1. Create the following users, groups, and group memberships:

-A group named admin

-A user harry who belongs to admin as a secondary group

-A user natasha who also belongs to admin as a secondary group

-A user sarah who must have access to an interactive shell on the system, and who is not a member of admin

-harry, natasha, and sarah should all have the password of "redhat@123?"

# groupadd admin

# useradd -G admin harry

# useradd -G admin natasha

# useradd sarah

# passwd

2. Copy the ﬁle /etc/fstab to /var/tmp. Conﬁgure the permissions of /var/tmp/fstab so that:

-The ﬁle /var/tmp/fstab is owned by the root user.

-The ﬁle /var/tmp/fstab belongs to the group root.

-The ﬁle /var/tmp/fstab should not be executable by anyone.

-The other users are able to read and write /var/tmp/fstab.

# cp -rfv /etc/fstab /var/tmp/

# chmod a-x,o=rw /var/tmp/fstab

Ownership:

`

1) User owner:

# chown username file/dir

2) Group Owner:

```

# chgrp groupname file/dir

ACL: [ Access Control List]

`

# setfacl -m u:harry:rw- file/dir

-m : modify

# setfacl -m g:grp:rw- file/dir

# getfacl file/dir

**Process Management:**

**````**

Process:

``

- Any running program is a process

- Multiple instances of the same program are processes.

- Shell is also a process

Process ID (PID)

````

- Each linux process is identified by unique id PID

- Every process has a Parent Process ID (PPID)

+ Except "init"

- When a process is loaded into memory there is structure:

+ stack

+ heap

+ data segment

+ code segment

[+] Stack: Used for static memory allocation.

[+] Heap: Used for dynamic memory allocation.

[+] Data: Stores any static or global variables if defined.

[+] Code: Is the instructions of the program.

Create a Process:

``

Terminal#1

``

# sleep 100

Terminal#2

``

# ps -la [ Show the processes with PID & PPID ]

Process States:

```

- The process is built and executing , so now...

- We enter the Process State Machine

- States of Process:

[N] New When a new process is being created

[R] Running Instructions are being executed

[W] Waiting The process is waiting for some event to occur

[R] Ready The process is waiting to be assigned to a processor

[T] Terminated The processes has finished execution and is exiting

Managing Processes:

`````

- Linux kernel tracks what each process is doing

- Process is assigned a priority

- Address space assigned to the process

- Files is the process allowed to access

- Is the process a ?:-

+ Parent

+ Child

+ Zombie

=> Sometimes the parent dies first

````

- If the parent just exits or dies, the child process is left running

- The child's PPID is no longer valid due to parent is gone.

- Linux calls these children processes "Zombies"

- In Linux a zombie is just a process in which the children are adopted by the "init" process

- The init process will eventually cleanup the zombie childrens

PPID[kill]

\\_\_\_\_\_\_\_\_\_\_\_PID[kill]

PPID[die]

\\_\_\_\_\_\_\_\_\_\_\_PID[zombie]

Process Termination:

`````

- When a parent forks a child, they can finish in any order( parent first or child first)

- Sometimes the parent process could encounter and error and die

- Sometimes the parent process will just wait around until the child processes all complete brfore exiting

+ it calls a wait() command

- There are variety of wait command.