Part 2: Useful commands

Master your Command Line

(Before it masters you)

Tejas Sanap

19th July, 2019

1. Introduction
2. Inside the file
3. Inside the directory
4. Manipulate Files and Directories
5. Customize your shell
6. References

Introduction

UNIX Philosophy

Focused on *modularity & reusability*.

It can be summarized as:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Basic Operations

- Search for text (in files).
  - `cat, head, tail, wc`
  - `grep`
- Search for files (in directories).
  - `ls`
  - `find, locate`
- Manipulate files and directories.
  - `cp, scp, rm, mv`
  - `rsync`

GNU Coreutils

The **GNU Core Utilities** are the basic file, shell and text manipulation utilities of the GNU operating system.
They are expected to be present on every operating system.
Previously, the core utilities were implemented by the following pacakages:

1. `fileutils`
2. `shellutils`
3. `textutils`

In 2003, these three packages were combined into the current `coreutils` package.

Globbing

- **Glob** is the common name for a set of Bash features that match or expand specific types of patterns.
- When used for filename matching globs are called **wildcards**.
- Wildcards cannot match filenames that **don't yet exist**.

Glob	Meaning
*	matches all characters, any number of times
?	matches all characters, but only once
[...]	character class
{... , ... , ...}	group patterns seperated by comma's

Example

```
$ for each in "$(ls -d /*)";
do (cd $each && mv ??  ../ && cd -); done
```

regex

A few simple regex characters:

Character	Meaning
^	Beginning of line
$	End of line
.	Single character
*	All characters
{n,m}	Occurance between minimum n and maximum m times

Inside the file

cat, head, cd , wc

Utilities to view file content

Example

```
cat -A -n -s torrent-trackers
```

Example

```
head -n 10 torrent-trackers
```

Example

```
cd , cd .., cd ~, cd -
```

Example

```
wc torrent-trackers
```

wc - Output

```
465 233 9585 torrent-trackers
```

newline, wordcount, bytes, filename

grep

`grep` prints line that matches a certain pattern.

Syntax

```
grep OPTIONS PATTERN INPUT_FILE_NAMES
```

Example

```
$ grep --color=always "anime" torrent-tracker
udp://tc.anime  reactor.ru:8082/announce
udp://tc.anime  reactor.ru:8082/announce
```

grep

The exit status of `grep` when:

- **line is selected** is **0**.
- **no line is selected** is **1**.
- **an error occurs** is **2**.

Useful `grep` options:

`-i`	ignore case
`-v`	invert matches
`-c`	count no. of matching lines
`-n`	prefix each line with line number
`-l`	print name of the file and suppress all other output
`-H`	print filename for each match
`-o`	print only the matched parts of a line
`-s`	suppress error messages
`--color`	color the matching content
`-a`	accept binary input
`--label=LABEL`	display input actually coming from `stdin` as input from file `LABEL`

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xf python_code.tar.gz
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep
main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -n main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -c main'
```

grep

Task

1. We have a tar file named `python_code.tar.gz`
2. We want to search for a function named `main`
3. But, without, extracting or decompressing the tar file

Example

```
$ tar -xzf python_code.tar.gz --to-command='grep -a
-H --label="$TAR_FILENAME" -c -s main'
```

Inside the directory

ls

`ls` displays directory contents.

Useful `ls` options:

- `--sort` `-S`, `-t`, `-X` Size, time, extension
- `--format` `-1`, `-m`, `-l` Horizontal, commas, long
- `-h` human readable
- `-g` don't display file owner
- `-G` don't display file group
- `-d` if argument is a directory, list only its name
- `-I` Ignore files matching pattern
- `--hide` Hide files matching pattern (overriden by `-a`)

`ls`

Task

1. List all the directories in the folder `find`
2. List the last five files/folders to be modified

Example

```
$ ls
```

`ls`

Task

1. List all the directories in the folder `find`
2. List the last five files/folders to be modified

Example

```
$ ls -d */
```

`ls`

Task

1. List all the directories in the folder `find`
2. List the last five files/folders to be modified

Example

```
$ ls -1t | head
```

find

`find` search for files in a directory hierarchy.

Syntax

```
find DIRECTORY EXPRESSION
```

find

`find` search for files in a directory hierarchy.

Syntax

```
find DIRECTORY TESTS ACTIONS
```

find

`find` search for files in a directory hierarchy.

Syntax

```
find DIRECTORY TESTS ACTIONS
```

Example

```
$ find . -name file1b1
```

find

`find` search for files in a directory hierarchy.

Syntax

```
find DIRECTORY TESTS ACTIONS
```

Example

```
$ find . -name file1b1
```

Useful global options:

`-maxdepth n` Descend at most `n` levels

`-mindepth n` Do not apply tests at levels less than `n`

find

Following TESTS are available:

Name	`-name, -iname, -path, -ipath`
Links	
Time	`-atime, -ctime, -mtime, -amin, -cmin, -mmin, -anewer, -cnewer, -mnewer, -newerXY, -used`
Size	`-size, -empty`
Type	`-type`
Owner	`-user, -group`
Mode Bits/ File Permissions	`-perm, -readable, -writable, -executable`
Contents	
Directories	
Filesystems	

find -TESTS

-path

```
$ find . -path '*/dir4a'
./dir1/dir1a/dir2c/dir3a/dir4a
```

-size

```
$ find . -size +5k
$ find . -size -5k
```

find files with some content

```
$ find . -name '*.[23]' | xargs grep -l anime
./dir1/dir1a/dir2c/dir3a/file4.2
./dir1/dir1b/file1b.3
```

find -TESTS

Task

Find files that were edited before:

1. 10 days.
2. 10 minutes.

find -TESTS

Task

Find files that were edited before:

1. 10 days.
2. 10 minutes.

-newerXY

```
$ find . -newermt "Jul 11"
```

-newerXY

```
$ find . -newermt "10:20"
```

find -TESTS

Task

Find and delete all files of a specific file type.

find -TESTS

Task

Find and delete all files of a specific file type.

-regex

```
$ find -regextype egrep -regex ".*(db|jpg)"
```

Use multiple tests

```
$ find . -type f \( -name "*.db" -or -name "*.jpg" \)
```

find -ACTIONS

Syntax

```
-execdir command {} ';'
                 file name
```

-execdir

```
$ find -name "*.db" -execdir rm {} ';'
```

find -ACTIONS

Syntax

```
-execdir command {} ';'
                    end of command
```

-execdir

```
$ find -name "*.db" -execdir rm {} ';'
```

locate

`locate` finds files by name.

It has two drawbacks:

1. It uses the database built using `updatedb`.
2. It does not check if the files still exist.

Useful `locate` options:

`-l, --limit` limit the no. of entries being displayed

`-b, --basename` match only the basename of the file

`-S, --statistics` display the database stats

Example

```
$ sudo updatedb
$ locate
```

Manipulate Files and Directories

Customize your shell

fortune & cowsay

- Let's add some star trek quotes
- fortune
- Cowthink and cowsay

References

References

1. UnixPin
2. `glob`:
   - 2.1 `man 7 glob`
   - 2.2 `man 7 regex`
3. `find`:
   - 3.1 Find History
   - 3.2 GNU Findutils - `info` –> Find

Questions?