# CST 8177 - Lab 9

## Automating Tasks: `lnuf`

| Student Name | Student number | Lab section |
|---|---|---|
| | | |

## Objectives

1. To learn and understand basic **bash** scripting language
2. To automate tasks
3. To develop basic PDL

## Lab Outcome

A good working understanding of how shell scripting works and how to create simple task automation

## In-Lab Demo -- Demo one of the scripts created in this lab

## Part I: Basic bash scripting

### Exercise #1: Creating a script

In this exercise, you will write a script called **`greet2`**.

Note: Scripts that are created on a local machine are typically stored in **`/usr/local/bin`** or **`/usr/local/sbin`** depending and who may run the script. Before you name your script verify with the command whereis if the name has already been used for a command. If it has, choose a different name for the script.

The script will contain a comment section with:

- the name of the script
- course number & name
- your name and student ID
- the purpose of this script

Note: Make certain you have the above information in every script file you create from now on.

Your script will:

- Ask the user for his or her name
- Greet the user using that name
- Ask the user what to do
- If 'date' or time', print the date and the time.
- Else if 'where", print the name of the machine.
- Else if 'files', print a list of all files in the user's home directory
- Else print the user's search path.

To accomplish this, follow the steps below:

Step #1

- Write and review the PDL you will need, using the outline above.
- Write a brief Test Plan for your script

Step #2

- Login as a user.
- Switch to the **`root`** account.
- Create a file in **`vi`** called **`greet2`**. Save and exit.
- Change the permissions so that everyone can execute the file.

Step #3

- Open the file and add the first line
- Add the starting comments

Step #4

- Use **`echo`** and **`read`** to get the user's name, both personal and family (i.e. Allan Turing); be sure to check the input
- Greet the user using **`printf`**
- Read one of the words from the notes above
- Based on that, perform the requested action
- Execute the script to ensure everything is working properly, using your Test Plan.

Step #5: Installing and testing the script

- Copy the script to the **`/usr/local/bin`** directory.
- Log in as a different user and execute the command, testing it against your Test Plan.

## Part II: Automating administrative tasks

### Search for orphan files (not owned by any account)

Specifications

Create a small script, PDL first, named **`lnuf`** (list **`nouser`** files) that accomplishes the following (be sure to create a standard script header).

The script searches for all files that have no user associated with them (hint: look at find(1)) and moves the files (see also xargs(1)) to the directory named **`nouser`** in the **`/var`** directory. If the directory does not exist, create it with permissions **`750`**.

Determine if the syslog service is running.

> Note: You can do this in more than one way. You may check for the existence of the service's pid file (**`/var/run/syslogd.pid`**) or use the return value of the **`service`** command. Don't check the **`ps`** output for the **`rsyslogd`** daemon. Why not?

If the syslog service is available, log a message at facility **`auth`** and priority **`warning`** using the standard log message format. The log file is located in the log directory and is named **`authlog`**. If syslog is not available, send the message only to **stderr**.

Note: To execute a command only if the previous command was successful, use the logical AND operator **&&**. Similarly, to execute only if the preceding command failed, use the logical OR operator **||:**

```
grep -q '^mail:' /etc/passwd && echo mail account found
grep -q '^obama:' /etc/passwd || echo no president here
```

The script must be scheduled to run monthly. Set it up as a system maintenance task, not as root's task, once you have verified that it is working. Change the date and test the complete set up.

Creating orphan files

Note that you will probably have to create several orphan files in several locations. Choose an unused UID/GID like 1000:1000:

```
Prompt# touch orphan
Prompt# ls -ln orphan    # -n shows UID/GID numbers
-rw-rw-r--. 1 500 500 0 2010-11-02 20:13 orphan
Prompt# chown 1000:1000 orphan
Prompt# ls -ln orphan
-rw-rw-r--. 1 1000 1000 0 2010-11-02 20:13 orphan
```

Result

Print out your script and all other material for inclusion in your Lab Book.

Record all changes to the rsyslog configuration file, if you need any:

________________________________________________________

Describe your scheduling setup, recording any entries made here:

________________________________________________________

You have created a new log file (**authlog**). Next you have to manage the number of log files kept and their frequency using ... (name the tool to use):

________________________________________________________

Identify any files that need to be modified to enable log rotation for the newly created log file and record your changes:

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________