In this lesson we will Learn

- Why we may want to use logging in your shell script

- Learn about syslog standards and how to generate messages that confirm to that standard

- we will also learn to create our own custom logging functions.

- If you wan to keep a record of what occurred during execution of script. We need to implement some sort of logging mechanism.

- Logs can store any type of information you want, the typically answer Who, what, When, Where and why something occurred.

- Logs can be useful when shell script is performing several actions it produces a lot of output that might scroll off your screen.

- Also, if you want to run your script unattended using a cron or some other means. You might want a way to look back and see exactly what happened and when it happened during the previous run.

- The Linux operating system uses the syslog standard for mass logging

- This allows programs and applications to generate messaged that can be captured, processed and stored by system logger.

- It eliminates the need for each and every application having to implement logging mechanism.

- This means we can take advantage of this logging system in our shell scripts.

- Before we start using logging lets briefly talk about how it works.

* The syslog standard used facilities and severities to categorize messages. Each message is labelled with a facility code and severity level.
* The various combination of facilities and severities can be used to determine how to handle a message.
  + **Facility:** Facilities are used to indicate what type of program or what part of the system the message is originated from.
    - **E.g.:**
      * Messages that are labelled with the keen facility originates from the Linux Kernel
      * Messages that are labelled with mail facility come from applications involved in mail handling.
  + There are various facilities, if your script is involved in handling mail we can use the mail facility for logging
  + **Facilities:**  kern, user, mail, daemon, auth, local0, local7
  + If its not clear what facility to use, we can simply use the user facility.
  + Also, the facilities ranging from local0 to local7 are to be used to create custom logs.
  + These facilities would be appropriate for custom written shell scripts.
  + **Severities:** emergency, alert, critical, error, warning, notice, info and debug.
  + The most severe message is an emergency message.

- These combinations of facilities and severities are used by system logger to handle these messages.

- Most messages are simply written to a file.

- Each distribution uses a slightly different ser of defaults and these logging rules are configurable and can be changed.

- We can find many messages stored in /var/log/messages on some distributions while other use /var/log/syslog

- For example: you will have to consult the documentation for the system logger that is in use on our system.

- Its typically one of syslogd, rsyslog, or syslog-ng. Although there are several other possibilities.

Logging with logger

By default it created user.notice message

$ logger “message”

$ logger -p local0.info

- -p: facility option

- local0: facility name

- .info: severity

$ logger -t myscript -p local0.info “Message”

* -t: tag

$ logger -I -t myscript “Message”

* -i: PID in message

$ logger -s -p local0.info “Message”

* -s: Display the message of log on screen

$ logger -i -t myscript “Message”

Aug 2 01:22:53 linuxsvr myscript[12986]: Message

We can also create a function in our script for logging.

Logit () {

Local LOG\_LEVEL=$1

Shift

MSG=$@

TIMESTAMP=$(date +”%Y-%m-%d %T”)

If [$LOG\_LEVEL = ‘ERROR’ ] || $VERBOSE

then

echo “${TIMESTAMP} ${HOST} ${PROGRAM\_NAME} [${PID}]: ${LOG\_LEVEL} ${MSG}”

fi

}

This function works like a log level be passed into it followed by a message.

It assigns the first thing passed into it is LOG\_LEVEL

Shift: It is used to shift the positional parameters to the left

This means the special variable $@ contains everything but the first positional parameters which we already used for our LOG\_LEVEL variable.