Command Line Cookbook

Ha.Minh

Published with GitBook

Table of Contents

A cookbook for using command line tools to do everyday job.

---

read online at:

- https://minhhh.gitbooks.io/command-line-cookbook/content/

download a .pdf, .epub, or .mobi file from:

- https://www.gitbook.com/book/minhhh/command-line-cookbook/details

contribute content, suggestions, and fixes on github:

- https://github.com/minhhh/cli-cookbook

References

- The Linux Cookbook
- Commandline fu

This chapter focus on using the shell to run and/or coordinate different programs together. Even though there are many different shell, we focus on `bash`, which is the standard on most Linux systems, including MacOS.

running-a-list-of-commands

Problem

You want to run a list of commands in order, sometimes in parallel. Sometimes you want to run a command only if another command succeeds or fails.

Solution

To run more than one command in order, simply type each command in the order you want them to run, separating them with a semicolon `;`

```
    echo 1; echo 2; echo 3;
    > 1
    > 2
    > 3
```

To run a command only if the previous ones succeed, we can use `&&`

```
    ls <file> && rm <file> -rf
```

To run a command only if the previous ones fail, we use `||`

```
    ls file &> /dev/null || echo "File not exist"
    > File not exist
```

To run several commands in parallel, you can run them as background process using `&` then `wait`

```
process1 &
process2 &
process3 &
process4 &
wait
process5 &
process6 &
process7 &
process8 &
wait
```

If you want to make sure that all processes succeeds together, you can use npm package `parallelshell`

```
parallelshell "echo 1" "echo 2" "echo 3"
```

References

- http://stackoverflow.com/questions/19543139/bash-script-processing-commands-in-parallel

This chapter focus on the tools for manipulating files and directories.

Watching a directory and execute command on file change

Problem

Watch a file sets or directory and run a command when anything is added, changed or deleted.

Solution

Use python watchdog module, which has a command line tool called `watchmedo`

```
watchmedo shell-command --recursive --command 'echo ${watch_event_type}' -w -W . \
| xargs -n 1 -I {} sh -c 'if [ "{}" = "modified" ]; then clear; make unittest; fi'
```

Alternatively, can use nodejs onchange module

```
onchange 'app/**/*.js' 'test/**/*.js' -- npm test
```

Listing file and folder sizes

Problem

You want to print the sizes of all files and folders in the current folder from largest to smallest

Solution

We simply run `du` command on each file and folder in the current folder then sort them using `sort`

```
ls -A | awk '{system("du -sm \""$0"\"")}'| sort -nr | head
```

To list only folders

```
ls -Al \
| egrep '^d' \
| awk '{printf $9; for (x=10; x <= NF; x  ) {printf " "$x;}; print ""}' \
| awk '{system("du -sh \""$0"\"")}'
```

To list only files

```
ls -Al | egrep -v '^d' \
| awk '{printf $9; for (x=10; x <= NF; x  ){printf " "$x;}; print ""}' \
| awk '{system("du -sh \""$0"\"")}'
```

References

- http://groups.google.com/group/comp.unix.shell/browse_thread/thread/aebcbd0591714584/5e496ed7cfbe6eb1
- http://en.wikipedia.org/wiki/Xargs
- http://www.cyberciti.biz/faq/linux-list-just-directories-or-directory-names/

Generate random file with particular size

Problem

You want to generate a random file used for testing with a particular size.

Solution

You can use `dd` to generate file with random content like this

```
dd if=/dev/urandom of=myFile.dat bs=64M count=16
```

Printing a file

Problem

We want to print a file with different representation. We also want to print various information related to the file.

Solution

For text file we can use various command like `cat` , `head` , `tail` , `more` , `less`

If we want to see file in hex format we can use `hexdump`

```
hexdump <file>
```

To print information about the file such as file type we can use `file` command

```
file <file>
```

To count the number of characters or lines, we use `wc`

```
# this shows number of line, words, character respectively
wc <file>

# show number of lines
wc -l <file>

# show number of words
wc -w <file>

# show number of characters
wc -c <file>
```

Splitting and merging files

Problem

We want to split a big file into smaller files and join them back later to the original file.

Solution

Use `split` to split file easily

```
# Default split will create xaa, xab, etc files
split <file>
> xaa
> xab
> xac
> xad

# Split with fixed number of files, numeric suffix of 3 digits, and prefix
split <FILE> -n 10 -a 3 -d <PREFIX>

# Split with fixed file size, numeric suffix of 3 digits, and prefix
split <FILE> --bytes=1000 -a 3 -d <PREFIX>
```

To merge splitted files, simply `cat` them together

```
cat prefix* > <NEWFILENAME>
```

Converting files to different format

Problem

You want to convert a file to/from different formats

Solution

`iconv` can be used to easily convert files from one character set to another

```
# convert from UTF-8 to ISO-8859-15/latin-1
iconv -f UTF-8 -t ISO-8859-15 <infile> > <outfile>
```

`recode` can do the same thing but `in-place`

```
recode UTF8..ISO-8859-15 <infile>
```

`recode` can also be used to convert line endings

```
# convert newlines from LF to CR-LF
recode ../CR-LF <infile>

# base64 encode file
recode ../Base64 <infile>
```

`recode` can also combine transform character set, line endings and encode

```
recode utf8/Base64..l1/CR-LF/Base64 <infile>
```

This chapter focuses on text manipulation.

Searching text from files

Problem

You want to search for text in a lot of files swiftly.

Solution

You can use `grep` or `egrep`

```
#list only file name
find . | xargs grep 'string' -sl
find / -type f -print0 | xargs -0 grep -l "test"

# print text and file name
grep -r "redeem reward" /home/tom

# egrep with regular expression
egrep "^\s+\"" file1

# grep excluding files
grep -ircl --exclude=*.{png,jpg} "foo=" *
grep -Ir --exclude="*\.svn*" "pattern" *
```

However the much better solution is to use `ag` or `ack`

```
ag -Q --smart-case --ignore=pack*.js --ignore=Code/tag \
--ignore-dir=build --ignore-dir=Code/JSON --ignore-dir=Tools --js "test"

ack -Q --smart-case "test" --js --ignore-file=match:/packed.*\.js/ \
--ignore-file=is:Code/tag --ignore-dir=build --js "test"
```

Removing duplicated lines

Problem

You want to remove duplicated lines in a file or from stdin.

Solution

You can combine `uniq` and `sort` to achieve this.

```
sort garbage.txt | uniq -u
cat garbage.txt | sort | uniq -u
```

Printing a range of lines

Problem

You want to print a range of lines from a file or from stdin, not the whole thing. For instance, you may want to print only the first 3 lines, or the last 5 lines, or everything except the first line, or everything except the last 2 lines.

Solution

First, we can count the number of lines in a file like this

```
wc -l <file>
cat <file> | wc -l
```

Print the first `n` line with `head`

```
head -n 10
```

Print last `n` line with `tail`

```
tail -n 10
```

Print everything except the first `n` line with `tail`

```
tail -n +7
```

Print everything except the last `n` line with `head`

```
head -n -2
```

Print from line `x` to line `y` with `sed`

```
sed -n "1,3p"
```

Converting tab to space

Problem

You want to convert tab to space

Solution

You can use `expand`

```
# convert tab to 4 space in all java files
find . -name '*.java' ! -type d -exec bash -c 'expand -t 4 "$0" > /tmp/e && mv /tmp/e
```

Comparing 2 text files

Problem

You want to compare 2 text files side by side.

Solution

Linux already has a tool to do this called `diff`

```
    diff file1 file2
```

The output will be something like this

```
1c1
< 1
---
> 2
```

where the `<` part is in the first file only and the `>` part is in the second file only.

If you want more visual diff you can use `colordiff`

Sorting lines based on a certain field

Problem

You want to sort a list of lines from a file or from stdin based on a certain field, provided all the lines follow the same format.

Solution

First, you can sort the whole line with `sort`. For instance, you can sort lines in `/etc/password`, which will sort by the username since the username is the first field in each line.

```
# sort password file by username
sort /etc/passwd
```

However, most of the time we want to sort the file based on a field in the middle, and/or some complex formula of the fields, for instance, the ratio between field 2 and field 3. In such cases, we will use `awk` to calculate the derived field then use `sort` on the final result

```
# sort based on field 2 / field 1 then print the result at the beginning of the line
cat somefile.txt | awk '{ratio = $2/$1; print ratio, $0;}' | sort -rnk1 | head
```

Real world example: Counting unique ip access in apache log in a month

```
grep Jan/2004 access.log | grep foo.php | \
awk '{ print $1; }' |  sort -n | uniq -c | \
sort -rn | head
```

Sorting lines based on a certain field 20

This chapter focuses on finding things in the file system.

Finding files based on size

Problem

You want to find the largest file or folders, maybe recursively.

Solution

Find the largest file/folder non-recursively OR sort files and folders by size

```
ls -A | awk '{system("du -sh \""$0"\"")}'| sort -hr | head
```

Find the largest file in a folder and all subfolders recursively

```
find . -type f -print0 | xargs -0 -n 1 du -sh | sort -hr | head

# display in block of 1024-byte
find . -type f -print0 | xargs -0 -n 1 du -sk | sort -nr | head
```

This command use `find` to search for all file recursively. The option `-print0` removes the need for `sed` to escape spaces since all fields now are separated by null character. `args -0` makes sure we use null separator.

Finding files based on size 22

This chapter introduces several simple admistratrative tasks. Most of these commands should be run using `root` account.

Shutting down

Problem

You want to shutdown system, sometimes immediately, sometimes at a certain time or after a certain duration.

Solution

Use the `shutdown` command with `root` privilege.

To immediately shut down and halt the system

```
sudo shutdown -h now
```

To immediately reboot the system

```
sudo shutdown -r now
```

You can optionally send a warning message to all user with `-c` option

```
sudo shutdown -h now "The system is being shut down now!"
```

To shut down the system at a certain time

```
# At 4.23 AM
sudo shutdown -h 4:23

# At 8.00 PM
sudo shutdown -h 20:00
```

To shut down and halt the system after a period of time

```
# In 5 minutes
sudo shutdown -h +5
```

To cancel a shutdown

```
sudo shutdown -c
```

Stuff that does not fit anywhere should go here