VE482 Lab 2

Liu Yihao 515370910207

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/text_copy.txt
    ```
  - List all the files contained in `dir`.
    ```
    ls dir -a
    ```
- Use the `grep` command to:
  - List all the files form `/etc` containing the pattern `127.0.0.1`.
    ```
    grep -rl '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 -rE '(liu|root)' /etc/passwd
    ```
- Use the `find` command to:
  - List all the files from `/etc` that have been accessed less than 24 hours ago.
    ```
    find /etc -atime 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.

  `>` redirects the standard output into a file.

  `>>` redirects and appends the standard output into a file.

  `<<<` redirects the contents on the right as the standard input of the command on the left.

  `>&1` redirects the standard output into standard output (meaningless).

  `2>&1 >` redirects the standard error into standard output, and redirects the origin standard output into a file.

- Explain the behaviour of the `xargs` command and of the | sign.

  `xargs` is used to build and execute command lines from standard input, by combining multi lines and extra spaces into a line with single spaces.

  The | sign pipes the standard output of the command on the left into the command on the right as the standard input.

- What are the `head` and `tail` commands? How to "live display" a file as new lines are appended?

  `head` and `tail` are used to get the first and last several lines of a file.

  Use the `-f` option of `tail` to "live display" a file as new lines are appended.

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

  `ps` is used to monitor the processes.

  `top` is used to monitor the CPU and RAM of processes.

  `free` is used to monitor the RAM.

  `vmstat` is used to monitor the RAM, IO and CPU in a period.

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

```
pkgin update       # Update the package repository
pkgin install name # Install a package
pkgin remove name  # Remove a package
pkgin upgrade name # Upgrade a package
pkgin search name  # Search a package
```

- What is the purpose of the commands `ifconfig`, `adduser`, and `passwd`?

  `ifconfig` is used to check the state of network.

  `adduser` is used to create a new user.

  `passwd` is used to set password for the current user.

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).

```
ssh root@localhost
```

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

  The default port is 22.

  On Minix3:

```
vi /etc/ssh/sshd_config
```

  and edit the option "Port".

  On Linux:

```
ssh root@localhost -p2222
```

- List and explain the role of each the file in the `$HOME/.ssh` directory. In `$HOME/.ssh/config`, create an entry for Minix 3.

  ```
  ls $HOME/.ssh
  ```

  shows the files

  ```
  config  id_rsa  id_rsa.pub  known_hosts
  ```

  `config` is the config file of ssh client.

  `id_rsa` is the private key of ssh client/server.

  `id_rsa.pub` is the public key of ssh client/server.

  `known_hosts` is the ssh servers logged before.

  On Linux:

  ```
  vi ~/.ssh/config
  ```

  and add these lines

  ```
  Host    minix
      HostName        localhost
      Port            2222
      User            root
  ```

  Then directly connect Minix 3 by

  ```
  ssh minix
  ```

- Briefly explain how key-only authentication works in SSH. Generate a key-pair on the host system and use it to log into Minix 3 without a password.

  First Alice (ssh client) generates a pair of key, and sends the public key to Bob (ssh server). Bob adds the key to a list of authorized hosts. When Alice wants to login, she sends the public key again and Bob uses principles of public key cryptography to verify the identity of Alice. If success, Bob and Alice can connect with a tunnel encrypted by their key pairs.

  ```
  ssh-keygen -t rsa
  scp -P2222 ~/.ssh/id_rsa.pub root@localhost:/root
  ssh minix
  cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
  exit
  ssh minix -i ~/.ssh/id_rsa
  ```

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`?

  `sh` is the origin shell of Unix, `bash`, `csh`, and `zsh` are different expansion of `sh`, they are all compatible with `sh`, but perhaps not compatible with each other.

- How to define and access variables?

```
var=1          # define a variable named var and assign it as 1
let var=var+1 # maths calculation
var=$var+1     # string concat
echo ${var}    # echo the variable
```

- What is the meaning of `$0`, `$1`,..., `$?`, `$!`?

  `$0` means `argv[0]` in C.

  `$1` means `argv[1]` in C.

  `$?` means the exit status of the last command.

  `$!` means the process id of the last command.

- How to define arrays and access or assign elements?

```
array=(var1 var2 var3 varN)
echo ${array[0]}
array[3]=var3
echo ${array[3]}
```

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

```
# if statement
if [ "$1" = "" ]; then
    echo 0
elif [ "$1" = 1 ]; then
    echo 1
else
    echo 2
fi

# switch statement
case $1 in
    1 )
        echo 1 ;;
    2 )
        echo 2 ;;
    * )
        echo 0 ;;
esac
```

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

```
# for-in loop
for file in * ; do
    echo $file
done

# c-style for loop
for ((i=0; i<10; ++i)); do
```

```
    echo $i
done
```

- How to write a `while` loop?

```
i=0
while [ $i -lt 10 ]; do
    echo $i
    let i=i+1
done
```

- What is the use of the `PS3` variable? Provide a short code example.

  PS3 is the select prompt `#?`, it is displayed in a select statement.

```
echo "Who is the strongest person in JI"
choices='xd qss gg xyy wgz'
select person in $choices; do
    if [ $person ]; then
        echo "$person is the strongest person in JI."
        break
    else
        echo "Invalid, select again."
    fi
done
```

- What is the purpose of the `iconv` command, and why is it useful?

  The `iconv` command reads in text in one encoding and outputs the text in another encoding. It is useful because there are various encoding in different operation systems, if one wants to use a file from another environment, the transform is important.

- Given a variable `$temp` what is the effect of `${#temp}`, `${temp%%word}`, `${temp/pattern/string}`.

  `${#temp}` means the length of `$temp`.

  `${temp%%word}` deletes `word` on the right of `$temp`.

  `${temp/pattern/string}` replaces `pattern` with `string` in `$temp`.

- 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.

  Regular expressions are some very useful expressions that I can never understand. However, there are also some masters in RegExp such as xyy, xtt and xd.

Two programming languages often used in conjunction with Bash are `sed` and `awk`.

- Provide a brief introduction to both of them, explaining how to use them and when they reveal to be the most helpful.

  Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

awk scans each input file for lines that match any of a set of patterns specified literally in prog or in one or more files specified as -f filename. With each pattern there can be an associated action that will be performed when a line of a file matches the pattern. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern. The file name - means the standard input. Any file of the form var=value is treated as an assignment, not a filename, and is executed at the time it would have been opened if it were a filename.

- 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 °C (e.g. AQ: 55 Temp: 24 °C).

```
curl --silent 'http://aqicn.org/?city=Shanghai&widgetscript&size=large' \
| sed ':t;N;s/\n//;b t' \
| sed 's/\([^\n]\+\)\\\">/AQ: /g' \
| sed "s/<\/div>\([^\n]\+\)10px;'>/ Temp: /g" \
| sed 's/<\/td>\([^\n]\+\)//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 wlp2s0 | awk '{if(NR==1){print $2}}'
```