Unix System Administration Logbook

Matthew Crabtree, B00414581, November 2021

Lab 01 - Introduction & Basic Commands

1. *Name and explain one qualifier selected by you for each of the three commands*
    1. `ls -h` prints human readable file sizes (e.g. 1K, 2G, etc.) when used in conjunction with `-s` and `l`
    2. `locate -c` prints the number of matching files instead of the absolute path (unless used in conjunction with `--print`)
    3. `cd -` an argument of a single hyphen is converted to `$OLDPWD` returning the user to the last directory they were in. If the directory change was successful, the absolute pathname of the new working directory is written to stdout.
    4. `pwd -P` prints the symbolically linked working directory as opposed to its symbolic location. See below.

```
# An example of pwd -P in an active session
mc@hearth ~/.config $ ls -lah | grep alacritty
lrwxrwxrwx  1 mc mc   36 Aug 29 16:50 alacritty ->
../dotfiles/config/.config/alacritty
mc@hearth ~/.config $ cd alacritty
mc@hearth ~/.config/alacritty (master) $ pwd
/home/mc/.config/alacritty
mc@hearth ~/.config/alacritty (master) $ pwd -P
/home/mc/dotfiles/config/.config/alacritty
```

2. *Who are you (what is your user name) in this setup?*

```
[student@UWS ~]$ whoami     # prints the username associated with the current
user's ID
student
```

3. *What type of information does `df` display?*

    `df` reports file system disk space usage; It displays the amount of free space available on a file system matching the name argument.

4. *What are the names of the different filesystems that are displayed? What is the mount-point for the `filesystem` source beginning `/dev/root`?*

```
[student@UWS ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/root              1048576    137428    911148  13% /
devtmpfs                255020       228    254792   0% /dev
tmpfs                   255180         0    255180   0% /dev/shm
tmpfs                   255180        24    255156   0% /tmp
tmpfs                   255180        48    255132   0% /run
/dev/sda1                 3963        31      3728   1% /media/disk1
```

- The names of the filesystems:

```
[student@UWS ~]$ df | awk '{print $1}'
Filesystem
/dev/root
devtmpfs
tmpfs
tmpfs
tmpfs
/dev/sda1
```

- `/dev/root` is mounted on the root directory `/`.

5. *What do the `Used` and `Available` columns stand for?*

`Used` is the number of used blocks on a system, and `Available` shows the number of available blocks. Blocks can be set to units of 1024 bytes, an overridden value using `-BG`, or 512 bytes if `POSIXLY_CORRECT` is set.

6. *What is the directory name displayed by the `pwd` command?*

`/home/student/subdir`

7. *Use the man command man ls to find out what information is given by ls -l? Try to figure out which column shows file size. What is the size of the newly created file, file2?*

- `ls -l` uses a long list format giving further info about each item
- `file1` is zero bytes in size
- `file2` is 20 bytes in size

8. *Use the `pwd` command to find out what happened and in which directory you currently reside? What is the meaning of the double dot `..` if used in conjunction with the `cd` command?*

```
[student@UWS subdir]$ cd ..     # change pwd to the parent directory
[student@UWS ~]$ pwd
/home/student                   # pwd is the parent dir
```

9. *Use the `pwd` command to find out what happened and your current directory*

```
[student@UWS ~]$ cd ./subdir    # from this dir, head to /subdir
[student@UWS subdir]$ pwd
/home/student/subdir            # pwd is now the subdir
```

The `.` in `cd ./subdir` represents the current directory. From this directory we moved into the sub-directory `subdir`.

10. *In which directory did the command `cd` just move you?*

```
[student@UWS subdir]$ pwd
/home/student/subdir            # currently in subdir
[student@UWS subdir]$ cd        # cd command with zero arguments ...
[student@UWS ~]$ pwd
/home/student                   # ... changes directory to the current user's
home dir
```

11. *Are there any hidden files in the root directory called `/`?*

```
[student@UWS ~]$ ls
Desktop  subdir
[student@UWS ~]$ ls -lah
total 24
drwxr-sr-x    4 student  student      138 Sep 14 11:25 .                # hidden
drwxr-xr-x    3 root     root          61 Sep 14 11:25 ..               # hidden
-rw-------    1 student  student      105 Sep 14 11:56 .bash_history    # hidden
-r--r-----    1 student  student     1004 Sep 14 11:25 .bashrc          # hidden
drwxr-xr-x    2 student  student       37 Sep 14 11:25 Desktop
drwxr-sr-x    2 student  student       81 Sep 14 11:55 subdir
```

Any directory name beginning with `.` represents a hidden directory. The listings for `.` and `..` represent the current and parent directories respectively, and are ommitted by default when running `ls` with no arguments.

12. *How many filesystems are there in total listed in the root directory called `/`?*

The wording of this question is a little ambiguous.

```
[student@UWS ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/root              1048576    137428    911148  13% /
devtmpfs                255020       228    254792   0% /dev
tmpfs                   255180         0    255180   0% /dev/shm
tmpfs                   255180        24    255156   0% /tmp
tmpfs                   255180        48    255132   0% /run
/dev/sda1                 3963        31      3728   1% /media/disk1
[student@UWS ~]$ df /
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/root              1048576    137428    911148  13% /
```

1. If the question is how many filesystems are mounted on the root directory, i.e. `/` then the answer is one: `/dev/root`
2. If, on the other hand, the question is how many filesystems are *contained within* `/` then the answer is four including `/dev/root`, given that `/dev`, `/tmp`, and `/run` are mount points for filesystems contained within immediate subdirectories within `/`

Lab 02 - Managing Files & Permissions

1. *Identify and note the line in `/etc/mtab` that contains the mount-information associated with the disk device.*

```
[root@UWS ~]# cat /etc/mtab | grep floppy
/dev/loop0 /media/floppy ext2 rw,relatime,errors=continue 0 0
```

2. *Note down the outputs of the commands `ls -l /dev/loop0` and `ls -l /media/floppy`. Compare the outputs and comment on what is actually achieved by the mount command that you have just performed.*

```
[root@UWS ~]# ls -l /dev/loop0
brw-------    1 root     root        7,   0 Sep 20 18:53 /dev/loop0
[root@UWS ~]# ls -l /media/floppy/
total 13
-rw-r--r--    1 root     root            61 Sep 13  2020 file.txt
drwx------    2 root     root         12288 Sep 13  2020 lost+found
```

The floppy disk image has been attached to `/dev/loop0` which, in turn, is mounted onto `/media/floppy`. The use of a loop device allows for loop mounting of images onto the overall filesystem as though they were physical devices.

The difference between the outputs stems from the fact that the disk image is mounted *via* the loop device onto `/media/floppy`, the logical point at which the contents of the image are mounted onto the overall filesystem.

3. *After mounting the virtual floppy, has `/etc/fstab` changed as well, or are all entries still the same?*

No changes. `fstab` defines filesystems to be mounted at boot, and *how* they're integrated as part of the overall filesystem.

4. *Why is it important not to have target media mounted while you try to create a file system?*

If mounted, the kernel may try to perform read or write operations on the target media. Unmounting the media ensures there is no risk of the operating system accessing the device and interfering with the creation of a new filesystem.

5. *How many `inodes` and `blocks` are created on the system?*

Assuming "the system" refers to the system as a whole:

```
[root@UWS ~]# df / -P
Filesystem           1024-blocks    Used Available Capacity Mounted on
/dev/root              1048576    138868    909708  13% /
[root@UWS ~]# find / -type f -print | wc -l
16203
```

- 1,048,576 blocks (~1.0G using `df / -h`)
- The number of inodes is contingent on the size of the filesystem, with a default configuration of one inode per 2048 bytes. Assuming this to be the case there are 524,288 inodes in `/dev/root`.

6. *How many blocks are reserved for the super user (root)?*

5% of the total capacity is reserved by the root user to prevent the system from failing if or when it runs out of available space. In this instance, 5% of 1,048,576 blocks is ~52,429 blocks.

7. *Manually run the program `fsck` on the virtual floppy. What is displayed by `fsck`?*

Given that the filesystem is mounted at present:

```
[root@UWS ~]# fsck /dev/loop0
fsck from util-linux 2.32.1
e2fsck 1.44.2 (14-May-2018)
/dev/loop0 is mounted.
e2fsck: Cannot continue, aborting.
```

Unmounting the filesystem results in the following output:

```
[root@UWS ~]# umount /dev/loop0
[root@UWS ~]# fsck /dev/loop0
fsck from util-linux 2.32.1
e2fsck 1.44.2 (14-May-2018)
/dev/loop0: clean, 12/184 files, 48/1440 blocks
```

8. *Note any files and directories that are on the floppy at this time.*

```
[root@UWS floppy]# ls -la /media/floppy
total 19
drwxr-xr-x    3 root     root          1024 Sep 21 10:37 .
drwxr-xr-x    3 root     root            60 Sep  9  2020 ..
-rw-r--r--    1 root     root            61 Sep 13  2020 file.txt
drwx------    2 root     root         12288 Sep 13  2020 lost+found
-rw-r--r--    1 root     root            11 Sep 21 10:37 tst.dat
```

9. *Can you think of any apocalyptic scenario that might produce an entry in the `lost+found` directory?*

An improper shutdown might result in orphaned inodes, or if a block device experiences some kind of hardware failure.

10. *Why would the kernel not allow you to dismount a filesystem that you are currently using?*

Pending IO operations should complete before a filesystem is unmounted. Linux, for example, buffers writing to disk in RAM until such a time as the kernel deems appropriate.

11. *Use the command cd /root, before issuing the command line umount /media/floppy again. Does the un-mounting operation work now?*

Yes!

```
[root@UWS ~]# cd /root
[root@UWS ~]# pwd
/root
[root@UWS ~]# umount /media/floppy
[root@UWS ~]# mount
/dev/root on / type 9p (rw,sync,dirsync,relatime,trans=virtio)
devtmpfs on /dev type devtmpfs (rw,relatime,size=255020k,nr_inodes=63755,mode=75
5)
```

```
proc on /proc type proc (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=666)
tmpfs on /dev/shm type tmpfs (rw,relatime,mode=777)
tmpfs on /tmp type tmpfs (rw,relatime)
tmpfs on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
sysfs on /sys type sysfs (rw,relatime)
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)
```

12. *Why can you still set on the floppy, even though you have just ejected the floppy disk?*

The mount point we used to access the image in the overall filesystem is a directory we created. Devices, virtual or otherwise, are usually mounted onto empty directories by convention.

13. *Explain why you no longer see the previous files* `lost+found` *and* `tst.dat` *on the mount-point?*

Because we've unmounted the image and its contained filesystem from the overall filesystem.

14. *Explain the difference between the two command lines given. What does the re-direction operator* `>` *do?*

```
[root@UWS ~]# echo '1234567890' test.dat
1234567890 test.dat
[root@UWS ~]# echo '1234567890' > test.dat
```

- The first command is doing little more than supplying standard input arguments to the `echo` programme, and `echo` is doing what it does: re-direct standard input to standard output.
- The second command uses the `>` redirection character which redirects the standard input on the left to a designated file on the right. The file is created if it doesn't already exists, and if it does, the content of the file is overwritten.

15. *(After performing the* `mv` *command) What happened to the original* `test.dat` *file?*

```
[root@UWS ~]# mv test.dat test_01.dat
[root@UWS ~]# ls -l
total 1448
drwxr-xr-x    2 root     root            37 Sep 21 13:01 Desktop
-rw-r--r--    1 root     root            11 Sep 21 13:13 test_01.dat
-rw-------    1 student  root       1474560 Sep 21 13:07 virtual_floppy.img
```

When using `mv` without specifying an alternate directory `mv` acts as though the file has been renamed. The inode of the file has not changed, instead the name of the directory entry associated with that inode has been modified.

16. *Examine the file* `test_01.dat` *with the* `cat` *command. What does the operator* `>>` *do?*

```
[root@UWS ~]# echo 'abcdefghi' >> test_01.dat
[root@UWS ~]# cat test_01.dat
1234567890
abcdefghi
```

The `>>` operator appends standard output of `echo` to the designated file.

17. *Based on the commands shown above: Explain the effect of the* `-i` *qualifier in the* `cp` *command.*

```
[root@UWS ~]# cp -i test_01.dat test_b.dat
[root@UWS ~]# cp -i test_01.dat test_b.dat
cp: overwrite 'test_b.dat'?
```

`cp`'s `-i` flag prompts the user if they wish to go ahead with the change if an overwrite will occur. The `-i` stands for 'interactive'.

18. *(After performing `cmp test_a.dat test_b/dat`) What is the output of the above command line, or is there no output at all?*

There is no output, indicating that there are no differences between the two files specified.

19. *What is the output of the above `cmp` command now and why?*

```
[root@UWS ~]# echo 'ei caramba' >> test_a.dat
[root@UWS ~]# cmp test_a.dat test_b.dat
cmp: EOF on test_b.dat
```

`cmp` reached the end of `test_b.dat` before reaching the end of file on `test_a.dat`.

```
student@UWS ~]$ od -c test_b.dat
0000000   1   2   3   4   5   6   7   8   9   0  \n   a   b   c   d   e
0000020   f   g   h   i  \n
0000025
[student@UWS ~]$ od -c test_a.dat
0000000   1   2   3   4   5   6   7   8   9   0  \n   a   b   c   d   e
0000020   f   g   h   i  \n   e   i       c   a   r   a   m   b   a  \n
0000040
```

20. *What is the output of the above command? Do you know another Unix command that does roughly the same?*

```
[student@UWS ~]$ cat test_a.dat
1234567890
abcdefghi
ei caramba
```

In cases like these with only a single argument `cat` will send the contents of a given file to standard out.put `less` will page through a file's content, `more` will print a file's content to standard output with a single argument, and `echo` can also be used in novel ways to print to standard output.

```
[student@UWS ~]$ less test_a.dat ; more test_a.dat ; echo "$(less test_a.dat)"
1234567890
abcdefghi
ei caramba
1234567890
abcdefghi
ei caramba
1234567890
abcdefghi
ei caramba
```

21. *What will cat do with the 3 files? Can you achieve the same result with the more command?*

```
[student@UWS ~]$ cat test_a.dat test_b.dat test_c.dat > test_abc.dat
[student@UWS ~]$ cat test_a.dat test_b.dat test_c.dat
1234567890
abcdefghi
ei caramba
1234567890
abcdefghi
Doh!
```

`cat` will concatenate the content of any given files and print their combined content to standard out. When the `>` operator is used along with a designated file name, the standard output of cat will be redirected to become the content of the file.

`more` will print to standard output the content of any files it's given, and on certain systems, will page through the results, and will allow the user to search using `/` while paging.

```
[student@UWS ~]$ more test_a.dat test_b.dat test_c.dat > more_test
[student@UWS ~]$ cat more_test
1234567890
abcdefghi
ei caramba
1234567890
abcdefghi
Doh!
```

22. *What does the `./` represent in the above command line and is it redundant here?*

`./` represents the current directory. The same result could have been achieved with the following command:

```
[student@UWS ~]$ cp test_abc.dat temp10/
```

23. *Try `cp test_abc.dat ./temp10/*` . What will be the result and does it differ from the previous `cp` command-line `cp test_abc.dat ./temp10/.` ?*

```
[student@UWS ~]$ cp test_abc.dat ./temp10/. && cat temp10/test_abc.dat
1234567890
abcdefghi
ei caramba
1234567890
abcdefghi
Doh!
```

We can see here that the file `test_abc.dat` has been copied to `temp10/` and the file name has been preserved.

```
[student@UWS ~]$ cp test_a.dat ./temp10/* && ls -lah temp10/
total 12
drwxr-sr-x    2 student  student       66 Sep 21 19:23 .
drwxr-sr-x    4 student  student      246 Sep 21 18:46 ..
-rw-r--r--    1 student  student       32 Sep 21 19:53 test_abc.dat
[student@UWS ~]$ cat temp10/test_abc.dat
1234567890
abcdefghi
ei caramba
```

This second command, particularly `cat`, demonstrates that `test_a.dat` has been copied to `temp10/` though the original content has been overwritten by that of `test_a.dat` while the original name has been preserved.

```
[student@UWS ~]$ cp test_a.dat ./temp10/. && ls -lah temp10/
total 16
drwxr-sr-x    2 student  student       93 Sep 21 19:23 .
drwxr-sr-x    4 student  student      246 Sep 21 18:46 ..
-rw-r--r--    1 student  student       32 Sep 21 19:57 test_a.dat
-rw-r--r--    1 student  student       32 Sep 21 19:53 test_abc.dat
```

Performing this third command shows that using the `/.` notation in `cp`'s destination argument preserves the source filename in the destination directory as opposed to the behaviour of the wildcard: overwriting the content of the first file it encounters, or creating a new file if none exists. Using `cp -i` will prevent any files from being overwritten.

24. *Has the `-i` qualifier for `rm` the same meaning as for the `cp` command?*

`rm -i` will prompt before each removal. `cp -i` will also prompt before overwriting a file.

25. *What does the asterisk sign " represent in the above expression?**

`*` represents a wildcard, which can represent anything in a given expression. In the following example we can use the wildcard as part of an argument for the `find` command to locate each of our `.dat` files:

```
[student@UWS ~]$ ls
Desktop       temp10        test_01.dat   test_a.dat    test_abc.dat  test_b.dat
   test_c.dat
[student@UWS ~]$ find . -name '*.dat'
./test_01.dat
./test_a.dat
./test_b.dat
./test_c.dat
./test_abc.dat
```

26. *Write down the permissions (in words) of `/etc/fstab` and `/etc/shadow` in your logbook, and explain why the latter is so highly restricted.*

```
[student@UWS ~]$ ls -la /etc/fstab
-rw-r--r--    1 root     root           334 Sep  9  2020 /etc/fstab
```

user	group	world
read, write	read	read

```
[student@UWS ~]$ ls -la /etc/shadow
-rw-------    1 root     root           360 Sep 22 09:09 /etc/shadow
```

user	group	world
read, write	n/a	n/a

`shadow` contains encrypted passwords for all users who are defined in the `passwd` file and is readable by none other than `root`, which is why I had to perform `su -` in the following code block before I could `cat` the content of `shadow`.

```
[student@UWS ~]$ cat /etc/shadow
cat: cant open '/etc/shadow': Permission denied
[student@UWS ~]$ su -
[root@UWS ~]# cat /etc/shadow
root:$5$nAjmGoNs$dTVSaUVPM3v/AzxeE7tt ... 99999:7::::::
-- snip --
student:$1$9JBdC6to$lVduSFgip ... 99999:7:::
```

27. *To which category, do you think the root-user belongs (user, group, world, or none of these)?*

Taking a look at the persmissions for a file such as `/etc/shadow`, and given that `root` was able to read its contents in the above example, we can surmise that `root` is a user and their persmissions are defined in the first three permissions columns (in this instance, read and write).

```
[student@UWS ~]$ ls -la /etc/shadow
-rw-------    1 root     root           360 Sep 22 09:09 /etc/shadow
```

28. *Check the entries in /etc/group to find out which users in your system belong to the group 'audio'.*

```
[student@UWS ~]$ grep audio /etc/group
audio:x:29:
```

`audio` has no members. We can surmise this by inspecting the properties of an alternate group:

```
[student@UWS ~]$ grep wheel /etc/group
wheel:x:10:root,student
```

29. *Use the ls –l command to find out the default user, group and world permissions for this newly created file.*

```
[root@UWS ~]# echo 'Donuts' > test_a.dat
[root@UWS ~]# ls -l test_a.dat
-rw-r--r--    1 root     root             7 Sep 22 10:18 test_a.dat
```

user	group	world
read, write	read	read

30. *How did the permission, user- and group-ownership change? Describe in your own words what user, group and world are allowed to do with the file. What is achieved by the commands `chgrp` and `chown`?*

```
[root@UWS ~]# ls -l test_a.dat
-rw-r--r--    1 root     root             7 Sep 22 10:18 test_a.dat
[root@UWS ~]# chmod 777 test_a.dat
[root@UWS ~]# chown student test_a.dat
[root@UWS ~]# chgrp www-data test_a.dat
[root@UWS ~]# ls -l test_a.dat
-rwxrwxrwx    1 student  www-data         7 Sep 22 10:18 test_a.dat
```

- `chmod 777` applied read, write, and execute privileges to the user, the the group, and world.
- The file's user has full read, write, and execute permissions. The same is true for any member of the file's designated group, as well as all other users.
- `chgrp` modified the file's designated group ownership, from `root` to `www-data`.
- `chown` modified the file's designated owner from `root` to `student`.

31. *Is a permission of `7` (in octal notation) a good idea to enable for world-access?*

No. Any user with filesystem access can execute a file with `world` permissions set to 7.

32. *Change the permissions in a way that the user can read and write, but the group and world can only read the file. Note the necessary command in your lab-book.*

```
[root@UWS ~]# chmod 644 test_a.dat && ls -l test_a.dat
-rw-r--r--    1 student  www-data         7 Sep 22 10:18 test_a.dat
```

33. *What is the most permissive access that you can grant in octal representation?*

`777`

34. *What is the default permission given to the new directory /root/temp11?*

```
[root@UWS ~]# ls -la | grep temp11
drwxr-xr-x    2 root     root            37 Sep 22 10:29 temp11
```

user	group	world
read, write, execute	read, execute	read, execute

35.
    - Which directory permission will allow you to create a files inside of it ? (eg: executing `touch` and `echo` commands)*

We can find out by cycling through the permissions, making a directory, assigning ownership to `root`, and setting full, read, read permissions:

```
[student@UWS Desktop]$ mkdir testdir
[student@UWS Desktop]$ sudo chown root testdir/
[student@UWS Desktop]$ sudo chmod 744 testdir/
```

```
[student@UWS Desktop]$ ls -l
total 4
drwxr--r--    2 root     student         37 Sep 22 10:50 testdir
[student@UWS Desktop]$ touch testdir/file1
touch: testdir/file1: Permission denied
[student@UWS Desktop]$ tree
.
-- testdir
1 directory, 0 files
[student@UWS Desktop]$ sudo touch testdir/file1
[student@UWS Desktop]$ sudo tree
.
-- testdir
    -- file1
1 directory, 1 file
[student@UWS Desktop]$ tree
.
-- testdir
1 directory, 0 files
```

At this point we can see that with `root` as the owner of the directory, `student` cannot view the contents of `testdir`, nor can it create files within the directory.

Adding `group` write permissions to `testdir` maintains `student`'s inability to view or create files in the directory:

```
[student@UWS Desktop]$ sudo chmod 764 testdir/
[student@UWS Desktop]$ ls -l
total 4
drwxrw-r--    2 root     student         59 Sep 22 10:50 testdir
[student@UWS Desktop]$ tree
.
-- testdir
1 directory, 0 files
[student@UWS Desktop]$ touch testdir/studentfile
touch: testdir/studentfile: Permission denied
```

Finally, adding execute permissions to the `group`, `student` is able to finally view and create files in the directory:

```
[student@UWS Desktop]$ sudo chmod 774 testdir/
[student@UWS Desktop]$ touch testdir/studentfile
[student@UWS Desktop]$ tree
.
-- testdir
    |-- file1
    |-- studentfile
1 directory, 2 files
```

36. *Which directory permission is necessary to run a binary executable located inside of it?*

Execute.

37. *Which directory permission is necessary to run a shell-script located inside of it ? (a shell script is an ASCII code!)*

It depends. `source myscript.sh` requires read permissions. In the instance of executing using `./myscript.sh` the kernel will abort execution after parsing a shebang `#!` if the user does not have execute permissions on that script.

38. *Interpret the machine response to this command based on your knowledge about the permissions of the directory. Does the setting of `733` allow `ls` to be performed?*

The permissions granted to the directory means that user `student` cannot read the contents of `temp11`.

```
[student@UWS root]$ pwd
/root
[student@UWS root]$ ls -la | grep temp11
drwx-wx-wx    2 root     root            91 Sep 22 10:29 temp11
[student@UWS root]$ cd temp11/
[student@UWS temp11]$ ls -la
ls: cant open '.': Permission denied
total 0
[student@UWS temp11]$ sudo ls -la
total 16
drwx-wx-wx    2 root     root            91 Sep 22 10:29 .
-- snip --
```

39. *Write down the kernel response for each of the three `more`-command lines given below.*

```
[root@UWS ~]# more /root/temp11/test_a.dat
Donuts
[root@UWS ~]# more /root/temp11/test_b.dat
Donuts
[root@UWS ~]# more /root/temp11/test_c.dat
more: /root/temp11/test_c.dat: No such file or directory
```

40. *Write down the response of the kernel to each of the three `cp`-command given above.*

```
[root@UWS ~]# cp /root/temp11/test_a.dat /home/student
[root@UWS ~]# cp /root/temp11/test_b.dat /home/student
[root@UWS ~]# cp /root/temp11/test_c.dat /home/student
cp: can't stat '/root/temp11/test_c.dat': No such file or directory
```

41. *Explain why you can copy a file, although you can't see it with the ls command.*

```
[student@UWS ~]$ ls -lah /root/ | grep temp11
drwx-wx-wx    2 root     root          91 Sep 22 10:29 temp11
```

Student has execute permissions which allow for the use of any file inside a directory but the user must specifiy an exact file name.

42. *Can you create a file that can be copied by group and world users, residing in a directory with a permission setting of `700`?*

Given the permissions of the directory, no.

```
[root@UWS ~]# touch temp11/q42.file
[root@UWS ~]# chmod 700 temp11/
[root@UWS ~]# ls -lah /root/temp11 | grep q42.file
-rwxrwxrwx    1 root     root           0 Sep 22 13:05 q42.file
[root@UWS ~]# exit
logout
[student@UWS ~]$ cp /root/temp11/q42.file /home/student
cp: cant stat '/root/temp11/q42.file': Permission denied
[student@UWS ~]$ sudo chmod 777 /root/temp11/q42.file
[student@UWS ~]$ cp /root/temp11/q42.file /home/student
cp: cant stat '/root/temp11/q42.file': Permission denied
```

43. *Anonymous ftp-servers only allow users to copy files when they know the file's name. Based on your observations above, which permission would you grant to an anonymous ftp server directory called `download_here` and a file that you create for public download called `download_me`? Which permission would you grant to a file within this directory that should not be downloaded by group or world, called `no_download`? Comment on the advantages of running an anonymous server.*

- `download_here`: 711 allowing full permissions for the owner, execute only for group and world.
- `download_me`: 751 allowing full access for the owner, read and execute for the group, and execute only for world.
- `no_download`: 700 allowing full permissions for the owner and no permissions for group or world.
- Anonymous servers allow for the sharing of files without having to create a user acocunt every time someone new wants to downlload a file. Permissions can be carefully configured to ensure only certain actions can be performed.

44. *Based on your fresh experience: Which of the following permission settings in octal representation are potentially dangerous when assigned to a directory: `700, 033, 633, 644, 755`? Comment on each one individually.*

- `700` creates a directory which is essenaitlly non-existent for anyone except the owner.
- `033` defines a directory with zero permissions for the owner, and write and execute permissions for groups and the public. This allows for the creation, renaming, deletion of files, potentially disastrous, but without read permissions someone would need to know the names of the files they wanted to interact with ahead of time. Unwise for the owner to have zero permissions.
- `633` allows the owner to read and write but not execute. The same security concerns exist as outlined for `033`.
- `644` allows read and write permissions for the owner, and read permissions for all others. Groups and world would not be able to read the contents of the directory.
- `755` allows full permissions for the owner, and read and execute permissions for group and world. Group and world would be able to read the contents of a directory and execute its content.

Lab 03 - Filesystem Hierarchy & User Management

1. Name the different directories under the `/` directory

```
[root@UWS /]# ls -lah | grep '^d'
drwxrwxrwx   22 root     root         501 Sep 28 18:11 .
drwxrwxrwx   22 root     root         501 Sep 28 18:11 ..
drwxr-xr-x    2 root     root        2.3K Nov 29  2020 bin
drwxrwxr-x    3 root     root          82 Dec 10  2020 boot
drwxr-xr-x    4 root     root        2.4K Sep 28 18:11 dev
drwxr-xr-x   17 root     root        1.3K Dec 10  2020 etc
drwxr-xr-x    3 root     root          61 Sep 28 18:11 home
drwxr-xr-x    6 root     root        1.6K Nov 29  2020 lib
drwxr-xr-x    2 root     root          57 Nov 29  2020 libexec
drwxr-xr-x    4 root     root          82 Sep  9  2020 media
drwxr-xr-x    2 root     root          37 Sep  9  2020 mnt
drwxr-xr-x    2 root     root          37 Sep  9  2020 opt
dr-xr-xr-x   60 root     root           0 Sep 28 18:11 proc
drwx------    3 root     root          85 Sep  9  2020 root
drwxr-xr-x    5 root     root         380 Sep 28 18:11 run
drwxr-xr-x    2 root     root        2.6K Dec 10  2020 sbin
dr-xr-xr-x   12 root     root           0 Sep 28 18:11 sys
drwxrwxrwt    5 root     root         160 Sep 28 18:11 tmp
drwxr-xr-x   10 root     root         253 Dec 10  2020 usr
drwxrwxr-x    3 root     root         289 Dec 10  2020 uwslabs
drwxr-xr-x    5 root     root         221 Nov 29  2020 var
```

Dir. Name	Contents
`/bin`	Contains critical programmes required by the system in order to function
`/boot`	Location for the bootable kernel and bootloader configuration
`/dev`	Access points for devices present in the system
`/etc`	Configuration files
`/home`	With the exception of root, the location of each user's home directory
`/lib`	Shared libraries for applications
`/media`	A default location for mounting devices
`/mnt`	An additional mount point for devices
`/opt`	A location where applications can be installed
`/proc`	Information about resources available to the system
`/root`	The root user's home directory
`/sbin`	Applications generally only available to the root user, daemon processes
`/sys`	Contains a `sysfs` filesystem, information about system hardware
`/tmp`	A location for temporary files generated, used, by applications
`/usr`	Contains executables, libraries, other system resources
`/var`	Contains files which are subject to change often, sysem logs, spools, etc.

2. *(After attempting to change to a parent directory from `/`) Explain why there is no difference.*

`/` has no parent folder, it is the top level of the file system.

3. *Are there any hidden files in the root directory? If yes: What are their names?*

Yes. `.fscmd`.

4. *Explain the meaning of the `.` ('dot') and the `..` ('double dot') in the command lines `cd .` and `cd ..`*

- `.` represents the current working directory
- `..` represents the parent directory
- `cd .` change directory to the present working directory
- `cd ..` change directory to the parent of the present working directory

5. *Are there any subdirectories in `/bin`?*

```
[root@UWS bin]# ls -d */
ls: */: No such file or directory
```

If `.` and `..` are considered directories then there are two, otherwise no.

6. *How many commands are in `/bin`? Write down and explain two commands that you already know.*

```
[root@UWS bin]# ls -ALd * | wc -w
103
```

List all files in the present directory, except `.` and `..`, list the referenced file for any symbolic link, and list the directory's entries instead of its contents, omitting the total. Pipe the result through to `wc` and print the number of words to standard out.

7. *Which of the four directories `/bin`, `/usr/bin`, `/usr/sbin`, `/opt` contains the most commands?*

```
[root@UWS /]# find /bin/ -type f -executable | wc -l
97
[root@UWS /]# find /usr/bin/ -type f -executable | wc -l
254
[root@UWS /]# find /usr/sbin/ -type f -executable | wc -l
46
[root@UWS /]# find /opt -type f -executable | wc -l
0
```

8. *Which of the four directories contains a large set of gnome-desktop related applications?*

```
[student@UWS usr]$ sudo find / -name '*gnome*' 2> /dev/null
Password:
/usr/share/bash-completion/completions/gnome-mplayer
```

It appears that the only gnome-related *file* appears in `/usr/share/bash-completion/completions`, and there appear to be zero executable *applications* on the filesystem.

```
[student@UWS usr]$ find / -name -executable '*gnome*' 2> /dev/null
[student@UWS usr]$
```

9. *Can you locate the `chroot` binary within `/usr/sbin`? Where does it point to?*

The question seems to imply that `chroot` should symbolically linked to an alternate location, however this appears to not be the case. The `chroot` binary is located in `/`

```
[student@UWS usr]$ sudo find / -executable -name 'chroot' 2> /dev/null | xargs
file
/usr/sbin/chroot: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked, interpreter /lib/ld-
linux.so.2, for GNU/Linux 4.12.0, stripped
[student@UWS usr]$ sudo find / -executable -name 'chroot' 2> /dev/null | xargs
file
"/usr/sbin/chroot: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked, interpreter /lib/ld-
linux.so.2, for GNU/Linux 4.12.0, stripped"
```

10. *What is the smallest size allocated to a directory file in `/usr`?*

A directory called `i586-buildroot-linux-gnu`.

```
[student@UWS usr]$ ls -lahSr /usr/ | grep '^d'
drwxr-xr-x    3 root     root          57 Nov 11  2020 i586-buildroot-linux-gnu
drwxrwxr-x    6 root     root         123 Dec 10  2020 local
-- snip --
drwxr-xr-x   11 root     root        4.2K Dec 10  2020 lib
drwxr-xr-x    2 root     root        6.8K Dec 10  2020 bin
```

11. *Why might `/usr/lib` be so large?*

```
[student@UWS usr]$ du -sh /usr/lib
68.0M   /usr/lib
```

Given that it contains shared libraries to be used by (potentially) all applications, it stands to reason that a significant amount of common utility will be located in `/usr/lib`, hence the overly large file size relative to other directories.

By comparison, on my own computer the same folder comes in at `6.0GB`.

12. *Which is the biggest standard directory in our system?*

```
[student@UWS /]$ sudo du -hc -d 1 2> /dev/null | sort -gr | grep M
136.0M  total
136.0M  .
111.4M  ./usr
5.8M    ./lib
5.0M    ./boot
4.6M    ./var
3.3M    ./sbin
2.5M    ./uwslabs
2.4M    ./bin
```

`/usr` appears to be the largest standard directory.

13. *What is the total size of our current system?*

The total size of the current system is 136 Megabytes.

14. *Try the command `cat /etc/passwd > /dev/stdout`. (The `cat` command displays (concatenates) the contents of a file to an output device such as the screen...) Explain why you see the contents of the file displayed*

```
[student@UWS /]$ cat /etc/passwd > /dev/stdout
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/false
-- snip --
nobody:x:65534:65534:nobody:/home:/bin/false
student:x:1000:1000:Linux User,,,:/home/student:/bin/bash
```

The content of the file is redirected from standard output, the default behaviour for `cat` and redirected to `stdout`, which then outputs an input stream to standard output. See below for a further example.

```
[student@UWS /]$ echo "hello world" > /dev/stdout
hello world
```

15. *If `stderr` is the standard channel for displaying error messages. Where is `stderr` directed to?*

Since the terminal functions in terms of text streams, and `stderr` outputs to standard output, `stderr` directs output to standard output as default behaviour.

16. *What is the meaning of the `-l` qualifier in the `grep` command?*

`grep -l` will only output matching filenames.

17. *Which files reference the IP-address?*

Assuming the solution uses the lopback address:

```
[student@UWS ~]$ sudo find /etc -type f -exec grep -l '127.0.0.1' {} \;
/etc/security/access.conf
/etc/hosts
```

Assuming otherwise:

```
[student@UWS ~]$ netstat -ie
Kernel Interface table
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.226.169  netmask 255.255.0.0  broadcast 10.5.255.255
        ether 02:8e:83:9a:5a:ff  txqueuelen 1000  (Ethernet)
        RX packets 3091  bytes 222020 (216.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 1535 (1.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2  bytes 140 (140.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 140 (140.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[student@UWS ~]$ sudo find /etc -type f -exec grep -l '10.5.226.169' {} \;      #
zero results
[student@UWS ~]$
```

18. *How would you make absolutely sure, you are examining the whole filesystem for e.g. the occurrence of a pattern like `127.0.0.1`?*

I'd alter the command to search from `/` rather than `/etc`:

```
[student@UWS ~]$ sudo find / -mindepth 0 > find_from_root
[student@UWS ~]$ sudo find /etc -mindepth 0 > find_from_etc
[student@UWS ~]$ wc -l find_from_root
20935 find_from_root
[student@UWS ~]$ wc -l find_from_etc
108 find_from_etc
```

19. *Locate the Linux commands `fsck` and `find` in the filesystem*

```
[student@UWS ~]$ find / -name 'find' -or -name 'fsck' 2> /dev/null
/sbin/fsck
/usr/bin/find
/usr/share/bash-completion/completions/find
/usr/share/bash-completion/completions/fsck
[student@UWS ~]$ whereis find
find: /usr/bin/find /usr/share/man/man1/find.1.gz
[student@UWS ~]$ whereis fsck
fsck: /sbin/fsck.ext4 /sbin/fsck /sbin/fsck.ext2 /sbin/fsck.ext3
/usr/share/man/man8/fsck.8.gz
```

- `/sbin/fsck`
- `/usr/bin/find`

20. *Locate the standar C library represented by the file* `libc.so.6`

```
[student@UWS ~]$ find / -name 'libc.so.6' 2> /dev/null
/lib/libc.so.6
```

21. *Locate and run the application* `dmesg` *using only the find command. Write down the full (find) command line that you used in your log-book.*

```
[student@UWS ~]$ find / -name 'dmesg' -executable 2> /dev/null -exec {} \;
[    0.000000] Linux version 4.12.0 (hecmargi@maximo) (gcc version 9.3.0 (Ubuntu
9.3.0-17ubuntu1~20.04) ) #2 Mon Nov 30 12:20:22 CET 2020
[    0.000000] CPU: vendor_id 'AuthenticX86' unknown, using generic init.
               CPU: Your system may be unstable.
[    0.000000] x86/fpu: x87 FPU will use FXSAVE
[    0.000000] e820: BIOS-provided physical RAM map:
-- snip --
[    5.947823] clocksource: Switched to clocksource pit
[    8.657060] random: crng init done
```

22. *What is the full size of the* `/home` *directory?*

```
[student@UWS ~]$ du -hc /home
4.0K    /home/student/Desktop
680.0K  /home/student
684.0K  /home
684.0K  total
```

684K

23. *Determine the number of shared libraries in* `/lib`*. Shared libraries end in* `.so.*` *(Where* `*` *represents a wildcard).*

```
[student@UWS ~]$ find /lib -name "*.so*" | wc -l
88
```

24. *Find out whether* `libreoffice` *has an entry in* `/var`*.*

It does not.

```
[student@UWS ~]$ sudo find /var -name 'libreoffice'
[student@UWS ~]$
```

25. *Deduce the role of `/var/log/messages` by reviewing its contents.*

```
[student@UWS ~]$ less /var/log/messages
-- snip --
Oct  2 11:58:09 UWS user.info kernel: console [hvc0] enabled
Oct  2 11:58:09 UWS user.info kernel: loop: module loaded
Oct  2 11:58:09 UWS user.info kernel: i8042: No controller found
Oct  2 11:58:09 UWS user.info kernel: NET: Registered protocol family 17
Oct  2 11:58:09 UWS user.info kernel: 9pnet: Installing 9P2000 support
Oct  2 11:58:09 UWS user.info kernel: registered taskstats version 1
Oct  2 11:58:09 UWS user.info kernel: VFS: Mounted root (9p filesystem) readonly
on device 0:14.
Oct  2 11:58:09 UWS user.info kernel: devtmpfs: mounted
Oct  2 11:58:09 UWS user.info kernel: Freeing unused kernel memory: 320K
Oct  2 11:58:09 UWS user.info kernel: Write protecting the kernel text: 3780k
Oct  2 11:58:09 UWS user.info kernel: Write protecting the kernel read-only data:
900k
-- snip --
```

It appears that the file is a place for kernel messages, information about the system, information about boot processes, critical information, etc.

26. *What information is given about the unsuccessful login attempt? Could you identify the hacking culprit at once?*

From `man ps`:

> ruser ... real user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

```
[student@UWS ~]$ tail /var/log/messages
-- snip --
Oct  2 13:35:36 UWS authpriv.notice su: pam_unix(su-l:auth): authentication
failure; logname=student uid=1000 euid=0 tty=console ruser=student rhost=
 user=root
Oct  2 13:35:38 UWS auth.notice su: FAILED SU (to root) student on console
[student@UWS ~]$ cat /etc/passwd | grep 1000
student:x:1000:1000:Linux User,,,:/home/student:/bin/bash
```

User `student` failed to switch user to root at 12:35:38 on the 2nd October.

27. *How does the `tail` command compare to the more command?*

`more` reads the entire input file before paging thorugh its content. `less` performs much the same functionality without the need to read the entire file. It also allows for a command mode which gives users opportunities to interact with the content, i.e. sending a file to `less` and searching for a specific pattern.

28. *What is the name and the size of the biggest filesystem entry of `/proc`? Do you have any idea what it may represent?*

```
[student@UWS proc]$ ls -lahS
total 4
-r--------    1 root     root     1023.9M Oct  2 14:08 kcore
drwxrwxrwx   22 root     root         501 Oct  2 10:57 ..
lrwxrwxrwx    1 root     root          11 Oct  2 14:08 mounts -> self/mounts
lrwxrwxrwx    1 root     root           8 Oct  2 14:08 net -> self/net
dr-xr-xr-x   60 root     root           0 Oct  2 10:58 .
dr-xr-xr-x    8 root     root           0 Oct  2 11:16 1
dr-xr-xr-x    8 root     root           0 Oct  2 11:16 10
-- snip --
```

`kcore` appears to be the largest file. We can find more information about it by using `file`:

```
[student@UWS proc]$ sudo file kcore
kcore: ELF 32-bit LSB core file Intel 80386, version 1 (SYSV), SVR4-style, from
loglevel=3 console=hvc0 root=root rootfstype=9p rootflags=trans=virtio ro TZ=UT
```

It's the kernel core!

29. What is the link between the `PID` of the running processes and the directory names within `/proc`?

There is a direct relationship in that there is a directory in `/proc` for each process ID running at the time. Consider the following example:

```
[student@UWS ~]$ ps -ef | grep sleep
student   1398  1117  0 14:25 hvc0     00:00:00 grep sleep
[student@UWS ~]$ sleep 600 &
[2] 1399
[student@UWS ~]$ ps -ef | grep sleep
student   1399  1117  1 14:26 hvc0     00:00:00 sleep 600
student   1401  1117  0 14:26 hvc0     00:00:00 grep sleep
```

- `sleep` is not currently running
- `sleep` is executed and sent to the background using `&` for 600 seconds
- querying `ps -ef` again shows `sleep` with a process ID of `1399`

```
[student@UWS ~]$ ls -lah /proc | grep 1399
dr-xr-xr-x    8 student  student        0 Oct  2 14:26 1399
[student@UWS ~]$ ls /proc/1399
auxv            cpuset          gid_map         mounts          oom_score_adj
-- snip --
[student@UWS ~]$ kill 1399
[2]-  Terminated              sleep 600
[student@UWS ~]$ ls /proc/1399
ls: /proc/1399: No such file or directory
```

- `ls` is run on `/proc` and fed to `grep` with a pattern matching the process ID
- a directory is found containing a number of files, directories, links, etc.
- the process is terminated using `kill` against the corresponding process ID
- running `ls` a second time on the directory reveals that the directory no longer exists. A direct relationship exists.

30. *What do you think will happen to each directory in /proc after the associated process has been killed?*

When a process is killed its corresponding directory entry in `/proc` will no longer exist.

31. *What is written in the `cmdline` file? Does this agree with the information as given by the command, `ps -ef`?*

each instance of `cmdline` contains the content of the `CMD` column for each entry returned by `ps-ef`.

32. *What is the status (check the contents of `/proc/1/status`) and the memory size (`VmSize`) that is used by the init process?*

```
[student@UWS ~]$ cat /proc/1/status | grep VmSize
VmSize:     2108 kB
```

33. *Name the different directories that are present in the `/media` directory.*

- `disk1`

34. *Now set in `/media/floppy` (having a formatted floppy with some data on it inserted).*

```
root@UWS ~]# cd /media/floppy/
[root@UWS floppy]# echo 'a dummy file' > testfile.dat
[root@UWS floppy]# ls -l
total 14
-rw-r--r--    1 root     root            61 Sep 13  2020 file.txt
drwx------    2 root     root         12288 Sep 13  2020 lost+found
-rw-r--r--    1 root     root            13 Oct  2 20:58 testfile.dat
```

35. *Why is the actual password depicted as `x`, although the password is not `x`?*

This indicates that the password is encrypted as part of `/etc/shadow`.

36. *What is the user identification (UID), home directory and login shell of the root-user?*

```
[student@UWS /]$ more /etc/passwd
root:x:0:0:root:/root:/bin/sh
```

- `0`
- `/root`
- `/bin/sh`

37. *What is the `UID` of the daemon called `uucp`?*

```
[student@UWS /]$ grep uucp /etc/passwd
[student@UWS /]$ cat etc/passwd
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
sync:x:4:100:sync:/bin:/bin/sync
mail:x:8:8:mail:/var/spool/mail:/bin/false
www-data:x:33:33:www-data:/var/www:/bin/false
operator:x:37:37:Operator:/var:/bin/false
nobody:x:65534:65534:nobody:/home:/bin/false
student:x:1000:1000:Linux User,,,:/home/student:/bin/bash
```

There appears to be no entry for `uucp` as part of the list of users.

38. *Determine the `UID` of student in the Debian system.*

`1000`

39. *How many users use the `/bin/sh` shell as their login shell?*

```
[student@UWS /]$ more /etc/passwd | grep '/bin/sh' | wc -l
1
```

- 1
- `wc -l` receives standard input and returns the number of lines present in the input.

Lab 04 - Process Management

1. *What is the difference between `ps` and `ps -ef`?*

- `ps` shows the active processes for the current user
- `ps -ef` shows all active processes

2. *Explain the meaning of the column headers `UID`, `PID`, `PPID` and `CMD`*

Header	Meaning
UID	User ID, who the process belongs to
PID	Process ID, the unique number assigned to a process when it's started up
PPID	Parent Process ID
CMD	The command line being executed

3. *Write down the process identifiers of at least two processes that have the PPID of 1, representing the init process*

```
[student@UWS ~]$ ps --ppid 1
  PID TTY          TIME CMD
   82 ?        00:00:00 syslogd
   84 ?        00:00:00 klogd
   92 ?        00:00:00 rpcbind
  112 ?        00:00:00 dhcpcd
  119 ?        00:00:00 dropbear
  126 ?        00:00:00 rpc.statd
  138 ?        00:00:00 rpc.mountd
  144 ?        00:00:00 crond
  174 ?        00:00:00 login
```

4. *Is the process `ps -ef` listed itself? If yes, what is its PID on the system?*

```
[student@UWS ~]$ ps -ef | grep student
root       174     1  0 19:42 ?        00:00:00 login -- student
student    192   174  0 19:44 hvc0     00:00:01 -bash
student    246   192  0 20:09 hvc0     00:00:00 ps -ef
student    247   192  0 20:09 hvc0     00:00:00 grep student
```

- Yes
- 246

5. *How many processes in total are currently being run by the root user?*

```
[student@UWS ~]$ ps -fu root | wc -l
45
```


6. *What is the `PID` of the terminal and the `nano` process?*

```
[student@UWS ~]$ nano &
[1] 253
[student@UWS ~]$ ps
  PID TTY          TIME CMD
  192 hvc0     00:00:02 bash
  253 hvc0     00:00:00 nano
  254 hvc0     00:00:00 ps

[1]+  Stopped                 nano
```

- nano: 253
- bash: 192

7. *What is the function of the ampersand `&` in the above command?*

`&` sends a process to the background.

8. *What is the PID and PPID of the terminal and the two processes `nano` and `vi`? What is the relation between the calling terminal shell and the two processes?*

```
[student@UWS ~]$ ps -ef | grep -e nano -e vi
student    253   192  0 20:11 hvc0     00:00:00 nano
student    255   192  0 20:16 hvc0     00:00:01 vi
```

`nano` and `vi` have PIDs 253 and 255 respectively, and they both share PPID 192, which is the PID of `bash`

9. *Did this kill the process `vi`?*

```
[student@UWS ~]$ kill 255; ps -fu student
UID        PID  PPID  C STIME TTY          TIME CMD
student    192   174  0 19:44 hvc0     00:00:02 -bash
student    253   192  0 20:11 hvc0     00:00:00 nano
student    255   192  0 20:16 hvc0     00:00:01 vi
student    275   192  0 20:20 hvc0     00:00:00 ps -fu student
```

No.

10. *Use the `man kill` command and check for the meaning of the qualifier `-9` to find out why the process has been killed this time?*

`-9` represents a `KILL` signal, which unconditionally and immediately halts the execution of a programme.

11. *What is the PPID and PID of each `top` process?*

```
[student@UWS ~]$ ps -ef | grep top
student    200   199  0 09:54 hvc0     00:00:00 top
student    202   201  0 09:54 hvc0     00:00:00 top
student    205   201  0 09:54 hvc0     00:00:00 grep top
```

- PIDs: 200, 199
- PPIDs: 202, 201

12. *What happened to the two `top` processes? Were they killed too? What is the new `PPID` of the two `top` processes? Can you therefore explain what happens to orphaned processes?*

```
[student@UWS ~]$ ps -ef | grep top
student    200   199  0 09:54 hvc0     00:00:00 top
student    207   199  0 09:55 hvc0     00:00:00 grep top
```

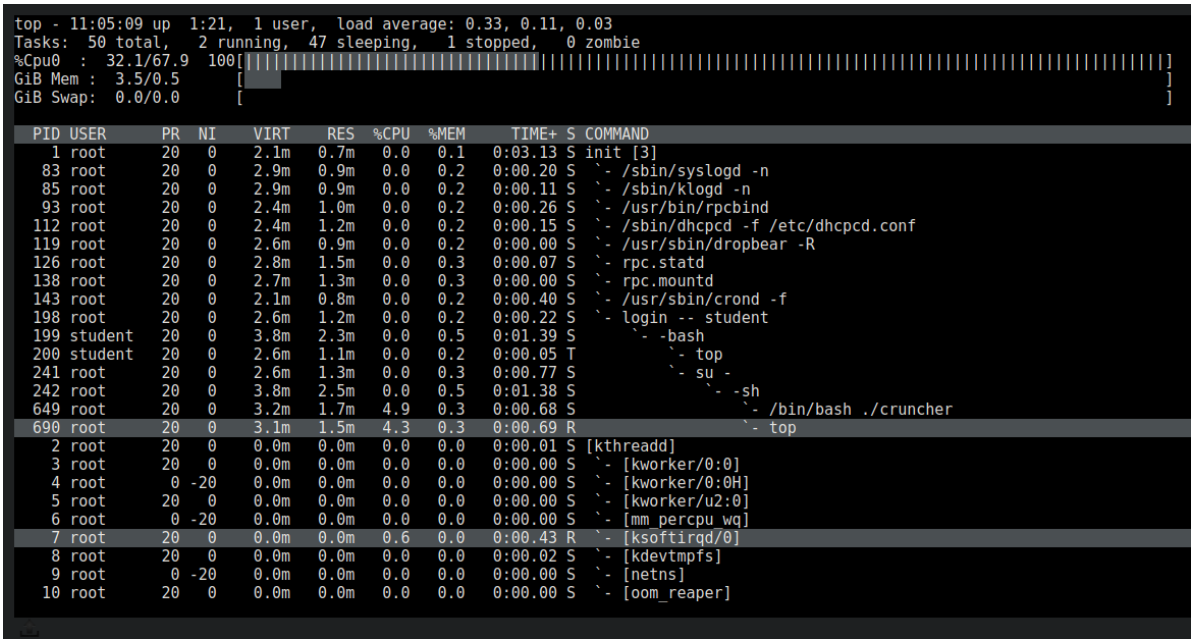
When the parent process was killed, the child was also killed. Process 200 remains in the background as its parent was untouched. If the question requires the overall parent to be killed, i.e. PID 199 `bash`, the terminal emulator would be killed along with its child processes, and the user would be required to log in.

An orphaned process is one whose parent has terminated but continues to execute. Running `ps -ax` and finding an absence of `PID -1` allows us to confirm that there are no current zombie or orphaned processes:

```
[student@UWS ~]$ ps -ax | head
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:02 init [3]
    2 ?        S      0:00 [kthreadd]
    3 ?        S      0:00 [kworker/0:0]
    4 ?        S<     0:00 [kworker/0:0H]
    5 ?        S      0:00 [kworker/u2:0]
    6 ?        S<     0:00 [mm_percpu_wq]
    7 ?        S      0:00 [ksoftirqd/0]
    8 ?        S      0:00 [kdevtmpfs]
    9 ?        S<     0:00 [netns]
-- snip --
```

If any of the `top` processes had been orphaned and adopted by `init` we'd see them with a PID of `1`.

13. *Describe the output that you see. How much approximate CPU time (in %) does the cruncher program use while it is still running?*



`top` appears in the terminal window with a cascading series of processes stemming from `init`, leading to `cruncher` and `top`, before a series of processes related to `[kthreadd]` appears. In CPU time %, `cruncher` appears to use between 3-5%.

14. *In what order does the `top` command display the processes?*

`top` displays processes in %CPU descending in a tree format, i.e. a `cruncher`, when using significant CPU time, will appear as a child of `init`. If `cruncher` is the most intensive process running at the time, `init` will appear at the top of the list despite its current %CPU reading of ~0.

15. *How many processes are running on your machine in total? How many of them are sleeping?*

```
[root@UWS ~]# top
top - 11:14:24 up  1:30,  1 user,  load average: 0.00, 0.03, 0.01
Tasks:  49 total,   1 running,  47 sleeping,   1 stopped,   0 zombie
-- snip --
```

- 49 total
- 47 sleeping

16. *The 'Tasks' row has space for zombie processes. Do you have any idea what a zombie in a Unix-like system might refer to?*

When a process is killed it informs its parent of its pending termination. If this is successful its PID will be removed from `ps`. If the parent fails to acknowledge the termination, for whatever reason, the process becomes a zombie process. A zombie process has been terminated, (and therefore) cannot be killed, and its resources have been freed. Either init adopts the zombie process, or the process is cleared on boot.

17. *What is the priority number of the program cruncher? Does it change at all?*

`20`, and the priority value does not appear to change throughout the programme's lifetime.

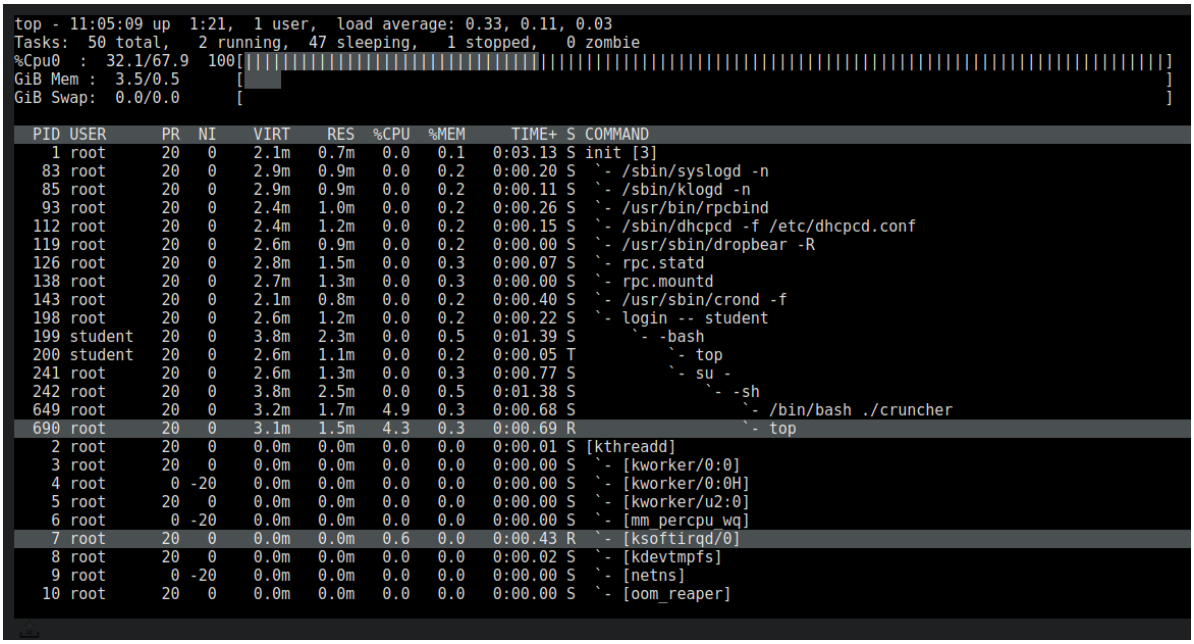
18. *What is the highest priority that any of the processes gets assigned?*

- BSD C Shell & standalone: 20
- System V C Shell: 39

19. *What priority level is given to the majority of the processes?*

20

20. *Is any swap space used at this time?*



`0.0GiB`

21. *Explain briefly what kind of information `vmstat` displays here? What do the two qualifiers: `10` and `4` stand for?*

Statistics pertaining to the system's virtual memory. `vmstat n` will cause the programme to refresh every `n` seconds, where 10 is every ten seconds, and 4 is every four seconds.

22. *What numbers change in the `vmstat` display? What does the number in the `r` column represent?*

With a number of `cruncher` programmes executing in the background:

- columns `r, in, cs, us, sy,` and `id` all change at least once.
- `r` represents the number of runnable processes which are either running, or waiting for run time.

23. *What happens, while running the application programs, to the last three entries shown below the cpu heading that are labelled: `us sy id`? To which value do they always add up? Can you interpret their meaning?*

The values of `us, sy`, and `id` each add up to 100. These numbers vary over time, with `id` appearching to change once at upon initial execution of the `vmstat` command, and when the background instances of `cruncher` appear to have each completed processing.

```
[root@UWS ~]# vmstat 10
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 4  0      0 495352      0   4952    0    0     0     0  904   78  9 15 77  0  0
 5  0      0 495352      0   4952    0    0     0     0 3608  368 36 64  0  0  0
 4  0      0 495384      0   4952    0    0     0     0 3626  369 37 63  0  0  0
 4  0      0 495260      0   4952    0    0     0     0 3636  366 39 61  0  0  0
 4  0      0 495352      0   4952    0    0     0     0 3638  370 38 62  0  0  0
 0  0      0 496160      0   4944    0    0     0     0 2753  276 28 47 25  0  0
```

24. *What happens to the number associated with the `us` column once all cruncher sessions have terminated?*

It rests at zero.

25. *What information does `uptime` provide?*

```
[root@UWS ~]# uptime
 11:55:33 up 12 min,  1 user,  load average: 0.44, 0.83, 0.46
```

The length of time that the system has been running, the number of logged in users, the time and date the system came online, and information about the system's average load.

26. *What is the name and the significance of the process that is displayed in the leftmost position*

`init`. Its significance is that it is the direct, indirect parent of all processes running on the system.

27. *Write down all linking processes between init and pstree. Could you obtain the same information using the ps command?*

`init --> login --> bash --> pstree`

A clumsy way of performing this would be to find the process ID of some programme and work backward:

```
[student@UWS ~]$ sleep 600 &
[1] 238
[student@UWS ~]$ ps -o ppid= -p 238
  192
[student@UWS ~]$ ps -o ppid= -p 192
  174
[student@UWS ~]$ ps -o ppid= -p 174
    1
```

28. *What are the nice numbers of some processes: e.g. init , kthread, cron and udevd?*

```
[student@UWS ~]$ top -b -n 1 | grep "NI\|cron\|init\|kthread\|udevd"
  PID USER      PR  NI    VIRT    RES  %CPU  %MEM     TIME+ S COMMAND
    1 root      20   0    2.1m   0.7m   0.0   0.1   0:02.84 S init [3]
  144 root      20   0    2.1m   0.8m   0.0   0.2   0:00.11 S  - /usr/sbin/crond
-f
  333 student   20   0    2.9m   1.0m   0.0   0.2   0:00.03 S          - grep
NI\|cron\|init\|kthread\|udevd
    2 root      20   0    0.0m   0.0m   0.0   0.0   0:00.01 S [kthreadd]
```

They're all 0.

29. *What is the nice number `NI` given by the system to the cruncher script?*

`20`

30. *Note the actual outputs for real, user and sys times for the cruncher script after it has finished executing.*

```
[root@UWS ~]#
real    0m15.022s
user    0m4.394s
sys     0m9.989s
```

31. *Note the real, user and sys times again and comment on any difference in the real time value*

```
[root@UWS ~]# time nice -n 19 ./cruncher > /dev/null
real    0m14.351s
user    0m4.453s
sys     0m9.710s
```

This second command was able to perform `0.671s` faster than the previous command, possibly due to the increased processor time enjoyed by the process as a result of the reduction in nice value.

32. *Why do you think, user and sys time are not so different when compared to the values obtained after the first run of cruncher?*

The processing time is the factor affected by the reduction in the process' `nice` value, the time it takes for the system to read the file and process system calls will be the same regardless of the `nice` value.

33. *Note the actual outputs for real, user and sys time again. Comment on any improvement in performance.*

```
[root@UWS ~]# time nice -n -20 ./cruncher > /dev/null
real    0m14.649s
user    0m4.582s
sys     0m9.965s
[root@UWS ~]#
```

There was no real effect on performance, presumably because there are so few active processes running on the system. Altering the `nice` value will only have significant impact on a busy system.

35. *Note the actual outputs of the time command for real, user and sys. How do the time differences between `nice 19` and `nice -20` compare to the time differences on a quiet system?*

```
[root@UWS ~]# time nice -n 19 ./cruncher > /dev/null &
[2] 9044
[root@UWS ~]# time nice -n -20 ./cruncher > /dev/null &
[3] 9110
[root@UWS ~]#
real    0m13.012s
user    0m4.497s
sys     0m8.321s

real    0m27.496s
user    0m4.467s
sys     0m9.770s
```

The `nice -20` command completed roughly twice as fast as the `nice 19` command.

36. *Note the output of the `renice` command.*

```
[root@UWS ~]# time nice -n 19 ./cruncher > /dev/null &
[2] 9850
[root@UWS ~]# renice -20 9850
9850 (process ID) old priority 0, new priority -20
[root@UWS ~]#
real    0m14.696s
user    0m4.178s
sys     0m9.481s
```

37. *Estimate the performance gain of the above action.*

After reconfiguring the nice value, the process runs about twice as fast as it would with its initial nice value.

38. *Which file contains a list of users who are allowed to submit requests to the `crontab` facility?*

If it exists, `cron.allow`.

39. *Which file contains a list of users who are NOT allowed to submit request to the `crontab` facility? If a user's name appears in both lists, which list takes precedence?*

`cron.deny`, and `cron.allow` respectively.

40. *Explain the result of the above command in detail.*

```
[root@UWS etc]# du -k /home
4       /home/student/Desktop
4       /home/student/.config/procps
8       /home/student/.config
20      /home/student
24      /home
```

This command lists the sizes of a directory and its subdirectory in a given unit - in this case, `KB`.

41. *What does the piping in the `sort –nr` and the `head –5` do to the `du -k /home` output? Explain in detail.*

- `sort –nr` sorts some input in reverse numerical order
- `head -5` receives a test stream and outputs the top five lines of that stream

42. *Note the output of the above command.*

```
[root@UWS etc]# crontab -l
5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 * * * * du -k /home | sort -nr | head
-5 >> /root/test.dat
```

43. *What happens to the size of `/root/test.dat` over time? Do you see any danger in having a steadily growing output file in a directory?*

The cronjob uses the append operator `>>`, meaning every five minutes another five lines will be added to the file, an disk space, while cheap, is finite!

44. *What would be the entry in `/tmp/entries` to perform the command once every day at `23:15`?*

```
15 23 * * * du -k /home | sort -nr | head -5 >> /root/test.dat
```

45. *What would be required to perform it at 4:30pm on the 21 st of January every year?*

```
30 16 21 1 * du -k /home | sort -nr | head -5 >> /root/test.dat
```

46. *Find an alternative `cron` syntax that could be used to specify that a job be performed every 5 minutes. Note the alternative notation as the answer in your logbook.*

```
*/5 * * * * * du -k /home | sort -nr | head -5 >> /root/test.dat
```

47. *In which different time-periods are the system `cron`-jobs ordered?*

```
[root@UWS etc]# ls -lah | grep cron
drwxr-xr-x    2 root     root          60 Nov 11  2020 cron.d
drwxr-xr-x    2 root     root          37 Nov 11  2020 cron.daily
drwxr-xr-x    2 root     root          37 Nov 11  2020 cron.hourly
drwxr-xr-x    2 root     root          37 Nov 11  2020 cron.monthly
drwxr-xr-x    2 root     root          37 Nov 11  2020 cron.weekly
```

- daily
- hourly
- monthly
- weekly

48. *Note an example of a `cron`-job service task that is found in the `cron.daily` directory.*

```
[root@UWS etc]# ls -lah cron.daily/
total 8
drwxr-xr-x    2 root     root          37 Nov 11  2020 .
drwxr-xr-x   17 root     root        1.3K Dec 10  2020 ..
```

There aren't any!

49. *What is the exact meaning of the `-r` character in the `crontab -r` command line?*

The -r option causes the current crontab to be removed.

Lab 05 - The Shell

1. *How many variables are currently defined in your shell? (Tip: use `wc -l`)*

```
[student@UWS ~]$ printenv | wc -l
14
```

2. What are the values for the shell variables `DISPLAY`, `LOGNAME`, and `PATH`?

```
[student@UWS ~]$ printenv | grep "DISPLAY\|LOGNAME\|PATH"
LOGNAME=student
PATH=/usr/local/bin:/bin:/usr/bin
```

`DISPLAY` does not appear to be set at this time.

3. *To which directory does the `cd` command set you now?*

```
student@UWS ~]$ cd
[student@UWS ~]$ pwd
/home/student
[student@UWS ~]$ HOME=/etc
[student@UWS student]$ export HOME
[student@UWS student]$ cd
[student@UWS ~]$ pwd
/etc
```

`cd` with no arguments changed the present working directory to `/etc`

4. *In which directories are the `fsck`, `which`, and `whereis` commands found?*

```
[student@UWS ~]$ whereis fsck which whereis cp
fsck: /sbin/fsck.ext4 /sbin/fsck /sbin/fsck.ext2 /sbin/fsck.ext3
/usr/share/man/man8/fsck.8.gz
which: /usr/bin/which /usr/share/man/man1/which.1.gz
whereis: /usr/bin/whereis /usr/share/man/man1/whereis.1.gz
cp: /bin/cp /usr/share/man/man1/cp.1.gz
```

5. *How many directories are searched for the occurrence of a program or command?*

`whereis` will search through the `$PATH` and `$MANPATH` variables for programmes, commands, and manual pages.

```
[student@UWS ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin
```

6. *What is the output if you try to run `whereis cp`?*

```
[student@UWS ~]$ whereis cp
cp: /bin/cp /usr/share/man/man1/cp.1.gz
```

7. *What is the `whereis` argument used to search only manuals?*

`-m`

8. *Which binary directories are omitted for student as opposed to root? Explain why this might be the case.*

```
/usr/local/bin:/bin:/usr/bin
[student@UWS ~]$ su -
[root@UWS ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

Any directories containing system binaries are omitted from the student user's path given that programmes in these binaries run themselves as root.

9. *What is the output if you try to run `ipmine`?*

```
[root@UWS ~]# cp /sbin/ip /root/ipmine
[root@UWS ~]# ipmine
-sh: ipmine: command not found
```

`/root` is not present in `$PATH` which is why it doesn't run.

10. *Explain the effect of the two commands `PATH=$PATH:.` and `export PATH` in your own words.*

`PATH=$PATH:.` assigns the current value of `$PATH` along with a colon and the present working directory to the `PATH` variable.
`export PATH` defines a shell environment variable which allows functions and parameters to be called in the current session.

11. *Does `ipmine` run now? If 'yes' explain why.*

Yes. the path environment variable has been updated to include the present working directory. We invoke the command and the shell searches for an appropriate binary within the path variable. It was found, so it was run.

12. *What command provides the same output as `echo *` ?*

`ls`

13. *Explain the two command lines above in respect to the fact that the shell expands wildcards*

`*` will match any group of zero or more characters. The shell will expand a single `*` to match any string as an argument. In the case of `*.dat` it will expand to only include string which match the given pattern.

14. *Explain the output produced by the FOUR command lines in detail.*

Each of the commands uses `?` which, in unix, will match a single character. In the case of `echo ???.dat` the only files which will be returned to standard out are those which contain three characters before the `.dat` suffix. The same is true for four, five, and six character filenames in the remaining three commands.

15. *What does the shell force the echo command 'to echo' if the expansion of wildcards does not produce any results, as in the case of the last command line?*

The shell will yield zero matching results and defer to standard `echo` functionality: feeding the programme a string to echo back to standard out.

16. *Do you know what type of files are found by the above command? Explain the function of the 3 metacharacters and the wildcard given within:* `[K][0-9][0-9]*`

- `[K]` will match the letter `K` exactly,
- `[0-9]` will match any digit in that range exactly, the same holds for the following `[0-9]`,
- `*` will match zero or more preceding characters.

17. *What actually happened to the existing file test.dat? What was the 1st task the shell performed once the above command line was issued?*

The shell would have looked through `$PATH` to find the `echa` application.

18. *Reflect on some potential dangers of this default behaviour*

I'm not sure I understand the question but I suppose a user might attempt to execute some command, programme a script, or automate some function which might attempt to run either a non-existent programme or the incorrect programme altogether which would result in unexpected or unintended behaviour. The shell assumes the user knows what they're doing.

19. *Explain the differences in the output of the above command lines*

`echo "$HOME"` will cause the shell to interpret and evaluate the value between the quotation marks which will result in the value of `$HOME` being presented to the use via echo. Inverted commas will simply have the shell return the string literal.

20. *Explain the output of the above command lines, if necessary use the man pages to understand the different output.*

```
[root@UWS ~]# echo `echo $HOME`
/root
[root@UWS ~]# echo `hostname`
UWS
[root@UWS ~]# echo `hostname -i`
127.0.1.1
```

In each statement the text within the backticks will be evaluated by the shell and it will replace the text contained by the backticks in the original statement. The result of `echo $HOME` in this instance is `/root`, which is then echoed by the first `echo` command.

21. *What is the function of the semicolon* `;` *in the above command line?*

The semicolon indicates the end of a complete statement. Anything following a semicolon will be interpreted by the shell as a new, independent statement.

22. *Would the script `envdisplay` run without a change of the permission settings? Note the current permission settings in octal representation.

No, given the lack of entry in the permission bit for the current user. The octal value is `644`

23. *Note the new permission settings of `envdisplay`, after the `chmod` command was issued, in octal representation. Why is the script executable now?*

The new permissions octal value is `755`. Permission bits have been granted to the file and can now be executed.

24. *Explain the output of `envdisplay`.*

The content of `envdisplay` is a series of commands which will print the content of several environment variables to standard out.

25. *Explain why you should only see the line `I did something wrong`, if your script does not perform as intended.*

`||` between two commands ensures that the second command will only execute if the first command fails to execute appropriately.

26. *Explain the output of `envdisplay`. In particular, why does the first invocation of `envdisplay` not produce the string test, yet the second one does?*

The first command is simply evaluated as a string and passed as an argument to the script to evaluate, which it will, as a string. Prefacing the variable name with a `$` symbol indicates a variable whose associated value should be evaluated and passed into the script.

27. *What is the value of `$HOME` after the script has finished running? What has happened?*

When a script is run the shell evaluates the shebang if one exists and spawns a new instance of that application, in our example, `/bin/bash/`. This new instance of bash runs and executes the script, and its environment variable is changed. When the script closes so too does that instance of bash, and the calling instance of bash remains untouched.

28. *What is the value of `$HOME` now? What happened?*

The current shell was used to execute the script which means the `$HOME` environment variable has been set to `/etc/`.

29. *Besides `/insecure` and `/home/crazy` are there any other unprotected directories? Try with at least one other insecure permission (`766, 733` or `722`).*

```
[root@UWS /]# find ./ -type d -perm 777 -print
./
./dev/shm
./home/crazy
./insecure
```

30. *What does the `-perm` in the find qualifier do?*

It searches for files and directories by a given set of permissions.

31. *What would be the find command line to change the directories from `777` to `755`?*

```
[student@UWS /]$ find ./ -type d -perm 777 -exec chmod 755 {} \;
```

32. *Does your shell script work?*

Yes, though there's no output.

33. *Comment on the different output.*

The only difference was the speed in execution which, I imagine, is due to the drastically reduced search parameter.

34. *Check whether your shell-script works by running it twice in a row (the 2nd time no directories should be found) or by checking the permissions using the ls -l command on the directories that were found to have the permission 777. What is the output of the script after the 1st and 2nd run?*

The directories which were found to have insecure permissions octal values had theri values changed to reflect the `-exec` argument and its subsequent command. The second run yielded no results given the lack of candidate files.

35. *Given a PID and nice number will `renice` a process with the given PID to the given nice number. This should be a one line shell-script using two input variables `$1` and `$2`, representing the nice number and the PID of the process. Try it out with your cruncher script.*

```
renice $1 $2
```

36. *Try to get the script to find the required `PID` if provided with only a process name only and a nice number.*

```
renice $1 `pidof $2`
```

Lab 06 - Package Management

1. *What is the meaning of `-o rw` in the command for mounting the floppy?*

Using `mount -o` will allow the user to select specific mounting options in a comma separated list, and `rw` will override the kernel default of mounting as read and will instead mount as read/write. `mount` will never attempt to read-only mount a filesystem when the `rw` argument is used.

2. *What is the meaning of the `-c` qualifier in the tar command?*

`-c` will create a new archive.

3. *What is the name of the tar file on the floppy?*

```
[root@UWS ~]# ls -lah /media/floppy/
total 39
drwxr-xr-x    3 root     root        1.0K Oct 28 19:40 .
drwxr-xr-x    3 root     root          60 Sep  9  2020 ..
-rw-r--r--    1 root     root       20.0K Oct 28 19:40 backuptest.tar
-rw-r--r--    1 root     root          61 Sep 13  2020 file.txt
drwx------    2 root     root       12.0K Sep 13  2020 lost+found
```

`backuptest.tar`

4. *Note down what was displayed by the tar command.*

```
[root@UWS ~]# tar -c /home/backuptest/ > /media/floppy/backuptest.tar
"tar: Removing leading `/' from member names"
```

There was no size displayed by the tar command, though the size of the tar is `20K`

5. *What is the meaning of the `-s` qualifier in `ls`? What does the `-k` qualifier achieve when used in conjunction with the `du` command?*

- The `-s` qualifier prints the allocated size of each - or in this case, a - file in blocks
- The `-k` qualifier in alongside `du` is equivalent to `--block-size=1K`

6. *What is the difference in size between the original directory and the backup on the floppy? Does `tar` perform any compression?*

```
[root@UWS ~]# ls -s /media/floppy/backuptest.tar
21 /media/floppy/backuptest.tar
[root@UWS ~]# du -k /home/backuptest/
20      /home/backuptest/
```

- The tar is a block larger in size.
- `tar` does not perform compression, it only archives.

7. *Examine `/tmp/home/backuptest` directory. Is this an exact copy of the original `/home/backuptest` directory?*

```
[root@UWS tmp]# diff /tmp/home/backuptest/ /home/backuptest/
[root@UWS tmp]#
```

`diff` presented no output, so they're the same

8. *Explain the meaning of the `xvf` qualifiers. Why is there no hyphen sign?*

The `xvf` qualifiers extract archive files verbosely.

9. *Why is there no output from the `cmp` command?*

`cmp` will only produce output if there's any difference between the passed files. Since they're identical, there's no output.

10. *Compare the date of creation of both files. Did the backup file inherit the same creation date?*

```
[root@UWS tmp]# ls -lisan /tmp/home/backuptest/world.dat
/home/backuptest/world.dat
   6747      4 -rw-r--r--    1 0        0               12 Nov  1 11:03
/home/backuptest/world.dat
   3884      4 -rw-r--r--    1 0        0               12 Nov  1 11:03
/tmp/home/backuptest/world.dat
```

The creation time of a file appears to be inaccessible, however both files share the same modified datetime which corresponds to the extraction datetime.

11. *Whilst examining `/tmp` do you see why tar removes the leading `/` of the path when creating an archive? Consider what would happen if the leading forward-slash was not removed by the `tar` command and you later attempted to un-tar the tarball (`tar` archive) in superuser mode*

- The path of the file we're archiving is relative.
- If the leading `/` wasn't omitted the application could potentially overwrite existing data.

12. *Check the size of the resulting file `/media/floppy/backuptest.tar.gz`. What is the compression rate now if you use the whole size of the directory `/home/backuptest` as a reference point?*

```
[root@UWS /]# tar -c /home/backuptest | gzip --verbose >
/media/floppy/backuptest.tar.gz
"tar: Removing leading `/' from member names"
"96.8%"
```

The compression ratio is 96.8%

```
[root@UWS /]# ls -lah /media/floppy/backuptest.tar.gz /home/backuptest
-rw-r--r--    1 root     root         674 Nov  1 11:29
/media/floppy/backuptest.tar.gz

/home/backuptest:
total 24
drwxr-xr-x    2 root     root          90 Nov  1 11:03 .
drwxr-xr-x    4 root     root          88 Nov  1 11:00 ..
-rw-r--r--    1 root     root        8.7K Nov  1 11:05 vmstat.dat
-rw-r--r--    1 root     root          12 Nov  1 11:03 world.dat
```

- The zipped tarball is 674 bytes in size

```
[root@UWS /]# du -k /home/backuptest /media/floppy/backuptest.tar.gz
20      /home/backuptest
1       /media/floppy/backuptest.tar.gz
```

- The zipped tarball is allocated a single block
- The original directory is assigned 20 blocks

13. *What means the `-p` option?*

`-p` extracts information about file permissions. It will preserve permissions on extraction.

14. *Why did the second command using `/home/backuptest/world.dat` as the destination not work?*

`tar` removed the leading `/` when the archive was initially created.

15. *Write a shell script that produces a zipped-tar version of any input directory called `dir_name`, whereby `dir_name` represents the relative directory name, not the absolute one. The output should be send to `/tmp` or a floppy device and should be of the form: `dir_name.tar.gz`.*

```
#!/bin/bash
file=`basename $1`
dir=`dirname $1`
tar -czf /media/floppy/`$file`.tar.gz -C $dir $file
```

16. *What does the `nsnake` package do?*

```
[root@UWS ~]# apt-cache search nsnake
nsnake - classic snake game on the terminal
```

17. *What does the number shown in the output of the command pidof represent?*

```
[root@UWS ~]# pidof nsnake
573
[1]+  Stopped(SIGTTOU)        /usr/games/nsnake
```

- `573` represents the process ID
- `[1]` represents the parent ID

18. *Verify that nsnake has been deleted by checking `/usr/games/nsnake` . What did you find this time?*

```
[root@UWS ~]# whereis nsnake
nsnake:
[root@UWS ~]# ls -lah /usr/games/nsnake
ls: /usr/games/nsnake: No such file or directory
```

19. *Briefly outline the functionality of the `md5sum` command.*

`md5sum` calculates MD5 cryptographic checksums.

20. *Did you get a match with the md5 checksum and if so what does this prove?*

```
[root@UWS ~]# md5sum hello.c
1813f4cdd3fcf986a25981f95c0dc0d2  hello.c
```

I'm not sure I understand the question. The above command simply presents the MD5 checksum for a file. The wording of this question '... get a match' implies some other file's MD5 sum would be tested against `hello.c`. If the aim of the exercise was to compare the sum of `hello.c` and its compiled `hello` counterpart, it stands to reason that the values would not match.

```
[root@UWS ~]# md5sum hello hello.c
8fcc5e999b835db0c79c570918539317  hello
6b9d1a1169e6ec3b160e5e5883765e7c  hello.c
```

If the aim was to compare the file to itself:

```
[root@UWS ~]# md5sum hello hello.c
8fcc5e999b835db0c79c570918539317  hello
6b9d1a1169e6ec3b160e5e5883765e7c  hello.c
```

It stands to reason that the sum would match. The point of checking an MD5 sum is to ascertain the integrity of a file after it's been through some change. A common practice is to check the sum of a file locally against some known value in order to ensure the file's integrity is valid.

21. *What is the difference in size between `hello.c` and `hello`?*

```
[root@UWS ~]# ls -lah | grep hello
-rwxr-xr-x    1 root     root        1.8K Nov  1 12:16 hello
-rw-------    1 student  root         108 Nov  1 12:16 hello.c
```

22. Can you explain the size differences?

The compilation process converts source code into object or machine code and comprises of pre-processing, compiling, assembling, and linking, though these steps are generally contingent on the compiler. An output binary is almost always significantly larger than the input source file as a result of this process.

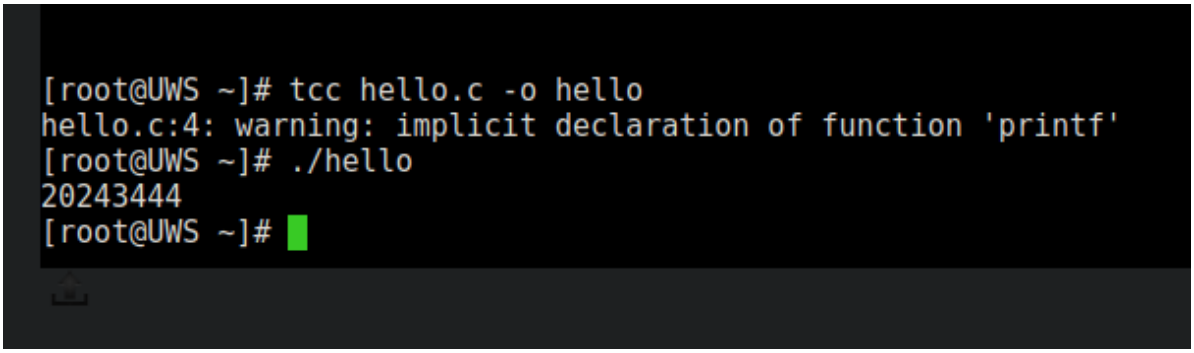
23. *Modify the `hello.c` to display your name and banner ID instead of "Hello, World!".*

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    printf("20243444\n");
    return 0;
}
```

24. *Compile and run the modified program.*

```
[root@UWS ~]# tcc hello.c -o hello
"hello.c:4: warning: implicit declaration of function 'printf'"
[root@UWS ~]# ./hello
20243444
```

25. *Do a screenshot and add it to your notes and lab book.*


[root@UWS ~]# tcc hello.c -o hello
hello.c:4: warning: implicit declaration of function 'printf'
[root@UWS ~]# ./hello
20243444
[root@UWS ~]#

Lab 07 - Configuration

1. *Describe what happened*

```
[root@UWS ~]# systemctl isolate rescue.target
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Give root password for system maintenance
(or type Control-D for normal startup):
sulogin: starting shell for system maintenance
```

- `systemctl isolate UNIT` starts the unit specified on the command line and its dependencies and stops all others
- `rescue.target` is the name of the unit to be loaded

2. *Is there any difference in the amount of free memory now available, and if yes, how much memory is freed in this mode?*

The first time this command was run:

```
[student@UWS ~]$ vmstat 5 5
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 480672      0  19320    0    0     0     0  252   21  8  4 88  0  0
 0  0      0 480672      0  19316    0    0     0     0  106    4  0  0 99  0  0
 0  0      0 480672      0  19304    0    0     0     0  102    4  0  1 99  0  0
 0  0      0 480704      0  19300    0    0     0     0  101    3  0  1 99  0  0
 0  0      0 480672      0  19296    0    0     0     0  101    3  0  1 99  0  0
```

Compared to the second time

```
[root@UWS ~]# vmstat 5 5 | awk '{print $4}'
free
484108
484108
484108
484108
484108
```

The difference is `-2436k`

3. *Is there any difference in the output and behaviour of such basic commands?*

```
[root@UWS ~]# ls -lah
total 24K
drwx------  4 root root  138 Nov 17  2020 .
drwxrwxrwx 20 root root  521 Nov  5 12:31 ..
-rw-------  1 root root   32 Nov  5 12:43 .bash_history
-r--r-----  1 root root 1004 Nov  5 12:32 .bashrc
drwxr-xr-x  3 root root   59 Nov  5 12:32 .local
drwxr-xr-x  2 root root   37 Nov  5 12:32 Desktop
[root@UWS ~]# cd .
[root@UWS ~]# cd Desktop/
[root@UWS Desktop]#
```

Nothing of note.

4. *What is the outcome of the command `init 0`?*

`init 0` shuts down the computer/

5. *How many different menu entries can you make out by studying the `/boot/grub/grub.cfg` file?*

```
[root@UWS ~]# cat /boot/grub/grub.cfg | grep menuentry
menuentry "Ubuntu, Linux 2.6.31-23-generic" {
menuentry "Ubuntu, Linux 2.6.31-23-generic (recovery mode)" {
```

Two. One for Ubuntu Linux, and its accompanying recovery mode.

6. *What is the image executable `vmlinuz` for the Linux kernel called? Please provide the absolute pathname.*

```
[root@UWS ~]# cat /boot/grub/grub.cfg | grep vm
        linux   /boot/vmlinuz console=hvc0 root=root rootfstype=9p
rootflags=trans=virtio ro
        linux   /boot/vmlinuz console=hvc0 root=root rootfstype=9p
rootflags=trans=virtio ro
[root@UWS ~]# file /boot/vmlinuz
"/boot/vmlinuz: Linux kernel x86 boot executable bzImage, version 4.12.0
(hecmargi@maximo) #10 Mon Oct 5 19:46:48 BST 2020, RO-rootFS, swap_dev 0x4,
Normal VGA"
```

7. *What is the size of the compressed kernel image?*

```
[root@UWS ~]# ls -lah /boot/vmlinuz*
-rw-rw-r-- 1 root root 5.0M Nov 19  2020 /boot/vmlinuz
```

`5.0M`

8. *What level is chosen to be the default runlevel?*

```
[root@UWS sbin]# ls -lah /etc/systemd/system/*.target
lrwxrwxrwx 1 root root 45 Nov 19  2020 /etc/systemd/system/default.target ->
../../../lib/systemd/system/multi-user.target
[root@UWS ~]# runlevel
N 3
[root@UWS ~]# ls -l /lib/systemd/system | grep runlevel3
lrwxrwxrwx 1 ... runlevel3.target -> ../../../../usr/lib/systemd/system/multi-
user.target
```

Calling the `runlevel` command immediately on reboot results in a returned equivalent run level of `3` which would indicate that `multi-user` is the default run level which is located at `/lib/systemd/system/multi-user.target`.

9. *What is the message displayed if you stop the networking daemon?*

```
[root@UWS ~]# systemctl stop systemd-networkd
Warning: Stopping systemd-networkd.service, but it can still be activated by:
  systemd-networkd.socket
```

10. *What is the output of the `ping localhost` command once you have stopped the networking daemon?*

`ping` executes normally.

```
[root@UWS ~]# ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.000 ms
^C
--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.000/0.000/0.000/0.000 ms
```

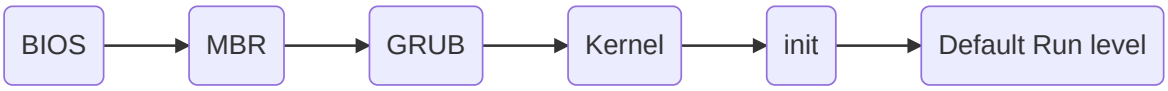
11. *What information was displayed after issuing the previous `systemctl` command?*

```
[root@UWS ~]# systemctl status systemd-networkd
● systemd-networkd.service - Network Service
   Loaded: loaded (/usr/lib/systemd/system/systemd-networkd.service; enabled;
vendor preset: enabled)
   Active: inactive (dead) since Fri 2021-11-05 14:19:11 UTC; 1min 30s ago
     Docs: man:systemd-networkd.service(8)
  Process: 91 ExecStart=/usr/lib/systemd/systemd-networkd (code=exited,
status=0/SUCCESS)
 Main `PID`: 91 (code=exited, status=0/SUCCESS)
   Status: "Shutting down..."

Nov 05 13:09:48 UWS systemd[1]: Starting Network Service...
Nov 05 13:09:51 UWS systemd-networkd[91]: Enumeration completed
Nov 05 13:09:51 UWS systemd-networkd[91]: request_name_destroy_callback n_ref=1
Nov 05 13:09:51 UWS systemd[1]: Started Network Service.
Nov 05 13:09:57 UWS systemd-networkd[91]: eth0: Gained carrier
Nov 05 14:19:10 UWS systemd[1]: Stopping Network Service...
Nov 05 14:19:11 UWS systemd[1]: Stopped Network Service.
```

Information pertaining to the status of the service, its process attributes, and any recent activity.

12. *Based on your knowledge of the boot process: Draw a diagram that shows the boot process from the very beginning (BIOS, LILO) to runlevel 2. Explain in your own words the complete boot process assuming you have to prepare the diagram for a potential client. Also investigate the standard System V runlevel settings and give some reasons why Debian mayhave deviated from this standard Research whether the standard Ubuntu, Redhat and Suse distributions adhere to the System V definitions of runlevels and note this in your answer.*



The question posits that the system will boot into runlevel 2 though this doesn't appear to be the default runlevel for the system (see Q.8).

The BIOS

- The system BIOS will perform a number of self-checks on the system and ensure overall integrity
- It then finds, loads, and executes a boot loader programme from the Master Boot Record
- The boot loader programme is given control of the system by the BIOS

Master Boot Record

- Located on the first sector of the drive
- MBR loads and executes the GRUB bootloader, or LILO if applicable

Grand Unified Bootloader

- Loads the system kernel into memory
    - generally the default system kernel
- GRUB allows more than one operating system to be loaded which the user can select

The Kernel

- the root filesystem is mounted
    - specified in `grub.conf`
- `init` is executed with a `PID` of 1
    - the kernel is initialised by `init`
    - `initrd` is a ram based filesystem used by the kernel until a full filesystem is mounted
    - this partition also allows for necessary drivers to be utilised allowing access to hardware, including drive partitions

init

- On `init` based systems, the `inittab` file is read and the run level is determined
- The default run level specifies the applications required in order to achieve that status
- Generally speaking, the default run level on servers is 3, and 5 for GUI environments

Default run level

- Services are executed, run by `init` in accordance with the requirements specified by the default run level
- Services are loaded in accordance with their sequence number
    - e.g. a process with a sequence number of 15 will run before another with a sequecne number of 35

13. *Stop and start again the `dropbear` service and verify that it is running. Check again the `PID` address and compare it against the previous one. Which `PID` value is higher?*

The previous value was 138, the new value is 158. The new value is higher.

14. *Why the `PID` assigned*

The question is unclear, but when a new process is generated the system adds the value of `1` to the current process count until it reaches the maximum `PID` value determined by `/proc/sys/kernel/pid_max`

15. *Is which situation the `PID` number assigned can be smaller?*

Once the `PID` value has reached the maximum the system will begin assigning `PID`s again from zero, unless there are `pid_max` number of concurrent processes running.

16. *Run the `pwd` command again and indicate the current directory.*

```
[student@UWS tmp]$ ssh localhost
Host 'localhost' is not in the trusted hosts file.
(ecdsa-sha2-nistp256 fingerprint sha1!! b7:2c:8e:ab:3e:ee:1d:12:6e:9f:03:da:b9:1
9:70:1d:31:af:51:46)
Do you want to continue connecting? (y/n) y
student@localhosts password:
[student@UWS ~]$ pwd
/home/student
```

17. *Explain why the current directory is not `tmp`*

We have logged into the remote session as `student` and, as such, have been placed into that user's home folder as normal.

Lab 08 - Networking

1. *Although in this virtual environment we only have on single machine "under the same network", what could be the reason to allocate a random MAC address in a virtual environment of more than one virtual machine? Consider what would happen if the same value was in use by more than one system.*

More than one MAC address would wreak havoc at layer two: neither of the virtual machines would be able to network effectively as a result of collision, packet loss. Duplicate MAC addresses are fine as long as they're separated by a router.

2. *What is the current IP-address given to the system and the MAC address allocated to the Virtual Box?*

```
[root@UWS ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.246.135  netmask 255.255.0.0  broadcast 10.5.255.255
        ether 02:0a:6d:c0:9c:47  txqueuelen 1000  (Ethernet)
        RX packets 3097  bytes 241199 (235.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 2295 (2.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2  bytes 140 (140.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 140 (140.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The IPV4 address is `10.5.246.135` and the MAC address is `02:0a:6d:c0:9c:47`.

3. *Summarize in your own words what the command `ifconfig` does.*

`ifconfig` is a tool for configuring network devices. Interfaces can be enabled and disabled by passing the `up` or `down` arguments respectively, and IP addresses can be assigned. With no arguments it displays the settings for each available ethernet adapter, with the `-a` argument it displays available and disabled adapters, and individual NICs can be accessed by passing their names as arguments, e.g. `ifconfig eth0`.

4. *How many individual systems can a subnet with netmask `255.255.240.0` accommodate?*

14

5. *What is the meaning of the double exclamation mark `!!` in Unix-like operating systems?*

`!!` repeats the last command.

6. *Explain the `ping` command in your own words. What do the summary entries min/avg/max/mdev represent?*

`ping` is a network diagnostic tool which allows the user to communicate with another computer by passing in the destination IP address. By default the programme sends an ICMP echo request message each second until the user interrupts operation. The min, avg, max, and mdev values represent the minimum, average, maximum, round trip time, mdev represents the standard deviation.

7. *Explain why in both cases, the summary output is very similar, for each of the min/avg/max and mdev values.*

The question is ambiguous. There exists no such environment variable as `IP_loc_machine`. Running `ping` against my own IP address both internal external yields no results.

```
[root@UWS ~]# IP_loc_machine
-sh: IP_loc_machine: command not found
[root@UWS ~]# ping 80.229.232.182
PING 80.229.232.182 (80.229.232.182) 56(84) bytes of data.
^C
--- 80.229.232.182 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2071ms
[student@UWS ~]$ ping 192.168.1.138
PING 192.168.1.138 (192.168.1.138) 56(84) bytes of data.
^C
--- 192.168.1.138 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13530ms
```

Further, there no such environment variable exists and there are no instructions to set an environment variable:

```
[root@UWS ~]# IP_loc_machine
-sh: IP_loc_machine: command not found
```

Below, the result when running `ping` against the local loopback and IPV4 address:

```
[root@UWS ~]# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.000 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.000 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2062ms
rtt min/avg/max/mdev = 0.000/0.000/0.000/0.000 ms
[root@UWS ~]# ping 10.5.67.79
PING 10.5.67.79 (10.5.67.79) 56(84) bytes of data.
64 bytes from 10.5.67.79: icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from 10.5.67.79: icmp_seq=2 ttl=64 time=0.000 ms
64 bytes from 10.5.67.79: icmp_seq=3 ttl=64 time=0.000 ms
^V^C
--- 10.5.67.79 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.000/0.000/0.000/0.000 ms
```

`127.0.0.1` represents a loopback address. Running `ping` against it is simply testing the NIC network stack, which is functionally identical to running `ping` against the local IP address of the device (in this case, `10.5.67.79`).

8. *Given that the loopback `127.0.0.1` is your `IP_loc_machine` kernel interface, why are both summary outputs not identical? Give a short explanation.*

The question makes no sense given the answer to question 7.

9. *Do you see an output? if not, review the network settings or restart the virtual machine.*

```
[root@UWS ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=168 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=175 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=168 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 168.000/170.333/175.000/3.316 ms
```

10. *How much longer are the times now for min/avg/max and mdev?*

The RTT when running `ping` against the local machine are zero. When running `ping` against the external IP `8.8.8.8` the times are `rtt min/avg/max/mdev = 168.000/170.333/175.000/3.316 ms`.

This question makes little sense in light of the answer to question 7. Please see Q7!

11. *Why are the times longer for the remote machine?*

Physics!

12. *Summarise the `route` command in your own words.*

The `route` command alows users to view and interact with the routing table.

13. *What are the Destinations, Gateways and Genmasks in the kernel?*

Destination represents the destination host or network, a gateway the address to access that destination, and genmask is the subnet mask of the destination network.

14. *How many packets have been received OK and transmitted error free by your loopback interface?*

```
[root@UWS ~]# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500     8022      0      0 0            66      0      0      0 BMRU
lo       65536       22      0      0 0            22      0      0      0 LRU
```

22

15. *Is your ethernet a point-to-point connection?*

```
[root@UWS ~]# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500     8022      0      0 0            66      0      0      0 BMRU
lo       65536       22      0      0 0            22      0      0      0 LRU
```

No.

16. *Are loopback and ethernet broadcasting?*

```
[root@UWS ~]# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500     8022      0      0 0           66      0      0      0 BMRU
lo       65536       22      0      0 0           22      0      0      0 LRU
```

A broadcast address has been set for `eth0`.

17. *How many packages have been dropped on your system by the ethernet interface?*

```
[root@UWS ~]# netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500     8022      0      0 0           66      0      0      0 BMRU
lo       65536       22      0      0 0           22      0      0      0 LRU
```

Zero.

18. *What are the local and Foreign addresses for the `ESTABLISHED` net-connection?*

```
[root@UWS ~]# netstat -na | head -2; netstat -na | grep 'ESTABLISHED'
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:53710         127.0.0.1:22            ESTABLISHED
tcp        0      0 127.0.0.1:22            127.0.0.1:53710         ESTABLISHED
```

19. *How many TCP (protocol based) connections are open at the moment?*

```
[root@UWS ~]# netstat --statistics
-- snip --
Tcp:
    1 active connection openings
    1 passive connection openings
    0 failed connection attempts
    0 connection resets received
    2 connections established
    747 segments received
    747 segments sent out
    0 segments retransmitted
    0 bad segments received
    0 resets sent
-- snip --
```

Two. One passive, one active.

20. *How many segments have been received and send via TCP/IP?*

```
[root@UWS ~]# netstat --statistics
-- snip --
Tcp:
    1 active connection openings
    1 passive connection openings
    0 failed connection attempts
    0 connection resets received
    2 connections established
```

```
    747 segments received
    747 segments sent out
    0 segments retransmitted
    0 bad segments received
    0 resets sent
-- snip --
```

747 received, 747 sent.

21. *Use your established knowledge to explain the command that you just issued on the remote system. What was the purpose of the semicolon?*

```
[root@UWS /]# cd / ; find /
-- snip --
[root@UWS ~]# netstat --statistics
-- snip --
Tcp:
    1 active connection openings
    1 passive connection openings
    0 failed connection attempts
    0 connection resets received
    2 connections established
    4548 segments received
    4548 segments sent out
    0 segments retransmitted
    0 bad segments received
    0 resets sent
-- snip --
```

Taking the question as wrote, the command `cd / ; find /` first performs `cd` with the root directory as the target, the semicolon separates the first command from the next command (which *will* be executed as `;` is not a conditional statement), and `find /` finds and displays the path of every file on the system.

22. *How many segments have been received and send via TCP/IP now? (I hope you see the difference.)*

```
[root@UWS /]# cd / ; find /
-- snip --
[root@UWS ~]# netstat --statistics
-- snip --
Tcp:
    1 active connection openings
    1 passive connection openings
    0 failed connection attempts
    0 connection resets received
    2 connections established
    4548 segments received
    4548 segments sent out
    0 segments retransmitted
    0 bad segments received
    0 resets sent
-- snip --
```

I do! 4,548 segments both received and sent out.

Lab 09 - Network File Systems

1. *Note the output line of the above command.*

```
root@UWS ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.239.36  netmask 255.255.0.0  broadcast 10.5.255.255
        ether 02:2f:be:6d:61:82  txqueuelen 1000  (Ethernet)
        RX packets 62  bytes 6157 (6.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 1372 (1.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2  bytes 140 (140.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 140 (140.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. *What permissions for user, group and world are given by the command `chmod 777` to the directory `/media/disk1`?*

```
[root@UWS disk1]# ls -lah /media | grep disk1
drwxrwxrwx    3 root     root        1.0K Nov 28 13:31 disk1
```

Full!

3. *What are the permission settings in octal representation for the newly created file `file.dat`*

```
[root@UWS disk1]# ls -lah /media/disk1/file.dat
-rw-r--r--    1 root     root          16 Nov 28 13:31 /media/disk1/file.dat
```

`644`

4. *Explain the settings, `rw` and `no_root_squash`. Use the man pages and the Internet to find out. Why might `no_root_squash` be a dangerous setting for a shared filesystem?*

`rw` allows for users to write to files and read from them. `no_root_squash` will allow the root users of remote machines to interact with the filesystem as though they had local root persmissions. `root_squash` should be used instead.

5. *Which port is used by `nfs`?*

```
[root@UWS disk1]# rpcinfo -p | head -1 ; rpcinfo -p $IP | grep 2049
   program vers proto   port  service
    100003    3   tcp   2049
    100003    3   udp   2049
```

Port `2049`, though the system provides no confirmation of this for some reason/

6. *Which different protocol types are supported by `nfs` in your system?*

TCP and UDP.

7. *Has the `nfs` service restarted?*

```
[root@UWS disk1]# /etc/init.d/S60nfs restart
Shutting down NFS mountd: OK
Shutting down NFS daemon: OK
Shutting down NFS services: OK
Stopping NFS statd: OK
Starting NFS statd: OK
Starting NFS services: OK
Starting NFS daemon: OK
Starting NFS mountd: OK
```

Yes. At least, it certainly looks that way.

8. *Does the `nfs` daemon connect to the same ports after restarting?*

Yes. Given that the default port configuration hasn't been reconfigured I see no reason why it wouldn't be the same.

9. *Besides `stop` and `start` what other input options does the shell-script `/etc/init.d/S60nfs` accept?*

`restart` and `reload`

10. *Note the contents of `file.dat` in your log-book*

```
[root@UWS imported_nfs]# cat file.dat
Welcome to socomp-09
```

11. *Do you have permission to edit the file `file.dat` yourself from the remote system?*

Yes, because I have root permissions and `no_root_squash` was configured earlier.

12. *In which system configuration file would you include a remote file system if you wanted it to be mounted at boot time?*

`/etc/fstab`

13. *Note the throughput of your Client/Server system for the nfs mount-options `rsize=1024` and `wsize=1024`, by performing the above command on your system and also noting the values for real, user and sys time*

```
[root@UWS mnt]# time dd if=/dev/zero of=/mnt/imported_nfs/zeroes.dat bs=1k
count=2k
2048+0 records in
2048+0 records out

real    0m1.997s
user    0m0.019s
sys     0m0.238s
```

This version of `dd` doesn't want to show throughput, but given a `real` time of about two seconds it looks ~1024kb/sec.

- real 0m1.997s
- user 0m0.019s

- sys 0m0.238s

14. *Why is the 'user' time so small? Please comment.*

`User` is the amount of CPU time in seconds used within the process outside the kernel, i.e. in user-mode code., not making system calls.

15. *Which activities will cause a non-zero system time?*

Any time a system call needs to be executed this will result in non-zero sysem time - whenever a process needs to spend CPU time in the kernel.

16. *Why do you think `rsize` and `wsize` are optional parameters for `nfs mount`, but when mounting an internal device, these parameters are not provided?*

`rsize` and `wsize` determine the size of the data chunks passed between a client and a server. These parameters are meaningless for any mount device other than one which exists across a network.

17. *Note the output of the od command. Explain, referring to the man pages, the meaning of the asterisk that appears in the 2nd line of the output.*

```
[root@UWS mnt]# od /mnt/imported_nfs/zeroes.dat
0000000 000000 000000 000000 000000 000000 000000 000000 000000
*
10000000
[root@UWS mnt]# hexdump /mnt/imported_nfs/zeroes.dat
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
0200000
```

The default output for `od` is as follows:

```
[root@UWS mnt]# od -A o -t oS -w16 /mnt/imported_nfs/zeroes.dat
0000000 000000 000000 000000 000000 000000 000000 000000 000000
*
10000000
```

Note that it doesn't contain the `-v` flag, `output-duplicates` which suppresses duplicate lines with an asterisk `*`.

```
[root@UWS mnt]# od -A o -t oS -w16 -v /mnt/imported_nfs/zeroes.dat
0000000 000000 000000 000000 000000 000000 000000 000000 000000
0000020 000000 000000 000000 000000 000000 000000 000000 000000
0000040 000000 000000 000000 000000 000000 000000 000000 000000
0000060 000000 000000 000000 000000 000000 000000 000000 000000
0000100 000000 000000 000000 000000 000000 000000 000000 000000
0000120 000000 000000 000000 000000 000000 000000 000000 000000
0000140 000000 000000 000000 000000 000000 000000 000000 000000
0000160 000000 000000 000000 000000 000000 000000 000000 000000
-- snip --
```

18. *Use `od -vc /mnt/imported_nfs/zeros.dat` as well and comment on the output.*

```
[root@UWS mnt]# od -vc /mnt/imported_nfs/zeroes.dat
0000000  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000060  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000100  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
```

The flags `-vc` force the printing of duplicate lines not suppressed by an asterisk, and each character is displayed with backslash escapes rendering them printable, escapable in some script.

19. *Comment on the behaviour of the real throughput time for increasing transfer block sizes, indicated by the sets of different `rsize` and `wsize` values. What is the optimal value for `rsize`, `wsize` for which the fastest real time throughput is achieved?*

`r/wsize =`	**1024**	**2048**	**4192**	**8192**	**16384**	**32768**
run: 1 (real)	1.978s	1.235s	0.779s	0.566s	0.461s	0.414s
run: 2 (real)	1.993s	1.155s	0.768s	0.561s	0.456s	0.408s
run: 3 (real)	2.043s	1.189s	0.767s	0.559s	0.453s	0.412s
avg:	2.005s	1.193s	0.771s	0.562s	0.46s	0.411s

The optimal size from the above table is `32768b` as it performs with the fastest average time.

20. *Comment on the fact, that the system time (sys) stays nearly constant*

The time it takes to execute the command is constant. The `rsize` and `wsize` has no bearing on execution time.