Linux Notes

TOC

Tips, Tutorials and Documentation

All Man topics Alphabetically

Links from SS64

- The Linix Documentation Project & Linux man pages online.
- Explain Shell - Enter an Ubuntu (bash) command-line to see the help text that matches each argument.
- Shell Check - Find bugs in your shell scripts.
- Shell Check - Static analysis tool - alerts for many common beginner's syntax errors.
- Command Linefu- head to head voting on best one line bash tricks. Lots of gold.
- Debian documentation
- The Linux Cookbook - Tips and Techniques.
- Google Shell Style guide - some short, sensible advice on coding style.
- Hyperpolyglot - Comparison of bash vs cmd vs PowerShell.
- Linux Shell Scripting Tutorial - LSST
- Unix/Linux/BSD Toolbox - Concise summary of commands.
- Administration shell scripts - Dawid Michalczyk.
- grymoire.com - home for UNIX wizards.
- Introduction to text manipulation on UNIX-based systems
- LWN - Linux Weekly News
- UbuntuForums - HowTo guides and forum
- Introduction to the Ubuntu terminal
- UNIX FAQ
- TCP and UDP port numbers - wikipedia
- JustLinux - Forum.
- System76 - OSHWA certified open source hardware

Bash

- Greg Wooledge Bash Guide and FAQ
- Bash Guide for beginners - Machtelt Garrels.
- GNU Bash Manual - gzip archives.
- Documentation and Examples - bash cook book.
- bash tips - Configure the bash terminal.
- Bash pitfalls - GreyCat's wiki.

WSL (Windows Subsystem Linux)

Install ZSH

ZSH in WSL

ZSH in VS Code (Mac)

Oh My ZSH

Antigen

ZSH, ZSH-antigen (package manager)

```
sudo apt-get install zsh zsh-common zsh-antigen

sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"

chsh -s $(which zsh)
```

1- installs ZSH, ZSH-common, and antigen (a theme manager)

Sample `.zshrc`

```
source /path-to-antigen/antigen.zsh

#Load the oh-my-zsh's library.
antigen use oh-my-zsh

#Bundles from the default repo (robbyrussell's oh-my-zsh).
antigen bundle git
antigen bundle heroku
antigen bundle pip
antigen bundle lein
antigen bundle command-not-found

#Syntax highlighting bundle.
antigen bundle zsh-users/zsh-syntax-highlighting

#Load the theme.
antigen theme robbyrussell

#Tell Antigen that you're done.
antigen apply
```

Setup and Running

Pressing escape

`Alt-2` gets out of the boot screen and gets to a prompt. `Alt-1` returns.

Work Computer

WSL - Windows Subsystem for Linux

alt 2639

File Structure

- `/` - **Root**
  - `bin` - **Binary** - biinaries are stored here, like *ls* and other things a regular single user might use.
  - `sbin` - System Binaries -- binary tools a system administrator might use.
  - `boot` - boot related files
  - `dev` - devices- devoces are mounted here. along with their drivers.
    - `sda`
    - `sdb` ... etc- hard disk mounting.
  - `etc` - Et cetera - "edit to configure" mneumonic- system wide settings- not per user settings
  - `home`
  - `lib`
    - `lib32` , `lib64` - libraries required for bin
  - `media` - Media directory, used to be called `mnt` or in general most OS'es manage media for you by mounting devices (typically removable devices) while the user can use mount. Things like USB sticks, external drives are found in `media` or `mnt` ;
  - `opt` - contains manually installed software from vendors resides. Usually user installed software doesnt go here. But User created software can go here.
  - `proc` - process - Psudofiles that contain info on system processes or resources. Each Process has its own "pseudofile". These are not actual files. CPU information is in here.
  - `root` - Root User Home Folder- not a typical user directory. Files stored here require root access. This was to make sure that if user directories were on another directory, it wouldnt be lost.

- `run` - variable- TempFS file system- like a temp directory. It is stored in Ram and lost when the computer reboots.
- `snap` -Snap Package directory- where Ubuntu stores the Snap packages.
- `srv` - Service Directory- usually empty, but if computer is a webserver- then website or FTP files are stored here.
- `sys` - System Files- contains Kernal pseudofiles. It is not permanent. Allows very low level access to kernal- for things like updating Video Card Settings.
- `tmp` - Temporary- this is for applications to use during a session- things like temproray copies of a Document in progress. A crashed program may lose access to the folder and it can begin to fill. Might need to be cleared by Root user in Single User Mode.
- `usr` - User or Unix System Resources - holds applications installed by the user. These must be considered non-essential.
    - `bin` - binaries
    - `sbin` - administration binaries
    - `lib` - libraries., also `lib32` and `lib64`
    - `src` Source - where programs installed from source are stored.
    - `local` - to indicate machine local users--
        - `bin` - binaries
        - `sbin` - administration binaries
        - `lib` - libraries., also `lib32` and `lib64`
        - `src` Source - where programs installed from source are stored.
    - `share` - The site for larger program installs.
        - `bin` - binaries
        - `sbin` - administration binaries
        - `lib` - libraries., also `lib32` and `lib64`
        - `src` Source - where programs installed from source are stored.
        - `local` - to indicate machine local users--
            - `bin` - binaries
            - `sbin` - administration binaries
            - `lib` - libraries., also `lib32` and `lib64`
            - `src` Source - where programs installed from source are stored.
- `var` - Variable -this folder contains files that are expected to grow in size
    - `log` - contains log files
    - `crash` contains information on crashes.
    - `spool` contains printer documents
    - `cache` contains cached folders
- `home` - links variably to each users home directory.
    - `.cache` - stores program cache
    - `.config` - Application (user specific) configuration settings. Desktop settings, etc.
    - `.themes` - Desktop Themes.
    - `.*` - hidden files for programs.

File permissions

Basically its { `d` / `-` }(`u` ser) `rwx` (`g` roup) `rwx` (`o` thers) `rwx` w/ `-` turning it off.
Access Levels

- None (-)
- read (r)
- write (w)
- execute (x)

You can see these by using the ls -l Unix command.

0123456789

```
drwxr-xr-x
```

```
-rwxr-xr-x
```

```
-rwxrwxrwx
```

The first character represents the entry type. The entry type for a directory is 'd'.

All files have a hyphen (-) as their entry type. The remaining nine characters indicate the permissions themselves in 3 groups of three.

1 through 3 show user (u) permissions;
4 through 6 are group (g) permissions;
7 through 9 are other (o) permissions (users who are not the owner and are not members of the group own the file/directory, a.k.a. "the world").

can also take away the read and write permissions by doing the following:

chmod o-rw /home/yourusername

CHMOD

to give the world permission to read and write to your home directory, then you would use the command below:

chmod o+rw /home/youruserna

Path

```
code ~/.bashrc
```

```
alias python ='/usr/bin/python3.7'
```

Useful commands

Show Toilet Fonts

```
for i in ${TOILET_FONT_PATH:=/usr/share/figlet}/*.{t,f}lf; do j=${i##*/}; toilet -d "${i%/*}" -f "$j" "${j%.*}"; done
```

requires toilet package

tput

tput setb 4 -sets the bg color

Quick Tips

A collection of short (1 liner or so) quick bash commands

- `sudo !!` redo last command but as root
- Open an editor to run a command
  `ctrl+x+e`
- create a super fast ram disk
  `mkdir -p /mnt/ram`
  `mount -t tmpfs tmpfs /mnt/ram -o size=8192M`
- Then test it by creating a blockfile
  `dd if=/dev/zero of=test.-0iso bs=1M count=8000`
  `rm test.iso`
  `mount -t tmpfs tmpfs /mnt/ram -o size=8192M`
  `cd ram/`
- Don't add command to history **(note the leading space)**
  `_ls -l` - Underscore added for clarity
  -Show the history
  `history`
  -clear the screen
  `clear`
- fix a really long command that you messed up
  `fc`
- tunnel with ssh (local port 3337 -> remote host's 127.0.0.1 on port 6379)
  `ssh -L 3337:127.0.0.1:6379 root@emkc.org -N`
  `bg`
  `redis-cli -p 3337`

Not sure I understand this fully. SSH is opened on 3337 to connect to 127...via 6379, followed by a login?I think the redis-cli is to demonstrate its success.

-quickly create folders
`mkdir -p folder/{sub1,sub2}/{subA,subB,subC}`
$/1/A,/1/B...3/C/

- Create folders 1 through 10 each with 10 folders in it.
  `mkdir -p folder/{1..10}/{1..10}`

-intercept stdout and log to file
`cat file | tee -a log | cat > /dev/null`
-exit terminal but leave all processes running
`disown -a && exit`
-Use the last term of the last command- !'! $mkdir - p/path/to/new/dir$" cd !$`

- Set the hostname
  `hostname`
  `hostname -f` fully qualified domain name

Set the timezone
`dpkg-reconfigure tzdata #Debian/Ubuntu`
`timedatectl list-timezones #CentOS or ArchLinux`
`timedatectl set-timezone 'America/Chicago'`

Find the appropriate zone file in /usr/share/zoneinfo/ and link that file to /etc/localtime. See the examples below for possibilities:
`ln -sf /usr/share/zoneinfo/EST /etc/localtime`
`ln -sf /usr/share/zoneinfo/US/Eastern /etc/localtime`

`/etc/hosts`
`mtr #track the speed of a connection between two connections`
`mtr --report`

- Check memory usage
  `free -m`
- Monitor IO usage with vmstat -runs a vmstat every second x 20
  `vmstat 1 20`
- Monitor processes, memory and CPU - commands for various distros below
  `apt-get install htop`
  `yum install htop`
  `pacman -S htop`
  `emerge sys-process/htop`
  - to run
    `htop`
  - quit with F10 or Q, run 'setup' with F2, Show tree view with F5

Kill App

press Ctrl+Z to suspend the script, then `kill %%` {style="font-style:Courier"}

The `%%` tells the bash built-in kill that you want to send a signal (`SIGTERM` by default) to the most recently suspended background job in the current shell, not to a process-id.

System Summary (inxi)

`inxi -Fxxxz`

Provides a summary of the system, often for sharing on forums and such.

- `-F` is "all", each x is a verbosity level
- `-H` will provide help
- `-A` audio
- `-C` cpu
- `-D` , -Dd Disk / optical

- `-G` graphics
- `-m` memory
- `-M`
- `-N` network
- `-Nn` advanced network
- `-P`
- `-R`
- `-S` system

```
inxi -G
```

Show the graphics info

apt

Commands

sudo apt-get update

sudo apt

apt-get:

- build-dep
- update - update the package lists
- upgrade - upgrade the packages
- auto-remove - clean out dead dependencies
- clean
- install
  - -f try and fix broken dependencies
  - -force carries strong warnings

E: Unable to correct problems, you have held broken packages.
apt-mark showhold

sudo apt-get install aptitude

apt list --installed

man - Get the Manual

`man` gets the manual

The manual has 8 main sections-

1. Executable programs or shell commands
2. System calls (functions provided by the kernel)
3. Library calls (functions within program libraries)
4. Special files (usually found in /dev)
5. File formats and conventions eg /etc/passwd
6. Games
7. Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8. System administration commands (usually only for root)
9. Kernel routines [Non standard]

Ok 9. This is the number in the [command] help >> see command(#) bit.

Ironically, `man man` gets help with the manual.

Less

Less is used to view man, its like linux notepad.

Less Doc
SS64 Brief Notes

frwrd/down/next	back/up/prev		action
g <	G >		TO beginning or End
f {space}	b	*	move one window
z	w	**	move (N1) lines (Nominally 1 window), #z sets N1
d	u	**	move (N2) lines (Nominally 1/2 window), #d/u sets N2
e j {Enter}	y k	*	move one *(N) line
{R arrow}	{L arrow}		Scroll Sideways
/pattern	?pattern		Start a search
n	N		Go to the next found item
M{letter}	m{letter}		Mark position (A...B...C.etc.) at bottom / top of page. ***
'$	'^		(Mark syntax) Go to End/Beginn.
t	T	*	Go to the next (*Nth) Tag
{ ([	]) }	*	Find the next (*Nth) open/close bracket. Search is rev for Open.
:n	:p	*	Examine the next (*Nth) file from command line ****

(*) the command can be preceded by an integer to repeat the command N times. E.g. `5e` moves forward 5 lines.
(**) the command can be preceeded by a number and repeated N times, and also this value is remembered for future commands. `d` moves forward "1/2 window", but `10d` moves forward 10 lines, and all future presses of `d` by itself will then move 10 lines. Half height and full height store separate values of N. You can then type `3z` and have the window advance move just 3 lines, it doesnt have to be more. Paired commands share a value.

Marking

(***) M{letter} marks the bottom and `m` marks at the top. You can go to a marker by pressing single quote `'` and the desired letter. `''` (Double single quote) goes back a mark (also `[Ctrl]x, [Ctrl]x`). Deleta a mark with `[Esc]M<letter>`

Upper case and lower case are distinct. Some are predefined.

Searching

`F` - to the end, `[Esc]F` advances towards end but stops if search string is found. You can also search adjacent files. See help.

supposedly typing /pattern or ?pattern kicks off a search but it didnt work. Can be (Needs to be?) preceeded by `!` Non, `*` search multiple files `@` start at first (/) or last (?) file, `[Ctrl]K` highlight only, `[Ctrl]R` Dont use regular expressions.
`/pattern` - Search forward, `?pattern` search back, `n` - next find, `N` Previous. `&pattern` to show only matching lines.

Maybe its not working bc it expects a regexp.

Moving files

`e:` will change to another file. So will `[Ctrl]X,[Ctrl]V`

`:x` goes to the first file, and `:d` deletes the current file. `:f` and `=` print the cile name.

`p(%)` or rather `p##` goes to the `##%` point in a file. p50 goes to the halfway point
`r` - rerun screen

`R arrow` - scroll right
`L arrow` - scroll left
`h` open help

Options

Miscellaneous Commands:
-flag Toggle a command line option [see OPTIONS below].
--name Toggle a command line option, by name.
_flag Display the setting of a command line option.
__name Display the setting of an option, by name.
+cmd Execute the less cmd each time a new file is examined.

!command Execute the shell command with ParseError: KaTeX parse error: Expected 'EOF', got '&' at position 52: …en current pos &̲mark X to shel…
VISUAL or $EDITOR.
V Print version number of "less".
[flag]^P (CONTROL-P) Suppress change message

OPTIONS:
Most options can be changed either on the command line, or from within less by using the - or -- command.
Options can be given in one of two forms: either a single character preceded by a -, or a name preceeded by --.

Short	Long	Breif Description
	FINDING	
-a	--search-skip-screen	Forward search, skips current screen.
-g	--hilite-search	Highlight only last match for searches.
-G	--HILITE-SEARCH	Don't highlight any matches for searches.
-#g		Opens at a particular line number
-h[n]	--max-back-scroll=n	Backward scroll limit.
-I, -I	--ignore-case, --IGNORE-CASE	Ignore case in searches.
		Ignore case in searches and in search patterns.
-j[n]	--jump-target=n	Screen position of target lines.
-p[pattern]	--pattern=pattern	Start at pattern (from command line).
-t[tag]	--tag=tag	Find a tag.
-T[tagsfile]	--tag-file=tagsfile	Use an alternate tags file.
-y[n]	--max-forw-scroll=n	Forward scroll limit.
	DISPLAY	
-c	--clear-screen	Repaint by scrolling/clearing.
-C	--CLEAR-SCREEN	Repaint by scrolling/clearing.
-D[xcolor]	--color=xcolor	Set screen colors. (MS-DOS only)
-m, -M	--long-prompt, --LONG-PROMPT	Set prompt style.
-n, -N	--line-numbers,	Use line numbers.
	--LINE-NUMBERS	
-P[prompt]	--prompt=prompt	Define new prompt.
-r, -R	--raw-control-chars, --RAW-CONTROL-CHARS	Output "raw" control characters.
-s	--squeeze-blank-lines	Squeeze multiple blank lines.
-S	--chop-long-lines	Chop long lines
-w, -W	--hilite-unread, --HILITE-UNREAD	Highlight first new line after forward-screen
-[z]n	--window=n	Set size of window.

Short	Long	Breif Description
-~	--tilde	Don't display tildes after end of file.
	FILE OPTIONS:	
-bn	--buffers=n	Number of buffers.
-B	--auto-buffers	Don't automatically allocate buffers for pipes.
-f	--force	Force open non-regular files.
-k[filename]	--lesskey-file=filename	Use a lesskey file.
-o[filename] , -O	--log-file=filename	Copy to log file (standard input only).
-Ofilename	--LOG-FILE=filename	Copy to log file (unconditionally overwrite).
	MISC OPTIONS:	
-?	--help	Display help (from command line).
-d	--dumb	Dumb terminal.
-e, -E	--quit-at-eof, --QUIT-AT-EOF	Quit at end of file.
-F	--quit-if-one-screen	
-J	--status-column	Show Status Column on Left
-K	--quit-on-intr	
-q, -Q	--quiet or --silent	Quiet the terminal bell.
-u, -U	--underline-special, --UNDERLINE-SPECIAL	Change handling of backspaces.
-V	--version	Display the version number of "less".
-xn,...	--tabs=n,...	Set tab stops
-X	--no-init	Don't use termcap init/deinit strings.
-" [c[c]], -dqcc	--quotes=[c[c]]	Set shell quote characters.
-L	--no-lessopen	Ignore the LESSOPEN environment variable
-#	--shift	Specifies the default number of positions to scroll horizontally
	--no-keypad	Disables sending the keypad initialization and deinitialization strings to the terminal
	--follow-name	
--		marks the end of option arguments. Any arguments following this are interpreted as filenames.
+		The remainder of that option is taken to be an initial command

more

Display output one screen at a time, less provides more emulation and extensive enhancements.

Syntax

```
`more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]
```

`

Options

Command line options are described below. Options are also taken from the environment variable MORE (make sure to precede them with a dash (``-'')) but command line options will override them.

- `-num` This option specifies an integer which is the screen size (in lines).
- `-d` more will prompt the user with the message "[Press space to continue, 'q' to quit.]" and will display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.
- `-l` more usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.
- `-f` Causes more to count logical, rather than screen lines (i.e., long lines are not folded).
- `-p` Do not scroll. Instead, clear the whole screen and then display the text.
- `-c` Do not scroll. Instead, paint each screen from the top, clearing the remainder of each line as it is displayed.
- `-s` Squeeze multiple blank lines into one.
- `-u` Suppress underlining.
- `+/` The +/ option specifies a string that will be searched for before each file is displayed.
- `+num` Start at line number num.

COMMANDS

Interactive commands for more are based on vi(1). Some commands can be preceeded by a decimal number, called k in the descriptions below. In the following descriptions, `^X` means control-X.

- `h` , `?` Help: display a summary of these commands. If you forget all the other commands, remember this one.
- `SPACE` Display next k lines of text. Defaults to current screen size.
- `z` Display next k lines of text. Defaults to current screen size. Argument becomes new default.
- `RETURN` Display next k lines of text. Defaults to 1. Argument becomes new default.
- `d` , `^D` Scroll k lines. Default is current scroll size, initially 1. Argument becomes new default.
- `q` , `Q` ,
  - `INTERRUPT` Exits the more command.
- `s` Skip forward k lines of text. Defaults to 1.
- `f` Skip forward k screenfuls of text. Defaults to 1.
- `b` , `^B` Skip backwards k screenfuls of text. Defaults to 1.
- `'` Go to place where previous search started.
- `=` Display current line number.
- `/pattern` Search for kth occurrence of regular expression. Defaults to 1.
- `n` Search for kth occurrence of last r.e. Defaults to 1.
- `!` , `:!` Execute in a subshell
- `v` Start up /usr/bin/vi at current line
- `^L` Redraw screen
- `:n` Go to kth next file. Defaults to 1.
- `:p` Go to kth previous file. Defaults to 1.
- `:f` Display current file name and line number
- `.` Repeat previous command

ENVIRONMENT

More utilizes the following environment variables, if they exist:

MORE This variable can be set with favored options to more.
SHELL Current shell in use (normally set by the shell at login

ls

ls

-a all
-i index number
-R Recurse
-s
-hyperlink

Converting Paths

~~~~
file=${file/C://c}
file=${file//\\//}
echo $file

#Rewritten As
file=${file/C://c}
file=${file//\\//}

#Using Filenames as an array
filenames=(
  'C:\Users\abcd\Downloads\testingFile.log'
  # ... add more here ...
)

for f in "${filenames[@]}"; do
  f="${f/C://c}"
  f="${f//\\//}"
  echo "$f"
done
#If you want to put the output into an array instead of printing, replace the echo line with an assignment:

  filenames_out+=( "$f" )

#Shell Function
win2lin () { f="${1/C://c}"; printf '%s\n' "${f//\\//}"; }

$ file='C:\Users\abcd\Downloads\testingFile.log'
$ win2lin "$file"
/c/Users/abcd/Downloads/testingFile.log
$ file='C:\Users\pqrs\Documents\foobar'
$ win2lin "$file"
/c/Users/pqrs/Documents/foobar

#Option3
file="$(echo "$file" | sed -r -e 's|^C:|/c|' -e 's|\\|/|g')"

#or
file="$(sed 's|^C:|/c|; s|\\|/|g' <<< "$file")"

#Or Using tr
file="/$(echo 'C:\Users\abcd\Downloads\testingFile.log'|tr '^C' 'c'|tr '\\' '/')"

file="/mnt/$(echo $file|tr '^C' 'c'|tr '^D' 'd'|tr '^E' 'e'|tr '^F' 'f'|tr '^G' 'g'|tr '^H' 'h'|tr 'q^I' 'i'|tr '^J' 'j'|tr '^K' 'k'|tr '^L' 'l'|

file="/mnt/$(echo $file|tr '^C' 'c'|tr '^D' 'd'|tr '^E' 'e'|tr '^F' 'f'|tr '^G' 'g'|tr '^H' 'h'|tr 'q^I' 'i'|tr '^J' 'j'|tr '^K' 'k'|tr '^L' 'l'|

### ZSH

[Note for WSL](https://stackoverflow.com/questions/49887469/why-wont-zshrc-load)

see /etc/zsh
nan
`chsh -s $(which zsh)`

`echo $0`

Installing "Oh My ZSH"
`sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"`

`open ~/.zshrc`

use `**` to descend into directories. `ls **/*.pyc`

Use globs for patterns to search for files. `ls *.(py|sh)`

Use glob modifiers ls -l \*(@)

`kill doc` to list running processes

Tab completion- navigate with cursor keys.

use `r` to repeat the last command with substitutions

```bash
touch foo.htm bar.htm
mv foo.htm foo.html
~~~~

```
r foo=bar
mv bar.htm bar.html
```

plugins:

```
ls ~\.oh-my-zsh\plugins\
```

```
ls ~\.oh-my-zsh\themes\
```

adb git-flow-avh powify
ant git-hubflow profiles
apache2-macports git-prompt pyenv
arcanist git-remote-branch pylint
archlinux gitfast python
asdf github rails
autoenv gitignore rake
autojump glassfish rake-fast
autopep8 globalias rand-quote
aws gnu-utils rbenv
battery go rbfu
bbedit golang react-native
bgnotify gpg-agent rebar
boot2docker gradle redis-cli
bower grails repo
branch grunt rsync
brew gulp ruby
bundler hanami rust
bwana helm rvm
cabal heroku safe-paste
cake history salt
cakephp3 history-substring-search sbt
capistrano homestead scala
cargo httpie scd
cask iterm2 screen
catimg iwhois scw
celery jake-node sfffe
chruby jenv shrink-path
chucknorris jhbuild singlechar
cloudapp jira spring
cloudfoundry jruby sprunge
codeclimate jsontools ssh-agent
coffee jump stack
colemak kate sublime
colored-man-pages kitchen sudo
colorize knife supervisor
command-not-found knife_ssh suse
common-aliases kops svn
compleat kube-ps1 svn-fast-info
composer kubectl swiftpm
copybuffer laravel symfony
copydir laravel4 symfony2
copyfile laravel5 systemadmin
cp last-working-dir systemd
cpanm lein taskwarrior
dash lighthouse terminalapp
debian lol terminitor
dircycle macports terraform
dirhistory magic-enter textastic
dirpersist man textmate
django marked2 thefuck

dnf mercurial themes
docker meteor thor
docker-compose minikube tig
docker-machine mix tmux
doctl mix-fast tmux-cssh
dotenv mosh tmuxinator
droplr mvn torrent
eecms mysql-macports transfer
emacs n98-magerun tugboat
ember-cli nanoc ubuntu
emoji ng ufw
emoji-clock nmap urltools
emotty node vagrant
encode64 nomad vagrant-prompt
extract npm vault
fabric npx vi-mode
fancy-ctrl-z nvm vim-interaction
fasd nyan virtualenv
fastfile oc virtualenvwrapper
fbterm osx vscode
fedora pass vundle
firewalld paver wakeonlan
forklift pep8 wd
fossil per-directory-history web-search
frontend-search percol wp-cli
fzf perl xcode
gas perms yarn
gb phing yii
geeknote pip yii2
gem pj yum
git pod z
git-auto-fetch postgres zeus
git-extras pow zsh-navigation-tools
git-flow powder zsh_reload

ZSH Profile

for wsl is /etc/zsh/zshrc

```
echo "Config"

ZSH=$HOME/.oh-my-zsh

ZSH_Theme="agnoster"
#ZSH_Theme="afowler"
#ZSH_Theme="af-magic"
#ZSH_Theme="alanpeabody"
#ZSH_Theme="amuse"
#ZSH_Theme="arrow"
#ZSH_Theme="astm"
#ZSH_Theme="avit"
#ZSH_Theme="bira"
#ZSH_Theme="blinks"
#ZSH_Theme="bureau"
#ZSH_Theme="cloud"
#ZSH_Theme="dallas"
#ZSH_Theme="dieter"
#ZSH_Theme="dogenpunk"
#ZSH_Theme="dpoggi"
#ZSH_Theme="evan"
#ZSH_Theme="flazz"
#ZSH_Theme="funky"
#ZSH_Theme="gallois"
#ZSH_Theme="gentoo"
#ZSH_Theme="geoffgarside"
#ZSH_Theme="gianu"
#ZSH_Theme="gnzh"
#ZSH_Theme="jtriley"
#ZSH_Theme="junkfood"
#ZSH_Theme="kfeitu"
#ZSH_Theme="kardan"
#ZSH_Theme="kolo"
#ZSH_Theme="lukerandall"
#ZSH_Theme="mh"
#ZSH_Theme="michelebologna"
#ZSH_Theme="miloshadzic"
#ZSH_Theme="muse"
#ZSH_Theme="nebirhos"
#ZSH_Theme="phillips"
#ZSH_Theme="pygmalion"
#ZSH_Theme="rixius"
#ZSH_Theme="rkj-repos"
#ZSH_Theme="smt"
#ZSH_Theme="steeef"
#ZSH_Theme="sunrise"
#ZSH_Theme="superjarin"
#ZSH_Theme="theunraveler"
#ZSH_Theme="tjkirch"
#ZSH_Theme="tonotdo"
#ZSH_Theme="wezm"
#ZSH_Theme="wedisagree"
#ZSH_Theme="wuffers"
#ZSH_Theme="xiong-chiamiov"
#ZSH_Theme="xiong-chiamiov-plus"
#ZSH_Theme="ys"
#ZSH_Theme="zhann"
#ZSH_Theme="wedisagree"
#ZSH_Theme="random"
#ZSH_THEME_RANDOM_CANDIDATES=(
    #"robbyrussell"
    #"agnoster"
#)

#(Will change per machine)
DEFAULT_USER="alt"

#Dots are pretty.
COMPLETION_WAITING_DOTS="true"

#Custom plugins may be added to ~/.oh-my-zsh/custom/plugins/
plugins=(git ruby history-substring-search rbenv python pip pylint vscode)

#Load the good stuff.
source $ZSH/oh-my-zsh.sh
```

```
#-------------------------------
#Exports
#---------------------------------
#Make sure we know which python is first.
#export PATH=/Library/Frameworks/Python.framework/Versions/2.7/bin:$PATH
#export ZSH=/Users/alt/.oh-my-ZSH
export EDITOR="nano"

autoload -U compinit colors vcs_info
colors
compinit

#Report command running time if it is more than 3 seconds
REPORTTIME=3
#Keep a lot of history
HISTFILE=~/.zhistory
HISTSIZE=5000
SAVEHIST=5000
#Add commands to history as they are entered, don't wait for shell to exit
setopt INC_APPEND_HISTORY
#Also remember command start time and duration
setopt EXTENDED_HISTORY
#Do not keep duplicate commands in history
setopt HIST_IGNORE_ALL_DUPS
#Do not remember commands that start with a whitespace
setopt HIST_IGNORE_SPACE
#Correct spelling of all arguments in the command line
setopt CORRECT_ALL
#Enable autocompletion
zstyle ':completion:*' completer _complete _correct _approximate

#------------------
#Prompt
#-----------------
zstyle ':vcs_info:*' stagedstr '%F{green}●%f '
zstyle ':vcs_info:*' unstagedstr '%F{yellow}●%f '
zstyle ':vcs_info:git:*' check-for-changes true
zstyle ':vcs_info:git*' formats "%F{blue}%b%f %u%c"

_setup_ps1() {
  vcs_info
  GLYPH="▲"
  [ "x$KEYMAP" = "xvicmd" ] && GLYPH="▼"
  PS1=" %(?.%F{blue}.%F{red})$GLYPH%f %(1j.%F{cyan}[%j]%f .)%F{blue}%~%f %(!.%F{red}#%f .)"
  RPROMPT="$vcs_info_msg_0_"
}
_setup_ps1

#Vi mode
zle-keymap-select () {
 _setup_ps1
  zle reset-prompt
}
zle -N zle-keymap-select
zle-line-init () {
  zle -K viins
}
zle -N zle-line-init
bindkey -v

#----------------------
#Alias
#----------------------

alias python='/usr/bin/python3.7'

#cd, because typing the backslash sucks
alias .='cd ../'
alias ..='cd ../../'
alias ...='cd ../../../'
alias ....='cd ../../../../'

alias del='echo Moving to ~/.Trash/ ...; mv -i $* ~/.Trash/'
#alias cdf='eval `osascript /Applications/Utilities/OpenTerminal.app/Contents/Resources/Scripts/OpenTerminal.scpt`'
#alias ls='ls -FG'
#alias dir='ls -FGl'
#alias ll="ls -l"
```

```
#Safe options
#alias rm='rm -i'
#alias mv='mv -i'
#alias cp='cp -i'
```

Dos2Unix

```
sudo apt install dos2unix
```

This is supposed to be a program for converting paths. Unknown more at this time.

Bash

Bash Reference Manual

Pipelines

Bash Setup

PATH

Path variables are in the home directory named

- `.bash_profile`
- `.profile`
- `.bashrc`

To see it you have to show hidden files - `ls -a`

Startup

```
sudo apt install screenfetch
```

The above has a bug.

Use:

```
wget https://raw.githubusercontent.com/KittyKatt/screenFetch/master/screenfetch-dev
chmod +x screenfetch-dev
./screenfetch-dev
#if the old one (above) was installed-
sudo rm /usr/bin/screenfetch
remname ./screenfetch-dev screenfetch
sudo mv screenfetch /user/bin
```

sudo apt install toilet
sudo apt install figlet

Usage

Run a bash shell script

A shell script is a text file containing one or more commands.

```
#!/bin/bash
#My example bash script
echo "Hello World"
```

The first line contains a **shebang** `#!` followed by the path to the shell, in this case bash - this acts as an interpreter directive and ensures that the script is executed under the correct shell.

The " `#` " character indicates a comment, so the shebang line is ignored by bash when running the script.

Next you need to make the script executable with chmod

```
$ chmod u+x my_script1.sh
```

You can now run the script by prefixing it with ./

```
$ ./my_script1.sh
```

Naming the file with an .sh extension is not required, but you may find it a helpful reminder that the file is a shell script.

If you will be writing a few shell scripts then it's worth creating a folder, perhaps called "scripts" and adding that to the system path:

```
$ mkdir -p $HOME/scripts
>
$ export PATH="$PATH:~/scripts"
>
```

Even better is to edit your .bash_profile file to include export PATH="$PATH:~/scripts" that will keep the "scripts" folder in your path every time you log in.

With the script saved in the folder, you can now run it with just:

```
$ my_script1.sh
```

If this returns an error like #: bad interpreter: No such file or directory that is typically an indication that the File encoding or line endings are wrong, use an editor like VI or just make sure the text is plain ASCII/Unicode and the line endings are classic UNIX LF (not Windows CR/LF or Mac CR)

Passing parameters to a shell script:

```
$ cat myscript2.sh
#!/bin/bash
echo 'Welcome' $0 'says' $1 $2

$ chmod a+x myscript2.sh
$ myscript2.sh Hello world
Welcome myscript2.sh says Hello world
```

Running scripts from the system path
A shell script is a text file containing shell commands. When such a file is used as the first non-option argument when invoking Bash, and neither the `-c' nor `-s' option is supplied, Bash reads and executes commands from the file, then exits. This mode of operation creates a non-interactive shell.

A shell script can be made executable by using the chmod command to turn on the execute bit. When Bash finds such a file while searching the $PATH for a command, it spawns a subshell to execute it.

In other words, executing `filename arguments` is equivalent to executing `bash filename arguments` if filename is an executable shell script. This subshell reinitializes itself, so that the effect is as if a new shell had been invoked to interpret the script, with the exception that the locations of commands remembered by the parent are retained by the child.

Most versions of Unix make this a part of the operating system's command execution mechanism. If the first line of a script begins with the two characters `#!', the remainder of the line specifies an interpreter for the program. Thus, you can specify Bash, awk, Perl, or some other interpreter and write the rest of the script file in that language.

The arguments to the interpreter consist of a single optional argument following the interpreter name on the first line of the script file, followed by the name of the script file, followed by the rest of the arguments. Bash will perform this action on operating systems that do not handle it themselves. Note that some older versions of Unix limit the interpreter name and argument to a maximum of 32 characters.

Bash scripts often begin with #! /bin/bash (assuming that Bash has been installed in /bin), since this ensures that Bash will be used to interpret the script, even if it is executed under another shell.

When Bash runs a shell script, it sets the special parameter $0 to the name of the file, rather than the name of the shell, and the positional parameters are set to the remaining arguments, if any are given. If no additional arguments are supplied, the positional parameters are unset.

Adding the following two set options to your scripts can be useful to catch undefined variables ("") and to exit if a command fails:

```
#!/bin/bash set -o nounset set -o errexit
```

Errors:

```
/bin/bash^M: bad interpreter: no such file or directory
```

This usually means the file has been edited and has Windows `<CR-LF>` instead of unix `<LF>` line endings

Quotes and Strings

Escape

A non-quoted backslash `\` is the Bash escape character. It preserves the literal value of the next character that follows, with the exception of newline. If a `\newline` pair appears, and the backslash itself is not quoted, the \newline is treated as a line continuation (that is, it is removed from the input stream and effectively ignored).

Single Quotes

Enclosing characters in single quotes **(`'`) preserves the literal value of each character within the quotes.** A single quote may not occur between single quotes, even when preceded by a backslash

Double Quotes

Enclosing characters in double quotes (`"`) **preserves the literal value of all characters within the quotes, with the exception of `$`, `` ` `` (backtick), `\`, and, when history expansion is enabled, `!`.**

When the shell is in POSIX mode, the `!` has no special meaning within double quotes, even when history expansion is enabled. The characters `$` and `` ` `` retain their special meaning within double quotes (see Shell Expansions). The backslash retains its special meaning only when followed by one of the following characters: `$`, `` ` ``, `"`, `\`, or newline. Within double quotes, backslashes that are followed by one of these characters are removed. Backslashes preceding characters without a special meaning are left unmodified.

A double quote may be quoted within double quotes by preceding it with a backslash. If enabled, history expansion will be performed unless an `!` appearing in double quotes is escaped using a backslash. The backslash preceding the `!` is not removed.

The special parameters `*` and `@` have special meaning when in double quotes

ANSI C Quoting

Words of the form $'string' are treated specially. The word expands to string, with backslash-escaped characters replaced as specified by the ANSI C standard. Backslash escape sequences, if present, are decoded as follows:

- `\a` - alert (bell)
- `\b` - backspace
- `\e`, `\E` - an escape character (not ANSI C)
- `\f` - form feed
- `\n` - newline
- `\r` - carriage return
- `\t` - horizontal tab
- `\v` - vertical tab
- `\\` - backslash
- `\'` - single quote
- `\"` - double quote
- `\?` - question mark
- `\nnn` - the eight-bit character whose value is the octal value nnn (one to three octal digits)
- `\xHH` - the eight-bit character whose value is the hexadecimal value HH (one or two hex digits)
- `\uHHHH` - the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value HHHH (one to four hex digits)
- `\UHHHHHHHH` - the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value HHHHHHHH (one to eight hex digits)
- `\cx` - a control-x character

Bash Commands

a

- `alias` Create an alias
  `apropos` Search Help manual pages (man -k)
- `apt-get` Search for and install software packages (Debian/Ubuntu)
- `aptitude` Search for and install software packages (Debian/Ubuntu)
- `aspell` Spell Checker
- `awk` Find and Replace text, database sort/validate/index

b

- `basename` Strip directory and suffix from filenames
- `bash` GNU Bourne-Again SHell
- `bc` Arbitrary precision calculator language
- `bg` Send to background

  `bind` Set or display readline key and function bindings
- `break` Exit from a loop
- `builtin` Run a shell builtin
- `bzip2` Compress or decompress named file(s)

c

- `cal` Display a calendar
- `case` Conditionally perform a command
- `cat` Concatenate and print (display) the content of files
- `cd` Change Directory
- `cfdisk` Partition table manipulator for Linux
- `chattr` Change file attributes on a Linux file system
- `chgrp` Change group ownership
- `chmod` Change access permissions
- `chown` Change file owner and group
- `chroot` Run a command with a different root directory
- `chkconfig` System services (runlevel)
- `cksum` Print CRC checksum and byte counts
- `clear` Clear terminal screen
- `cmp` Compare two files
- `comm` Compare two sorted files line by line
- `command` Run a command - ignoring shell functions •
- `continue` Resume the next iteration of a loop •
- `cp` Copy one or more files to another location
- `cron` Daemon to execute scheduled commands
- `crontab` Schedule a command to run at a later time
- `csplit` Split a file into context-determined pieces
- `curl` Transfer data from or to a server
- `cut` Divide a file into several parts

d

- `date` Display or change the date & time
- `dc` Desk Calculator
- `dd` Convert and copy a file, write disk headers, boot records
- `ddrescue` Data recovery tool
- `declare` Declare variables and give them attributes
- `df` Display free disk space
- `diff` Display the differences between two files
- `diff3` Show differences among three files
- `dig` DNS lookup
- `dir` Briefly list directory contents
- `dircolors` Colour setup for `ls'
- `dirname` Convert a full pathname to just a path
- `dirs` Display list of remembered directories
- `dmesg` Print kernel & driver messages
- `du` Estimate file space usage

e

- `echo` Display message on screen •
- `egrep` Search file(s) for lines that match an extended expression
- `eject` Eject removable media
- `enable` Enable and disable builtin shell commands
- `env` Environment variables

- `ethtool` Ethernet card settings
- `eval` Evaluate several commands/arguments
- `exec` Execute a command
- `exit` Exit the shell
- `expect` Automate arbitrary applications accessed over a terminal
- `expand` Convert tabs to spaces
- `export` Set an environment variable
- `expr` Evaluate expressions

f

- `false` Do nothing, unsuccessfully
- `fdformat` Low-level format a floppy disk
- `fdisk` Partition table manipulator for Linux
- `fg` Send job to foreground
- `fgrep` Search file(s) for lines that match a fixed string
- `file` Determine file type
- `find` Search for files that meet a desired criteria
- `fmt` Reformat paragraph text
- `fold` Wrap text to fit a specified width.
- `for` Expand words, and execute commands
- `format` Format disks or tapes
- `free` Display memory usage
- `fsck` File system consistency check and repair
- `ftp` File Transfer Protocol
- `function` Define Function Macros
- `fuser` Identify/kill the process that is accessing a file

g

- `gawk` Find and Replace text within file(s)
- `getopts` Parse positional parameters
- `grep-` Search file(s) for lines that match a given pattern
- `groupadd` Add a user security group
- `groupdel` Delete a group
- `groupmod` Modify a group
- `groups` Print group names a user is in
- `gzip` Compress or decompress named file(s)

h

- `hash` Remember the full pathname of a name argument
- `head` Output the first part of file(s)
- `help` Display help for a built-in command
- `history` Command History
- `hostname` Print or set system name
- `htop` Interactive process viewer

i

- `iconv` Convert the character set of a file
- `id` Print user and group id's
- `if` Conditionally perform a command
- `ifconfig` Configure a network interface
- `ifdown` Stop a network interface
- `ifup` Start a network interface up
- `import` Capture an X server screen and save the image to file
- `install` Copy files and set attributes
- `iostat` Report CPU and i/o statistics
- `ip` Routing, devices and tunnels

j

- `jobs` List active jobs •
- `join` Join lines on a common field

k

- `kill` Kill a process by specifying its PID
- `killall` Kill processes by name

l

- `less` Display output one screen at a time
- `let` Perform arithmetic on shell variables •
- `link` Create a link to a file
- `ln` Create a symbolic link to a file
- `local` Create a function variable •
- `locate` Find files
- `logname` Print current login name
- `logout` Exit a login shell •
- `look` Display lines beginning with a given string
- `lpc` Line printer control program
- `lpr` Off line print
- `lprint` Print a file
- `lprintd` Abort a print job
- `lprintq` List the print queue
- `lprm` Remove jobs from the print queue
- `lsattr` List file attributes on a Linux second extended file system
- `lsblk` List block devices
- `ls` List information about file(s)
- `lsof` List open files
- `lspci` List all PCI devices

m

- `make` Recompile a group of programs
- `man` Help manual
- `mkdir` Create new folder(s)
- `mkfifo` Make FIFOs (named pipes)
- `mkfile` Make a file
- `mkisofs` Create an hybrid ISO9660/JOLIET/HFS filesystem
- `mknod` Make block or character special files
- `mktemp` Make a temporary file
- `more` Display output one screen at a time
- `most` Browse or page through a text file
- `mount` Mount a file system
- `mtools` Manipulate MS-DOS files
- `mtr` Network diagnostics (traceroute/ping)
- `mv` Move or rename files or directories
- `mmv` Mass Move and rename (files)

n

- `nc` Netcat, read and write data across networks
- `netstat` Networking connections/stats
- `nice` Set the priority of a command or job
- `nl` Number lines and write files
- `nohup` Run a command immune to hangups
- `notif` y-send Send desktop notifications
- `nslookup` Query Internet name servers interactively

o

- `open` Open a file in its default application
- `op` Operator access

p

- `passwd` Modify a user password
- `paste` Merge lines of files
- `pathchk` Check file name portability
- `Perf` Performance analysis tools for Linux
- `ping` Test a network connection
- `pgrep` List processes by name
- `pkill` Kill processes by name
- `popd` Restore the previous value of the current directory
- `pr` Prepare files for printing
- `printcap` Printer capability database
- `printenv` Print environment variables
- `printf` Format and print data
- `ps` Process status
- `pushd` Save and then change the current directory
- `pv` Monitor the progress of data through a pipe
- `pwd` Print Working Directory

q

- `quota` Display disk usage and limits
- `quotacheck` Scan a file system for disk usage

r

- `ram` ram disk device
- `rar` Archive files with compression
- `rcp` Copy files between two machines
- `read` Read a line from standard input •
- `readarray` Read from stdin into an array variable •
- `readonly` Mark variables/functions as readonly
- `reboot` Reboot the system
- `rename` Rename files
- `renice` Alter priority of running processes
- `remsync` Synchronize remote files via email
- `return` Exit a shell function
- `rev` Reverse lines of a file
- `rm` Remove files
- `rmdir` Remove folder(s)
- `rsync` Remote file copy (Synchronize file trees)

s

- `screen` Multiplex terminal, run remote shells via ssh
- `scp` Secure copy (remote file copy)
- `sdiff` Merge two files interactively
- `sed` Stream Editor
- `select` Accept keyboard input
- `seq` Print numeric sequences
- `set` Manipulate shell variables and functions
- `sftp` Secure File Transfer Program
- `shift` Shift positional parameters
- `shopt` Shell Options
- `shutdown` Shutdown or restart linux
- `sleep` Delay for a specified time

- `slocate` Find files
- `sort` Sort text files
- `source` Run commands from a file '.'
- `split` Split a file into fixed-size pieces
- `ss` Socket Statistics
- `ssh` Secure Shell client (remote login program)
- `stat` Display file or file system status
- `strace` Trace system calls and signals
- `su` Substitute user identity
- `sudo` Execute a command as another user
- `sum` Print a checksum for a file
- `suspend` Suspend execution of this shell •
- `sync` Synchronize data on disk with memory

t

- `tail` Output the last part of file
- `tar` Store, list or extract files in an archive
- `tee` Redirect output to multiple files
- `test` Evaluate a conditional expression
- `time` Measure Program running time
- `timeout` Run a command with a time limit
- `times` User and system times
- `touch` Change file timestamps
- `top` List processes running on the system
- `tput` Set terminal-dependent capabilities, color, position
- `traceroute` Trace Route to Host
- `trap` Execute a command when the shell receives a signal •
- `tr` Translate, squeeze, and/or delete characters
- `true` Do nothing, successfully
- `tsort` Topological sort
- `tty` Print filename of terminal on stdin
- `type` Describe a command •

u

- `ulimit` Limit user resources •
- `umask` Users file creation mask
- `umount` Unmount a device
- `unalias` Remove an alias •
- `uname` Print system information
- `unexpand` Convert spaces to tabs
- `uniq` Uniquify files
- `units` Convert units from one scale to another
- `unrar` Extract files from a rar archive
- `unset` Remove variable or function names
- `unshar` Unpack shell archive scripts
- `until` Execute commands (until error)
- `uptime` Show uptime
- `useradd` Create new user account
- `userdel` Delete a user account
- `usermod` Modify user account
- `users` List users currently logged in
- `uuencode` Encode a binary file
- `uudecode` Decode a file created by uuencode

v

- `v` Verbosely list directory contents (`ls -l -b')
- `vdir` Verbosely list directory contents (`ls -l -b')

- `vi` Text Editor
- `vmstat` Report virtual memory statistics

w

- `w` Show who is logged on and what they are doing
- `wait` Wait for a process to complete •
- `watch` Execute/display a program periodically
- `wc` Print byte, word, and line counts
- `whereis` Search the user's $path, man pages and source files for a program
- `which` Search the user's$path for a program file
- `while` Execute commands
- `who` Print all usernames currently logged in
- `whoami` Print the current user id and name (`id -un')
- `wget` Retrieve web pages or files via HTTP, HTTPS or FTP write Send a message to another user

x

- `xargs` Execute utility, passing constructed argument list(s)
- `xdg-open` Open a file or URL in the user's preferred application.
- `xz` Compress or decompress .xz and .lzma files
- `yes` Print a string until interrupted
- `zip` Package and compress (archive) files.
- `.` Run a command script in the current shell
- `!!` Run the last command again
- `###` Comment / Remark

Bash Syntax

Arguments

Shell parameters can be a name, a number, or one of the special characters listed below. For the shell's purposes, a variable is a parameter denoted by a name.

A parameter is set if it has been assigned a value. The null string is a valid value. Once a variable is set, it can be unset only by using the `unset` builtin command.

A variable can be assigned to by a statement of the form `name=[value]`

If value is not given, the variable is assigned the null string (*declaration*). All values undergo *tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal* (detailed below). If the variable has its *integer attribute* set, then value is subject to arithmetic expansion even if the `$((...))` expansion is not used. Word splitting is not performed, with the exception of `"$@"` as explained below. Filename expansion is not performed.

Positional Parameters
A positional parameter is a parameter denoted by one or more digits, other than the single digit 0. Positional parameters are assigned from the shell's arguments when it is invoked, and can be reassigned using the `set` builtin command. Positional parameter **N** can be referenced as `${N}`, or as `$N` (when N consists of a single When a positional parameter consisting of more than a single digit is expanded, it must be enclosed in braces.

Positional parameters can not be assigned to with assignment statements. The `set` and `shift` builtins are used to set and unset them. The positional parameters are temporarily replaced when a shell function is executed.

Special Parameters
The shell treats several parameters specially. These parameters can only be referenced; assignment to them is not allowed.

- `*` Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, it expands to a single word with the value of each parameter separated by the first character of the **IFS** special variable. That is, `"$*"` is equivalent to `"$1c$2c..."`, where `c` is the first character of the value of the **IFS** variable. If **IFS** is unset, the parameters are separated by spaces. If **IFS** is null, the parameters are joined without intervening separators.
- `@` Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, each parameter expands to a separate word. That is, `"$@"` is equivalent to `"$1" "$2" ...`. When there are no positional parameters, `"$@"` and `$@` expand to nothing (i.e., they are removed).
- `#` Expands to the **number of positional parameters** in decimal.

- `?` Expands to the exit status of the most recently executed foreground pipeline.
- `-` **(A hyphen.)** Expands to the current option flags as specified upon invocation, by the set builtin command, or those set by the shell itself (such as the `-i' option).
- `$` Expands to the **process ID of the shell.** In a () subshell, it expands to the process ID of the invoking shell, not the subshell.
- `!` Expands to the **process ID of the job most recently placed into the background,** whether executed as an asynchronous command or using bg
- `0` Expands to the **name of the shell or shell script.** This is set at shell initialization. If Bash is invoked with a file of commands, $0 issettothenameofthatfile.IfBashisstartedwiththe`-c' option, then$0 is set to the first argument after the string to be executed, if one is present. Otherwise, it is set to the filename used to invoke Bash, as given by argument zero.
- `_` **(An underscore.)** At shell startup, set to the **absolute filename of the shell or shell script being executed** as passed in the argument list. Subsequently, expands to the last argument to the previous command, after expansion. Also set to the full pathname of each command executed and placed in the environment exported to that command. When checking mail, this parameter holds the name of the mail file.

Arrays

Bash provides one-dimensional array variables. Any variable can be used as an array; the declare builtin will explicitly declare an array. There is no maximum limit on the size of an array, nor any requirement that members be indexed or assigned contiguously. Arrays are zero-based.

An array is created automatically if any variable is assigned to using the syntax

```
name[subscript]=value
```

The subscript is treated as an arithmetic expression that must evaluate to a number greater than or equal to zero. To explicitly declare an array, use

```
declare -a name
```

The syntax

`declare -a name[subscript]` is also accepted; the subscript is ignored. Attributes can be specified for an array variable using the declare and readonly builtins. Each attribute applies to all members of an array.

Arrays are assigned to using compound assignments of the form: `name=(value1 ... valuen)`, where each value is of the form `[[subscript]=]string`. If the optional subscript is supplied, that index is assigned to; otherwise the index of the element assigned is the last index assigned to by the statement plus one. Indexing starts at zero.

This syntax is also accepted by the declare builtin. Individual array elements can be assigned to using the `name[subscript]=value` syntax introduced above.

Any element of an array can be referenced using `\${name[subscript]}`.

The braces are required to avoid conflicts with the shell's filename expansion operators.

If the subscript is `@` or `_`, the word expands to all members of the array name. These subscripts differ only when the word appears within double quotes. If the word is double-quoted, ${name[*]} expands to a single word with the value of each array member separated by the first character of the IFS variable, and $name[@] expandseachelementofnametoaseparateword.Whentherearenoarraymembers,`$ *{name[@]} `expands to nothing. This is analogous to the expansion of the special parameters @ and` . ParseError: KaTeX parse error: Expected '}', got '#' at position 2: {#name[subscript]…{name[subscript]}. If subscript is `@' or _', the expansion is the number of elements in the array.

Referencing an array variable without a subscript is equivalent to referencing element zero.

The `unset` builtin is used to destroy arrays. unset `name[subscript]` destroys the array element at index subscript.

`unset name`, where `name` is an array, removes the entire array. A subscript of `*` or `@` also removes the entire array.

The `declare`, `local`, and `readonly` builtins each accept a `-a` option to specify an array.

The contents of the directory stack are also visible as the value of the `DIRSTACK` shell variable.

Bash Snippets

Generate a menu

```
select fname in *;
do
        echo you picked $fname \($REPLY\)
        break;
done
```

I think there needs to be a semi-colon after line 3- otherwise it keeps looping?

Rename a file

There is no rename command...? rename is a program in Apt that is vastly more complicated.

```
mv file1.ext file2.ext
```

Alias syntax

```
alias mycommand !:1 !:2
```

Then `!:1` references the arguments from the input. Add to profile any aliases.

yes

Source of Yes Info

Typing `yes` causes 'y' to be printed over and over.

This is intentional. Any word after `yes` is repeated infinite time. `yes test` prints "test" over and over.

You can use `yes` command to prevent from overwriting the existing file or forcefully overwrite the existing file. In the following commands, the first command is used to prevent the overwrite and the second command is used to overwrite the file without any permission.

```
cat hello.txt
cat sample.txt
cp -i sample.txt hello.txt
yes n | cp -i sample.txt hello.txt #prevent overwiting by saying no to each
yes | cp -i sample.txt hello.txt #force overwrite by saying yes to each.
```

You can use `yes` command to run any script multiple times in the command line. In this example, `yes` command is used to run while loop repeatedly ten times. Here, `yes` command will continuously send the numeric value from 1 to 10 to the loop and the loop will print the values in regular interval of one second.

```
$ yes "$(seq 1 10)" | while read n; do  echo $n; sleep 1; done
```

```
#Example 5
#!/bin/bash
#Read the value passed from yes command
read string

#check the string value is empty or not
if [ "$string" == "" ]; then

echo "Empty value is passed by yes command"
else
newstr="The value passed by yes command is $string"
echo $newstr
fi

#Which is read and run by:
$ yes "" | bash yes_script.sh
>>>Empty value is passed by yes command
#or
$yes testing | bash yes_script.sh
>>>The value passed by yes is testing
```

Or write a string over and over into a file, for "testing" or something:

```
yes 'Add this line for testing' | head -50 > testfile
```

Pi Notes

```
sudo raspi-config
```

passwd

```
ssh Hostname@IP
```

pi-HOLE now

now pihole
Hostname = pihole
user = pi
pass = summer1!
ssh pi@pihole
network configuration?

/etc/network/dhcpcd.conf

Renpptomg

```
systemctl reboot -i
```
or
```
sudo reboot
```

List network interface

```
ip link show
```

Show interface summary in a table
```
netstat -i
```

```
ifconfig
```

"Unable to resolve hostname "hostname"

```
sudo nano /etc/hosts
#add:
127.0.0.1 hostname
```

Change enx***{MAC address} to eth0

Get regular net names back

(eth0) instead of the MAC address.

One of these works. Not sure which.

add `net.ifnames=0` to `/boot/cmdline.txt`

to
```
/etc/udev/rules.d/70-persistent-net.rules
```
add
```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="b8:ac:6f:65:31:e5", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth
```

or

to `/etc/udev/rules.d/80-net-name-slot.rules` add `ln -s /dev/null`

or

```
#an almost functioning bash script
sudo -i
tempvar='SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="b8:ac:6f:65:31:e5", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*"
echo $tempvar >> sudo nano /etc/udev/rules.d/70-persistent-net.rules

echo "/etc/udev/rules.d/80-net-name-slot.rules" >> ln -s /dev/null

echo "net.ifnames=0" >> /boot/cmdline.txt

cat /lib/udev/rules.d/73-usb-net-by-mac.rules
#note /lib dir instead of /etc
nano /lib/udev/rules.d/73-usb-net-by-mac.rules

#make
#the
#change


cp /lib/udev/rules.d/73-usb-net-by-mac.rules /etc/udev/rules.d/

exit
```

/lib/udev/rules.d/73-usb-net-by-mac.rules
Change:

```
ACTION=="add", SUBSYSTEM=="net", SUBSYSTEMS=="usb", NAME=="", \
ATTR{address}=="?[014589cd]:*", \
TEST!="/etc/udev/rules.d/80-net-setup-link.rules", \
IMPORT{builtin}="net_id", NAME="$env{ID_NET_NAME_MAC}"

#to

ACTION=="add", SUBSYSTEM=="net", SUBSYSTEMS=="usb", NAME=="", \
ATTR{address}=="?[014589cd]:*", \
TEST!="/etc/udev/rules.d/80-net-setup-link.rules", \
IMPORT{builtin}="net_id", NAME="eth0" #note change
```

Then run:

```
cp /lib/udev/rules.d/73-usb-net-by-mac.rules /etc/udev/rules.d/
```

and finally

add `net.ifnames=0` to `/boot/cmdline.txt` -- I added it above

/etc/dhcpcd.conf
interface eth0
static ip_address=192.168.1.3/22
static routers=192.168.1.254
static domain_name_servers=127.0.0.1
1982.168.2.1
192.168.1.254
1.1.1.1
1.0.0.1
8.8.8.8
8.8.4.4
68.69.156.9
68.94.157.9
99.99.99.53
99.99.99.153

FILES
/etc/dhcpcd.conf
Configuration file for dhcpcd. If you always use the same options, put them here.
/usr/lib/dhcpcd/dhcpcd-run-hooks
Bourne shell script that is run to configure or de-configure an interface.

/usr/lib/dhcpcd/dev
/dev management modules.
/usr/lib/dhcpcd/dhcpcd-hooks
A directory containing bourne shell scripts that are run by the above script. Each script can be disabled by using the -C, - -nohook option described above.
/var/lib/dhcpcd/duid
Text file that holds the DUID used to identify the host.
/var/lib/dhcpcd/secret
Text file that holds a secret key known only to the host.
/var/lib/dhcpcd/interface-ssid.lease
The actual DHCP message sent by the server. We use this when reading the last lease and use the files mtime as when it was issued.
/var/lib/dhcpcd/interface-ssid.lease6
The actual DHCPv6 message sent by the server. We use this when reading the last lease and use the files mtime as when it was issued.
/var/lib/dhcpcd/rdm_monotonic
Stores the monotonic counter used in the replay field in Authentication Options.
/run/dhcpcd.pid
Stores the PID of dhcpcd running on all interfaces.
/run/dhcpcd-interface.pid
Stores the PID of dhcpcd running on the interface.
/run/dhcpcd.sock
Control socket to the master daemon.
/run/dhcpcd.unpriv.sock
Unprivileged socket to the master daemon, only allows state retrieval.
/run/dhcpcd-interface.sock
Control socket to per interface daemon.

sudo (Superuser DO)

Get a root shell `sudo -i`
sudo -s for a non-login shell.
run a non interactive shell using root access: `sudo bash -c 'echo something >> /etc/somewhere/file'`

Print a file to the terminal

in otherwords see whats in a file

```
cat [file]
```

Append a line to a file

`>` Writes to a file (overwrites)
`>>` Appends

So youd think

```
sudo nano /home/test.txt
hello world
#Save and exit
"hello again" >> /home/test.txt
#would work, but it doesnt.  Permission denied

sudo echo "hello again" >> /home/test.txt
#correct syntax at least, still permission denied.   This is because sudo applies to echo not the write to file

#so either get a root shell
sudo -i
#or use `tee`

echo "hello again" | sudo tee /home/test.txt
echo "hello again" | sudo tee -a /home/test.txt
```

Setting up Linux

Adding Windows Network Share--

CURL https://getmic.ro | bash

```
sudo apt-get installl nautilus-share
```

`sudo apt autoremove` cleaned up 400 mb

$need GCC and fortran

sudo apt-get update
sudo apt install gcc
suro apt install gfortran

sudo add-apt-repository ppaLmarutter/c2d4u3.5

$to sources add R

nano /etc/apt/sources.list
dep https://cloud.r-project.org/bin/linux/ubuntu bionic-cran35/

sudo add-apt-repository ppa:marutter/c2d4u
sudo add-apt-repository ppa:marutter/c2d4u3.5

sudo apt-get install r-base-dev

```
deb http://ppa.launchpad.net/marutter/c2d4u/ubuntu bionic main
deb-src http://ppa.launchpad.net/marutter/c2d4u/ubuntu bionic main
deb http://ppa.launchpad.net/marutter/c2d4u3.5/ubuntu bionic main
deb-src http://ppa.launchpad.net/marutter/c2d4u3.5/ubuntu bionic main
```

~~~
sudo nano /etc/default/grub
go to
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
change to
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash video=hyperv_fb:2560x1600"
then write out ([[Ctrl]]+[[O]])
sudo update-grub
sudo apt-get install linux-image-extra-virtual

sudo apt-get update
sudo add-apt-repository ppa:git-core/ppa
sudo apt update;
sudo apt install git
git clone https://github.com/jterry75/xrdp-init.git ~/xrdp-init





```bash
sudo apt-get update
sudo add-apt-repository ppa:git-core/ppa
sudo apt update; sudo apt install git
git clone https://github.com/jterry75/xrdp-init.git ~/xrdp-init
~~~

1.2 Make the scripts executable and run them...

```
cd ~/xrdp-init/ubuntu/18.04/
sudo chmod +x install.sh
sudo ./install.sh
reboot
```

1.3 Run script again to finish setting up VM

```
cd ~/xrdp-init/ubuntu/18.04/
sudo ./install.sh
Shutdown Ubuntu VM
```

Register Ubuntu Session ~ run this command on host PC using powershell with admin rights:

```
Set-VM -VMName YourUbuntuVMNameHere -EnhancedSessionTransportType HvSocket
Restart VM
```

https://github.com/craigcabrey/luminance

git clone https://github.com/craigcabrey/luminance.git ~/luminance
sudo apt-get install libgtk-3-dev
sudo apt install python3-pip

Themes
To perform a vanilla installation of Gnome desktop execute the following linux command:

sudo apt install gnome-session gdm3
In order to install the full Ubuntu Gnome desktop use the tasksel command. In case the tasksel command is not available on your system you can install it by:

$ sudo apt install tasksel
Once the tasksel command is installed, begin the Gnome desktop installation by executing:

$ sudo tasksel install ubuntu-desktop
To start Gnome session on a system without a current graphical user interface (GUI), login to your console and execute:

$ sudo service gdm3 start

VNC Server

VNC Howto
sudo apt install vnc4server xfce4 xfce4-goodies

#other things
sudo apt install gnome-tweaks
sudo apt install snapd
sudo apt install htop
sudo apt install
sudo apt install
sudo apt install

Paper Theme
sudo add-apt-repository -u ppa:snwh/ppa

sudo apt-get install paper-icon-theme

sudo dpkg -i paper*.deb

sudo apt-get install -f
tool

#deepin
sudo add-apt-repository ppa:leaeasy/dde
sudo apt-get update
sudo apt-get install dde
sudo apt install dde dde-file-manager
sudo apt install deepin-gtk-theme

Kubuntu
https://linuxconfig.org/how-to-install-kde-plasma-desktop-on-ubuntu-18-04-bionic-beaver-linux

sudo tasksel install kubuntu-desktop
sudo apt install sddm

f the sddm is already installed than try to reconfigure it in order to make sddm the default display manager for your system:

$ sudo dpkg-reconfigure sddm

Resize LVM

```
sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) universe"
sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu xenial universe"

sudo apt-get update
sudo apt-get install system-config-lvm
```