Exercise Sheet 2

Exercise 1 (Classifications of Operating Systems)

1. At any given moment, only a single program can be executed. Give the technical term for this operation mode.
2. Name one singletasking operating system.
3. Name multitasking operating system.
4. Name one single-user operating system.
5. Name one multi-user operating system.
6. Describe what half multi-user operating systems are.
7. Describe the difference between 8 bit, 16 bit, 32 bit, and 64 bit operating systems.
8. Name the essential criteria of real-time operating systems.
9. Name the two types of real-time operating systems.
10. Name four typical application areas of real-time operating systems and assign each application area to one of the categories of subtask 6.
11. Describe the structure of a thin kernel.
12. Describe the structure of a nano kernel.
13. Describe the structure of a monolithic kernel.
14. Describe the structure of a microkernel.
15. Describe the structure of a hybrid kernel.
16. GNU HURD implements a...

    ☐ monolithic kernel ☐ microkernel ☐ hybrid kernel
17. Linux implements a...

    ☐ monolithic kernel ☐ microkernel ☐ hybrid kernel
18. MacOS X implements a...

    ☐ monolithic kernel ☐ microkernel ☐ hybrid kernel
19. Windows NT4/Vista/XP/7/8/10 implements a...

☐ monolithic kernel ☐ microkernel ☐ hybrid kernel

20. Name one advantage and one drawback of monolithic kernels.
21. Name one advantage and one drawback of microkernels.
22. Name one advantage and one drawback of hybrid kernels.
23. Your colleague recommends you to relocate frequently used server daemons, such as web server, email server, SSH server and FTP server, from user mode to kernel mode. How do you feel about this idea? Give reasons for your answer. Explain a benefit and one drawback.
24. Describe what Single System Image means.

Exercise 2 (Basic Linux/UNIX commands)

Which command is used to...

1. check the man pages?
2. print out the present working directory in the shell?
3. create a new directory?
4. navigate to a directory?
5. print out the content of a directory in the shell?
6. create an empty file?
7. try to determine the content of a file?
8. concatenate the content of files with other files and can also be used to print out the content of a file?
9. print out lines from the end of a file in the shell?
10. print out lines from the beginning of a file in the shell?
11. copy files or directories to a different location?
12. move files or directories to a different location?
13. delete files or directories?
14. delete an empty directory?
15. place a string in the shell?

16. modify the permissions of the file or directory?
17. change the password of a user?
18. terminate a session (and thus shell) and allows to specify the return value of the shell script?
19. reboot the system?
20. shut the system down?
21. create a new user?
22. delete a user?
23. modify a user?
24. print out the group memberships of a user?
25. create a new group?
26. delete a group?
27. change a group?
28. change the user ($\Longrightarrow$ ownership) which is associated with a file or directory?
29. change the group which is associated with a file or directory?
30. create a link?
31. search a file for lines which contain a search pattern?
32. print out a list of running processes in the shell?
33. bring a process running in the background of the shell, into foreground?
34. bring a process into the background of the shell?
35. kill (terminate) a process?
36. kill (terminate) a group of processes?
37. specify the priority of a new process?
38. modify the priority of an existing process?
39. print out the process tree in the shell?

Exercise 3 (Permissions / Access Rights)

The source of this tutorial is:
`http://www.ws.afnog.org/afnog2012/unix-intro/presos/permissions-exercises.pdf`

Notes

- Commands preceded with `$` imply that you should execute the command as a general user and not as root.
- Commands preceded with `#` imply that you should be working as root with `sudo`

REFERENCE

If you look at files in a directory using `ls -al` you will see the permissions for each file and directory. Here is an example:

```
drwxrwxr-x    3 bnc   bnc      4096 Feb 25 09:49 directory
-rwxr--r--   12 bnc   bnc      4096 Feb 16 05:02 file
```

The left column is important. You can view it like this:

```
Type User Group Other Links Owner Group  Size  Date   Hour  Name
=====================================================================
d    rwx  rwx   r-x   3     bnc   bnc    4096  Feb 25 09:49 directory
-    rwx  r     r     12    bnc   bnc    4096  Feb 16 05:02 file
```

The directory has `r` (read), `w` (write), `x` (execute) access for the User (= Owner) and Group. For Other it has `r` (read) and `x` (execute) access.

The file has `r` (read), `w` (write), `x` (execute) access for User and `r` (read) only access for everyone else (Group and Other).

You can change permissions with the `chmod` command. `chmod` uses a base eight (octal) system to configure permissions. Or, you can use an alternate form to specify permissions by column (User/Group/Other) at a time.

Permissions have values like this:

```
Letter   Permission   Value
============================
r        read         4
w        write        2
x        execute      1
-        none         0
```

Thus you can give permissions to a file using the sum of the values for each permission you wish to give for each column. Here is an example:

```
Letter   Permission                  Value
=======================================
---      none                       0
--x      execute                    1
-w-      write only (rarely used)   2
-wx      write and execute (rare)   3
r--      read only                  4
r-x      read and execute           5
rw-      read and write             6
rwx      read, write, and execute   7
```

This is just one column. Since we have three areas of permissions (User, Group, Other), it looks like this, if you want to specify all 3 sets:

```
Permissions  Numeric     Description
             equivalent

-rw-------   600         User has read & write permission.
-rw-r--r--   644         User has read & write permission.
                         Group and Other have read permission.
-rw-rw-rw-   666         Everyone (User, Group, Other) has
                         read & write permission (dangerous?)
-rwx------   700         User has read, write, execute permission.
-rwxr-xr-x   755         User has read, write, execute permission.
                         Rest of the world (Other) has read & execute
                         permission (typical for web pages or 644).
-rwxrwxrwx   777         Everyone has full access (read, write,
                         execute).
-rwx--x--x   711         User has read, write, execute permission.
                         Group and world have execute permission.
drwx------   700         User only has access to this directory.
                         Directories require execute permission to
                         access.
drwxr-xr-x   755         User has full access to directory. Everyone
                         else can see the directory.
drwx--x--x   711         Everyone can list files in the directory,
                         but Group and Other need to know a filename
                         to do this.
```

1.) CHANGING FILE PERMISSIONS

If you are logged in as the root user on your machine please do the following to become a normal user.

```
# exit
```

Your prompt should change and now include a `$` sign.

```
$
```

Please check your username with the command `whoami`:

```
$ whoami
```

Please create a file and set permissions of the file in various ways.

```
$ cd
$ echo "test file" > working.txt
$ chmod 444 working.txt
```

What does that look like?

```
$ ls -lah working.txt
```

Because the file has no write permission for the owner, the owner can still change the file's permissions. This way, the owner can change the permissions of the file at any time to have write access again.

```
$ chmod 644 working.txt
```

Or, you can do this by using this form of `chmod`:

```
$ chmod u+w working.txt
```

Note: When you type these commands you should be able to use the tab key for command completion once you've typed the `w` in the file name `working.txt`. This will save you a lot of time. It's highly recommended!

To remove the read permission of a file for the user you would do

```
$ chmod u-r working.txt
```

Or, you can do something like this:

```
$ chmod 344 working.txt
```

You probably noticed that you can use the - (minus) sign to remove permissions from a file. Try reading your file:

```
$ cat working.txt
```

Now you cannot read your own file. Please make the file readable again by you with the `chmod` command. Look at your reference at the start of these tutorial to figure out what permissions are required.

2.) PROGRAM EXECUTION, PRIVILEGES AND SUDO

As a general user you can see that there is a file called `/etc/shadow`:

```
$ ls /etc/shadow
```

But, you cannot see its contents:

```
$ less /etc/shadow
```

Figure out the permissions of this file.

As a general user, however, you can see the content of the `/etc/shadow` file if you do the following:

```
$ sudo less /etc/shadow
```

What is `sudo`? Read about it:

```
$ man sudo
```