Homework #1

Due Time: 2018/3/11 (Sun.) 22:00
Contact TAs: `vegetable@csie.ntu.edu.tw`

Submission

- Compress all your files into a file named **HW1_[studentID].zip** (e.g. `HW1_bxx902xxx.zip`), which contains two folders named **[studentID]_NA** and **[studentID]_SA** respectively.
- **Folder [studentID]_NA** should contain a pdf file named **na.pdf** of all your answers in *Network Administration Part*.
- **Folder [studentID]_SA** should contain a pdf file named **sa.pdf** for SA section 1 and three indicated scripts for SA section 2.
- Submit your zip file to Ceiba.

Instructions and Announcements

- Discussions with others are encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (the URL of the web page you consulted or the people you discussed with) on the first page of your solution to that problem.
- Problems below will be related to the materials taught in the class and may be far beyond that. Try to search for additional information on the Internet and give a reasonable answer.
- Some problems below may not have standard solutions. We will give you the points if your answer is followed by reasonable explanations.
- **NO LATE SUBMISSION OR PLAGIARISM IS ALLOWED.**

Network Administration

1. Internet Protocol Stack: 5-layer Model

(a) What information is used to identify the packet source and destination in Transport, Network and Link layer, respectively? (5%)

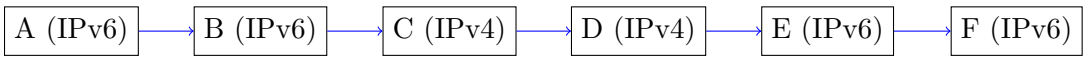
(b) For each layer in the 5-layer model, give an example of what protocol belongs to it and briefly explain its functionality. (10%)

2. IP

(a) Theoretically, how many address can IPv4 and IPv6 provide, respectively? (Without considering extensions such as private IP and NAT.) (2%)

(b) We know that an A record in the DNS server maps a domain name to an IPv4 address. If we want the DNS server to support IPv6, what type of record should be added? (3%)

(c) It will take a long time for the world to migrate from IPv4 to IPv6. In this transition state, devices that support IPv6 and devices that only support IPv4 will coexist and must work with each other. For example, suppose that A wants to send an IPv6 datagram to E, but some intermediate routers (C, D) only support IPv4. In such situation, please briefly explain how tunneling may help to resolve the problem. (5%)



3. Wireshark

"Wireshark is a free and open source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education." , Wikipedia. In this section, we' ll familiar ourselves with the basic usage of this useful tool.

First, perform the following steps.

1) Open Wireshark and start capturing packets.
2) Open your browser and go to headhunt.com.tw/pages/login.aspx.
3) Try to login with username `nasahw1` and password `wireshark`.
4) Stop capturing packets in Wireshark.

For the following questions, please also provide the display filter you applied as a part of your answer.

(a) Filter out all dns related packets. What is the IP address of the dns server that your machine is querying? (5%)

(b) Filter out dns packets that involve querying headhunt.com.tw. What is the corresponding IP address of headhunt.com.tw? (5%)

(c) Filter out all packets whose destination IP address is the one you found in the previous question. What protocols are used by these packets you filtered out? (5%)

(d) Among the packets that you filtered out in the previous question, can you find a packet that contains the password Wireshark? What is the protocol of this packet? (5%)

(e) What protocol should be used to replace HTTP so that sensitive data sent through the Internet will be protected? (5%)

System Administration

1 Short Answer Questions

1.1 More Permissions! (5%)

During the class, we mentioned the directory permission, which was described to be more complex than file permission.
Also, as you should recall, when we went through the file flag (which appears when we use `ls -l` to check file attributes), there are three types of files: `-` for normal files, `d` for directories, and `l` for symbolic links.

(1) Please fill in the table (T or F), in which you should compare different permission setting on directories, and also give a brief conclusion about it.

Note: subdir is the directory under dir.

Dir Permission	Delete Rename Create files	List dir (ls)	Read file content	Write file content	cd dir	cd subdir	List subdir (ls)	Access subdir files
`---` (0)								
`-w-` (2)								
`r--` (4)								
`rw-` (6)								
`--x` (1)								
`-wx` (3)								
`r-x` (5)								
`rwx` (7)								

(2) Please describe the permission on symbolic link (How does it work?)

(3) There are some special permissions we haven't mentioned in class: `setuid`, `setgid` and `sticky bits`. Please explain the usage of them.

UPDATE:
For (1), the permission of files and subdir is 'rwx'.

1.2 Deeper, deeper (3%)

As you should still remember, when we were writing the hello-world script, the first line of the script is `#!/usr/bin/env bash`.

(1) What does that mean?

(2) Why don't we just use something like `#!/usr/bin/bash` ? Which is better? Give your explanation.

1.3 Copy Monster (2%)

In shell scripting you can combine all kinds of command to achieve any task, but we only covered some of them in class. Maybe you've noticed that there is a command called `rsync`, which is "a fast, versatile, remote (and local) file-copying tool".
Please compare it with the `cp` command taught in class: Which is better? When should we use `rsync`, and when should we use `cp`? Explain in your own words.

2 Shell Scripting

Before you start writing shell scripts, make sure you are aware of these rules:

- You're allowed to use the standard shell (POSIX, sh) or GNU bash.
  If you use the standard shell, the first line of your script must be `#!/bin/sh`.
  If you use bash, the first line of your script must be `#!/usr/bin/env bash`.
- Before submission, test each script with the `shellcheck` command, available on CSIE workstations; we'll also test your scripts on CSIE workstations, so please make sure your scripts run fine on the workstations!
- You might find that the sample output is hard to match exactly. Don't worry, we won't check your result with automatic checker, but read your scripts one by one and grade it by your understanding of shell scripting. Little mismatch is ok, what's important is to make sure your scripts behave correctly.

2.1 SSH Ninja (15%)

Sometimes you want to run a task across all workstations (linux1∼linux15, oasis1∼oasis3, bsd1), and it can be tedious to repeatedly enter password and commands in each SSH session.
An obviously smarter solution is to write a script automates that for you. Write a shell script satisfying the following requirements:

- Name: `deploy.sh`
- Usage: `./deploy.sh [username] [command] ...`
- Sample output:

```
$ ./deploy.sh b04902010 echo "hi"
======== linux1 ========
b04902010@linux1.csie.ntu.edu.tw's password:
hi
========================
======== linux2 ========
b04902010@linux2.csie.ntu.edu.tw's password:
hi
========================
......
========================
======== oasis1 ========
b04902010@oasis1.csie.ntu.edu.tw's password:
hi
========================
......
========= bsd1 =========
Password for b04902010@bsd1.csie.ntu.edu.tw:
hi
========================
```

- Subtask:
  1. Satisfy the least requirement (log into every workstations and execute the command): 5%
  2. Use SSH key to login: 5%
  3. Use SSH key, login with only type SSH passphrase once: 5%
  4. (Bonus) Use terminal multiplexer to deploy each process in distinct local sessions (you have to make it a controllable option): 3%
- Note:
  Here is an example of controllable option:

  ```
  # Just login and execute command sequentially
  ./deploy.sh b04902010 echo "hi"
  # Login and execute command in distinct sessions
  ./deploy.sh -d b04902010 echo "hi"
  ```

  If you want to get the bonus, you have to write some comment in your code to let us know how you implement the option.
- Keywords:
  For loop, SSH key, SSH agent, tmux, screen
- **UPDATE:**
  1. The word passphrase in subtask 2 is the one for SSH key; if you use SSH key to login, the output will become like this:

     ```
     Enter passphrase for key '/home/b04902010/.ssh/ws_ecdsa':
     hi
     ```

     And if you want to get the credit for subtask 3, you have to make sure we only need to type the passphrase once for the whole process.
     Also, since that you need to login with SSH key, the usage should become like below:

     ```
     ./deploy.sh [username] [key] [command] ...
     ```

  2. For subtask 2 & 3, you can assume that the key is already on the workstations (you don't need to generate them and put them into the '.ssh' directory.)
  3. For the bonus, you can just let them run in the detached sessions, and close the sessions after they're finished. You don't need to record the output of each sessions and print them to STDOUT or STDERR.
  4. For this problem, we'll test your script on workstations, too. However, you should assume that you're on a remote machine, use the script to run some task across the workstations.

2.2 Do You Want Some Receipts? (10%)

Every two months, we check our receipts for possible lottery prize. Write a shell script to read the winning numbers and receipt numbers, and do the checking for us.

- Name: `lottery.sh`

- Usage:
  `lottery.sh [the file of winning numbers] [the file of receipt numbers]`
- Input format:
```
# the file of winning numbers
75350343 # special prize
67035249 # grand prize
03696891 # first prize 1
79882491 # first prize 2
77486437 # first prize 3
055 # additional sixth prize 1
816 # additional sixth prize 2
292 # additional sixth prize 3
# the file of receipt numbers
67035249
ac-80986891
AG09182055
blablablabla
```
- Output format:
```
# Always use plural is ok
The number of receipts: 27
The number of valid receipts: 20
The number of winning lotteries: 10
The winning money: 2044400
Winning Lotteries:
1. 67035249 (1) $2000000 # number (index) $money
2. ac-80986891 (2) $1000
```
- Subtask:
  No subtask. You have to do all the things right to get the credit.
- Note:
  1. There won't be any comment (e.g. `# grand prize`) in input files; also, don't output any comments.
  2. Valid receipt number will be in one of the the formats below:

     `[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]`
     `[A-Za-z][A-Za-z]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]`
     `[A-Za-z][A-Za-z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]`

  3. Receipt number not in one of the three formats above is considered invalid.
  4. You should output the winning lottery number in the same format as in the input file.
- **UPDATE:**
  The indexes of lotteries in sample output are "line numbers in sample input".

2.3 They Took Our Jobs! (15%)

Well, it's annoying that when you're using one of the workstations, the workstation is so overloaded that even opening an editor seem to take forever. Even though it doesn't help, you want to find out which grade of students consumes the most resources (CPU and memory) on workstation.
(To ease the problem, you only need to calculate the values for `bxx` (undergradate), `rxx` (master student), and `dxx` (PhD student); for other users, you should sum the resource, and print them out as the group `others`.)

- Name: `jobs.sh`
- Usage:
```
If no option is given, simply print the grade that consumes the most resource
(according to CPU usage first, then memory usage).
OPTIONS:
-l, --list          listing all grades in descendant order of consuming resource
(according to CPU usage first, then memory usage)
-m, --mem           print the usage of memory (in KB)
-c, --cpu           print the usage of CPU (in %)
-h, --help          print this help message
```
- Sample output:
```
$ ./jobs.sh # default setting
GROUP
others
$ ./jobs.sh --list
GROUP
others
b99
b90
b93
b75
r93
r03
```

```
r01
d97
b95
b94
b91
b89
b02
$ ./jobs.sh -lm # You have to deal with this kind of arguments!
GROUP  MEM(KB)
others 59844872
b99    1317804
b90    971996
b93    1125264
b75    1213200
r93    547664
r03    561712
r01    978000
d97    511372
b95    585376
b94    649516
b91    435860
b89    513980
b02    1822112
$ ./jobs.sh --list -mc
GROUP  CPU(%) MEM(KB)
others 4.6    59844872
b99    1.3    1317804
b90    1.3    971996
b93    1.2    1125264
b75    0.7    1213200
r93    0      547664
r03    0      561712
r01    0      978000
d97    0      511372
b95    0      585376
b94    0      649516
b91    0      435860
b89    0      513980
b02    0      1822112
$ ./jobs.sh -h
jobs.sh [OPTION...]
-l, --list         listing all grades in descendant order of consuming resource
(according to CPU usage first, then memory usage)
-m, --mem          print the usage of memory (in KB)
-c, --cpu          print the usage of CPU (in %)
-h, --help         print this help message
$ ./jobs.sh blablabla
jobs.sh: Extra arguments -- 'blablabla'
Try 'jobs.sh -h' for more information.
$ ./jobs.sh -j
```

```
getopt: invalid option -- 'j'
jobs.sh [OPTION...]
-l, --list         listing all grades in descendant order of consuming resource
(according to CPU usage first, then memory usage)
-m, --mem          print the usage of memory (in KB)
-c, --cpu          print the usage of CPU (in %)
-h, --help         print this help message
```

- Subtask:
  1. Your script can deal with the long options (`--list, --mem, --cpu, --help`), and it can output correct result: 10%
  2. Your script can deal with both the long and short options(`-l, -m, -c, -h`, also their aggregation form!): 5%
  3. (Bonus) Your script can deal with wrong options and output some instruction respectively: 2%
- Note:
  1. You're allowed to create temporary files under `/tmp`, and you can use the command `mktemp` to get a usable path. But don't forget to clean these temporary files before the script exits.
  2. To get the list of running processes on workstation, you can use `ps aux`; the percent of CPU usage is the third column, and memory usage is the fourth column of output.
  3. *I leave a clue about a useful tool in sample output, find it out!*
- **UPDATE:**
  For the case that `-h` (`--help`) and other options exist simultaneously, you can either output help message or just see it as an error. The implementation won't affect your grade, and we'll grade it only by your understanding of shell scripting (for this case).