Play With Bash

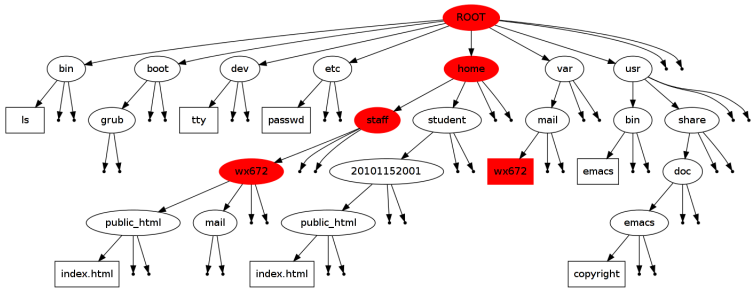
WANG Xiaolin

September 7, 2016

目录

1 Basic Bash Command Line Operations

1.1 Understanding The File System



1.2 Must Known Commands

- **Simple:**

```
ls, cd, pwd, mkdir
cp, mv, rm, ln, chmod
cat, echo, less, more
man, info, help
type, which, whereis
wc, sort, uniq
ps, w, top, free
du, df
ssh, scp
date, cal
```

- **Advanced:**

```
grep, find
tar, gzip, 7z
diff, patch
```

1.3 CLI shortcuts

- **Ctrl-a**: beginning of line
- **Ctrl-e**: end of line
- **Ctrl-f**: forward
- **Ctrl-b**: backward
- **Ctrl-n**: next
- **Ctrl-p**: previous
- **Ctrl-r**: reverse search
- **Ctrl-u**: cut to beginning
- **Ctrl-k**: kill (cut to end)
- **Ctrl-y**: yank (paste)
- **Ctrl-d**: delete a character

1.4 Examples

- `>` — output to a file

  ```
  date  >  file1
  ```

  Check what's in `file1`:

  ```
  cat  file1
  ```

- `>>` — output a string

  ```
  echo Hello, world
  ```

  output to `file2` rather than `STDOUT` (screen)

  ```
  echo 'Hello, world!' >> file2
  cat file2
  echo 'Hello again, world!' > file2
  cat file2
  ```

Q: Can you explain the difference between > and >>?

- `cat` — concatenate files

```
cat file1
cat file2
cat  file1 file2
cat file1 >> file2
cat file2
cat file1 > file2
cat file2
cat >> file2
cat file2
cat > file2
cat file2
```

Q: Can you explain the above commands?

- `ln` — one file can have several names (shortcuts)

```
ln -s file1 file11
ls -l file*
```

- `head` — list the head (first 10 lines) of file1?

```
head file1
```

- `tail` — list the tail (last 10 lines) of file1?

```
tail file1
```

- `cp` — copy

```
cp  file1  file111
ls  -l file*
cp file* /tmp
ls -l /tmp
```

When copying directories, you need the '`-a`' option:

```
mkdir testdir
cp -a testdir /tmp
```

- `mv` — move/rename

```
mv  file1  file1111
ls  -l  file*
mv file* /tmp
ls -l /tmp/file*
```

Q: Any differences between **move** and **rename**?

- **file types**, **file modes**, and **file permissions**

  Given the following `ls -l` output:

```
lrwxrwxrwx 1   root root 23     May 17   2010 /usr/bin/emacs -> /etc/
alternatives/emacs*
-r--r--r-- 1  root sys  418   Oct 13 16:25 /etc/passwd
drwxrwxrwx 10  bin bin  1024  Oct 15 20:27 /usr/local/
-r-sr-xr-x 1  root bin  28672 Nov  6  1997 /usr/sbin/ping
```

  Tell me:

  1. the file type of the above files?
  2. the owner and group of the above files?
  3. the permissions for the owner, group, and all ”other” users of the above files?

- `chmod` — change **file mode**

```
chmod  777  file1  &&  ls  -l  file1
chmod  000  file1  &&  ls -l file1
chmod  a+rwx  file1  &&  ls  -l  file1
chmod  a-rwx  file1  &&  ls  -l  file1
chmod  755  file1  &&  ls  -l  file1
chmod  700  file1  &&  ls  -l  file1
chmod  777  file1  &&  ls  -l  file1
chmod  go-rwx  file1  &&  ls  -l  file1
chmod  600  file1  &&  ls  -l  file1
chmod  u+x  file1  &&  ls  -l  file1
```

- `wc` — word count

```
wc  -l  file1
```

2 Shell Basics

2.1 Shabang

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/python
#!/usr/bin/php
```

2.2 Shell variables

```
echo  $PATH
echo  $PWD
echo  $HOME
echo  $USER
```

Try command `env`, `set`, `unset`

2.3 PATH

```
PATH="./:$PATH"
echo $PATH
```

2.4 Background and foreground jobs

- To run a command in the background

  ```
  emacs &
  google-chrome &
  ```

  Show background jobs:

  ```
  jobs
  ```

- To push a foreground process into background?

  ```
  Ctrl-Z
  bg %1
  ```

2.5 Processes

```
ps ux
ps aux
top
w
```

2.6 File types

1. normal files
2. directories
3. links
4. block devices
5. character devices
6. pipes
7. sockets

Hint: the last four can be found in the `/dev` directory

2.7 Special files

- `/dev/null`

```
ls  >  /dev/null
cat  log* nullfile  2>  /dev/null
cat  log* nullfile  &>  /dev/null
```

- `/dev/zero`

```
dd  if=/dev/zero  of=/tmp/testfile  bs=1k  count=1000
```

- `/dev/random`

```
echo $(( `od -An -N2 -i /dev/random` % 1000 ))
```

2.8 Soft links and hard links

```
ln -s /tmp/a /tmp/aa
ls -l /tmp/a*
ln /tmp/a /tmp/aaa
ln /tmp/aa /tmp/aaaa
ls -li /tmp/a*
```

2.9 Getting help

```
man -k music player
apropos music player
info tar
```

2.10 Advanced commands and concepts

2.10.1 Pipe — chaining commands together

```
man ls | head
man ls | head | tail -3
cat file1 | head -20 | tee file5
```

2.10.2 finding a file

```
find  /  -name  ls
type  ls
which  ls
whereis ls
find  /etc  -type d -name "rc*"
find ~ -name "*~" | xargs rm
```

2.10.3 grep — finding lines in files

```
grep stud /etc/passwd
man cp | grep -B2 -A2 recur
```

2.10.4 Single-quotes and double-quotes

```
a=alpha;  b="$a";  c='$a'
echo  a  b  c
echo  $a $b $c
echo '$a $b $c'
echo "$a $b $c"
```

2.10.5 Wildcard charactors

```
mkdir  tmp  &&  cd  tmp
for  ((i=0;i<101;i++));  do  touch  f$i;  done
ls  f*
ls  f?
ls  f??
ls  ??
ls  ?8*
ls  *0
```

2.10.6 Command alias

```
alias
alias la='ls -a'
```

```
alias rm='rm -i'
which rm
```

2.10.7 STDIN, STDOUT, STDERR, and redirection (>, >>, <)

- Redirect STDOUT into a file

```
ls  >  listing
cat  listing  >  listing2
cat  listing*  >  listing3
cat  listing* >> listing3
```

- Redirect STDIN from a file

```
cat  <  listing
sort  <  listing
```

```
#!/bin/bash
while  read  LINE
do
  case  $LINE  in
  *root*)  echo  $LINE ;;
  *stud*)  echo  $LINE ;;
       *)  echo  "I don't care." ;;
  esac
done  <  /etc/passwd
```

- Redirect STDERR into a file

```
touch  realfile
ls  nullfile  realfile

ls  nullfile  realfile  2>  log2
ls  2>  log  nullfile  realfile
```

- Redirect both STDERR and STDOUT into a file

```
ls  nullfile  realfile  &>  log3
ls  nullfile  realfile  > log3  2>&1

diff  log*

cat  nullfile  realfile  &>  log4
cat  &>  errorlog  <  nullfile realfile
```

2.10.8 Initial files

- .bashrc, .bash_profile, .profile

```
vim  .bashrc
source  .bashrc
.  .bashrc
```

2.10.9 tar

```
tar  cvf  myarchive.tar  /etc/termcap  /etc/passwd
tar  tvf  myarchive.tar
tar  xvf  myarchive.tar
```

With compression:

```
tar  zcvf  myarchive.tgz  /etc/termcap  /etc/passwd
tar  zxvf  myarchive.tgz
tar  ztvf  myarchive.tgz
```

2.10.10 gzip

```
gzip file1
zcat file1.gz
gunzip file1.gz
```

2.10.11 System info

```
mount
uname -a
dmesg
lspci
lsusb
lsmod
```

2.10.12 Job scheduling

- at

```
at  11:00
at>  date >> $HOME/date.out
at>  type CTRL-D to quit
at -l
```

- crontab

```
crontab -e
*/2 * * * * date  >>  $HOME/date.out
crontab -l
```

3 Bash Programming Examples

3.1 for VAR in LIST; do SOMETHING; done

```
for i in 1 2 3 4 5; do echo $i; done
for ((i=1;i<6;i++)); do echo $i; done
```

```
for i in 1 2 3 4 5; do echo $((i*i)); done
for ((i=1;i<6;i++)); do echo $((i*i)); done
```

```
for i in 1 2 3 4 5; do j=$((i*i)); echo $i $j; done
for ((i=1;i<6;i++)); do j=$((i*i)); echo $i $j; done
```

```
#!/bin/bash
# check disk usage.
for f in /home/students/*
do
  du -cks $f | grep -v total
done | sort -n | tail -10
```

```
for f in /home/stud/*; do du -b $f; done | sort -n | tail -10
du -b /home/stud/* | sort -n | tail -10
```

```
for f in *jpg; do convert $f -resize 1280x -gravity center -crop 1280x768+0+0 `basename
```

3.2 if TEST; then COMMANDS; else OTHERCOMMANDS; fi

3.2.1 Comparisons

```
if [ $a -lt 10 ];   then a=$(($a+1)); echo $a; else echo "a is too large."; fi

```

```
if [[ $a -lt 10 ]]; then a=$(($a+1)); echo $a; else echo "a is too large."; fi

if (("$a" < 10));   then a=$(($a+1)); echo $a; else echo "a is too large."; fi
```

```
#!/bin/bash
# This is a simple string comparision script.
#
# 1. Use '[[' instead of '[' whenever possible.
# 2. Don't use '[  ]' with '<', '>'.
# 3. '-eq -le -ge -lt -gt' are for arithmetic comparisons
# 4. '< >' for string comparisons
#
if [ -z "$2" ]; then
    echo Usage: $0 '<string1> <string2>'
elif [[ "$1" > "$2" ]]; then
    echo $1 is bigger than $2.
elif [[ "$1" = "$2" ]]; then
    echo $1 is equal to $2.
else
    echo $1 is smaller than $2.
fi
```

```
if [[ $(ls | wc -l) -gt 10 ]]; then echo "messy!"; else echo "clean!"; fi
```

- Other Comparison Operators

3.2.2 Test exit status

```
#!/bin/bash
for f in *.sh
do
    if grep -q while $f; then
^^Iecho "$f: while loop found\!"
    else
^^Iecho "$f: no while loop."
    fi
done
```

```
if grep -q while while.sh ; then echo "While loop found."; else echo "no while loop"; fi
```

```
for f in *.sh; do if grep -q while $f; then echo "$f: while loop found\!"; else echo "$
```

3.2.3 See your `.bash_profile`

```
# include .bashrc if it exists
if [ -f ~/.bashrc ]; then
   . ~/.bashrc
fi
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
   PATH=~/bin:"${PATH}"
fi
```

3.3 `while` CONDITION; `do` SOMETHING; `done`

```
while true; do mpg123 song.mp3; done

while true; do mpg123 `find ~/ -iname "*.mp3"`; done
```

```
#!/bin/bash
x=0
while [ $x -lt 10 ]      #  [ ]
do
    y=$x
    while [[ $y -ge 0 ]]   #  [[ ]]
    do
^^Iecho -n $y          # no newline
^^Iy=$((y-1))          # y--
    done
    echo
    x=`echo "$x + 1" | bc` # x++
done
```

3.3.1 `read` — Read a line from STDIN

```
while read LINE; do echo "what I typed is: $LINE"; done
```

```
#!/bin/bash
while read LINE
do
    case $LINE in
^^I*root*) echo $LINE ;;
^^I*stud*) echo $LINE ;;
^^I*) echo "I don't care." ;;
    esac
done < /etc/passwd
```

3.4 case VAR in PATTERN) COMMANDS ;; esac

```
#!/bin/bash
printf "Play a game?"
read YN
case $YN in
  [yY]|[yY][eE][sS]) exec bb ;;
^^I^^I  *) echo "Maybe later." ;;
esac
```

```
#!/bin/bash
YN=yes
printf "Play a game?[$YN]"
read YN
: ${YN:=yes}
case $YN in
  [yY]|[yY][eE][sS]) exec bb ;;
^^I^^I  *) echo "Maybe later." ;;
esac
```

3.5 Command line arguments ($0, $1, $2..., $#, $@)

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i;
  printf("You said:\n\t");

  for(i=1; i<argc; i++)
    printf("%s ",argv[i]);

  printf("\n\n\targc = %d\n", argc);

  for(i=0; i<argc; i++)
    printf("\targv[%d] = %s\n",i,argv[i]);

  return 0;
}
```

Listing 1: An example C program:

```bash
#!/bin/bash

echo "You said:"

echo -e "\t$@"
echo
echo -e "\targc = $#"
echo -e "\targv[0] = $0"

i=1
for arg in $@; do
    # printf "arg[i]is%s""arg"
    echo -e "\targv[$i] = $arg"
    let i++
done
```

Listing 2: An equivalent bash script:

3.6 Arrays

Set random wallpaper:

```
#!/bin/bash
### demonstrate ARRAY and RANDOM ###

files=($HOME/pics/2009summer/wallpapers/2009summer-1280x768/*.jpg)

```

3.7 GUI

```
#!/bin/bash
while NAME=`zenity --entry --text="Your name?"`
do
    zenity --info --text="Hello, $NAME\!"
done
```

3.8 `/etc/init.d/*`

Check files in `/etc/init.d/` directory to see how shell scripts can be used seriously. Emacs 24.5.1 (Org mode 8.2.10)