Unix Shell Reference Card

Syntax

A command is a series of words separated by spaces. If a filename contains a space, enclose it in single or double quotes or escape each space with a backslash. You escape shell special characters with a backslash too.

Most Unix commands support **qualifiers** which usually start with a minus sign; some useful examples are shown in this document. Read the relevant manual page to explore all supported qualifiers and what they do. Most commands are shown taking a single *`file`* argument but can usually be invoked with many files.

All the commands listed here are normally available under both Linux and macOS, though a few need to be installed via **Homebrew** on a Mac.

Control characters and shortcuts

complete the current command or filename	⟨`tab`⟩
halt the current command	`C-c`
stop the current command (see **Processes**)	`C-z`
delete next character; end-of-file; exit (**dangerous**)	`C-d`
erase previous word typed	`C-w`
erase entire line	`C-u`

Streams

ordinary output from a program (normally goes to your terminal window)	**stdout**
error message output from a program	**stderr**
input stream to a program	**stdin**

Redirection and pipes

save **stdout** of *cmd* in *file*	`cmd >file`
append **stdout** of *cmd* to *file*	`cmd >>file`
make **stdout** of *cmd* disappear	`cmd >/dev/null`
make *cmd* read from *file*	`cmd <file`
make **stdout** of *cmd1* the **stdin** of *cmd2*	`cmd1 | cmd2`
output to *file* as well as to **stdout**	`cmd | tee file`

Adrian F. Clark ⟨`alien@essex.ac.uk`⟩ [last edit: 2020/01/08]

Filenames

filename structure	`/dir1/dir2/dir3/dir4/file.ext`
the current directory	`.`
the parent directory	`..`

Filenames starting with / are **absolute** and work in all directories; others are **relative** to the current directory. Filenames can contain **any** characters except / though you should avoid `|*?#"'!` Apart from compilers, programs generally don't care about file extensions.

Wildcards in shell commands

your login directory	~
a single character	`?`
any file	`*`
files whose names end with `.txt`	`*.txt`

File and directory commands

directory listing	`ls`
formatted directory listing with hidden files	`ls -al`
show current directory	`pwd`
show the directory hierarchy	`tree`
create directory *dir*	`mkdir dir`
change directory to *dir*	`cd dir`
copy *file1* to *file2*	`cp file1 file2`
copy directory tree *dir1* to *dir2*	`cp -r dir1 dir2`
rename *file1* to *file2*	`mv -i file1 file2`
delete *file*	`rm file`
securely delete *file*	`shred file`
delete empty directory *dir*	`rmdir dir`
force the deletion of directory *dir* (**dangerous**)	`rm -rf dir`
con**cat**enate *file* to **stdout**	`cat file`
view *file* a screenful at a time	`less file`
show the first few lines of *file*	`head file`
show the last few lines of *file*	`tail file`
create or update *file*	`touch file`
make *f2* a **symbolic link** to *f1*	`ln -s f1 f2`

File permissions

Files have **r** (read), **w** (write) and **x** (execute) permissions, and they are separate for **user** (owner), **group** and **others**.

make *file* executable for all	`chmod +x file`
make *file* readable only by you	`chmod go-r file`

For further information, explore `man chmod`.

Processes

run *cmd* in the background	`cmd &`
detach current command from terminal session	`C-z`
list stopped commands	`jobs`
bring most recently stopped command to foreground	`fg`
bring stopped command %*n* to foreground	`fg %n`
kill stopped command %*n*	`kill %n`
display your running processes	`ps`
kill process *pid*	`kill pid`
kill all processes containing *name* (**dangerous**)	`killall name`
display running processes	`top`
display running processes on CPU cores	`htop`

Useful information

show the manual page for *cmd*	`man cmd`
show the date and time	`date`
show this month's calendar	`cal`
show who you are logged in as	`whoami`
show who is doing what	`w`
show possible locations of *cmd*	`whereis cmd`
show the filename *cmd* will run	`which cmd`
show disk space usage	`du`
show disk usage of current directory tree	`du -sh .`
report the type of data stored in *f*	`file f`
output text strings in (binary) *file*	`strings file`
show the values of environment variables	`env`

System information

show how long since booting	`uptime`
show kernel information	`uname -a`
show disk usage	`df -h`
show network configuration	`ifconfig`
translations of MAC onto Ethernet addresses	`arp -a`
show routing table	`netstat -ern`
show network connections	`netstat -ta`
show network statistics	`netstat -i`
list open files	`lsof`
list processes using TCP port 1337	`lsof -i tcp:1337`

Command history

Use the arrow keys to recall commands and edit them, then hit ⟨return⟩ to run the edited command.

output previous commands	`history`
search for a recent command	`C-r`
re-run command *n*	`!`*n*
recall command *n* for editing	`!`*n*`:p`
repeat recent command starting with *text*	`!`*text*
repeat the previous command	`!!`

bash

initialization file	`~/.bashrc`
initialization file for login shells	`~/.profile`
set environment variable	`export VAR=value`
delete environment variable	`unset VAR`
create command alias *dir*	`alias dir='ls'`
delete command alias *dir*	`unalias dir`
make **stderr** go to the same place as **stdout**	*cmd* `2>&1`

iterate over a set of files:

```
  for f in *.pdf
  do
    echo "$f"
  done
```

csh and tcsh

initialization file	`~/.cshrc`
set environment variable	`setenv VAR value`
delete environment variable	`unsetenv VAR`
create command alias *dir*	`alias dir 'ls'`
delete command alias *dir*	`unalias dir`
make **stderr** go to the same place as **stdout**	*cmd* `>&`

iterate over a set of files:

```
  foreach f (*.pdf)
    echo "$f"
  end
```

Networks

output the name of the machine you're using	`hostname`
see whether *host* is alive	`ping` *host*
output the route to *host*	`traceroute` *host*
output DNS information for *host*	`dig` *host*
output owner information for *domain*	`whois` *domain*
discover information about *user*	`finger` *user@host*
download *url* to your machine	`wget` *url*
continue a stopped download of *url*	`wget -c` *url*

login on *host*	`ssh` *host*
login on *host* as *user*	`ssh` *user@host*
run *cmd* on *host* as *user*	`ssh` *user@host cmd*
copy your key to *user* on *host*	`ssh-copy-id` *user@host*
copy *file* to *host*	`scp` *file user@host:dir*
copy remote *file* to *dir*	`scp` *user@host:file dir*
backup to *dir* on *there*	`rsync -arv ~` *there:dir/*

Programming

edit *file* using the One True Editor ;-)	`emacs` *file*
edit *file* using the standard Unix editor	`vi` *file*
edit *file* using a simple editor	`nano` *file*
typical C compilation	`gcc -o prog prog.c mod.c -lm`
typical C++ compilation	`g++ -o prog prog.cc -lm`
run *prog* from your current directory	`./prog`
build library *libmy.a*	`ar -rv` *libmy.a* `*.o`
make *libmy.a* ready for use	`ranlib` *libmy.a*
(to use `-lmy` on `gcc` and `g++` commands)	
execute compilation instructions in `Makefile`	`make`
typical Java compilation	`javac prog.java`
run Java `class` file	`java prog`
run the Python program in *file*	`python` *file*

Bundling up files into archives

build *zipfile* containing *files*	`zip` *zipfile files*
unpack *zipfile*	`unzip` *zipfile*
build a compressed tar-file	`tar zcvf` *tarfile files*
unpack *tarfile*	`tar zxvf` *tarfile*

Printing

check the status of the print queue	`lpq`
print *file* on *queue*	`lpr -P`*queue file*

Searching

list all filenames containing *text*	`locate` *text*
search for *text* in *file*	`grep` *text file*
recursively search for *text* in *dir*	`grep -r` *text dir*
look for *text* in output of *cmd*	*cmd* `| grep` *text*
search *file* for *regex*	`egrep` *regex file*

Regular expressions

start of line	`^`
end of line	`$`
start of word	`\<`
end of word	`\>`
start or end of word	`\b`
any character	`.`
dot character	`\.`
zero or one time	`?`
one or more times	`+`
zero or more times	`*`
character class	`[ ]`
whitespace character	`\s`
negated character class	`[^ ]`
match *a* or *b*	*a*`|`*b*

Text manipulation

These commands are often used as *filters* in a pipeline, reading from **stdin** and writing to **stdout**. Read the commands' manual pages to understand how to use them in anger.

sort lines into alphabetical order	`sort`
remove duplicate lines	`uniq`
count characters, words and lines	`wc`
number lines in output	`nl`
number lines starting with *alias*	`nl -b p^alias`
transliterate characters in *from* to *to*	`tr` *from to*
remove all vowels from input	`tr -d aeiou`
output differences between *f1* and *f2*	`diff` *f1 f2*
compare directory trees *d1* and *s2*	`diff -qr` *d1 d2*
report common lines in *f1* and *f2*	`comm` *f1 f2*
(see its manual page for interpreting its output)	
output lines in reverse order	`tac`
remove parts of lines	`cut`
merge files	`paste`
join lines of two files having a common field	`join`
check the spelling in *file*	`spell` *file*
interactively check the spelling in *file*	`ispell` *file*