Lab 5: **The Linux Labs – Standard Linux Commands**

CSI4103 – Web Application Software Design

Faculty of Engineering – University of Ottawa

Objective:

Understand and practice how to use linux command line instructions. This is useful when working on server backends through a terminal window (e.g. using telnet or an SSH connection). This part of the lab continues on with a review of basic linux commands that are needed when manipulating processes on a linux system.

Instructions:

Filesystem Management

1. Login as a regular user (not root) in console #1 and "***su -***" to become root.
2. Switch to console #2 and login as a regular user in console #2 and "***su -***" to become root. Switch to console #3 and login as a regular user in console #3 and leave it as is for later.
3. Pay attention to which console you are asked to use to do any parts of the exercises. Mistakes will surely end up in errors and possibly cause problems along the way
4. To manage a filesystem, we need to know how it is set up. Commands to do this include:
    - *The* ***fdisk*** command to either create, change or list partitions on the system
    - ***mke2fs*** to format a newly created or old ext2 filesystem
    - ***mkbootdisk / mkfs / mkfs.ext2 / mkfs.minix / mkinitrd / mkpv / mkraid / mkswap*** – all specialized versions of ***mke2fs*** for specific application
    - ***mount*** to show what partitions are currently mounted.
    - Edit the ***/etc/fstab*** file to find out which partitions are to be mounted automatically and how
    - ***debugfs*** to allow fine examination and debugging of an ext2 filesystem (*use ONLY if you know what you are doing, as you may destroy information in the filesystem*)
    - ***dumpe2fs*** to view the current settings of an ext2 filesystem
    - ***tune2fs*** to change the current settings of an ext2 filesystem
5. Let's try an exercise. You should be fairly familiar with ***fdisk***, ***mke2fs*** and ***mount***, as well as the structure of the ***/etc/fstab file***, so we'll skip these. *IF you are not comfortable or familiar, it is suggested that you get acquainted with them before you begin this lab.*
6. Pointers:

Using ***mount –a*** will remount every file system listed in the ***/etc/fstab***, should you have made a change to it recently.

7. Let's try fine-tuning the file system that is already there.

Note: Follow the instructions. Some of what you are about to do can damage your system beyond repair if not done correctly. Keep in mind; most of the utilities that you are using will display more

than one screen of information, so you may have to filter out what you are looking for or execute the command again.

All of this next exercise is to be done in Console #1. Some of these commands take a while to execute.

> ***debugfs /dev/sda1*** *–*
> *debugfs 1.35 (28-Feb-2004)*
> *debugfs:*

`stats` – to view the current filesystem statistics

> Which directory is this filesystem currently mounted?_________________________
>
> When was the filesystem created? _______________________________________
>
> When was the filesystem last mounted? ___________________________________
>
> ***ls /*** – to list the root directory. Try other directories as well.
> What information is displayed? (Do not record the output, interpret the result)
>
> ____
>
> ____

`ncheck` – use the ncheck command with several of the large numbers displayed the ls command above. What does this command do? __________________________________

`imap` – use the imap command to display the information associated with several directories. What does this command do? ____

How many free inodes are available? __________

How many free blocks? __________

? – to view the commands available for debugfs and ***q –*** to exit debugfs
How could you use debugfs to recover a damaged filesystem?

____

____

____

dumpe2fs /dev/sda1 > ~/sda1info – keep this for future reference

dumpe2fs /dev/sda1 | less

Examine the information given here, especially in the first page of information. This is how your current filesystem is configured for this chosen partition.

- What's the maximum mount count value for this partition? _________

- What's the inode size for this partition? ___________________

- Which user is allowed to use the reserved blocks in the filesystem? ___________

- Which group is allowed to use the reserved blocks in the filesystem? ___________

  tune2fs –c 1 /dev/sda1 – this will set the maximum mount count to 1
  reboot – keep an eye on the boot process here

1. What happens when the system reboots and tries to mount ***/dev/sda1***?

   ____

Use ***tune2fs*** to set the maximum mount count back to the original value

Record the command used here: ____

How could you use dumpe2fs and tune2fs?

____

____

____

Process management

1. To manage processes on a system, you need to be able to both view and access those processes. Commands to do so include:

   - ***ps*** to list current running processes (parent & child) on the entire local system
   - ***kill*** to manipulate any process using it's process ID (PID)
   - ***killall*** to manipulate any process using it's process name
   - ***Ctrl-Z*** to stop execution of a process running in the foreground and push it to the background in a wait-state.
   - ***Ctrl-C*** to kill a process currently running in the foreground.
   - ***jobs*** to list current processes, with associated ***jobid***, started by this particular login or shell
   - ***bg %jobid*** to move ***jobid*** to the foreground
   - ***fg %jobid*** to start suspended ***jobid*** currently in the background

- ***top*** / ***gtop*** to view currently running processes and the resources associated to them (***gtop*** is the Gnome/KDE version of this tool)
- ***free*** – generates information about how much free memory is available on the system
- ***vmstat*** – generates information about how virtual memory is used
- ***procinfo*** - reports both memory and CPU usage along with some hardware related info

You won't necessarily be told which command to use, but try a variety of them to see if you can locate the best one to accomplish the tasks that are asked of you.

2. Using the utilities above, answer the questions below:

What is the name of the process that is currently using up the most memory? _______

What is the name of the process that is currently using up the most CPU? _______

How much memory is on your system and how much of it is used? ______ / ______

What is the PID of the `ntpd` daemon? ___

- Is it a parent process or a child process? _________
- If it is a child process, what's the parent process' name? _________

Which process would be affected if you typed in the command "***kill 1***"? What could you do before typing it, to make sure that it is not destructive?

____

What do you think would happen? ____

What does actually happen? ____

3. Now, let's examine process management. Follow these steps closely:

Console #1: ***cd ~***
Console #1: ***view .bash_profile*** (What command is view?)
Console #1: move the cursor down a few lines
Console #2: ***ps -ef | grep find***

- Does this list the process you started in console #1? _____

Console #1: ***Ctrl-Z*** (suspend the process)

- Does this list the process you started in console #1? _____

Console #1: ***find / -name *.conf***
Console #1: ***Ctrl-Z*** (suspend the process)
Console #1: ***jobs***

- How many processes are "stopped" on console #1? _____

Console #2: ***ps -ef | grep find***

- Does this list the process you started in console #1? _____

Console #2: ***jobs***

- Does this list the process you started in console #1? _____

Console #1: ***fg %1***
Console #1: Is the process still running ? _____
Console #1: Is your cursor in the same place? _____
Console #1: jobs

- How many jobs are still listed now? ____

Console #1: ***kill %2***
Console #1: ***ps –ef | grep find***

- Is the find process still running? ____

Console #1: ***man find &***
Console #1: ***bg %1*** - make sure you do this while the man page is loading

- Does the man page come up? ___

Console #1: ***fg %1***

- Does the man page come up? ___

4. Play around some more with interrupting, killing and restarting processes. This is one of the few critical skills needed to effectively manage a Linux server or system.

Job Scheduling

1. In this section, you will be creating and modifying ***crontab***, ***at*** and ***nohup*** jobs to observe how each scheduling system works. This is a critical skillset in a Linux administrative position, since these mechanisms allow for automating a very large portion of daily/weekly routine workload.
2. Before you go on, however, you'll need to verify that your system date & time is set properly. This is important as you are going to be trying to use timed processes, and if the date and time are not what you expect them to be, you may not get the desired results.

   Use the ***date*** command to view the current system date & time
   Use the command ***date -s="time month day"*** to set the date and/or time
   Check the ***man*** pages for the ***date*** command to see other options available
3. Once this is done, you're ready to go on.
4. Let's start with the ***crontab*** job scheduling system. This command is used to schedule jobs that will run periodically, using a schedule based on minute / hour / day-of-the-month / month / day-of-the-week. Jobs submitted using ***crontab*** will run at the scheduled time until they are removed from the cron queue.

   Switch back to Console #1, if you aren't already there. You'll need to make sure that the cron daemon, `crond`, is running:

   ps -ef | grep crond

   If you are not already superuser

   su -

   Now let's look at the queued cron jobs for root:

crontab command

crontab -u root -l

- Does ***root*** have any ***cron*** jobs currently in the queue? ________

Let's create a cron job for root:

cd ~
vi root.cron

Let's put in a reminder of the structure, since this is not something you will use very often:

# min hour dom month dow command

Now, create a couple of cron jobs that will run a command in about 5 minutes on console #2 and 10 minutes on Console #1 - ***i.e.*** if today if Friday, and it is 9:00 am, you want to run the job on console #2 at 9:05 am and the job on console #1 at 9:10 am

5 9 * 11 5 echo "Hey, it's morning!" > /dev/tty2
10 9 * 11 5 echo "Hey, it's Friday!" > /dev/tty1
:wq

Now, let's submit the job

crontab -u root root.cron
crontab -u user -l

Switch to Console #2 and wait for a few minutes.

What happens at the end of the allocated time on Console #2? ____

Does the first message show up on Console #1? ______

Does the second message eventually show up in Console #1? _____

Does the second message show up in Console #2? _____

Now, since you don't want these jobs to run every day at this time, you want to remove the cron jobs. There's 2 ways of doing this: the first is to remove ALL cron jobs at the same time; the second requires you edit the cron file and resubmit it to modify the cron jobs in queue.

crontab -u root -r
crontab -u root -l

- Did it remove all the cron jobs in queue for root? _______

crontab -u root root.cron
crontab -u root -l

You should now have the cron jobs in queue again. So let's modify it the second way:

vi cron.root

Delete both lines below the commented structure line

:wq
crontab -u root root.cron
crontab -u root -l

- Did it remove all the cron jobs in the queue for root? _______

Take the time to think about a process you would like to run periodically on your system and insert it into the ***root.cron*** file and submit it to the cron system. Observe what happens over time with this job. And example of a command might be something like:

0 4 * * 1,3,5 find / -name core | xargs rm -f {}

This command will delete any file called ***core*** (generated by a core dump of any kernel or application problem) that can be found on the system at 4 am every Monday, Wednesday and Friday of the year.

at Command

1. Now let's look at the ***at*** command. The ***at*** command is designed to run a job at a specific date & time, but it will do so once and only once. In other words, once the job has been run at the specified date/time, it will not run ever again. So, ***at*** is used for single run jobs that need to be run at specified date/time, but that you want to be able run without you having to be at the console, or even logged into the system.

   The ***at*** command will allow you to specify what commands you want to run and when, but it does require a slightly different approach to do it. The at command expects information formatted as:

   at 23:22 Nov 17 -f filename-to-run

   The ***at*** command can either be pointed to a file, which contains all the commands you wish to ***execute***, or the list of commands can be given to it one by one. Using a file does have the advantage of being able to run the job again with little hassle, but this is a rare instance. The more common approach is a single instance set of commands that need to be run.

   To accomplish this, you need only specify the time & date when you wish the set of commands to be run, and the ***at*** command will then let you give it each command you want to run. Once you have typed in all the command, a ***Ctlr-D*** tells the ***at*** command you are done and to submit the job to the cron daemon. It will then store them and execute them in the exact order you specified them when the appropriate time & date occur.

   at [*current time+5 mins in hh:min format****] Nov [****today's date****]***
   > echo "Hey ! this works!"
   > ls -la ~
   > (*Ctrl-D****)***
   warning: commands will be executed using /bin/sh
   job 2 at 2000-11-17 9:25

   Wait for a few minutes and you should see the results of the command. The at command also allows you to manage jobs in the queue.

```
at [current time+1 hr in hh:min format] Nov [today's date]
> echo "Hey ! this works!"
> ls -la ~
> (Ctrl-D)
warning: commands will be executed using /bin/sh
job 2 at 2000-11-17 10:27
```

To list the current at jobs in the queue

```
atq
2        2000-11-17     9:25     a
```

To remove a job in the queue

```
atrm 2
atq
```

Now, if you are not at the console or logged in, results of the at job will not be sent to you. To do this, you would have to use the ***-m*** option for the ***at*** command, which sends you a mail stating the ***at*** job was completed, whether it executed properly or not.

If you want, try scheduling an ***at*** job, using the ***-m*** option, to run shortly, then log out, wait until the expected time for the job, and log back it to see if it indeed ran.

Alternately, you can schedule an ***at*** job to run over the next few days or prior to the next lab session, using the ***-m*** option, and see if you get a mail from the cron system about the job.

nohup Command

1. Lastly, you want to explore the ***nohup*** command. This command simply allows for running a command with all hangup signals ignored, allowing the command to continue to run in the background after you log out.

   The structure for the nohup command is quite simple:

nohup command args &

The ***&*** is critical to ensure the job gets put into the background.

Should there be any results that would have been sent to the screen while logged in, they will be sent to a file called ***hohup.out*** in the user's directory.

Try it out. Switch to Console #2

```
nohup find / -name *.doc &
exit (until logged out of Console #2)
```

wait a few minutes then log back into console #2

- Is there aa nohup.out file? _______
- What's in it? ______

Take the time to experiment on your own with commands you could run with nohup while not logged into the console and see what happens for yourself.

Run Level / Daemon Management

1. To configure your system to ensure that the required processes and capabilities are activated when a person log's into the system using any particular ***init*** runlevel, you need to configure the system to understand which processes should run at those particular ***init*** runlevels. The commands to help do so include:

   - ***ntsysv*** to manage which processes/daemon are to be run for a specific init runlevel
   - ***tksysv*** the Gnome/KDE version of ***ntsysv***, with some added beeps-n-whistles
   - ***chkconfig***, which emulates the ntsysv and tksysv capabilities but from the command line script files in ***/etc/rc.d/init.d*** to run, start, stop, restart and others for each system process/daemon available in every ***init*** runlevel (path may be different in other Linux implementations)

2. Let's try a simple example of how this works:

   ps –aux | grep lpd – verify that the ***lpd*** service is running
   ntsysv 3 – edit services running for init runlevel 3
   Scroll down and turn off ***lpd*** (spacebar to toggle * off)
   Scroll down and turn on ***smb*** (spacebar to toggle * on)
   Use [***Tab***] to go down and select '***Ok***'
   shutdown –r now – reboot the system

   - Is ***lpd*** running? ____
   - Is ***smb*** running? ____

Reverse the process above (i.e. add ***lpd*** and remove ***smb*** to init runlevel 3) before shutting down your computer.

3. Take the time to play with ***ntsysv***, ***tksysv*** and ***chkconfig***. These tools are invaluable in managing the different services and daemons that are supposed to run, or not, in each runlevel for Linux.

The Linux Automounter

1. There is also another tool of interest within Linux, called the "***automounter***". This particular tool allows for on-demand mounting of removable media or network filesystems. It can be configured to mount any available removable media or network filesystem when and only when the user has need of it, and release the mount once it is no longer of use. This helps to minimize the amount of memory and kernel processing time required for filesystems that may or may not be immediately useful.

There are 2 Linux autmounters:

- ***amd*** - is user level and requires no kernel support
- ***autofs*** – requires kernel support, but it simpler to administer

2. Take the time to examine and play with the ***autofs*** system and its files.

If you do not have it installed (use ***ls /etc/rc.d/init.d/autofs*** to verify), then you need to install the RPM file.

Test to see if it really does automount a floppy and/or a CD-ROM or a USB key memory device on demand from the console (not X Windows). Ensure that it's loaded in your particular init runlevel for it to work. You may have to restart the system for it to work, after you make any changes.

3. The ***autofs*** automounter, the preferred mechanism in Linux, needs to be loaded in the specific init runlevels you want it to be available, otherwise it will not work. It uses several files:

/etc/auto.master – overall automounter configuration, which contains entries like:

Mount Point	**Map**	**Options**
/misc	/etc/auto.[name]	--timeout 60

where **Mount Point** is the point where the autofs file system associated with **Map** will be mounted on the local file system.

/etc/auto.[name] – the mount point map file, with entries like:

Key	**Mount Options**	**Location**
cd	-fstype=iso9660,ro	:dev/cdrom
floppy	-fstype=auto	:dev/fd0

/etc/rc.d/init.d/autofs – the actual start/stop/status script file

4. Read the ***man autofs*** for more information on how to create an automount file system for removable media or network file systems.