switched around, to view partition UUIDs use:

```
blkid
```

Given the UUID of the partition, change the udisks command parameters to:

```
/usr/bin/udisks --mount /dev/disk/by-uuid/1234567890ABCD
```

Or, one can edit the filesystem table file (*/etc/fstab*).

### changing file ownership on a USB drive not formatted to FAT32:

The drive will not allow you to write to it since you are not the owner of files on that filesystem (the *root* account is). Chances are, while FAT* does not support the concept of user ownership, the existing filesystem does (e.g. ext4), causing the problem. Use the "chown" command to change the ownership to your account. With the drive mounted, do:

```
sudo chown james /media/james/kingston32gb
```

This changes the files mounted to */media/james/kingston32gb* to be owned by user *james*. You will then be able to both read and write files to the device.

### getting multi-monitor settings with nvidia-settings to stick after reboot:

I had an issue with three display devices having their positions reset every reboot (two standard monitors, plus an Oculus Rift). The Oculus Rift would always be reset to be positioned between the two monitors. Using nvidia-settings and saving to */etc/X11/xorg.conf* never seemed to work. Bring up a terminal and run:

```
gksudo nvidia-settings
```

Set the desired display configuration. Then, save *xorg.conf* configuration settings to:

```
/usr/share/X11/xorg.conf.d/xorg.conf
```

You may notice when browsing the directory */usr/share/X11/xorg.conf.d/* that a file *xorg.conf.backup* was instead saved. You can copy or rename this file using *gksudo nautilus* using a GUI, or use the terminal command

```
cd /usr/share/X11/xorg.conf.d/
sudo cp xorg.conf.backup xorg.conf
```

On restart, the monitor settings should be remembered! (Tested on Ubuntu 13.04.)

## removing overlay-scrollbar from default Ubuntu installation:

The selection zone to bring up the overlay scrollbar I have issues with. It appears when the cursor is over the left side of the window edge, but is displayed on the right. When moving the cursor to the right to manipulate the control, it often disappears - I find this frustrating.

Fortunately, it is easy to disable the "overlay scrollbars". On Ubuntu 12.04:

```
gsettings set org.gnome.desktop.interface ubuntu-overlay-scrollbars false
```

and to bring them back:

```
gsettings reset org.gnome.desktop.interface ubuntu-overlay-scrollbars
```

For Ubuntu 12.10 and 13.04, to disable them:

```
gsettings set com.canonical.desktop.interface scrollbar-mode normal
```

and to bring them back:

```
gsettings reset com.canonical.desktop.interface scrollbar-mode
```

The command should take effect immediately. Previously, I removed the package using "*sudo apt-get remove overlay-scrollbar*". While that worked, warnings would appear about the missing "overlay-scrollbar" GTK module when running applications.

## fixing the fontconfig warning when running applications:

When running my Qt applications, under application output I would always see the message: *Fontconfig warning: "/etc/fonts/conf.d/50-user.conf", line 9: reading configurations from ~/.fonts.conf is deprecated.* This can be fixed by doing the following:

```
mkdir ~/.config/fontconfig
mv ~/.fonts.conf ~/.config/fontconfig/fonts.conf
```

The change should take effect immediately -- applications will no longer complain about it. (Tested on Ubuntu 13.04.)

## setting up an environment for making documents with TeXworks and LaTeX:

On a new Ubuntu installation, to produce academic papers I install TeXworks software, the latexmk script, and some LaTeX packages, using the following:

```
sudo apt-get install texworks latexmk texlive-latex-recommended texlive-latex-
extra
```

In TeXworks, Edit>Preferences>Typesetting. Add latexmk processing tool with the program *usr/bin/latexmk* and the following command line arguments:

```
-e
$pdflatex=q/pdflatex $synctexoption %O %S/
-pdf
$fullname
```

Check "view PDF after running", press OK. For default - set to latexmk.

### switching between qt4 and qt5 with qtchooser:

I reinstalled Ubuntu 13.04 on a machine (comes with qt4) overtop a 13.10 installation (comes with qt5). I found when building through qtcreator that there were errors in relation to using the qt5 libraries rather than qt4. The program "qmake" is a symlink to the program "qtchooser", which was configured to use qt5. To set it back to qt4, the /usr/share/default.conf symlink needs to point to qt4.conf:

```
cd /usr/share/qtchooser
sudo rm default.conf
sudo ln -s qt4.conf default.conf
```

### open a terminal through nautilus:

Use the following command to install the extension:

```
sudo apt-get install nautilus-open-terminal
```

### youtube issues with chromium:

On an Ubuntu 12.04 installation with chromium, youtube videos often fail to load with the message "This video is currently unavailable". The fix (in my case):

```
sudo apt-get remove chromium-codecs-ffmpeg
sudo apt-get install chromium-codecs-ffmpeg-extra
```

### hostnames, avahi and .local:

You may find you are unable to connect using the hostname to another computer on the network (e.g. for a computer whose hostname is "whirringbox"):

```
ping whirringbox
ping: unknown host whirringbox
```

Ubuntu (and others) include software called avahi, "zero-configuration networking software" which includes support for something called mDNS (or the "multicast Domain Name System"). On Ubuntu installations, there will be a program called avahi-daemon running. For example, if we run on whirringbox:

```
ps -ax | grep avahi
```

this displays

```
  684 ?          S       0:00 avahi-daemon: running [whirringbox.local]
  687 ?          S       0:00 avahi-daemon: chroot helper
 8350 pts/0     S+      0:00 grep --color=auto avahi
```

The purpose of the *avahi-daemon* process is to make the computer accessible on the local network. Supporting mDNS, *avahi-daemon* waits for a message on the local network requesting the host with name "whirringbox.local" to identify itself. Then, the *avahi-daemon* replies to this message, providing the IP address of whirringbox. So in general, to connect to an Ubuntu machine by hostname for an mDNS-supporting system, you need to add ".local":

```
ping whirringbox.local
PING whirringbox.local (192.168.0.11) 56(84) bytes of data.
64 bytes from whirringbox.local (192.168.0.11): icmp_req=1 ttl=64 time=0.543 ms
64 bytes from whirringbox.local (192.168.0.11): icmp_req=2 ttl=64 time=0.150 ms
64 bytes from whirringbox.local (192.168.0.11): icmp_req=3 ttl=64 time=0.152 ms
```

I find this a great alternative to the idea of setting up static IP addresses and messing with */etc/hosts*.

## linker errors using webkitwidgets:

When attempting to add a webpage viewer to my Linux Qt5 application (Ubuntu 14.04) via a "QWebView", part of the "webkitwidgets" module (QT += webkitwidgets), I was missing some development libraries which were causing numerous linker errors (libraries which could not be found). Many related to gstreamer, but there were a few others. Installing the following few packages fixed this for me:

```
sudo apt-get install libgstreamer0.10-dev
sudo apt-get install libxslt-dev
sudo apt-get install libgstreamer-plugins-base0.10-dev
```

## convert a PDF into a series of images:

Use the command line utility pdftoppm:

```
pdftoppm -png slides.pdf slide_imgs/slide
```

where -png defines the format (e.g. png, jpg) can be specified, slides.pdf is the input file, and slide_imgs/slide is the prefix for each filename (this puts slides into a slide_imgs subdirectory).