Assessment Tool

Name:	Gabinete, Keith Ginoel S.	**Lab Section:**	CD-7L

The exercise is divided into **four** parts. Make sure to answer and double-check all parts before submitting. The goal of this exercise is to explore the numerous shell commands apart from the commands and features discussed in the lab.

Part 1. Basic Linux Commands

Run the following commands and explain their usage (i.e. brief description of command, options used, operators, output). You may check their manual using the man command.

#	Commands	Description
1	`passwd`	- changes password for user accounts
2	`date`	- displays the current date and/or time or sets the system date
3	`hostname`	- shows or sets the system's host name
4	`Arch`	- prints machine architecture
5	`uname -a`	- prints ALL system information (e.g. kernel name, nodename, kernel release, kernel version, etc.)
6	`uptime`	- tells how long the system has been running
7	`whoami`	- prints the username associated with the current effective user ID
8	`who`	- prints information about users who are currently logged in
9	`last`	- shows a listing of last logged in users
10	`w`	- displays information about the users currently on the machine, their processes (tells what they are doing)
11	`top`	- displays Linux processes
12	`man`	- displays the user manual of any command we can run on shell/terminal
13	`wget https://file-examples-com.github.io/uploads/2017/02/file_example_CSV_5000.csv`	- tries to download the file from the given link < https://file-examples-com.github.io/uploads/2017/02/file_example_CSV_5000.csv>; returns an error as the file isn't found
14	`touch`	- updates the access and modification times of each file to the current time; or can simply be used to create a file (touch <filename>
15	`history`	- views/displays the previously executed commands
16	`clear`	- clears the terminal screen
17	`cal 2013`	- shows/displays the calendar of the year 2013
18	`cal 9 1752`	- shows the calendar only for the month of September of the specified year (1752)

19	`echo hello world`	- prints/displays “hello world”
20	`echo {con,pre}{sent,fer}{s,ed}`	- prints/displays the expansion of an expression into multiple text strings from each pattern defined {con,pre}, {sent,fer}, and {s,ed} - prints/displays the resulting expansions: consents, consented, confers, confered, presents, presented, prefers, and prefered
21	`echo 9+4`	- displays/prints 9+4

22	`bc -l`
	`9+4`
	`quit`
		bc -l: opens up a command line calculator (bc is an arbitrary precision calculator language) and defines the standard math library (with option command -l)

23	`echo 9+4 | bc -l`	- prints the sum of 9 and 4; (bc -l opens up a calculator and with the ‘|’ command, 9+4 is passed as an input to the opened calculator)
24	`yes please` *`(press Ctrl+C to exit)`*	- outputs/prints “please” repeatedly until killed (Ctrl+C)
25	`sleep 6`	- pauses (the terminal) for 6 seconds - (commands entered in the terminal will be executed once the sleep command execution is finished)
26	`time sleep 6`	- just like the command in the previous number, time sleep 6 pauses (the terminal) for 6 seconds but displays additional information about resources used by the sleep command
27	`ifconfig`	- is used to configure the kernel-resident network interfaces
28	`ps aux`	- ps displays information about a selection of the active processes - with aux, ps displays all processes in a user-oriented format options: a – list all processes with a terminal (tty) u – display user-oriented format x – list all processes owned by you a and x being together in a single command would list ALL processes (includes processes with a terminal (tty) and processes that you owned)
29	`man pwd > pwdman.txt`	- exports the user manual information for the command pwd into a file named pwdman.txt (would create it if it doesn’t exist and would overwrite it if it does exist)
30	`whereis gcc`	- locates the binary, source, and manual page files for the command ‘gcc ‘
31	`printenv`	- displays/prints all the values of environment variables (with format: name=value)
32	`less`	- reads an input text file and displays it - similar function with ‘more’ but has many more features

		- Unlike other reading commands, ‘less’ does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi
33	`more`	- is a filter for paging through text one screenful at a time - is slower than the ‘less’ command as ‘more’ loads the file first before displaying it in the terminal
34	`tail`	- outputs the last part of files - prints the last 10 lines of a file specified by it (tail <filename>) to standard output
35	`cat`	- concatenates files and print on the standard output
36	`tee`	- reads from standard input and writes to standard output and files
37	`stat pwdman.txt`	- displays the (file) status of the file named pwdman.txt
38	`wc pwdman.txt`	- prints newline, word, and byte counts for the file named pwdman.txt
39	`sort pwdman.txt`	- sort the lines of the text file pwdman.txt and displays them in the terminal (original contents of pwdman.txt will not be modified)

Part 2. File System

Provide the commands needed to do the following instructions. Make sure to answer the following in order.
Note: All instructions must be done in one command

#	Instruction	Command and Explanation
1	Create a folder in the Desktop named **ayi**.	a. change current working directory to Desktop first ex. (cd /mnt/c/Users/ASUS/Desktop) or just open the terminal/linus shell directly from the Desktop b. use mkdir command to make a folder in the current working directory mkdir ayi
2	Go to the newly created folder.	cd ayi - cd changes the current working directory to ayi
3	Move the given file (**exer.tar.gz**) to the current directory.	- use single dot (.) or single dot with forward slash (./) for destination directory to indicate that we want to move the file into our current working directory command: mv /mnt/c/Users/ASUS/Downloads/exer.tar.gz ./
4	Untar the file **exer.tar.gz**. *(Explain the options you used)*	tar -xf exer.tar.gz Explanation: - tar is an archiving utility - option x is used to indicate that we want to extract the files contained in exer.tar.gz - option f is used to indicate the path/directory where we want to extract/untar the file (in this case the current working directory) - option C can be added if we want to untar/unpack the file on a different directory than our current working directory Ex. mkdir extracted (create a new folder named extracted) tar -xf exer.tar.gz -C extracted (extracts/untar the file in the folder named extracted)
5	List all the files of the untarred directory.	ls exer - ls is used to list directory contents
6	Show the contents of the file **meow** in the terminal.	cat exer/meow - cat reads data from a file and gives its content as output
7	Copy the file **meow** to the current directory and name it **purr**.	cp exer/meow ./purr - cp <src> <dest>/<newFileName> - cp copies the file from source <src> to the destination directory <des> - can add text after <dest> to indicate that you want to rename the file with text after copying
8	Compile the C file named **loop.c**.	gcc exer/loop.c (will create a new file in the current

		directory named a.out) or gcc -o loop exer/loop.c (will create a new file in the current directory named loop)
9	Run the program (do not exit).	./a.out or ./loop (if the second method is used to compile the c file) - results to an infinite loop when executed
10	Suspend the running program.	Ctrl + C - sends a signal to the process, causing it to terminate immediately
11	Put the program in the background.	./a.out & (returns 1666 as PID in my linux shell) Or ./loop & (returns 1669 as PID in my linux shell) PID = process identification
12	Check the current jobs.	ps -e (option e is a standard syntax to see every process on the system) or ps aux
13	Terminate the program you ran in the background.	(kill PID) kill 1666 or kill 1669 - 'kill' command terminates/kills a running process
14	Create two directories named **pet** and **food** on the current directory.	mkdir pet; mkdir food - command ';' separates two commands and execute them sequentially
15	Copy the file named **meow** to the directory **pet**.	cp exer/meow ./pet - cp copies the meow file in exer folder (exer/meow) to the pet folder in the current working directory (./pet)
16	Determine the file type of the file named **loop.c**.	file exer/loop.c - command 'file' determines the file type of a file - returns loop.c: C source, ASCII text - C source indicates the file type (c)
17	Update the access timestamp of the file **me**.	touch exer/me - touch command updates the access and modification times of the file 'me' to the current time
18	Delete the directory named **food**.	rm -rf food - rm command: removes files or directories - option r: remove directories and their contents recursively - option f: (--force); ignore nonexistent files and arguments, never prompt

19	Go to the Desktop directory.	cd .. - since our working directory is ayi, which is a folder inside our Desktop, we can go back to the parent directory (which is the Desktop) of our current working directory by simply typing the command cd with double periods (cd ..) or just perform cd with the entire Desktop's path/location In my case: cd /mnt/c/USERS/ASUS/Desktop/
20	Delete the directory named **ayi**.	rm -rf ayi (same explanation as number 18)

Part 3. File Permissions and Ownership

Provide the command needed for the following situations.

1. Trying to create a directory named `$cmsc125` using the command `mkdir $cmsc125` would result in a missing operand error. What should be used to create the directory named $cmsc125?

   Add backslash '\' before $cmsc125 to preserve the literal value of the next character that follows (in this case: the dollar sign '$')
   New command: mkdir \$cmsc125

2. Use the `find` command to display all files in your home directory that is greater than 1 kilobyte.

   find ~ -type f -size +1k
   - command 'find' searches for files in a directory hierarchy
   - tilde '~' to indicate that we want to perform the find command in our home directory
   -> -type f to indicate that we only want to search for files and not for directories, etc.
   -> -size +1k to indicate that we only want to search for files that are larger (+) than 1 kilobyte (1k)

3. Change the file permission of the file (given to you) named **script.sh** and run it (it should print Hello World).

   ```
   -r-xr-xr-x 1 teioh teioh 32 Feb  9  2016 script.sh
   ```

   file script.sh is originaly readable for me, so I can just run the file directly and it will print 'Hello World'

   ```
   teioh@teioh:/mnt/c/Users/ASUS/Desktop/125/exer$ ./script.sh
   Hello World
   ```

   however, if I want script.sh to be writable for all users (owner, group, others), then I will do something like this:

   ```
   teioh@teioh:/mnt/c/Users/ASUS/Desktop/125/exer$ chmod ugo+w script.sh
   teioh@teioh:/mnt/c/Users/ASUS/Desktop/125/exer$ ls -l
   total 0
   -rwxrwxrwx 1 teioh teioh 74 Feb  9  2016 loop.c
   -rwxrwxrwx 1 teioh teioh 36 Feb  9  2016 me
   -rwxrwxrwx 1 teioh teioh 15 Feb  9  2016 meow
   -rwxrwxrwx 1 teioh teioh 32 Feb  9  2016 script.sh
   ```

   chmod command is used to change the file permissions of script.sh
   ugo indicates that we want to change permissions for users (u), groups (g) and others (o)
   +w means to add (+) writable (w) permission to ugo

4. Using `ls`, how will you print all the files in your current directory including all the files in its subdirectories?

   ls -R
   -option R lists subdirectories recursively

5. If you use `ls -l` in any directory, it would print something similar to the output below:

   ```
   -rw-rw-r--  1  user  user     0  Feb   9  14:52    file.txt
   ```

   a. What does each column represent?

      1st column: file/directory permission
      2nd column: number of links to the file
      3rd column: owner of the file/directory
      4th column: group name where the file/directory belongs (or simply owner group)
      5th column: file/directory size in bytes

6th column: created/modified date
7th column: file/directory name

b. Explain the meaning of `-rw-rw-r--`.

-rw-rw-r-- can be divided into: (-, rw-, rw-, r--)
Meanings:
- >>> file is just a regular file
rw- >>> file is readable and writable but not executable by owner
rw- >>> file is readable and writable but not executable by group
r-- >>> file is readable only by others (neither writable nor executable)

Part 4. Basic System Administration Tasks

Provide the commands needed to do the following instructions.

1. Some commands in Linux cannot be done unless you are using the terminal as the root user. From the current user account, how will you become the root user?

sudo su
- enter the password for the current user if prompted

```
teioh@teioh:~$ sudo su
[sudo] password for teioh:
root@teioh:/home/teioh#
```

2. Once you are the root user, add another user account named <yourinitials>.

adduser ksgabinete

- I didn't use useradd (as it does not prompt the user to enter a new password for the newly created account)
- I also added the newly created account ksgabinete to the sudoer list with command 'usermod -aG sudo ksgabinete' so I can switch back to being a root user. (usermod -> modifies a user account, -aG -> appends user to a group, sudo -> group name)

```
root@teioh:~# adduser ksgabinete
Adding user `ksgabinete' ...
Adding new group `ksgabinete' (1002) ...
Adding new user `ksgabinete' (1002) with group `ksgabinete' ...
The home directory `/home/ksgabinete' already exists.  Not copying from `/etc/skel'.
adduser: Warning: The home directory `/home/ksgabinete' does not belong to the user you are currently creating.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ksgabinete
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
root@teioh:~# usermod -aG sudo ksgabinete
```

3. Switch (or login) to the newly created account.

su ksgabinete

```
root@teioh:~# su ksgabinete
ksgabinete@teioh:/root$ whoami
ksgabinete
ksgabinete@teioh:/root$
```

4. Change the password of the newly created account.

passwd

```
ksgabinete@teioh:/root$ passwd
Changing password for ksgabinete.
Current password:
New password:
Retype new password:
passwd: password updated successfully
ksgabinete@teioh:/root$
```

5. Remove the newly created user account.

deluser ksgabinete

(must login as the root user first)

```
ksgabinete@teioh:/root$ sudo su
[sudo] password for ksgabinete:
root@teioh:~# deluser ksgabinete
Removing user `ksgabinete' ...
Warning: group `ksgabinete' has no more members.
userdel: user ksgabinete is currently used by process 2730
/usr/sbin/deluser: `/sbin/userdel ksgabinete' returned error code 8. Exiting.
root@teioh:~#
```