CST 8177 - Lab 8

Automating Tasks: **`homebak`**

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 the **`homebak`** script and explain its commands.

Part I: Basic bash scripting

Exercise #1: Creating a script

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

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 a 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. Personal scripts are often placed in **`~/bin`**.

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:

- Greet the user using the user's login name
- Print the date and the time.
- Print the name of the machine.
- Print a list of all files in the user's home directory using the environment variable that contains the user's home directory.
- Print the search path using the environment variable that contains the user's search path.

To accomplish this follow the steps below:

Step #1

- Login as a user.
- Switch to the **root** account.
- Create a **bin** directory in root's home directory and change into it.

Step #2

- Create a file in **vi** called **greetings** insert the first line:
  ```
  #! /bin/bash
  ```
- This is the line that indicates to the shell that this is file to be interpreted by **/bin/bash**. Broken down into its components, this line uses:
  - **#!** - indicates a script file (hash-bang line)
  - **/bin/bash** - the path in which the shell bash resides

Step #3

Add the lines indicated below. Ensure that your script shows your OWN information!

```
# Name: greetings
# Course: CST8177 - Linux 2
# (your name) - Student ID (your id)
# Purpose: greet the user and display some information
```

Step #4

- Greet the user using the user's login name. Use printenv(1) to identify the environment variable that contains the user's login name
  Tip: Execute **printenv** and use **grep** to search for the entries that contain the account name.
- Use the **echo** command to display the content of the environment variable that contains the user's login name.
  Example: **echo $USER**
- Save and exit when you are done.
- Change the permissions so that everyone can execute the file.
- Execute the script to ensure everything is working properly.

Step #5

Next, you want to print the date and time. This information can be obtained using the date(1) command, with some options thrown in for formatting to make it look good. Re-open your file and add the next command:

Example: Enter the following line exactly as shown below.

```
echo "It is $(date +%A" "%e" "%B" "%Y", "%R" "%p)."
```

Note: Notice that we put the **date** command inside round brackets and preceded by a **$** sign, to tell the shell that this is a command to be executed outside the scope of the **echo** command and for its **stdout** to be returned.

Now run it to make sure it works properly.

Step #6

Next you want to display a list of all files in the user's home directory using the appropriate environment variable.

Save the file, exit and run it again to make sure it works properly.

Step #7

You want to show the search path.

Note: To display the search path use the echo command to display the contents of the environment variable used for the search path: **`PATH`**.

Save the file, exit and run it again to make sure it works properly.

Step #8

Display the time and date again after waiting 1 minute (note: don't actually wait for 1 minute - that's a very long time; test it with only 3 to 5 seconds instead).

Note: Use the sleep(1) command.

Save the file, exit and run it again to make sure it works properly.

Step #9: Installing and testing the script

Copy the script to the **`/usr/local/bin`** directory.

Log in as a different user and execute the command.

Step #10

The PDL or description for this script is simple because the flow of execution (flow control) is sequential.

```
START greetings
     DISPLAY user name
     DISPLAY date
     SHOW user's home directory
     SHOW current search path
     WAIT for a few seconds
     DISPLAY date
END greetings
```

The complete script is shown below:

```
# Name: greetings
# Course: CST8177 - Linux 2
# (your name) - Student ID (your id)
# Purpose: greet the user and display some information
echo "Hi $LOGNAME!"
echo "It is $(date +%A", "%B" "%e" "%Y)."
ls $HOME
echo $PATH
sleep 3
date
```

Part II: Automating administrative tasks

Back up all home directories at regular intervals

Specifications

Create a small script, named **`homebak`** (for home backup) that accomplishes all of the following (create a script header as above):

- Send a message to all users currently logged in (use the wall(1) command) to warn them of the impending operation and wait 5 minutes before proceeding (Note: While working on the script use a 5-second wait period).
- Prevent users from logging in during the backup (see nologin(5)). When should this login prevention start, and when should it end?
- Backup all home directories to **`/var/local/backup`** in a single gzipped tar archive. Use a file name that includes the month and day of the backup, specifically: **`home-back-MM-DD.tgz`**.
- Submit a message to the system logger at facility **`user`** and priority **`info`**. The log file you will use is to be located in the default log directory and named **`userlog`**. You may have to add an entry to the logging rules.
- Note: When logging, identify the name of your utility and keep the format of the log message as close as possible to the standard log message format.
- The backup is scheduled to occur weekly. Set it up as a system maintenance task in **`/etc/crontab`**, NOT as root's task.

Solution

- Write a point-form description of this script: describe WHAT or WHY, but not HOW.

  Example: State that the script is preventing users from logging in, but not how this is implemented: "PREVENT user logins"
- Write a Test Plan
- Develop your script from the description; for testing, replace long-running commands with faster forms and modify commands that create big files so they create smaller ones.
- Execute your Test Plan, confirming that any test output is correctly formed.
- Remove your replacement statements and run your Test Plan again, confirming that all files and records are correctly formed.
- Add your weekly task to the system maintenance facility. Test this by adjusting the date/time for entry and then restoring it.

Result

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

Record the output of an **`ls -l`** for your new home backup **`.tgz`** file:

Record the entry in **`rsyslog.conf`** file if you needed to add one:

Describe your scheduling setup, recording any entries made here:

You may have had a problem with the scheduling. How did you fix it?

You have created a new log file (**`userlog`**). 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:
