UM–SJTU Joint Institute

Introduction to Operating Systems
(VE482)

Laboratory Report

Lab 2

Name: Ji Xingyou ID: 515370910197
Date: 22 September 2017

1 Basic shell

- Use the mkdir, touch, mv, cp, and ls commands to:
  - Create a file named test.

```
touch test
```

  - Move test to dir/test.txt, where dir is a new directory.

```
mkdir dir
mv test /dir/test.txt
```

  - Copy dir/test.txt to dir/test_copy.txt.

```
cp dir/test . txt dir/test copy . txt
```

  - List all the files contained in dir.

```
ls -a /dir/
```

- Use the grep command to:
  - List all the files from /etc containing the pattern 127.0.0.1.

```
grep '127.0.0.1' etc/*
```

  - Only print the lines containing your username and root in the file /etc/passwd (only one grep should be used)

```
grep    E  'root|HappyBirthdayHoney' /etc/passwd
```

- Use the find command to:
  - List all the files from /etc that have been accessed less than 24 hours ago.

```
find /etc -type f -ctime  -1
```

  - List all the files from /etc whose name contains the pattern netw.

```
find /etc -name ``$netw$''
```

- In the bash man-page read the part related to redirections. Explain the following signs >, >>, <<<, > &1, and 2 > &1 >. What is the use of the tee command.
  tee: read from standard input and write to standard output and files. **>:** redirect the standard output to a file and override the original file
  >>: redirect the standard output to a file and append to the original file
  <<<: redirect the content behind it to the standard input of the content ahead of it
  > &1: redirect the standard output to standard output
  2 > &1 >: redirect the standard error information to the standard output

```
tee [OPTION]... [FILE]...
```

-a: append to the given files and do not overwrite

-i: ignore interrupt signals

-p: diagnose errors writing to non pipes

- Explain the behaviour of the xargs command and of the | sign.
  xargs: build and execute command lines from standard input

  |: transport the output of command 1 to command 2

```
command 1 | command 2
```

- What are the head and tail commands? How to live display a file as new lines are appended?
  head: output the first part of files

```
head [OPTION]... [FILE]...
```

tail: output the last part of files

```
tail [OPTION]... [FILE]...
```

live display:

```
tail -f
```

- How to monitor the system using ps, top, free, vmstat?

```
ps
```

```
top -hv | -bcHioSs -d secs -n max -u|U user -p pid -o fld -w [cols]
```

```
free [OPTION]
```

```
vmstat [OPTION] [DELAY [COUNT]]
```

- In Minix 3, how to manage softwares (install, remove, update. . .)?

```
pkgin install [NAME]
```

```
pkgin remove [NAME]
```

```
pkgin update
```

- What is the purpose of the commands ifconfig, adduser, and passwd?
  ifconfig: configure a network interface

  adduser: add a user to the system

  passwd: change user password

2 Working on a remote server

- Setup an SSH server on Minix 3. From Linux (using ssh) or Windows (using Putty) log into Minix 3. Note: the network need to be properly setup on the Virtual Machine (VM).

- What is the default SSH port? Change this port for port 2222. Log into Minix 3 using this new SSH server setup.
  The default SSH port is 22.

- List and explain the role of each the file in the $\$HOME/$.ssh directory. In $\$HOME/$.ssh/config, create an entry for Minix 3.
  id_rsa.pub: public key for the ssh connection
  id_rsa: private key for the ssh connection

3 Basic Bash scripting

- What should be the first line of a Bash script?

```
#!/bin/bash
```

- What are the main differences between sh, bash, csh, and zsh?
  Some of them are better.

- How to define and access variables?
  define:

```
course_name=``ve482''
```

  access:

```
$course_name
```

- What is the meaning of $\$0, \$1, ..., \$?, \$!$?
  $0: the name of the file to be executed

  $1: first parameter in the script

  $?: show the exit status of the last command

  $!: the ID number of the last process in the background system

  $#: show the number of the current parameters

- How to define arrays and access or assign elements?
  define:

```
array_name=(value_1 ,value_2 ,... ,value_n)
```

  access:

```
${array_name[index]}
```

  assign:

```
${array_name[index]}
```

- How to perform if and switch statements? Provide an example.

if:

```
if condition
then
    command1
    command2
    ...
    commandN
else
        command
fi
```

example:

```
a=10
b=20
if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less then b"
else
   echo "no matches"
fi
```

switch:

```
case value in
mode 1)
    command1
    command2
    ...
    commandN
    ;;
mode 2
    command1
    command2
    ...
    commandN
    ;;
esac
```

example:

```
echo 'Please input a number between 1 and 4:'
read input
case $input in
    1)  echo 'You input 1'
    ;;
    2)  echo 'You input 2'
    ;;
    3)  echo 'You input 3'
    ;;
```

```
    4)  echo 'You input 4'
    ;;
    *)  echo 'Invalid input'
    ;;
esac
```

- What are the various syntaxes of a for loop? For each type write a sample code.
  for:

```
for ((i=0;i<10;++i))
do echo $i
done
```

```
for i in {1..10}
do echo $i
done
```

```
for i in $(seq 1 10)
do echo $i
done
```

- How to write a while loop?
  while:

```
while condition
do
    command
done
```

- What is the use of the PS3 variable? Provide a short code example.
  PS3 variable is used to customize the prompt information.
  example:

```
PS3="Select a day (1-4): "
select i in mon tue wed exit
do
    case $i in
        mon) echo "Monday";;
        tue) echo "Tuesday";;
        wed) echo "Wednesday";;
        exit) exit;;
    esac
done
```

- What is the purpose of the iconv command, and why is it useful?
  iconv: For the given file, translate its content from one encoding to another.
  It is useful because for example, if we want to get some data online, we can use iconv to translate them into the correct encoding.

- Given a variable $temp what is the effect of ${#temp}, ${temp%%word}, ${temp/pattern/string}.
  ${#temp}**:** output the length of the variable
  ${temp%%word}**:** remove the first word in temp
  ${temp/pattern/string}**:** change the first pattern in temp to string

- Search what are regular expressions and how to use them in a grep or find command. Give some simple examples based on files and keywords used in exercise 2 of assignment 2.

- Provide a brief introduction to both of them, explaining how to use them and when they reveal to be the most helpful.
  sed: stream editor for filtering and transforming text

```
sed [OPTION]... {script-only-if-no-other-script} [input-file]
```

  awk: pattern scanning and text processing language

```
awk [-F fs] [-v var=value] ['prog'|-f progfile] [file ...]
```

- Using curl or wget retrieve information on shanghai air quality and pipe it to sed which should parse the output in order to display the information in the terminal following the format below AQ: value Temp: value oC (e.g. AQ: 55 Temp: 24 oC).

```
curl --silent 'http://aqicn.org/?city=Shanghai&widgetscript&size=large&id=52b39
d71decf07.20261781' | sed 's/.*title=\\"Moderate\\">/AQ: /g' | sed 's/<\/div.*/
/g'

curl --silent 'http://aqicn.org/?city=Shanghai&widgetscript&size=large&id=52b39
d71decf07.20261781' | sed "s/.*style\='font-size:10px;'>/Temp: /g" | sed 's/<\
/td><\/tr><\/table>.*//g' && echo    e     \u00B0C
```

- Pipelining the output of ifconfig to awk return only the ip address of your current active network connection (the active network interface can be passed to ifconfig).

```
ifconfig wlp3s0 | awk '/inet /{print $2}'
```