CST8177 - Lab #7

Student Name	Student Number	Lab section

Service Management and System Logging

Objectives

- To learn more about how to manage the system logs
- See how **crond** and **rsyslogd** interoperate.
- Begin to see how system services work together (daemon, config file, client, system logs). All the services you will learn about next semester follow this model.

In-Lab Demo - Display and explain all the entries in the **rsyslog** config file.

Preparation

Modify **/boot/grub/grub.conf** (Note: first, copy **grub.conf** to **grub.conf.backup** Just In Case) to add two stanzas matching your current one. To the end of the **kernel** statement for one of the copies add **3**; at the end of to the other, add **single** (or **s**, or **1**). Adjust the **title** statement for each stanza to be more descriptive (it's just text). You may also have to remove the statement **hiddenmenu** to permit the display of the menu.

Boot into runlevel 3 and log in as **root** to test your new **grub.conf**. Next, test single-user mode to check it as well. Return to runlevel 5.

Supporting commands:

- **runlevel** - To display the previous and current runlevel
- **telinit n**, where **n** is a runlevel number - To switch runlevel
- **pgrep** and **ps** with **grep** – To search for a running process
- **chkconfig** - To manage runlevel services
  - **--list [service]** - To list the state of one or all services in all runlevels; a state can be **on** or **off**
  - **--level n service on** - To change the state of a runlevel service to **on** in the specified runlevel
  - **--level n service off** - To change the state of a runlevel service to **off** in the specified runlevel
  - **--add service** - To add the service to the runlevels based on default settings from the runlevel scripts that exist
  - **--del service** - To remove the service from all runlevels
- **uname** - To display basic system information
- **service** – To manage system service daemons

Exercise #1: Identify system architecture

- Identify the kernel version using **`uname -a`**

  _____

- Compare it with the kernel file name in the **`/boot`** directory.

  _____

- Are they the same version? **[Y / N]**

Exercise #2: Identify runlevel for the **`atd`** daemon

- What is the current runlevel? How did you find it?

  _____

- Should **atd** be running at this level? **[Y / N]**

- Is it? **[Y / N]**

- How did you determine these things?

  _____

  _____

- Stop the **`atd`** daemon and then start it up again.

  _____

  _____

Section A – The system logging daemon: rsyslogd

Linux and UNIX systems have a bunch of processes that are not attached to any terminal, and when they have errors to report they don't have a console to dump them to. In the very early days of UNIX development, each program would dump out errors into its own file. As you can imagine, this got to be unmanageable, as the files would be found (or rather, be hidden) all over the place.

A service was created to handle error logging: the service is called **rsyslog** (originally just **syslog**). The daemon is called **rsyslogd**, which uses the configuration file **/etc/rsyslog.conf**.

Exercise #1: Working with the **rsyslog** service

Note: The utility **/sbin/rsyslogd** is the daemon; the service started during startup is called **rsyslog**. In this lab, **rsyslog** always refers to the service and **rsyslogd** to the daemon. However, the whole thing is often just called syslog.

Runlevel configuration

Is the **rsyslog** service currently running? **[Y / N]**

Is the **rsyslogd** daemon currently running? **[Y / N]**

In which runlevels is the **rsyslog** service started?

Disable the **rsyslog** service in runlevel 3 using the **chkconfig** command.

Is the **rsyslog** service still running? **[Y / N]**

Is the **rsyslogd** daemon still running? **[Y / N]**

Enable the **rsyslog** service in runlevel 3 using the **chkconfig** command.

Record the name of the **rsyslog** link (use the absolute path) that is used in the runlevel directory of runlevel 3:

Record the name of the **rsyslog** script (use the absolute path) that is executed when the service is activated:

Managing a service

The **service** command allows you to start, restart, stop, and get the status of a service, its daemon. It is usually a short script that facilitates the task of managing a service. The syntax of the command is:

service service_name start|restart|stop|status

Record the status of the **rsyslog** service using the **service** command:

Record the command line you use:

What is the pid (only) of your **rsyslogd** process? How did you find it?

Show the **service** command to stop the **rsyslog** service, and then stop it:

Record the status of the **rsyslog** service and the command used:

Show the command and the pid (only) of your **rsyslogd** process (daemon):

Show the **service** command to start the **rsyslog** service, and start it:

_____

Record the status of the **rsyslog** service and the command used:

_____

Show the command and the pid (only) of your **rsyslogd** process (daemon):

_____

If your **rsyslogd** pid has changed during the steps above, briefly explain why:

_____

_____

_____

Section B – Log files

Exercise #1: Viewing log files

To view the log file you can use the **less** command or, if the log file is very long and you are only interested in the most recent log data, the **tail** command. The **tail** command defaults to displaying the last 10 lines of a file (just as head defaults to the first 10) but you can modify this behaviour by providing the number of lines as an option to the command.

Example #1: view the whole log file in pages

```
less /var/log/maillog
```

Example #2: view the last 10 lines of the log file

```
tail /var/log/maillog
```

Example #3: view the last 20 lines of the log file

```
tail -20 /var/log/maillog
```

Example #4: follow (**-f**) the tail of the log file as lines are added (**^C** to exit when done)

```
tail -f /var/log/maillog
```

View the log files below and note the type of information recorded. Note: If your log files do not have any content, check your rotated log files. They will have the same base name modified by the date of rotation or a sequence number.

Log rotation is necessary since each log file grows over time and will soon become an inconvenient size for viewing or worse, fill the disk. When each log file qualifies, that log file is renamed to an archive name and a new log file created. When there are more than some limit of archived log files (there's a set of rules to look at), the oldest is deleted.

- **/var/log/secure**
- **/var/log/messages**
- **/var/log/dmesg**

Note: Execute the command **dmesg**. Do you notice a difference between the content of the log file and the output of the command? You should.

The utility **dmesg** retrieves the contents that are currently stored in the kernel ring-buffer, while the log file is created at startup and is only a record of the last startup process.

Exercise #2: Creating and analyzing log data

In this exercise, create a new account and view the resulting log entries.

- Record enough of the last line in **`/var/log/secure`** to identify it

  _____

- Create a new user account (skip the **`/home`** set up)

  _____

- Set a password for the new user.

  _____

- View the log file **`/var/log/secure`** and record the new log entries.

  _____

  _____

  _____

- Remove the new user

  _____