Linux/UNIX Shell Cheat Sheet

Shells

`Bourne Shell (sh)`	Standard UNIX shell. Not used.
`C Shell (csh)`	Influential shell in interactive use. Not used.
`Bourne-Again Shell (bash)`	A massively improved version of `sh`.
`tcsh`	Improved version of csh.
`Korn Shell (ksh)`	Compatible with `sh`, adding `csh` features.
`Z Shell (zsh)`	Feature-packed shell, most similiar to `ksh`.
`ash` and `dash`	Minimalistic shells compatible with `sh`.
`fish`	A "friendly" shell, focused on interactive use.

`bash` is by far the most prominant. Virtually every UNIX system has a Bourne-compatible shell (`bash`, `ksh`, `ash`, or `dash`).

Input/Output (I/O) Redirection

Every "file" opened by a program is assigned a number, called a file descriptor (FD). This includes the keyboard and console. The three file descriptors used in every UNIX program are:

FD	Name	Normally Connected With
0	Standard Input	Keyboard
1	Standard Output	Screen
2	Standard Error	Screen

Bash/Bourne Shell I/O

command `>`*file*	Redirect standard output to *file* instead of screen.
command `>>`*file*	Append standard output to *file*.
command `<`*file*	Use *file* for standard input instead of the keyboard.
command num`>`*file*	Redirect file descriptor to a *file*.
command num`>>`*file*	Redirect file descriptor to the end of *file*.
command num1`>&`*num2*	Redirect FD *num1* to FD *num2*.
command `2>&1 >`*file*	Redirect standard error and output to *file*.

C Shell I/O

command `>`*file*	Redirect standard output to *file* instead of screen.
command `>>`*file*	Append standard output to *file*.
command `<`*file*	Use *file* for standard input instead of the keyboard.
command `>&` *file*	Redirect standard error and output to *file*.

Redirection of Program I/O

command1 `|` *command2*	Redirect standard output of *command1* to standard input of *command2*.

This is a powerful technique that deserves some examples:

`ps aux	less`	View a potentially long output through `less`.
`ls -lt|head`	Use the `head` program to display the first ten lines.
`export	grep -i ssh`	Use `grep` to search through a command's output.

Environmental Variables

All programs in Linux/UNIX are loaded with a set of named data called the environment. Some programs use this to modify their behavior. By convention, the names are capitalized.

Manipulating Variables in Bash

`export`	List environmental variables.
`export` *VARNAME*`=`*value*	Set *VARNAME* to *value*.

Manipulating Variables in C Shell

`setenv`	List environmental variables.
`setenv` *VARNAME value*	Set *VARNAME* to *value*.

Using Variables in Bash / C Shell

...`$`*VARNAME* ...	Run the command with the value of *VARNAME*.
`echo $`*VARNAME*	Example of above, prints the value of *VARNAME*.

Useful Variables

`PATH`	Colon-seperated list of locations for commands.
`PAGER`	Program used to display long files (e.g., by `man`).
`EDITOR`	Program used to edit files, (e.g., by `ipython`).
`HOME`	The path to the home directory.
`DISPLAY`	Where to access an X Server (used for graphical programs.)

Aliases

`alias` *newalias*=``*commands ...*''	Set an alias in Bash.
`alias` *newalias commands ...*	Set an alias in C Shell.

Aliases can be created with the names of existing commands, or new ones. For example (in bash):

`alias rm='rm -iv'`	Always confirm before removing files.
`alias ls='ls -G'`	Make `ls` colorful (on BSD systems).
`alias cup-holder='eject /dev/cdrom'`	Provide yourself a little humor.
`alias rsys='rsync -avuz --de...'`	Shorten long commands.

Startup Files

These files contain places to put aliases and environmental variables, as well as any other code you'd like run at startup.

Two distinctions need to be made: login shells are run once you login, as opposed to ones run afterwards (e.g., running `bash` after logging in). Interactive shells are those you run commands with, as opposed to ones that are run from scripts.

While Linux terminals run in graphical systems are considered non-login, interactive shells; Mac OS X's Terminal.app runs as an interactive login shell.

Bash Startup

If a login shell, `~/.bash_profile`, `~/.bash_login`, and `~/.login` are check. The first one to exist is loaded.

If an interactive non-login shell, `~/.bashrc` is loaded if it exists.

See http://wiki.bash-hackers.org/scripting/bashbehaviour#quick_startup_file_reference for more information.

Simplified Bash Startup

Many Linux distributions take the approach recommended in the Bash documention of just using `~/.bashrc` for all interactive use. To do this, put the following in `~/.bash_profile`:

```
if [ -f ~/.bashrc ];
        then . ~/.bashrc;
fi
```

C Shell Starup

`~/.tschrc` and `~/.cshrc` are checked, and the first one that exists is loaded.

`~/.login` is also loaded for login shells.

Wildcards

... `*` ...	Replace `*` with all files in current directory.
... `*.c` ...	Complete part of filename: replace with files ending in `.c`.
.....`?`...	Replace `?` a single leter from files in current directory.

Recalling History

`Up Arrow`	Go through previous commands
...`!!`...	Replace `!!` with entire last command

Modified from https://wch.github.io/latexsheet/