5. Multi-users, permissions and system monitoring

More than one user can be operating the computer at the same time. For example, if your computer is attached to a network or the Internet, remote users can log in via ssh (secure shell) and operate the computer. A good example is the high-performance computer in the Computer Engineering department (aiken.ce.pdn.ac.lk), where many users will be working on the machine at the same time. In order get the feeling of such environment, today you are supposed to work on aiken.ce.pdn.ac.lk for this chapter.

To connect to Aiken from a Linux computer see the steps at
https://faq.ce.pdn.ac.lk/index.php?action=artikel&cat=1&id=6&artlang=en-us

To connect to Aiken from a Windows computer see the steps at
https://faq.ce.pdn.ac.lk/index.php?action=artikel&cat=1&id=7&artlang=en-us

5.1 Who are my neighbours

5.1.1 who, what, last, etc.

Let's see what manual pages exist related to 'who':

```
$ man -k  who
[...]
w (1)          - Show who is logged on and what they are doing.
w.procps (1) - Show who is logged on and what they are doing.
who (1)        - show who is logged on
who@ (1)       - prints the list of active users on a remote host.
whoami (1)    - print effective userid
[...]
```

The commands `who` (who is logged in), `w` (what they are doing) and `whoami` are interesting for us:

```
$ who -H
NAME     LINE  TIME             COMMENT
user0  pts/0 2015-02-14 16:26 (1.2.3.4)
user1  pts/1 2015-02-14 16:47 (5.6.7.8)
user1  pts/2 2015-02-14 17:32 (5.6.7.8)

$ who am i
user1 pts/1 2015-02-14 16:47 (5.6.7.8)

$ w
 17:34:06 up  8:15,  3 users,  load average: 0.08, 0.09, 0.12
USER  TTY   FROM    LOGIN@  IDLE  JCPU   PCPU  WHAT
user0 pts/0 1.2.3.4  16:26 49:26  1:13   0.39s sshd: pi [priv]
user1 pts/1 5.6.7.8  16:47  6.00s 2.17s  0.39s sshd: pi [priv]
user1 pts/2 5.6.7.8  17:32  1:26  1.21s  1.21s -bash
```

```
$ whoami
user1
```

As you can see in the example above, who -H prints headers to the columns. `who am i` is a special version of who showing details of one’s own session. The output of w above shows that two users are logged in remotely, user1 has two sessions open.

The command `last` (last logged in) lists the users logged in last.

```
# last –5
user1  pts/1  1.2.3.4 Sat Feb 14 17:50   still logged in
user1  pts/0  1.2.3.4 Sat Feb 14 17:47   still logged in
reboot system 3.18.7+ Sat Feb 14 09:13 – 17:56  (00:10)
user0  pts/2  5.6.7.8 Sat Feb 14 17:32 – down   (00:13)
user5  pts/2  8.7.6.5 Sat Feb 14 17:03 – 17:24  (00:20)
```

The user ‘reboot’ is a special kind of a user. It shows that the local time is 17:56 and the machine has been running since 09:13 the same day.

5.1.2 finger

The command `finger` gives you some information about other users, in the system:

```
$ finger e00000
Login: e00000            Name: First Name Surname
Directory: /home/e00000  Shell: /bin/bash
Office: PR21, 34 56       Home Phone: 012 345 67 89
Last login Mon Dec 29 19:33 (IST) on pts/0 from 10.20.30.40
Mail last read Sun Feb 15 00:14 2015 (IST)
```

Section 5.1 was derived from the handout prepared by Dr Visvanath Ratnaweera for the Unix one Course in the year 2017.

5.2 File Permissions

On a Linux system, each file and directory is assigned access rights for

1. the **owner (u)** of the file,
2. the **members of a group (g)** of related users, and
3. **others (o)**.

Rights can be assigned to

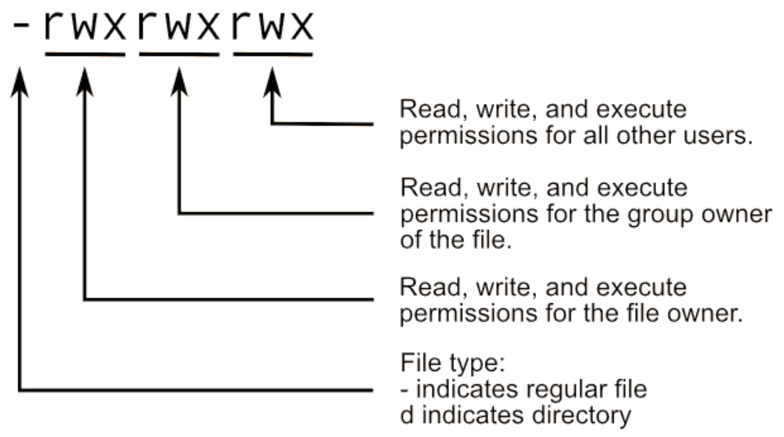
1. **read (r) a file**,
2. **write (w) a file**, and
3. **execute (x)** a file (i.e., run the file as a program).

To see the permission settings for a file, we can use the **`ls`** command. As an example, we will look at the **`bash`** program which is in the **`/bin`** directory:

```
hasindu@aiken:~$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1021112 May 16  2017 /bin/bash
```

- The file "/bin/bash" is owned by the superuser "root"
- The file is owned by the group "root"

In the diagram below, we see how the first column of the listing (`-rwxr-xr-x`) is interpreted. It consists of a character indicating the file type, followed by three sets of three characters that convey the reading, writing and execution permission for the owner, group, and everybody else.


- rwx rwx rwx
Read, write, and execute permissions for all other users.
Read, write, and execute permissions for the group owner of the file.
Read, write, and execute permissions for the file owner.
File type:
- indicates regular file
d indicates directory

Hence for **/bin/bash**,

- The superuser has the right to read, write, and execute this file
- Members of the group "root" can also read and execute this file
- Everybody else can read and execute this file

5.2.1 chmod

The `chmod` command is used to change the permissions of a file or directory. To use it, you specify the desired permission settings and the file or files that you wish to modify.

There are two methods to use `chmod`:

- using letters
- using numbers

The following examples will explain the usage of `chmod` using letters.

- First create some empty files:

```
hasindu@aiken:~/temp$ touch file1 file2 file3 file4
hasindu@aiken:~/temp$ ls -l
total 0
-rw-r--r-- 1 hasindu lecturer 0 Feb  3  2018 file1
-rw-r--r-- 1 hasindu lecturer 0 Feb  3  2018 file2
-rw-r--r-- 1 hasindu lecturer 0 Feb  3  2018 file3
-rw-r--r-- 1 hasindu lecturer 0 Feb  3  2018 file4
```

- Add owner execution permission:

```
hasindu@aiken:~/temp$ chmod u+x file1
hasindu@aiken:~/temp$ ls -l file1
-rwxr--r-- 1 hasindu lecturer 0 Feb  3 06:07 file1
```

- Add others write & execute permission:

```
hasindu@aiken:~/temp$ chmod o+wx file2
hasindu@aiken:~/temp$ ls -l file2
-rw-r--rwx 1 hasindu lecturer 0 Feb  3 06:07 file2
```

- Remove group read permission:

```
hasindu@aiken:~/temp$ chmod g-r file3
hasindu@aiken:~/temp$ ls -l file3
-rw----r-- 1 hasindu lecturer 0 Feb  3 06:07 file3
```

- Add read, write and execute to everyone:

```
hasindu@aiken:~/temp$ chmod ugo+rwx file4
hasindu@aiken:~/temp$ ls -l file4
-rwxrwxrwx 1 hasindu lecturer 0 Feb  3 06:07 file4
```

In this tutorial, we will not discuss how `chmod` can be used with numbers. If you are interested see the following links.

- http://linuxcommand.org/lc3_lts0090.php
- https://help.ubuntu.com/community/FilePermissions

5.3 Directory Permissions

The meaning of the r, w, and x attributes is different for directories:

- **r** - Allows the contents of the directory to be listed if the x attribute is also set.

- **w** - Allows files within the directory to be created, deleted, or renamed if the x attribute is also set.
- **x** - Allows a directory to be entered (i.e. `cd dir`).

You can use `chmod` to set permissions to directories the same way you did for files.

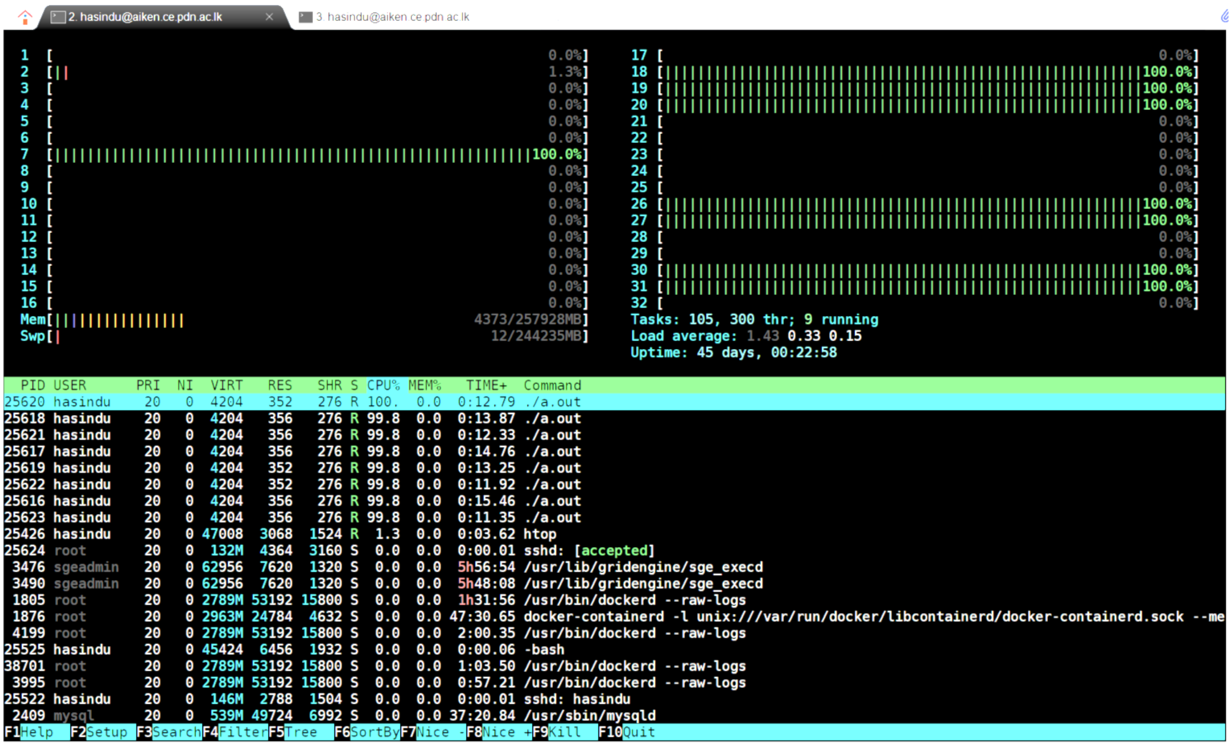
Section 5.2 and 5.3 of this tutorial was derived from http://linuxcommand.org/lc3_lts0090.php and https://help.ubuntu.com/community/FilePermissions

5.4 System monitoring

In this section, we will look at two useful commands namely `htop` and `time`.

5.4.1 htop

In Windows, you should be familiar with the task manager which you can use to monitor the system. When you are working on the Linux command line, you can use the command `top` or `htop` to view the currently running processes and their statistics. *htop* is colourful and more interactive than *top*. Following is a screenshot of `htop`.


2. hasindu@aiken.ce.pdn.ac.lk   3. hasindu@aiken.ce.pdn.ac.lk

```
  1  [                                                     0.0%]     17 [                                                     0.0%]
  2  [||                                                   1.3%]     18 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  3  [                                                     0.0%]     19 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  4  [                                                     0.0%]     20 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  5  [                                                     0.0%]     21 [                                                     0.0%]
  6  [                                                     0.0%]     22 [                                                     0.0%]
  7  [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]     23 [                                                     0.0%]
  8  [                                                     0.0%]     24 [                                                     0.0%]
  9  [                                                     0.0%]     25 [                                                     0.0%]
  10 [                                                     0.0%]     26 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  11 [                                                     0.0%]     27 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  12 [                                                     0.0%]     28 [                                                     0.0%]
  13 [                                                     0.0%]     29 [                                                     0.0%]
  14 [                                                     0.0%]     30 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  15 [                                                     0.0%]     31 [||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  16 [                                                     0.0%]     32 [                                                     0.0%]
  Mem[|||||||||||||||                             4373/257928MB]     Tasks: 105, 300 thr; 9 running
  Swp[|                                             12/244235MB]     Load average: 1.43 0.33 0.15
                                                                     Uptime: 45 days, 00:22:58

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
25620 hasindu    20   0  4204   352   276 R 100.  0.0  0:12.79 ./a.out
25618 hasindu    20   0  4204   356   276 R 99.8  0.0  0:13.87 ./a.out
25621 hasindu    20   0  4204   356   276 R 99.8  0.0  0:12.33 ./a.out
25617 hasindu    20   0  4204   356   276 R 99.8  0.0  0:14.76 ./a.out
25619 hasindu    20   0  4204   352   276 R 99.8  0.0  0:13.25 ./a.out
25622 hasindu    20   0  4204   352   276 R 99.8  0.0  0:11.92 ./a.out
25616 hasindu    20   0  4204   356   276 R 99.8  0.0  0:15.46 ./a.out
25623 hasindu    20   0  4204   356   276 R 99.8  0.0  0:11.35 ./a.out
25426 hasindu    20   0 47008  3068  1524 R  1.3  0.0  0:03.62 htop
25624 root       20   0  132M  4364  3160 S  0.0  0.0  0:00.01 sshd: [accepted]
 3476 sgeadmin   20   0 62956  7620  1320 S  0.0  0.0  5h56:54 /usr/lib/gridengine/sge_execd
 3490 sgeadmin   20   0 62956  7620  1320 S  0.0  0.0  5h48:08 /usr/lib/gridengine/sge_execd
 1805 root       20   0 2789M 53192 15800 S  0.0  0.0  1h31:56 /usr/bin/dockerd --raw-logs
 1876 root       20   0 2963M 24784  4632 S  0.0  0.0 47:30.65 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --me
 4199 root       20   0 2789M 53192 15800 S  0.0  0.0  2:00.35 /usr/bin/dockerd --raw-logs
25525 hasindu    20   0 45424  6456  1932 S  0.0  0.0  0:00.06 -bash
38701 root       20   0 2789M 53192 15800 S  0.0  0.0  1:03.50 /usr/bin/dockerd --raw-logs
 3995 root       20   0 2789M 53192 15800 S  0.0  0.0  0:57.21 /usr/bin/dockerd --raw-logs
25522 hasindu    20   0  146M  2788  1504 S  0.0  0.0  0:00.01 sshd: hasindu
 2409 mysql      20   0  539M 49724  6992 S  0.0  0.0 37:20.84 /usr/sbin/mysqld
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

System-wide CPU usage:

The numbers from 1 to 32 on the above figure represents the number of CPUs (more accurately, the number of CPU threads) in the system. The progress bar next to them represents the load of each CPU.

System-wide memory usage:

Below the CPU progress bars, you will see the memory progress bars (shown as **mem**).

Information on processes:

Htop will list all the running processes on a system and information about each process.

For detailed information, you can visit http://www.deonsworld.co.za/2012/12/20/understanding-and-using-htop-monitor-system-resources/

5.4.2 time

Time command can be used to find how much time is taken for a command to complete. This tool can be very useful when you want to measure the runtime for a program that you write in your programming course. Let's take an example.

The following command can be used to get the disk usage of your home directory.

```
hasindu@aiken:~$  du -d1 -h ~
……
1.3G    /home2/hasindu/soft
4.0K    /home2/hasindu/temp
8.0K    /home2/hasindu/.distlib
8.0K    /home2/hasindu/.gnome2
436K    /home2/hasindu/.java
78G     /home2/hasindu
```

Now let's say you want to measure the time taken by this command to run. You should run the command as follows.

```
hasindu@aiken:~ $ time du -d1 -h ~
……
real    1m17.206s
user    0m0.324s
sys     0m4.722s
```

Under “real” it says that it took 1 minute and 17.206 seconds to complete the command. If you are enthusiastic, check what are the values under “user” and “sys”.

5.4.3. Few other useful commands for system monitoring

I will leave this as an exercise for you to figure out what the following commands do.

- `df -h`
- `ifconfig`
- `netstat`
- `iostat`
- `cat /proc/cpuinfo`

5.5 Assignment 5

This assignment should be done on *aiken.ce.pdn.ac.lk* logged in through SSH. You should record what you do using *ttyrec* and submit the *ttyrec* recording. If you have any doubts you should ask them in the discussion forum.

1. Print the system date.
2. Print ‘who am i’.
3. Record the ‘who’ and ‘what’ of the others who are logged in.
4. Get the last 10 logins into the system.
5. Create a folder named *linux_assignment5* in your home directory.
6. Inside *linux_assignment5* create a file named *test.txt*
7. Check what the default permissions are for *test.txt.*
8. Remove the *read* permission for the owner of *test.txt.* Try reading the contents of the file.
9. Add the read permission back to the owner of *test.txt*. Now try reading the contents of the file.
10. Remove the *write* permission for the owner of *test.txt*. Try writing to *test.txt*.
11. Add the *write* permission back for the owner of *test.txt*. Now try writing to *test.txt*.
12. In the same *linux_assignment5* directory write a “hello world” program in C in a file called *hello.c*. Compile the program using *gcc* and create a binary file called *hello.out*.
13. Check what the default permissions are for *hello.out*.
14. Remove the executable permission for the owner of *hello.out*. Try running the *hello.out* program.
15. Add the executable permission back to the owner of *hello.out*. Now try running *hello.out.*

16. Check the default permissions for the *linux_assignment5* directory you created in step 6.
17. Remove all the *read, write and executable* permissions for the owner of *linux_assignment5.*
18. Try entering into the directory *linux_assignment5.*
19. Now add the *executable* permission to *linux_assignment5* for the owner and try entering the directory.
20. Now try listing the contents of the folder *linux_assignment5.*
21. Add the *read* permission to *linux_assignment5* for the owner and try listing the contents of the folder.
22. Try creating a new file inside the directory *linux_assignment5.*
23. Add the *write* permission to *linux_assignment5* for the owner. Now try creating a new file.
24. Set your *linux_assignment5* folder such that only you have access. (Remove all group and other permissions)
25. Check if you can access few of your colleagues' *linux_assignment5* folder.