UNIX Shell Cheatsheet

Johann Dréo

UNIX Shell Cheatsheet

Graphical Server
Shell
Executables
Libraries
Operating System
Drivers
Hardware

Flavors of UNIX:
- Linux
- FreeBSD
- Darwin (MacOSX)
- Cygwin
- WSL (Windows)
- …

Flavors of Shell:
- Bash
- Zsh
- Fossil
- …

UNIX wisdom: **everything is a file**.
Shell wisdom: **everything is a text stream.**
Script wisdom: **everything is a string**

bash interactivity:

Ctrl-D	exit
Ctrl-L	clear
Ctrl-R	recover command (history)
Ctrl-U	erase line before cursor
Ctrl-W	erase word before cursor
Ctrl-←	move one word left
Ctrl-XX	move at beginning and back
Ctrl-_	undo last edit

Basics

Everyday commands:
```
man
ls
cp
mkdir
cd
mv
rm
```

less:

q	quit
/	search
n	next
↑↓	move

Options convention:
- cmd [options] ARG [OPTARG]
- Short: -h -v [or] -al
- Long: --help --verbose
- Arguments: --opt[=]<value>

Examples: $ ls -a -l ..
$ ls -al ~

List files:

ls -l	long listing format
ls -a	all files (even hidden)
ls -h	human-readable
ls -t	sort by modification date
ls -S	sort by size

Remove files

rm -r	recursive
rm -f	force
rm -i	interactive

Change directory:

cd ..	above
cd /	file system root
cd ~	home directory

Wildcards:

*.txt	ending with .txt
A	containing A
A*.txt	between A and .txt
A?.txt	one character after A

Copy files

cp -r	recursive
cp -a	archive
cp -i	interactive

Misc files:

rename "s/X/Y/" *.ext	rename files
ln -s <target> <link>	symbolic link

Examples:

```
history|cut -d " " -f 5|sort|uniq -c|sort -n -r|head -n 10|sed "s/^\s*//"
```
10 most used commands

```
head -n 1 A.csv > out.csv && tail -n+2 -q *.csv >> out.csv
```
merge two CSV files, not duplicating headers

```
(cmake .. && make) ; zenity --info --text "Ended at $(date)"
```
notify when some (here C/C++) build command ended

Streams

Display files:

cat	concatenate
less	pager
head	beginning of file
tail	end of file

Look at files:

wc	word (and line & char) count
grep	filter matching text
find	search for files
locate	search for file names (fast)

Count:

wc -w	number of words
wc -l	number of lines
wc -m	number of characters

Filter:

```
grep [OPTIONS] PATTERN [FILE…]
```

grep -v	filter *out* matching lines
grep -E	use regexp
grep -n	show line numbers
grep -A	show n lines *after*
grep -B	show n lines *before*

Search:

find . -name "x"	by name from current directory
find .. -type f	files from directory above
find ~ -type d	directories from home

Text streams:

cat A > B	redirect
cat A >> B	append
cat A B > C	concatenate
cat A|grep stuff	pipe

```
cat ~/.bashrc | grep alias | less
```
display bash aliases in the pager

Standard outputs:

cmd > out.txt	redirect standard output
cmd 1> out.txt	idem
cmd 2> err.log	redirect standard error
cmd &> all.txt	redirect all
cmd 2>&1 |…	redirect before piping
…|tee -a out	print *and* redirect

```
cmd 2> /dev/null 1> out.txt
```
suppress errors and redirect output

Manipulate streams:

uniq	remove duplicates
sed	replace text
cut	extract columns
sort	order lines

Sort lines:

sort -r	reverse order
sort -n	numeric order
sort -f	ignore case

Get columns:

cut -d<c>	use "c" as delimiter
cut -f<n>	get the nth field
cut -f<n-m>	get fields nth to mth

Replace text:

sed "s/X/Y/g"	replace matching
sed "s,X,Y,g"	regexp "X" by "Y"
sed -i	replace in-place
sed -e	multiple replace

Execute on content:

xargs	run cmd for each item of piped stream
find -exec	run cmd on each matching file

Execute on each file:

find -exec cmd {} \;	execute cmd on found file
find -print0	print found filenames
find -delete	delete found files

```
find /tmp -depth -name core -type f -delete
```
delete files named "core" in /tmp

```
find ~ -type f -name *.sh -exec ls -l {} \;
```
find shell scripts under home and show their state

Execute on each item:

xargs -L 1	for each line of input stream
xargs -I {}	replace {} with input argument

```
cat /etc/passwd|cut -d: -f1|sort|xargs -L1 echo
cat /etc/passwd|cut -d: -f1|sort|xargs -I{} echo {}
```
display users registered in the system

Execute

Shell interactivity:

cmd	run a command
./scr	run a script (forgets changes)
. scr	run a script (keep changes)
Ctrl-C	stop a running command
Ctrl-Z	pause a command
fg	foreground a paused command

Processes:

ps aux	show running processes
kill	ask to stop
kill -9	force kill
nice	run with +/- priority
renice	change priority
at	run at a date
sleep	wait before next command

Detach processes:

background=depends on the session
detached =does not depends on session

Ctrl-Z, bg	pause, run in background
cmd &	run in background
nohup cmd	run detached
tmux	run detachable shell(s)
Ctrl-b d	detach the shell(s)
tmux attach	get back the detached shell(s)

Chain commands:

A ; B	A then B
A && B	if A successful, then B
A || B	if A on error, then B

Logical tests:

&&	logical AND
||	logical OR
!	logical NOT

Returned information:

$?	returned code (0=OK, >0=error)
$!	Process ID

Shell basic syntax:

```
# Comment
x="A"                 put "A" in variable x
printf "${x}"         print content of x
z="${x:-B}"           if x is unset, put "B" in z
declare -a x=()       declare x as an array
x+=("A" "B")          put two elements in x
printf "${x[0]}"      access the array
```

Shell calls:

#!/bin/bash	run this script with bash
cmd	call cmd
x=$(cmd)	capture cmd standard output
x=$((1+2))	integer arithmetics

Binary conditions:

```
if [[ "$x" == "$y" ]]; then
    # Strings are equals
fi
if [[ "$x" -eq "$y" ]]; then
    # Numeric values are equals
fi

-ne  not equal
-gt  greater than
-ge  greater or equal
-lt  lesser than
-le  lesser or equal
```

Special variables:

$0	program name
$1	1st argument
…	…
$*	all arguments
$#	number of arguments
$IFS	Internal File Separartor

Functions:

```
alias f="cmd -opt"    cmd shortcut
function f() {…}      function definition
f "A" "B"             function call
if f "A" "B";then…    test exit code

function g() {
    local str="${1}"
    if [[ -z "$str" ]]; then
        return 1 # error
    else
        # return by var
        ret="$str"
        # return by stdout
        printf "$str"
        return 0 # OK
    fi
}
printf $(g "First")
if g "Second"; then
    printf "$ret"
fi
```

Unary conditions:

```
if [[ -n "$x" ]]; then
    # string is not empty
    # variable is not unset
fi

-z  string is empty
```

Loops:

```
for x in "$X"; do
    # iterate over split X
done

for ((i=0; i<10; i++)); do
    # numeric counter i
done

i=0
while [[ "$i" -ne 10 ]]; do
    i=$((i+1)) # same
done
```

File conditions:

```
if [[ -w "$fname" ]]; then
    # file is writable
fi

-r  is readable
-f  is a file
-d  is a directory
-x  is executable
```

Read file line by line:

```
while IFS='' read -r line || [[ -n "$line" ]]; do
    printf "$line\n"
done<"$filename"
```

Uselful commands:

date -I	sortable date
mktemp	temp file
mktemp -d	temp dir

Environment

Environment variables:

printenv	show current config
export	configure an env. variable
PATH	where the shell finds commands
HOME	path to user's directory
PWD	current working directory
LOGNAME	current user name
SHELL	current shell
LANG	current language
HISTIGNORE	command not saved in history
EDITOR	default text editor

```
export PATH="$PATH:/my/path"
```
add a directory to the default path

~/.bashrc

. ~/.bashrc	reload config
alias	shortcuts
function	complex cmd
PATH	exported env. var.
PYTHONPATH	
…	

Sysadmin

Core sysadmin:

sudo	execute as root
chmod	change permissions
chown	change owner
time	measure execution time
curl	download from the web
sha256sum	file fingerprint
ifconfig	ethernet interfaces
iwconfig	wifi interfaces
hostname	current host
df -h	disk free
free -m	free memory
du -h -d <n>	disk usage

chmod/chown:

chmod [ugoa][+-][rwx]	**u**ser	**r**ead
chmod -R	**g**roup	**w**rite
	other	e**x**ecute
chown user[:group]	**a**ll	
chown -R	recursive	

Archives:

```
zip -r f.zip *
unzip -d f f.zip
tar cfz f.tgz *
tar xfz f.tgz
    … j f.tar.bz2
    … J f.tar.xz

dtrx
```

Remote

SSH:

ssh [<user>@]<host>[:/<path>] [cmd]	
ssh -X	allows graphical apps
ssh -L	setup a tunnel
scp -r	copy across network
ssh-keygen	generate keys
ssh-copy-id	copy public key on host

~/.ssh/config:

```
Host brocante
    Hostname la.brocante.fr
    User louis
    IdentityFile ~/.ssh/id_louis

Host derriere
    Hostname vieille.local
    ProxyJump brocante
```

Interface

Python scripts as pipes:

```
def write(something, stream=sys.stdout):
  try:
    stream.write(something)
    stream.flush()
  except IOError:
    try:
      stream.close()
    except IOError:
      pass

def pipe(in, out, callable, *args):
  while True:
    try:
      item = in.readline()
    except UnicodeDecodeError:
      continue
    except KeyboardInterrupt:
      break
    if not item:
      break
    write( callable(item, *args), out )
```

Useful config in ~/.bashrc:

```
alias ..="cd .."  # go one dir up
alias ll="ls -al"  # list all
alias lk="ls -lSr" # list by size
function md() {    # create directory,
  mkdir $1         # then move in
  cd $1 }
# use up/down to search history
# (instead of Ctrl-R)
if [[ $- == *i* ]]; then
  bind '"\e[A": history-search-backward'
  bind '"\e[B": history-search-forward'
fi
alias upgrade="sudo apt update &&
  sudo apt dist-upgrade -y &&
  sudo apt --purge autoremove -y &&
  sudo apt autoclean -y"
export PATH="$PATH:$HOME/.local/bin/"
```

Additional tools:

liquidprompt	smart prompt
exa	better "ls"
rsync	better "scp"
colout	add colors to text streams
autojump	smart "cd"
bash-completion	for common commands