Computing and Web Development with Foundation

**Programme:** Computing & Web Development

**Module:** CPU5003

**Module Title:** UNIX

**Assignment Title:** UNIX Administration Facility

**Student Identification Number:** 1401501

**Tutor Name: Rob Campbell**

**Date of Submission: Friday 9th December 2016**

Contents Page

[1. Abstract 3](#_Toc468733167)

[2. Introduction 4](#_Toc468733168)

[3. System Administration 4](#_Toc468733169)

[4. UNIX Scripting 4](#_Toc468733170)

[5. Script Structure 9](#_Toc468733171)

[6. Conclusion 11](#_Toc468733172)

[7. Appendices 12](#_Toc468733173)

[Appendix One 12](#_Toc468733174)

[Appendix Two 12](#_Toc468733175)

[Appendix Three 13](#_Toc468733176)

[Appendix Four 13](#_Toc468733177)

[Appendix Six 14](#_Toc468733178)

# Abstract

The following report documents a brief history of the UNIX operating system and its uses. Included within the report is a guide to the process and utilisation of shell scripting. An overview of the shell scripting methodology is defined with a number of UNIX commands and utilities being demonstrated. Further documentational evidence is provided to show the results of these advanced features and system programs as they would appear to the end user.

# Introduction

UNIX is a multi-tasking, multi-user operating system developed by Bell Labs research centre in the 1970’s by a number of computer scientists namely Ken Thompson, Dennis Ritchie, Brian Kernighan and a number of others. Initially the UNIX system was used within academic institutes and later became a commercial grade operating system. UNIX has seen a number of additions since its creation as users developed and added their own utilities to the program increasing the functionality. The UNIX environment became essential in the development of the Internet and saw the transformation of computers being used a central hub within networks. UNIX and its derivatives are now widely used within a vast amount of technology products including the operating system IOS and OSX as used by Apple.



# System Administration

The UNIX operating system is extremely versatile, secure and robust. There are a number of computerised devices on which it can be deployed, from washing machines to company servers. Each device will require the operating system to perform a different task or application. A good example of the systems versatility is the administration and upkeep of a company network system.

It is essential the system is maintained to ensure system integrity, security and performance. Many companies adopt a multi-tiered approach to these administration tasks where repetitive tasks can be written into a script. A script can be a basic collection of important tasks, and is run on a daily basis. A demonstration and explanation of how this could be created and used for basic administration tasks will be shown below.

# UNIX Scripting

Scripting or shell programming is a great way to call commands and functions through the execution of a script. Scripts are written to perform repetitive tasks or scheduled tasks which can be run at any time. Creating a script is very much the same as executing a command within the command line interface (CLI).

An example of a simple script is shown in fig 1 with the results shown in appendix one.

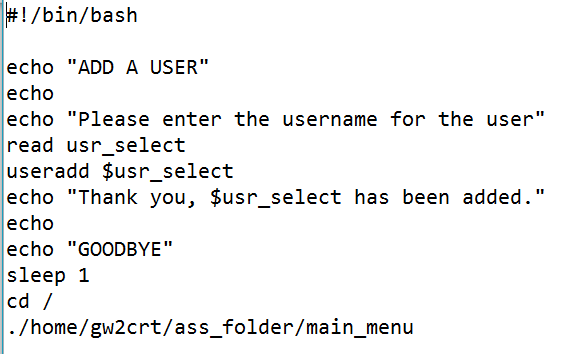


Figure 1 - Example Add User

From the example above you can see the structure of the script file, each line is executed one by one in the way they are written. This script is set to run the useradd command which adds a user to the system. This works by firstly asking for the username to be inputted which is stored into a variable named usr\_select. The command useradd is then executed with the variable being used in place of a username.

This enables anybody to add a user to the system without knowledge of the command line.

The next example shown in fig 2 shows a more advanced script which removes a user from the system.

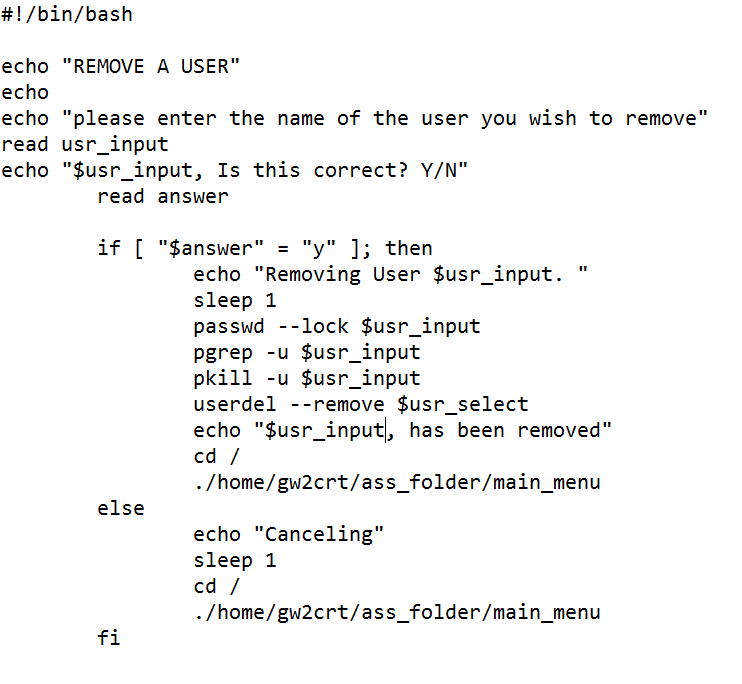


Figure 2 - Example User Remove

In the same way as the previous script this is asking the administrator for the users name to be removed. This again is written in a way which enables anybody to remove a user from the system without prior knowledge of the command line. Once the user enters the user name to be removed it is stored into a variable and used in conjunction with other commands.

The user is prompted to confirm their action as this action cannot be undone. The IF represents a conditional statement which must be met before the process flow of commands will continue, if this isn’t met the process flow jumps to the ELSE statement and performs those commands instead.

If confirmation is made the next command is called which locks the entered user account, this stops any further activity from the account and enables the user to be removed. PGREP then runs a search on all the active processes which match the username to be removed, followed by PKILL which terminates any active processes enabling USERDEL to fully remove the user, the --REMOVE addition ensures all files and folders relating to the user are fully removed. The output of this script can be seen in appendix two

Example fig 3 shows the process of using an iteration statement to check if there are no other users logged onto the system before performing a system reboot.

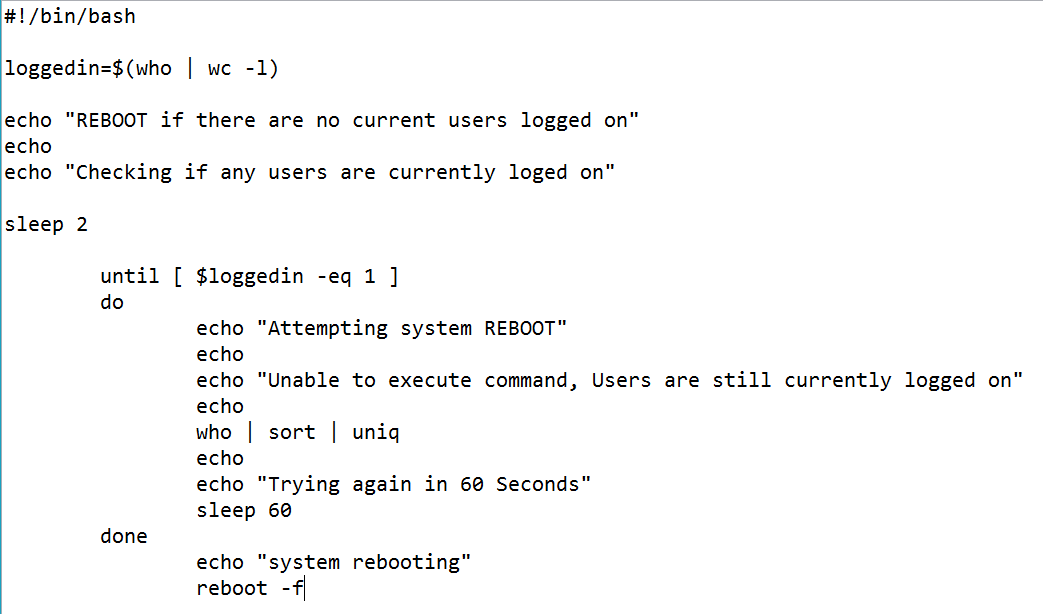


Figure 3 - Iteration Statement

When the script is initially started the variable LOGGEDIN is assigned with the value from the WHO command which then pipes the information to a word count giving the result as a single integer number. This integer is used against a conditional statement which will iterate until the condition is true. Using an UNTIL statement, the variable is constantly tested until the condition it is met. In this instance the statement loops until LOGGEDIN equals the value of 1. The DO aspect of the UNTIL statement presents the administrator information about who is currently logged into the system by running the WHO command. Once the UNTIL condition is met, the process flow will jump out of the loop and execute the reboot command appendix three shows the script in action.

A further example of an iteration condition is the FOR statement. Fig 4 shows the use of a FOR statement to loop through the results, displaying them to the administrator.

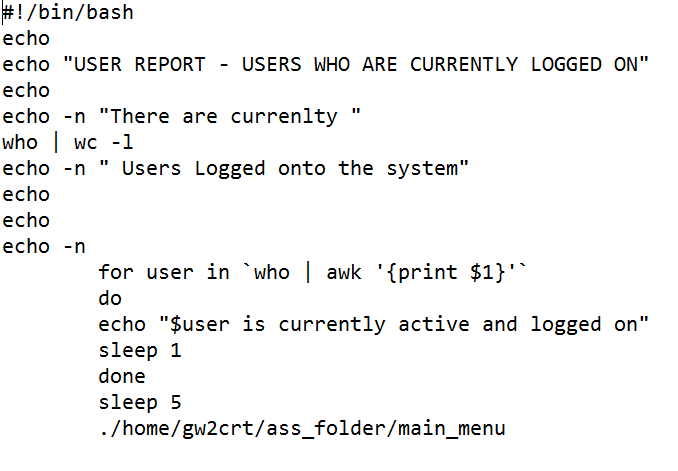


Figure 4 - Iteration and AWK example

The addition of AWK is used for this example which processes row and column information. This gives the administrator a way to present data, it is ideal for report generating or removing information which is irrelevant to the end user. In the example fig 4, the first column of information is selected and displayed on screen by using the AWK ’{print $1}’ command, the result of this script is shown in appendix four.

Further use of the AWK command is shown in fig 5, this example combines the use of AWK with GREP which is a powerful search program.

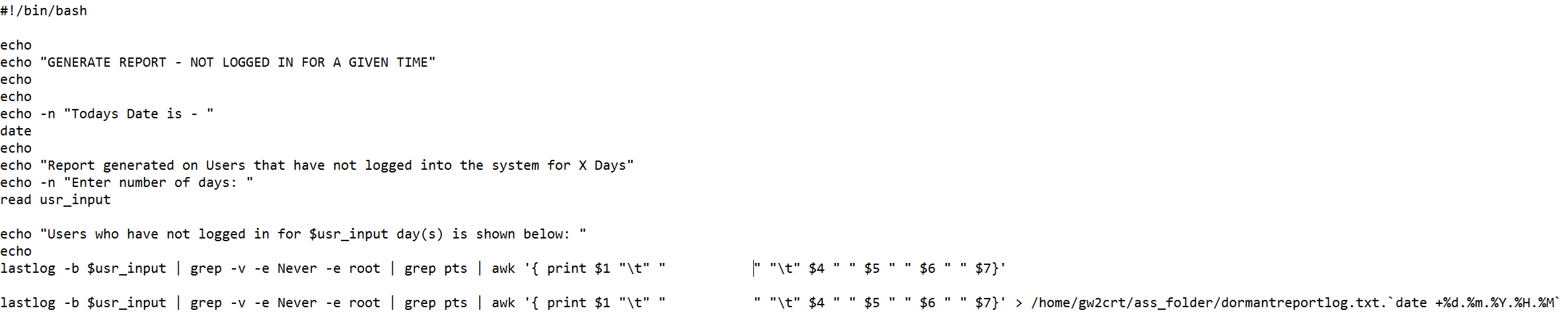


Figure 5 - Using Grep with Awk

The use of GREP in this instance is to search for users who have not logged in for a number of days. The administrator is shown today’s date with the date command and then asked to enter the number of days as their search term. Using the ‘lastlog’ command and the administrator’s value a list of users are shown. The addition of the pipe command sends the data into a GREP command which is searching to omit any results with the word never and root. The file is then piped to another GREP command which returns only data with ‘pts’ contained within the data structure. This data is then presented to the administrator with the use of the AWK command and sent to a text file which can be used within a report at a later stage.

# Script Structure

Creating a user-friendly menu driven system requires the use of case statements which take the users response and execute the corresponding script. The case statement used within the main menu system is shown below in fig 6.



Figure 6 - Menu Drive system using a Case statement.

This example shows the administrator the initial menu from which they can select to accomplish their given task. The user is welcomed with their username with the use of $USER and shown the date. The following echo statements return the contained text to the users screen giving them a number of options. Using a read command, the program flow is halted until an option is selected. A confirmation message is given to the user and executed with the case statement. The user input is tested against the number within the case statement and if matched it will execute the command contained within. If there are no matches an error message is displayed asking the user to try again. The option to generate a user report is shown in appendix six with a list of related commands.

# Conclusion

UNIX is an extremely powerful system giving the user a way to interact with the hardware through the use of the shell which in turn is directly linked to the system kernel. The UNIX has a large number of commands and utilities at the user’s disposal, these utilities can be further extended via the use of third party applications.

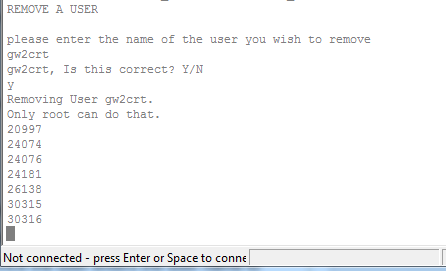
Through the use of the shell or system interface environment users are able to enter tasks, commands or prompts in order to manipulate the system into producing a result. This enables the use of shell scripting to be executed. Lists of commands are stringed together and interpreted and executed by the system producing results which can be automated giving great flexibility to the system administrators.

# Appendices

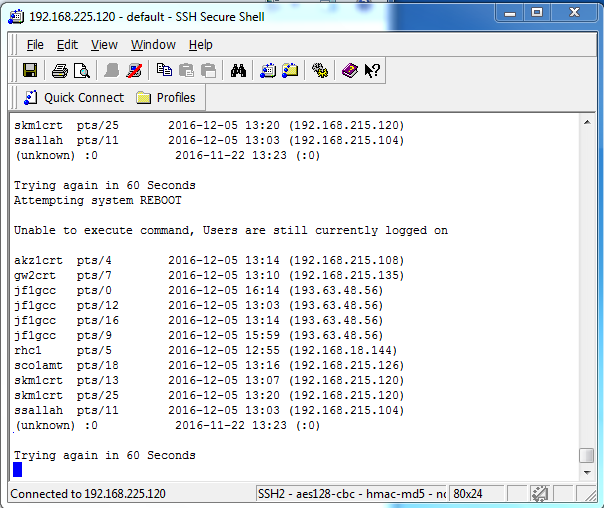
### Appendix One



### Appendix Two



### Appendix Three



### Appendix Four

