TASK - 1

Q NO1:

/proc/:

/proc/ directory is also known as running window on Linux Kernel because, it:

- tells which processes are running on Linux Kernel.
- the process IDs of processes running on Kernel.
- contains different directories and files.

Version:

Version is a file which contains details of Kernel;

* how Kernel was built
* who built the kernel
* which machine built the Kernel.
* compilation date and time of Kernel.
* compiler release.

/proc/cmdline:

* cmdline in /proc directory, is a file which contain commands which Linux Kernel pass to itself, in boot time.
* This automatic passing of commands are managed through boot loaders/boot managers.
* Boot managers could be of multiple types; GRUB, LILO, etc.
* Boot time is the time taken from switching on the system/device to the time it takes to become ready to operate.

/proc/uptime:

It contains two fields;

(1) the amount of time (in secs) in which the process is running

(2) the amount of time it spent being idle.

/proc/cpuinfo:

It gives information of CPU:

* the model of CPU.
* speed of CPU.
* family of CPU.
* cache of CPU.

/devices:

* It contains major number of characters and block special files.

* There is a numeric value against every file, which is related to the process / Linux Kernel.

These include:

1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ttyprintk
6 lp
7 vcs
10 misc
13 input
21 sg
29 fb
89 i2c

/proc/modules:

* This file contains a test list of modules which are loaded by Linux Kernel / system.
* Modules are junk of codes, which are executed by linux when required.

QNO2:

cmdline:

- It is a file which is present in the directory of almost all the processes running on Linux Kernel.
- It contains information of the commands which we pass to run a particular process.
  - It contains those arguments / commands as strings, which we pass whenever we run a file in bash shell.
    - If no command is passed, then it simply contains bash.

Environ:

- It is a file which exist in the directories of almost every process.
- It contains the environmental variables of processes.

also has a long listing containing different fields, which are separated by colons (:).

- It contains the environmental variables in fields which are needed by processes to run.

Limits:

- also exist in the directories of all processes.
- It contains the information of resources which the process can utilize.
- These resources include:
  - → number of files that can be opened when process runs.
  - → number of files that can be locked through that process (memory that can be utilized)
  - → It has fields; which contains soft limits, hard limits, and units.

/proc/cmdline:

* cmdline

stat:

- This file gives the information of the status of processes.
- It has multiple fields, such as:
  - → PID of that process.
  - → Name of process.
  - → State of process:
    - s = sleeping
    - r = running
    - z = zombie
    - t = stopped
    - x = grid
    - d = waiting for uninterrupt
- stat contains information in machine program readable format.

status:

- status file contains information of status of processes, in human readable format.

statm:

- It provides information about memory usage.
- It gives us different columns, which contain information of pages.

contains different numeric values;

1) size of program
2) resident size
3) group size, etc.

- These values are not usually accurate due to optimization of Kernel.

Q No3:

/proc/pid/fd:

- File descriptor directory contains different links, three of them are default/standard links.
- Files which we open in it, come as a link in it.

0 → stdin (linked with keyboard)

1 → stdout (linked with terminal)

2 → stderr (linked with terminal as well)

- If fd of the PID of bash is open, and we redirect its output, then it will basically be written at terminal.

/proc/pid/task:

- This file must present in the PIDs of every process.
- Task file is used for management of threads.
- It contains numeric files (tids) named after their pids.
- If the process in single threaded, it contains only one file/directory (tid).

  And, if the process is multi-threaded, it contains multiple files/directories (tids).
- Task directory must contain the numeric file (tid) of the PID of process in which we are currently in.
- TIDs contain all the content of PID except the task directory.

Usage in Linux System:

* fd directory is used for management of descriptors.
* Task directory is used for management of Threads.

Syscall:

NO4:

syscall:

In every PID of process, there is a file "syscall", which contains nine numeric values.

- First numeric value is the count of number of system calls made by the process durings its execution/running.
- A few numeric values refer to the arguments and registers.
- Some numeric values refer to the number of count of stack pointer and program counter.
- If a process is blocked, but its system call is not blocked, then '-1' will appear against first numeric value in 'syscall' file.
- File is present only if kernel was configured with CON_FIG_HAVE_ARCH_TRACEBOOK

TASK - 2

QNO1:

DAEMON:

Daemon is used for management of cron jobs which are running on your system. Individually, all cron jobs are daemon; they all have a separate task to perform.

To check daemons running on your system, command is:

```
ps -ajx
```

QNO2:

At and Batch commands are used for one time execution.

At command	Batch Command
★ It is used for execution of a job at a specific time.	★ When system load drops below 80, and the command that you've scheduled through batch command will execute.

Q3:

Cron and Anacron commands both execute recursively (again and again at a specific time).

Difference between

Cron	Anacron
1) Minimum Granularity Level	
1 min	1 day
2) Management and Scheduling	
managed and scheduled by regular user.	managed and scheduled by root user.
3) Expectations	
Cron expects that system is 24/7 up (running). It only execute jobs when system is always in running.	Anacron expects that system needs to be up (running). If system is down, it waits until it is up and then execute those jobs which were scheduled during down

QNO4:

sudo service --status -all | grep

QNO6:

crotab -e ⟶ to edit crotab file

30 18 * * 0 task/command

QNO7:

crotab -e ⟶ to edit crotab file

45 15 1 * * task/command

QNO8:

cron.allow and cron.deny in /etc:

* Both of these files tell us about the users which can execute/run the cron jobs.
* If you want to deny most of the users, so that they cannot run cron jobs then you must make "cron.allow"

and write name of the users whom you want to allow, and rest will be denied.

* If you want to allow most of the users, so they can run cron jobs, you must make "cron.deny" file and write names of users you want to deny, and the rest will be allowed.

* If both of these files do not exist, then only root will be able to run cron jobs.

* If both of these files exist, then cron.deny file will be ignored and those users which are written in cron.allow file will be able to run/execute cron jobs.

QNO9:

steps:

```
cd /home
sudo mkdir tmp
sudo touch ay.txt tty.txt ayt.tt
sudo crontab -e
```

2

* * * * * rm /home/tmp/a*.txt

QNo10:

*/10 9-17 * * * task/command

QNo5:

sudo sysv-rc-conf --level 3 5 cron on

↳ This command no longer exists.

TASK - 3

QNO1:

Binary Packages

★ It is collection of files bundled into a single file containing some:

i) executable file

ii) man pages for that package.

iii) copyright information

iv) configuration and install scripts.

↳ These scripts copy the executable and libraries and corresponding man pages to specific location.

Source Packages

★ It contains the source file, containing

i) source code file

ii) README & INSTALL

↳ two text files (info about how to install package)

iii) AUTHORS (contain info about authors)

iv) configure (script that checks our system for required software)

v) makefile.am & makefile.in

↳ which is used for compilation process of source file.

★ A binary package contains machine code, of the software (that we wish to install) for a specific platform, such as am64, i386, ia64, aem64, mips, powerpc, spare, etc.

★ A source package is eventually converted into a binary package for a specific platform.

QNO2:

PACKAGE MANAGER:

→ Package manager is a program which is used to install, remove, upgrade and manage package on UNIX based system.

Formats of Binary Package:

There are three important formats of binary package:

(i) ***.deb

(ii) ***.rp

(iii) ***.pkg

.deb:

- This is a format of binary packages with extension ".deb"

- This format is used by debian and distributions derived from debian like:
  - * Ubuntu
  - * kali-linux
  - * mint

- It is mostly used for installation of software on Ubuntu or kali linux.

- package managers for debian format are:
  1) apt
  2) dpkg

  } → command line interface

  3) aptitude
  4) synoptic

Q NO3:

$ sudo apt-get install cmatrix

Q No4:

Downloading Source package hello

```
~ $ sudo mkdir source
~ $ cd source
```

```
/home/rayan/source $ wget http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

```
/home/rayan/source $ tar xzf hello-2.10.tar.gz
~ $ cd hello
```

```
/home/rayan/source/hello-2.10 $ ./configure
```

```
/home/rayan/source/hello-2.10 $ ls
```

```
/home/rayan/source $ make
```

'make' command not found.

```
/home/rayan/source $ sudo apt install make
```

```
/home/rayan/source $ make
```

```
/home/rayan/source $ make install
```

```
/home/rayan/source $ hello
```

/home/rayan/source $ cat README

/home/rayan/source $ man hello

/home/rayan/source $ sudo make uninstall

Downloading Source Package asciiquarium

/home/rayan/source $ wget http://www.robobunny.com/project/asciiquarium/asciiquarium.tar.gz

/home/rayan/source $ tar xzf asciiquarium.tar.gz

/home/rayan/source $ ls

/home/rayan/source $cd asciiquarium_1.1/

/home/rayan/source $ ls

/home/rayan/source $ sudo cp asciiquarium /user/local/bin/

/home/rayan/source $ sudo chmod 0755 /usr/local/bin/asciiquarium

/home/rayan/source $ sudo apt install animation

/home/rayan/source $ asciiquarium

SCENERIO BASED TASK

TASK - 1

sudo crontab -e

→ server starts at 8:30 on every monday and tuesday:

```
30 8 * * 1,2 /home/rayan/myfolder/server.out
```

→ server closes at 9:55 on monday and tuesday:

```
55 9 * * 1,2 killall server.out
```

→ server again start at 2:10 on every monday and tuesday:

```
10 14 * * 1,2 /home/rayan/myfolder/
              server.out
```

→ server closes at 3:35 on every monday and tuesday:

```
35 15 * * 1,2 killall server.out
```