CST8177 - Lab #6

Student Name:	**Student Number:**	**Lab section:**

Process Management

Objectives

- To use process management tools
- To learn how to use job scheduling tools

Lab Outcome

- An understanding of how to manage running processes

In-Lab Demo: List the steps required to submit a weekly system job.

Section A - Process management

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

Process management commands

`ps`	to list existing processes on the local system
`jobs`	to list existing background jobs
`fg` and **`bg`**	to move jobs between the background and foreground
`kill`	to send a signal to a process using the PID or job number
`killall`	to send a signal to all processes using the process name
`Ctrl-C`	to kill a process currently running in the foreground (also **`^C`**)
`top`	to view currently running processes and the resources associated to them (use **`q`** to quit, **`h`** for help)
`pgrep`	look up processes, based on the name or on other attributes
`pstree`	to display process relationships (hierarchy)

Resource management commands

`free`	to generate information about how much free memory is available on the system
`vmstat`	reports information about processes, memory, paging, block IO, traps, and cpu activity.

Try a variety of the commands in the questions below to locate the best one for accomplishing the tasks requested.

Exercise #1: Using process management utilities

Using the utilities above, or other suitable commands, answer the questions below:

- What is the PID of the **`rsyslogd`** daemon?
  Record the most suitable utility to use: ____________________ __________________

- Is **`rsyslogd`** a parent process or a child process?
  Record the most suitable utility to use: ______________________ _________________

- If **`rsyslogd`** is a child process, what's the parent process's name/PID?
  Record the most suitable utility to use: _______________________ ________________

- What is the name of the process that is currently using up the highest **`%MEM`**?
  Record the most suitable utility to use: ________________________ ________________

- Assume that the PID of **`foobar`** is **`99`**. Further assume that the command line **`kill 99`** has no effect (what is **`kill`**'s default signal?). To kill the process foobar you have to use the following command line: _____________________ _________________________________

- Write the key combination that kills a process that runs in the foreground. ____

Exercise #2: Working with job control

- In this exercise you are working on more than one terminal window.
  Note: To determine the device name of your terminal, use the **`tty`** command.

Job control commands

`jobs`	to list current processes started by this particular shell
`CTRL-Z` or **`^Z`**	to stop execution of a process running in the foreground
`bg %`*`jobnum`*	to move a suspended job into the background
`fg %`*`jobnum`*	to move a background job to the foreground
`kill %`*`jobnum`*	to terminate a background job

At terminal #1

- **`cat`**

  starts a copy of **`cat`** reading from **`stdin`**
- **`CTRL-Z`**

  suspend the **`cat`** process while it is running
- **`find / -name '*.conf' >& /dev/null`**

  starts a long-running **`find`**
- **`CTRL-Z`**

  as above, but do this fairly quickly, since it's not that long-running
- **`jobs`**

  List all the background jobs for this terminal or **`tty`** device

At terminal #2

- **`ps -ef | grep find`**

  Does this list the process you started in terminal #1? Determine the process state of find: ____________________ ____________________

- **`pgrep find`**

  compare with the **`ps`/`grep`** combination above; which do you prefer and why? ____

  ____

- **`jobs`**

  Does this list the processes you stopped in terminal #1? _________________________

At terminal #1

- **`kill %1`**

  terminate the **`cat`** process. Press ENTER and describe what happens:

  ____

- **`bg %2`**

  check job **`%2`**; Is it still **`find`**? ____

- **`fg %2`**

  Is the **`find`** process still running? ____

- **`cat &`**

  run the **`cat`** again, starting it directly in the background

- **`jobs`**

  How many jobs are listed? With what job number(s)? _____________ _____________

- Terminate all remaining jobs with a single command:

  ____

Section B - Job Scheduling

In this section, you create and modify **`crontab`** entries to observe how the normal scheduling system works. You will find some good information about the format and content of the **`crontab`** in **`man 5 crontab`**., and information about the **`crontab`** command in **`man 1 crontab`** (yes, they are different). Be sure to check anacrontab(5) and anacrontab(8) as well.

Supporting material: date and time settings

Before working with a scheduling system, we need to know how to view and modify the system date and time. To check the system date and time maintained by the kernel, use the **`date`** command. With no arguments, it displays the current values. You can also modify the date/ time as root

Note: The date/time argument can be presented as either a string or as a numeric argument. See **man date** for the complete details.

- Examples using a string To change the date and time using a string argument use the **date** command with the **-s** option.
  - *Example #1*: set the time using the current date

    ```
    date -s "12:00"
    ```
  - *Example #2*: set the date using the current year

    ```
    date -s "Feb 16"
    ```
  - *Example #3*: set the complete date and time

    ```
    date -s "Feb 16 2004 12:00:59"
    ```
- Examples using a numeric argument using the standard format **mmddhhmm[[cc]yy][.ss]** without **-s**:
  - *Example #4*: set the date and time using the current year

    ```
    date 02161200
    ```
  - *Example #5*: no century number (bad) followed by setting the date with century (good) and time

    ```
    date 0216120004
    date 021612002004
    ```
  - *Example #6*: set the complete date and time, with seconds

    ```
    date 021612002004.45
    ```

Note: To check the date and time that is maintained by the hardware use the **hwclock** command, but use it carefully on a real (not virtual) machine.

Exercise #1: Cron scheduling

Do this exercise first as a normal user and again as **root**.

The **crond** daemon is used to schedule jobs that run periodically, using a schedule based on minute / hour / day-of-month (dom) / month / day-of-week (dow). Jobs submitted usingthe system **crontab** file in **/etc** will run at the scheduled time until they are removed from the **cron** queue.

Verify that the **cron** daemon, **crond**, is running and show the command used: ______________ ________________________________

Analyzing a **cron** job

```
# min hour dom month dow [userid] command
0,*/15 8-20  24,25 12 * root echo "From crontab" > /root/out
```

What does this **cron** job do? Change the date/time in order to execute it now.

____

____

Creating / submitting a **`cron`** job as a user

The **`crontab`** utility is used to allow users to submit jobs to **`cron`**. The **`crontab`** utility can perform a number of actions, such as:

`-l`	list current **`cron`** job file
`-e`	edit/create **`cron`** job file
`-r`	remove **`cron`** job file

See **`man 1 crontab`** for the switches to perform **`crontab`** command actions. For more details on **`cron`** syntax, see **`man 5 crontab`**.

Creating/Modifying a **`cron`** job

- Create a **`cron`** job with the **`crontab`** command. You are now in a specialized **`vi`** (see? I told you!) set to edit only your userid's own **`crontab`** file.

  Note: The **`su`** command will cause the **`crontab`** command to operate on the **`crontab`** file for root. If root, specify the user name in the command, as:
  `crontab -u` ***userid*** **`-e`**.

- Create a **`cron`** job, based on the example above. (you must follow **`cron`** syntax) to create a file in your **`~/cron`** directory (which you must first create). Make your time delay only a minute or two for the current day.

  Tip: Put in a comment with the **`crontab`** syntax as your first entry :
  ```
  # min hour dom month dow command
  ```

- After leaving the **`crontab`** command, verify that you now have a **`crontab`** file with your input (as root, look at **`/var/spool/cron/`*****`userid`***; now **`cat`** the file and verify the content. Is that what you put in?). ________________________________

- View the result of the entry executing. Did it work? ___________________________

  Note: You don't have to restart the service for the new configuration to take effect. The **`crond`** daemon checks each of its directories every minute)

Removing a **`cron`** job

- Now remove the **`cron`** job(s).
- Verify that the job(s) have been removed.
  Is it gone from **`/var/spool/cron/`**? ____

`Crontab` access

You should experiment with the files **`/etc/cron.allow`** and **`/etc/cron.deny`** to control who can submit **`cron`** jobs using the **`crontab`** command.

The format of both access control files is one line per user, specifying the userid. The **`cron`** control files are read each time a user tries to create/modify a **`cron`** job. If the file **`cron.allow`** exists, only users listed in it are allowed to use **`cron`**, and the **`cron.deny file`** is ignored. If **`cron.allow`** does not exist, all users listed in **`cron.deny`** are not allowed to use **`cron`**. *Note*: The root user can always use **`cron`**.

Exercise #2: Viewing the system `cron` jobs

- View the **`cron`** configuration file that lists the **`cron`** jobs used for system maintenance with the anacron daemon: **`less /etc/anacrontab.`**

  Historically the file **`/etc/crontab`** contained configuration entries like those we've seen above which called the **`run-parts`** command on all the files in the **`cron.{daily,weekly,monthly}`** directories. These jobs are now run indirectly through **`anacron`** to prevent conflicts between **`cron`** and **`anacron`**. See **`man 5 anacron`** and **`man 8 anacron`** on how to managethe job execution.

- Describe the purpose of any one of the **`anacrontab`** entries.

  Note: The utility **`/usr/bin/run-parts`** is a shell script that takes a directory name as an argument. Its purpose is to execute every executable file that is located in the given directory. Therefore, no extra entry needs to be added into the system files.

- Describe your own **`crontab`** entries (both user and root), whether each was successful or not, and why.

Section C - Service management

Supporting commands

`runlevel`	display the previous and current runlevels
`telinit` ***`n`***,	switch to runlevel ***`n`***
`uname`	display basic system information
`chkconfig`	manage runlevel services with the following options
`--list [`*`service`*`]`	list the state of one or all services in all runlevels; a state can be **`on`** or **`off`**
`--level` ***`n service`*** **`on`**	change the state of a runlevel service **`on`** in the specified runlevel
`--level` ***`n service`*** **`off`**	change the state of a runlevel service **`off`** in the specified runlevel
`--add` ***`service`***	add the service to the runlevels based on the defaults
`--del` ***`service`***	remove the service from all runlevels

Exercise #1: Viewing a runlevel service

- List all runlevels where the **`crond`** service runs: ________________________________

  Record the command line: ____

- Record the name of the **`crond`** startup script (show the <u>absolute</u> path) used in the directory for runlevel 3 (**`/etc/rc.d/rc3.d/`** for runlevel 3): ____
  *Tip: use **`ls`** with **`grep`***

  Record the command line: ____

- Record the name of the **`crond`** startup script (show the <u>absolute</u> path) that is executed when the service is activated: ____

  Record the command line: ____

Exercise #2: Turning a service on & off

- Deactivate the **`crond`** service in runlevel 3.

  Record the command line: ____

- Verify that the changes you made are in effect.

  Record the command line: ____

- Reactivate the **`crond`** service in runlevel 3.

  Record the command line: ____