Unix Command Reference

Frequently used Commands

Words in `monospace` type are commands and should be typed as they are printed

Words in **bold** type should be substituted with the appropriate filename or directory

Unix is case-sensitive — *UPPER* and *lower*case letters have different meanings

General

`man` **`command`**
Display the Unix manual entry describing a given command

`apropos` **`command`**
Locate commands by keyword

`alias` **`name1 name2`**
Create command alias *name*

`alias` **`name`**
Display command alias

`unalias` **`name`**
Remove command alias *name*

`passwd`
Change password

`quota`
Display amount of disk space used

`df`
Show available system disk space

`du`
Show disk space being used up by folders

`bc`
Basic calculator

`bc obase=16 255`
Displays `FF`

`bc ibase=16 obase=10`
Hex to Dec

`date`
Display date & time

`cal` **`month year`**
Show calendar

`whoami`
Display current user

`history`
Display recent commands

`!!`
Repeat last shell command

`!`**`string`**
Repeat last shell command that began with string

`!` **`n`**
Repeat recent shell command *n*

`Ctrl/p` (previous)
Move up in history list

`Ctrl/n` (next)
Move down in history list

`Ctrl/b` (backward)
Cursor left to edit command

`Ctrl/f` (forward)
Cursor right to edit command

`Ctrl/d` (delete)
Delete character in command

`clear`
Clear terminal screen

`lock`
Lock terminal

`reset`
Reset / initialize terminal

`set`
Show environment

`env`
Show current settings

`sentenv name` **`v`**
Set env var to value *v* (csh/tcsh)

`export name="`**`v`**`"`
Set environment variable to value *v* (ksh/bash)

`exit`
Terminate current session

`script`
Make a typescript of everything printed on the terminal

`sudo /usr/libexec/locate.updatedb`
Update the locate database

File System Navigation

`*`
Wild card: match zero or more characters

`?`
Wild card: match zero or one character

`.`
Shorthand for the current directory

`..`
Shorthand for the parent of the current directory

`~`
Home directory

`~`**`username`**
Home directory of user *username*

`cd` **`dir`**
Change to directory *dir*

`cd`
Return to home directory

`pwd`
Display working directory

`file` **`file`**
Determine file type

`du -ks *|sort -nr|more`
Show all directory sizes in order, largest first

`ls`
List the contents of the current directory

`ls` **`dir`**
List the contents of the directory *dir*

`ls -l`
Show permissions, owner, size, and other file info

`ls -a`
Show all files, including (hidden) files that begin with a dot

`ls -R`
Show files recursively, for all subdirectories

`ls -d`
List directories like other *files*, without displaying their contents

`ls -k`
List file sizes in kilobytes

`ls -X`
Sort files by file extension

`ls -1`
Display the listing in 1 column

`ls -t`
Show files in time order, newest to oldest

`ls -l | grep "^d"`
List all directories in the current directory without any of the files

`ls -l | grep ^d | wc -l`
Find the number of subdirectories in the current directory

`ls -ls|sort -nr|more`
List files by size, largest first

Data Manipulation

`mkdir` **`dir`**
Create new directory *dir*

`cp` **`file1 file2`**
Copy *file*(s)

`cp` **`file dir`**
Copy *file*(s) into a *dir*ectory

`cp -r` **`dir1 dir2`**
Copy a *dir*ectory and, recursively, its subdirectories

`mv` **`file dir`**
Move *file* to directory *dir*

`mv` **`dir1 dir2`**
If directory *dir2* exists, move *dir1* into *dir2*; otherwise, rename *dir1* as *dir2*

`mv` **`file1 file2`**
Rename *file1* as *file2*

```
#!/bin/sh
for i in *
do
echo $i
mv $i `basename $i`.ext
done
```
Rename a number of files

`rm` **`file`**
Remove *file*

`rm -f` **`file`**
Force, remove files without prompting

`rm -r` **`file`**
Remove files, directories, and recursively, any subdirectories

`rmdir` **`dir`**
Remove empty directory *dir*

`vi` **`file`**
Vi fullscreen editor

`emacs` **`file`**
Emacs fullscreen editor

`pico` **`file`**
Pico text editor

`wc` **`file`**
Count lines, words, & chars

`cat` **`file`**
List contents of *file*

`more` **`file`**
Display contents of a *file* one screen at a time

`less` **`file`**
Opposite of more

`head -n` **`file`**
Display first *n* lines of *file*

`tail -n` **`file`**
Display last *n* lines of *file*

`cmp` **`file1 file2`**
Compare two *file*s

`diff` **`file1 file2`**
Show *file* differences

`cp` **`file1 file2`**
Copy file *file1* into *file2*

`sort` **`file`**
Display the *line*s of text *file* alphabetically

`sort -r` **`file`**
Sort in reverse order

`sort -n` **file**
Sort numerically (2 before 10)

`sort +n` **file**
Sort on n+1st field

`cat` **file1 file2 > file3**
Concatenate *file1* & *file2* into *file3*

`split [-n]` **file**
Split *file* into *n*-line pieces

`grep` **sample file**
Output lines that match *sample* string or pattern

`grep -i`
Case-insensitive search

`grep -n`
Show the line # along with the matched line

`grep -v`
Invert match: find all lines that do not match

`grep -w`
Match entire words, rather than substrings

`touch` **file**
Update the timestamp on a file, if the file doesn't exist, touch creates an empty file

I/O Redirection

The shell expects input from; and sends output to, a terminal. To write command output to files or read input from files, redirection is used. UNIX defines three I/O units with corresponding file descriptors:

```
0: stdin  (standard input)
1: stdout  (standard output)
2: stderr  (standard error)
```

prog `>` **file**
Redirect (write) stdout of *prog* to *file*

prog `>>` **file**
Append stdout of *prog* to *file*

prog `<` **file**
Read stdin for *prog* from *file*

prog `<` **file1** `>` **file2**
Read stdin for *prog* from *file1*, redirect stdout to *file2*

prog `2>`**file**
Write stderr of *prog* to *file*

prog `2>&1`
With file descriptor: write stderr of **prog** to stdout

cmd1 `|` **cmd2**
Pipeline: use *cmd1*'s output as input for *cmd2*

cmd1 `&&` **cmd2**
cmd2 is executed only if the execution of *cmd1* ends up successfully

cmd1 `||` **cmd2**
cmd2 is executed only if the execution of *cmd1* does not end up successfully

cmd1 `;` **cmd2**
Execute *cmd2* after execution of *cmd1* stopped

`nohup` **command** `<` **file.in** `>>` **file.out**`&`
'No hangup': execution of *command* will continue even if the user logs off the system (exit). Run *command* in the bakground (&), taking input from *file.in* and appending output to *file.out*.

Permissions

`-rwxr-xr-x`
Directories have a *d* in the first column; regular files have a *-*.
The remaining 9 characters indicate the owner, group, and world permissions of the file.
An *r* indicates that the file is readable; *w* is writable, and *x* is executable.
A dash in the column instead of a letter means that particular permission is turned off.
t `is the` 'sticky bit' for directories; prevents files from being deleted by anyone other than the owner.
s is the 'setuid-bit' for files; execute a program using the owner's permissions (rather than those of the one who calls it).

Setting Permissions with Letters

`chmod` *u+rwx,go+rx* **file**
u is the user's (owner) permissions; *g* is the group permissions, and *o* is world (other) permissions.
The *+* sign turns the stated permissions on; a *-* sign turns them off.
Directories should always have, at least for the owner, the *x* permission set.
A directory doesn't have to be readable for the web server to read and execute files within that directory. Only the files themselves must be readable.

Numeric Permissions

`chmod` **711 file**
Change permissions on a *file*.
The first number translates to permissions by the owner (logon account). The second is permissions for the group (a possibly empty group of logon accounts). The third is permissions for everyone.

```
0: ---  (no permissions)
1: --x  (executable only)
2: -w-  (writable only)
3: -wx  (writable and executable)
4: r---  (readable only)
5: r-x  (readable and executable)
6: rw-  (readable and writable)
7: rwx  (readable, writable, and executable)
```

File Compression

`compress` **file**
Reduce the size of a *file*

`uncompress` **file**
Restore a compressed *file*

`tar cf - /home/file | compress >` **file.tar.Z**
tar and compress a *file*

`tar cf - /home/file | gzip >` **file.tar.Z**
tar and gzip a *file*

`ls -al | awk '$0!~/^d/ {print $9}' | xargs tar cvf` **archive_name**`.tar`
Archive only regular files in a directory, omitting subdirectories and hidden files

Make an index file of the contents of the tar file

`tar cvf - /home/file 2>`**file**`.idx | compress >` **file.tar.Z**
For sh, ksh

`(tar cvf - /home/file | compress > /file.tar.Z) >&`**file**`.idx`
For csh

A simple backup script

sh:
```
% pico ~/bin/backup.sh

#!/bin/sh
echo "Backup of Folder:"
tar cvf - /home/file 2>file.idx |
gzip > home/file.tar.Z
```

Save the script in ~/bin

```
% chmod +x ~/bin/backup.sh
```
Make it executable

```
% rehash
```
Force the shell to rebuild its list of known executables

Networking & Communications

`who`
List logged in users

`finger` **user**
Display *user* information

`chfn`
Change finger information

`ping` **host**
Send ICMP ECHO_REQUEST packets to network **host**s

`telnet` **hostname**
Connect to another remote system using the telnet protocol

`ssh` **host**
`rsh` **host**
Log into and execute commands on a remote machine

`lpr -P printer` **file**
Output *file* to line printer

`mail` **user**
Send mail to *user*

`biff` **y/n**
Instant notification of mail

Process Control

`sleep` **n**
Sleep for *n* seconds

`jobs`
Display list of jobs

`Ctrl/c`
Interrupt process / stop execution of a command

`Ctrl/d`
End of typed input (End of File Key)

`Ctrl/q`
Start / resume terminal output

`Ctrl/s`
Stop terminal output

`Ctrl/z`
Suspend execution of a command

`ps`
Show process status statistics

`ps aux`
Show complete process listing

`top`
Show system usage statistics dynamically; stop with *q*

`kill -9` **n**
Remove process *n*

`stop %`**n**
Suspend background job *n*

command`&`
Run *command* in background

`bg %`**n**
Resume background job *n*

`fg %`**n**
Resume foreground job *n*

Maintained by Alexander Becker
Published by kokhaviv press
www.kokhavivpublications.com/help/unix/
print 0.92 / 2004-10-19