

Electronic Submission Coversheet

TO BE COMPLETED BY STUDENT

Important – choose one of the following statements
(DELETE TWO THAT DO NOT APPLY):

☒ This is my FINAL submission for this assignment.

By electronically submitting this work, I certify that:

- This assignment is my own work
- It has not previously been submitted for assessment
- Where material from other sources has been used it has been acknowledged properly
- This work meets the requirement of the University's ethics policy

Student Name: Matthew Utin

Email: Outinm77@solent.ac.uk

Student Number : (Q10266577)

Faculty: MARTEC

Level of study: 6

Course title: Network Security Management_L3_Grp 1

Unit title: Project (PRJ600)

Assignment title: Project Report

Assignment tutor: REMOVED

Word count: 9,323

Learner request for feedback:

TO BE COMPLETED BY STAFF

Tutor feedback:

Areas of Strength:

Areas for Improvement:

Grade mark:

Submitted on time (Y/N):

Tutor signature:

Date:

Southampton Solent University

**Maritime and Technology Faculty
Technology School**

BSc (Hons) Network Security Management.

2015

Matthew Utin

**“Creating a Linux administration script using the
“BASH” scripting language.”**

Supervisor : REMOVED
Date of presentation : 08th May 2015

Southampton Solent University
Maritime and Technology Faculty
Technology School

BSc (Hons) Network Security Management
PRJ154 Project, Project (PRJ600)
Assessment: Project Report
Academic Year 2014-15

Name: Matthew Utin

Project Title: Creating A Linux administration script using the
“BASH” scripting language.

Tutor: REMOVED

Date: 08th May 2015

Acknowledgments

Warren Earle, the project supervisor, has given outstanding support throughout the project in delivering professional information and motivation.

Finally, Neville Palmer, lecturer, for introducing Linux administration as a subsection of Network Operating Systems (CEN164), this has helped with the creation of this project.

Abstract

This project was created to explore and resolve a problem with a slow workflow in the ACME Company, due to performing daily administrative system task manually e.g. adding new users, creating new user groups and removing users. A solution was explored and then developed, to speed up the workflow of performing these tasks with the use of script automation to solve and address the current problem. This project also identifies which scripting language is best suited through the use of analysis, from script comparisons with the justification of the use of BASH script. The final implemented script was then packaged and submitted with a user feedback survey, to gain a user's response from the administrative script. The data collected has shown that the script was successful in showing that system automation is speeding up, workflow when performing system tasks. Moreover recommendations are given in script automation with future work and research.

Table of Contents

1. Introduction	1
1.1 Background	1
1.1.1 Introduction	1
1.1.2 Justifying the use of BASH within this project	2
2. Aim	2
3. Objectives	3
4. Option analysis	4
4.1 Introduction	4
4.1.1 Justifying the solution	4
4.1.2 Option 1 (Human interaction, with system automation)	4
4.1.3 Option 2 (Non- human interaction, with system automation)	5
4.1.5 Conclusion	5
5. Specification	6
5.1 Introduction and requirements	6
5.1.1 Conclusion	7
6. Literature Review	7
7. Methodology	12
7.1 Was the methodology followed in this project?	12
7.1.1 Conclusion	13
8. Design	13
8.1 Introduction	13
8.1.1 An overview of the design of the completed script	13
8.1.2 Adding a new user to the system	16
8.1.3 Adding a new user group	16
8.1.4 Show all users on the system and home directories	17
8.1.5 Removing a user from the system	18
8.1.6 Showing the manual file	19
8.1.7 Showing the about page	19

8.1.8 Showing the change log of all changes made to the script	20
8.1.9 Adding users from a file	21
8.2.1 Project Progress and Pilot Study	21
8.2.2 NASM	21
8.2.3 Python	22
8.2.4 BASH	22
8.2.5 Project Planning	23
8.2.6 Project Monitoring	24
8.2.6 Project Control	24
8.2.7 Conclusion	24
9. Issues arising from implementation and Tests	25
9.1 Introduction	25
9.1.1 Implementation	25
9.1.2 Tests	26
A.1. Creating new a regular user	26
A.2. Creating a root/admin user	26
A.3. Adding a user-group to a user	27
A.4. Creating a new user-group	27
A.5. Showing user-groups that a user is linked to	27
A.6. Showing all user-groups on the system	27
A.7. Showing all users on the system	27
A.8. About	28
A.9. Remove a user-group from a system user	28
A.10. Remove a user-group from system	28
A.11. Remove User with the home directory or just the user account	28
A.12. Quit!	29
A.13. The code element: options=(“”) script case opt selector	29
9.1.3 Final testing	29
9.1.4 User survey results	31
9.1.5 User survey results conclusion	32

9.1.6 Conclusion	32
10. Evaluation and Conclusions	33
11. Recommendations	35
12. References and Bibliography	36
13. IEEE References	44
14. Appendices	45
APPENDIX A: Gantt chart	A-1
APPENDIX B: Risk Table	B-1
APPENDIX C: Script Diagram and Pseudocode	C-1
C.1. Pseudocode	C-2
APPENDIX D: Scrum Methodology	D-1
D.1 Scrum project plan	D-2
APPENDIX E: NASM CODE	E-1
APPENDIX F: Python Script	F-1
APPENDIX G: BASH “BETA - Test” SCRIPT v1	G-1
APPENDIX H: BETA test script results	H-1
APPENDIX I: Final script build 2.0	I-1
APPENDIX J: Final script manual (MAN) file	J-1
APPENDIX K: Script addusers database file	K-1
APPENDIX L: Project planning level of achievement table	L-1
APPENDIX M: User Survey Results	M-1
M.1 Participant 1	M-1
M.2 Participant 2	M-3
M.3 Participant 3	M-5
M.4 Participant 4	M-7
M.5 Participant 5	M-9
M.6 Participant 6	M-11
M.7 Participant 7	M-13
M.8 Participant 8	M-15
M.9 Participant 9	M-17

List of Tables

Description	Page Number
Table 1: Shows the requirements of the project.	6
Table 2: Table showing the risk that could arise during the project.	B-2
Table 3: Methodology comparison.	D-1
Table 4: This table shows the Project planning and level of achievement.	L-2

List of Figures

Description	Page Number
Figure (1): Showing the prompt message if the script is not run as root.	14
Figure (2): XTerm terminal loading the script.	15
Figure (3): Gnome-terminal loading the script.	15
Figure (4): Adding a new user account to the system using the script.	16
Figure (5): Adding a new user group to the system, using the script.	17
Figure (6): Showing all users on the system with their home directory.	17
Figure (7): Showing a user being removed from the system.	18
Figure (8): Showing the man file being loaded from the script.	19
Figure (9): The about page being loaded into the terminal.	20

Figure (10): Shows the script executing the whiptail command line interface.	20
Figure (11): Showing the user add code function from a file, within the script.	21
Figure (12): Start of the completed Gantt chart.	A-1
Figure (13): End of the completed Gantt chart.	A-2
Figure (14): Diagram showing, how the Script will perform.	C-1
Figure (15): Linux project Scrum methodology.	D-3

Acronyms

Acronym	Meaning	Page Number
ACME	American Companies Make Everything	1
ANSI	American National Standards Institute	14
ASCII	American Standard Code for Information Interchange	9
BASH	Bourne-again shell	1
CPU	Central processing unit	22
GNOME	GNU Network Object Model Environment	11
HEX	Hexadecimal	19
IEEE	Institute of Electrical and Electronics Engineers	8
KDE	K Desktop Environment	26
NASM	Netwide Assembler	1
OS	Operating System	9
PERL	Practical Extraction and Report Language	2
SUSE	(Is a German acronym), Software und System-Entwicklung (software and systems development)	25
UNIX	Uniplexed Information and Computing System	8

1. Introduction

1.1 Background

The ACME Company is a small newly established company, based within the west midlands, that has two running servers both using Ubuntu. They want outside help to develop a running Linux Script, so that the system administrators can cut the time in completing, daily server tasks for example, adding new users and work groups to their network of computers.

1.1.1 Introduction

This is a brief introduction of the project. This project was created and selected because, ACME would like to create a Linux Script that can add new users and perform administrative tasks within the Linux (Ubuntu) system (Ask Ubuntu, 2014). So that the system administrators employed by the ACME company, can cut the time in completing, daily server tasks. This will be done using the BASH scripting language, (Arachnoid.com, 2014) to cut the time and cost because, it is a free scripting language and is available in all Linux distributions without the need for configuration.

This project will also explore in detail, at what other languages could have been used and why BASH was chosen instead of using the other different types. Here are the other languages that have been identified and reviewed within this project below:

- Python, this is a high-level object-oriented programming language. (Python.org, 2014).
- NASM e.g. (The Netwide Assembler), this is a low-level assembly code language. (Nasm.us, 2015).

The main skills that would be used to complete this project would be, improvisation e.g. (This would be used to work with the tools a valuable and if needed modify them to work or to be used to serve a specific function), time management e.g. (This would be used

to keep track of all the tasks that needed to be completed) and objective thinking e.g. (Thinking outside the box. To solve problems that are faced when implementing command line code into a scripting language).

1.1.2 Justifying the use of BASH within this project

The reason to pick BASH Script as the language to use and not others e.g. Perl Script (MOISE, D.L. and K. WONG, 2006) and the python Programming language (Python.org, 2014), is because, in most Linux distributions BASH Script is pre-installed so there is no need to go through setting up all of the files to run the Scripting language, also configuration to get it to run correctly for example Python. Perl Script is in most Linux operating systems, but can sometimes be long winded when trying to create a running script. Even though Python is also a good choice to use, as it is an object orientated programming language but due to the time to set up a working platform, that is useable, it would take too long to implement it for the ACME Company. So the main reason BASH was chosen was because it is fast, easy to implement and does not need to be compiled, to be useable unlike C based languages, (Oak.cs.ucla.edu, 2014).

2. Aim

The aim is to create a Linux user script that will be used by a Linux system administrator, to modify and preform command line tasks much more easily. Here is a list of some of the tasks below that the administrator performs daily on the system. These are going to be implemented within the BASH script.

- Creating regular users and root/admin users.
- Adding a user group to a user.
- Creating a new user group.
- Showing the user groups that a user is linked to on the system.
- Viewing all user groups on the system.
- Viewing all users on the system.

- Removing a user group from a user.
- Removing user group from system.
- Removing a user and home directory or just the user account from the system.
- Making a file executable.

This script is going to be more user-friendly, than just typing in countless amounts of command line code and just hoping that the input into the system is in the correct syntax to make it execute correctly to achieve the task. This is why script automation is a must when keeping a better workflow to complete tasks.

3. Objectives

1. To research and identify the most useful commands, that are used within Linux, for system administration.
2. Find out how to implement those command line commands into a Linux BASH Script.
3. Design and create a Linux script and test/debug the script to see if it has any problems or if there could be any improvements.
4. Trial out a small sample for testing in a beta form to see if there are any unforeseen problems in the testing stage that can be fixed, also any improvements that could be made to the script.
5. After all of the script has had the fixes that are needed, this would be the time to finish the script off and package it for install, with the correct software licences and help files for download.

4. Option analysis

4.1 Introduction

The option analysis will explain why the different script options were used within this project when creating the script, also justifying the reason for script automation within the parameters of the projects structure, the aspects that the option analysis will explore are human interaction with system automation and non-human interaction with system automation.

4.1.1 Justifying the solution

The reason for creating this project is, because the Linux terminal commands are sometimes hard to implement and execute if you do not know the correct syntax (SOLOMON, A., D. SANTAMARIA and R. LISTER, 2006). Linux commands also have the problem that if you type in the wrong spelling let's say in the Terminal and you try to execute the command you will not be able to, or even worse the command, let's say "userdel user01" executes but deletes the wrong user. This can happen if you have to keep on typing out this command countless times.

4.1.2 Option 1 (Human interaction, with system automation)

The solution to this is system automation, this will be done using a Linux script that will perform theses commands, in a more safe and friendly user environment. This will tell the user if the command is correct and will show instructional interactions within the script to help with the task the user wants to achieve.

From this article (Advanced Bash-Scripting Guide, An in-depth exploration of the art of shell scripting, part 1 Introduction by Mendel Cooper, 10 Mar 2014). The author describes why Bash Script can be powerful as a scripting language. ("If that were not enough, internal shell commands, such as testing and loop constructs, lend additional power and flexibility to scripts.").

4.1.3 Option 2 (Non- human interaction, with system automation)

The second part to this project is to read in large amounts of data from a file and then use this collected data to perform a system task, this would be reading in user information from a data file and then using this data to add and apply new users to the system without manually typing in every user by hand, diminishing the users performance and workflow.

In the article How to Write a Basic Shell Script by (Oak.cs.ucla.edu, 2014). It is explained how the use of BASH Scripts are used to automate the Linux Command line. (“If you find that you execute a sequence of commands frequently in your command line, you can write a shell script to automate this process.”).

4.1.5 Conclusion

To conclude script system automation is a must when cutting down working time when completing a task on a server or system because, why do something manually when there script automation could be implemented. The two options discussed above make good points on the different types of automation when applied to systems. One being that non-human interaction scripts are slowly becoming the future of things to come because of the expanding use of artificial intelligence systems deciding when to run a tasks.

5. Specification

5.1 Introduction and requirements

This project is going to be specified within the parameters of the development and environment. Ubuntu is the main requirement for developing the script as this is the operating system that the script is going to be run on. The final script is also going to be using Ubuntu specific commands. The script will be written to perform Linux administrative tasks on the operating system. The main requirements for the script are shown in the bullet points and table below:-

<u>Requirements</u>	<u>Why?</u>
A Linux operating system mainly (Ubuntu)	To run the script and for future development.
The Bash scripting language needs to be installed on the system correctly	So the bash scripting code can be run within the Linux terminal.
System Administrative privileges on the system	The script can only be run if the person using the script is an Administrator.
Code editing software e.g. (Kate)	To develop the code.
Linux Terminal	The terminal is like a compiler and executes the code within the script.

Table (1). Shows the requirements of the project.

“There are no costs associated for the requirements needed to create the script, because the requirements are all open source and free to use.”

These are the main elements that are specified within this project.

- Create regular users and root/admin users.
- Binding a user group to a user.
- Create a new user group.
- Viewing the user groups that a user is linked to on the system.
- Viewing all user groups on the system.

- Viewing all users on the system.
- The Removal of a user group from a user.
- The Removal of a user group from system.
- The Removal of a user and home directory or just the user account from the system.
- Changing a file to be executable.

View: (Appendix L), to see the Project planning level of achievement table.

5.1.1 Conclusion

The outlined specifications for this project were used to give a base to work from a list of requirements to follow, this helped in the creating of the script in its beta stage of development to see what worked best, also it helped in defining the final script within this project.

6. Literature Review

In the article How to Write a Basic Shell Script by (Oak.cs.ucla.edu, 2014). It is explained how to use BASH Scripts to automate the Linux Command line. (“If you find that you execute a sequence of commands frequently in your command line, you can write a shell script to automate this process.”).

From this article Advanced Bash-Scripting Guide, An in-depth exploration of the art of shell scripting, part 1 Introduction by (Mendel Cooper, 2014). The author describes why Bash Script can be powerful as a scripting language. (“If that were not enough, internal shell commands, such as testing and loop constructs, lend additional power and flexibility to scripts.”).

From this Book (Robbins, A R, 2010. bash Pocket Reference, Help for Power Users and sys Admins. 1st ed. USA: O'Reilly Media Inc.) Shows some great examples of how to automation system tasks with useful command line shell code written in BASH script. This book has helped with many aspects of the finished script from the use if some of the basic commands that are used within the Linux system.

From the following IEEE conference publication (MCNABB, A., J. LUND and K. SEPPI, 2012. Mrs: MapReduce for Scientific Computing in Python. High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion , pp.600-608.) Shows how Python scripting can adapted to perform some extraordinary tasks, when it is put to its use of object oriented programming, it also shows how python could also no longer be called a scripting language and now is a fully automated programming language.

In this conference publication (SOLOMON, A., D. SANTAMARIA and R. LISTER, 2006. Automated Testing of Unix Command-line and Scripting Skills. Information Technology Based Higher Education and Training, 2006.ITHET '06.7th International Conference, pp.120-125.) It shows how Linux and Unix system automation is a must and why it should be implemented for cutting down work time. It also talks about different types of shell scripting languages, for example BASH, C shell and Perl script.

So why was scrum chosen for this project well from this conference publication (OVERHAGE, S., SCHLAUDERER, S., BIRKMEIER, D. and MILLER, J., 2011. What makes IT personnel adopt scrum? A framework of drivers and inhibitors to developer acceptance, System Sciences (HICSS), 2011 44th Hawaii International Conference on, 2011, IEEE pp1-10google.), It puts it in easy terms to why software developers choose to use Scrum for project management, when creating new software. The main reason why it is chosen is because Scrum has different parts one in creating the pre list of software additions and the other three aspects of completion sessions, in which can be changed or move backwards if a problem is uncouneted, this is slightly different to the waterfall method.

From the following website article (Python For Beginners, How to use Loops in Python. 2012) it shows how easy it is to set up a loop within the programming language, also how

to execute OS system shell code. This was used in the project to create the Python script test code.

From this online book (MENDEL, C., Advanced Bash-Scripting Guide The Linux Documentation Project, 2014.) It explains how to add advanced feathers to any basic BASH script and shows how easy it is to implement with great examples. This shows the main point that BASH is quick to implement and can be capable to a programming language at times.

From this conference publication (HE, K., X. XU and Q. YUE, 2008. A secure, lossless, and compressed Base62 & Base 64 encoding. Communication Systems, 2008.ICCS 2008.11th IEEE Singapore International Conference on, 761-765.) It pushes the point of encoding documents and explains why base 64 can help do this, also giving a description of what is base 64. You can see this from this extract from the conference (“ $d_1b^{n-1} + d_2b^{n-2} + \dots + d_nb^0$, $0 \leq d_i < b$.”) the equation shows how base 64 also known as MIME Base64 uses a form of radix but with the use of a minimum of two characters or more than two characters to encode. This will also have the adding of padding showing this with “==” for one byte or “=” for two bytes. This is calculated in with the following equation (“ $4[n/3]$ ”) to determine the size of padding used.

This website explains (Unix-manuals.com, ASCII Table - table of ASCII codes. 2001) How to use ASCII within BASH this was used to create the BASH script menus within the script.

This website explains (Docs.cs.up.ac.za, System Call Table. 2004) how to use system calls within Linux, this was used to help build the NASM script, so that it would function correctly.

In this web article (Cyberciti.biz, HowTo: Linux / UNIX Create a Manpage., 2010) They show how to set up and create a manual help file in Linux. Here is a useful extract from the web article (“You can add a few more other sections such as EXIT STATUS,

ENVIRONMENT, FILES, and HISTORY etc. The table below shows the section numbers of the manual followed by the types of pages they contain.”)

The following website (Isle of Man Training Course Company. Agile Development., 2011) gave a good explanation to what the agile methodology is and where it is used here is an extract from the site (“Agile development delivers a working application, free of all known bugs, at the end of each development iteration. The resulting alpha release is then delivered to the client for testing and feedback.”) This is like SCUM but Scrum includes this as standard with a waterfall type flow incorporation. This helped when choosing the correct methodology for this project.

This short web article from the following website (Isle of Man Training Course Company. Waterfall Development Methodology, 2011) explains in a nut shell how the waterfall methodology is used, this also helped when choosing the correct methodology that fits correctly within this project. Here is a brief extract from the following website (“It differs from the agile development model by seeking to fully describe the application in written documents before any code is written (the implementation phase).”) This is also like SCRUM and is encountered within script but with the advantage of change later on within the next stages if there needed to be that is.

From this website (How to Add Users to the Linux OS from a Text file, HowtoForge - Linux Howtos and Tutorials. 2014.) This website helped finalize the add users script from a text or database file as it gave some good tips on how to implement the script to edit the system users. Here is an extract from the website (“useradd -g \$usr_group -m \$usr_name”) that shows the basic code to adding a new user with a group to the system. This was then worked into the finished script with modifications.

From the following website (Bash:tip_colors_and_formatting - FLOZz' MISC, 2014) Had some great information on how to implement colour codes within Linux and what terminals are compatible, here is an extract from the website (“The colours number 256 is only supported by vte, GNOME Terminal, XFCE4 Terminal, Nautilus Terminal, Terminator.”) this explains the information of what different terminal compatibility’s

are used in the 256 colour range. This was used within the script to brighten up the menus, so that it is easier to read for the users.

From the following Journal Critical analysis of the Scrum Project Management Methodology. By (FT N IL IONEL, N, 2008) It goes on to talk about the different well rounded aspects of using the scrum methodology to oppose to using a different methodology to run a project that will be creating some type of software. It also gives some good advantages to project management flexibility. Here is an extract from the Journal (“The SCRUM methodology, on the other hand, is designed to be flexible all along the project life. It provides control mechanisms for planning a release, and then for managing the project’s variables as it progresses.”) As you can see it gives a great argument in why to use SCRUM to manage a software project. Because of the use of flexibility.

From this website article Compiling an assembly program with Nasm. (Kioskea, 2014) It shows how to create the NASM assembly code and how to compile it the correct way, from code to executable, this site was useful in setting up the NASM code and running tests to get it to run correctly. Here is an extract of information about NASM from the article (“An assemble will turn your low-level coding, using mnemonics, into machine language that can be understood by the processor.”) as you can see it gives a straightforward definition of what the assembler dose. And here is one of the command line commands to compile the NASM code from the article (“Use the following command line to assemble your source file:nasm -f elf test.asm”) this then compiles the code but it will then need the command to execute the file.

7. Methodology

The reason the scrum methodology was chosen for this project, is because it is in within the main structure of this project as “SCRUM”, (FT N IL IONEL, N, 2008; OVERHAGE, S., SCHLAUDERER, S., BIRKMEIER, D. and MILLER, J., 2011; Cohn, M. 2014.) is used in software development in creating a new product that will have different components that would need to be completed in different segments and that will be in a timed schedule, SCRUM also has the advantage of the use of product backlog, sprint backlog and a burn-down chart. This is useful when reviewing what parts of the Linux Script in this project needs to be completed. SCRUM also takes in the advantage of up front planning for example like the “WATERFALL”, (Isle of Man Training Course Company, Waterfall Development Methodology 2011.) methodology that uses the same approach but does not use the cyclical that SCRUM uses to put the requirements for the software in highest order, this is also used in the “AGILE”, (Isle of Man Training Course Company, Agile Development 2011; Cprime.com 2014.) methodology, using a multi changing scope that could be changed if needed, with the highest order, of the different segments within project plan. See (Appendix D).

7.1 Was the methodology followed in this project?

The current methodology SCRUM has been followed all the way through this project when creating the script, this was due to the methodology’s flexibility when developing software. For example when creating the different aspects of the script when following SCRUM methodology it gave time to debug the script within the SCRUM methodology cycles of development, also when one section is in development the other section will also be in debugging, so the project will always be in constant development not impacting on the workflow.

7.1.1 Conclusion

The methodology used for this project has helped in the creation of the final script also in the help of debugging the script when it was in development, this was because the scrum methodology takes into account the debugging stage of development, moreover the methodology was used throughout the projects life cycle and there was not any deviation from the latter.

8. Design

8.1 Introduction

The design will convey how the project was created and what aspects were needed to be used to finish the project, the main sections that are going to be explained is the “overview of the design of the script”. This will show how the script parts are used, also showing how they should perform with user interaction. The section “Project Progress and Pilot Study” will explain the aspects of Pilot study and will talk about the script comparison results. The following sections “Project Planning”, “Project Monitoring” and “Project Control” are going to explain the project’s control and planning metrics.

8.1.1 An overview of the design of the completed script

This section is going to look at how the overall script has been developed and is going to show the thought behind the scripts functionality and design, explaining the process in depth and showing how the script links in different script technologies and methods to perform specific system tasks.

The first section that is going to be shown is the first load menu of the Linux administration script, but before this can be talked about. The methods off the script start-up must be discussed, because the script has some error checking when executed.

When the script is executed the script will initialize some code to check if the user running the script is logged in as an administrator in the current terminal session, if the user is not logged in as an administrator or a root user, then the script will print out a message to the user to tell them that this script can only be run if you are administrator.

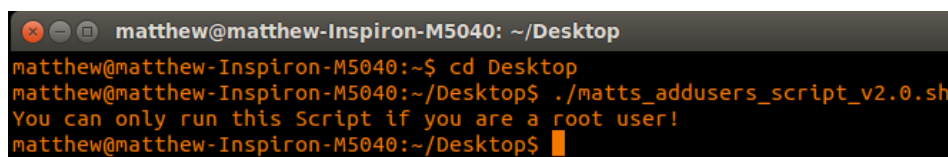
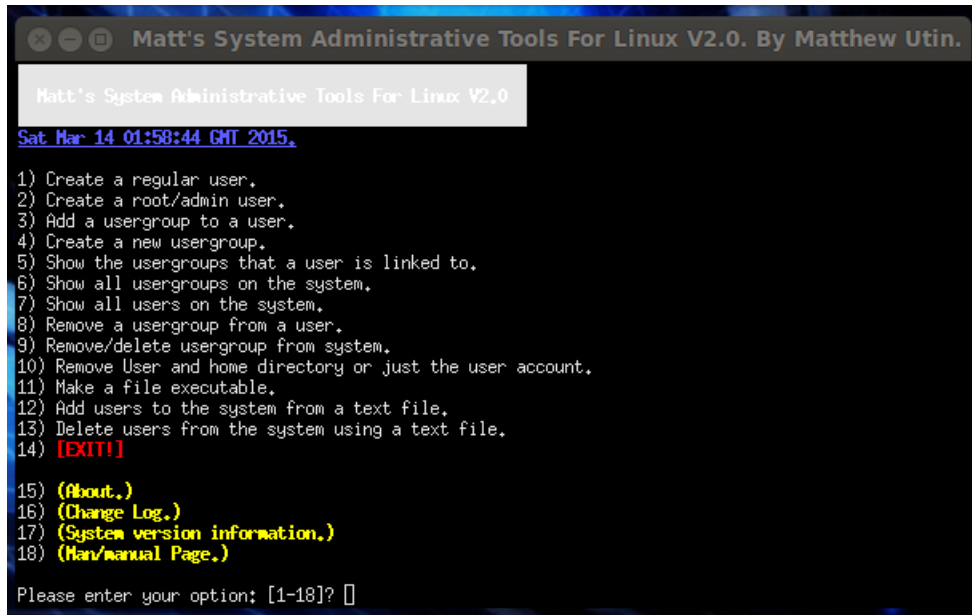
A terminal window titled 'matthew@matthew-Inspiron-M5040: ~/Desktop'. The prompt is 'matthew@matthew-Inspiron-M5040:~\$'. The user enters 'cd Desktop'. The prompt changes to 'matthew@matthew-Inspiron-M5040:~/Desktop\$'. The user enters './matts_addusers_script_v2.0.sh'. The script outputs the message 'You can only run this Script if you are a root user!' followed by a new prompt 'matthew@matthew-Inspiron-M5040:~/Desktop\$'.

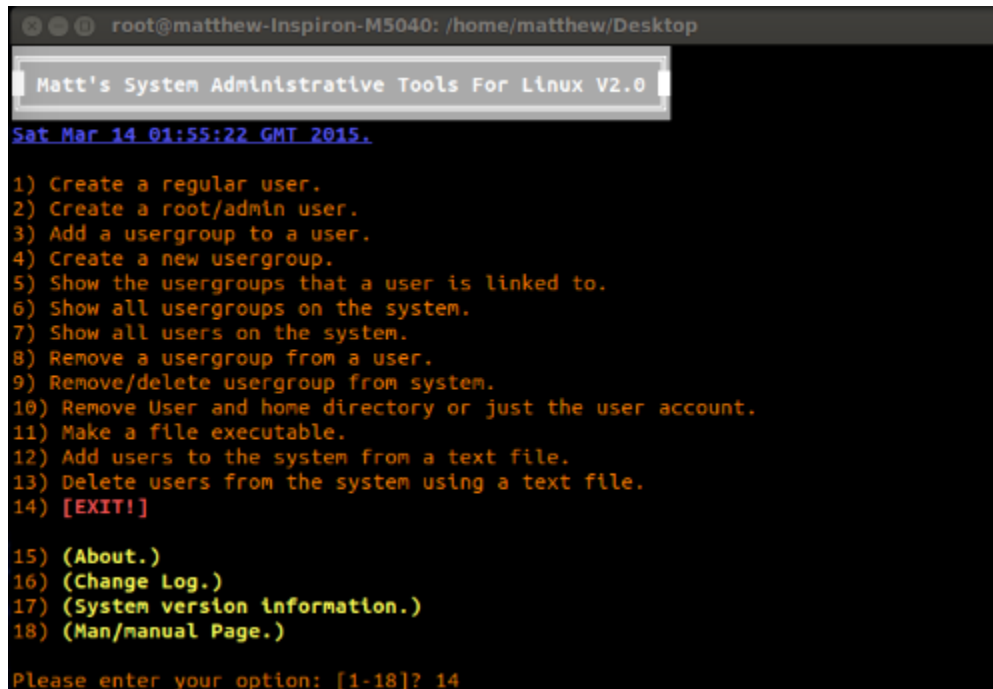
Figure (1): Showing the prompt message if the script is not run as root.

The next stage is the execution of the script, on load it will set the height and width of the current terminal session window size, also the script will print out the main selection menu so that the user can see what options they would like to select to perform a task on the system. The menu is also compatible across the “gnome-terminal” (Help.gnome.org, 2014), the Ubuntu Linux default terminal and “XTerm” (Dickey, T. 2014). The XTerm terminal can also run terminal colour ANSI Escape sequence codes (Ascii-table.com, 2015; Ezust, A. 2014; Flozz, M. 2014.), So when the script loads in XTerm terminal the date and time is set to blink as it can read in the custom ANSI Escape sequence code blink, the blink code is only supported in XTerm, but the colour codes work correctly in the gnome-terminal moreover when the script starts up it sets the terminal console windows title with “Matt’s System Administrative Tools For Linux V2.0. By Matthew Utin.” If this wasn’t set the window title would be blank or showing the default system text set by that system. This can be viewed in the image below:-

A screenshot of an XTerm terminal window. The title bar reads "Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.". The terminal content shows the script's title, a timestamp "Sat Mar 14 01:58:44 GMT 2015.", a numbered list of 14 commands, and options 15 through 18. The prompt "Please enter your option: [1-18]? " is at the bottom.

```
Matt's System Administrative Tools For Linux V2.0
Sat Mar 14 01:58:44 GMT 2015.
1) Create a regular user.
2) Create a root/admin user.
3) Add a usergroup to a user.
4) Create a new usergroup.
5) Show the usergroups that a user is linked to.
6) Show all usergroups on the system.
7) Show all users on the system.
8) Remove a usergroup from a user.
9) Remove/delete usergroup from system.
10) Remove User and home directory or just the user account.
11) Make a file executable.
12) Add users to the system from a text file.
13) Delete users from the system using a text file.
14) [EXIT!]
15) (About.)
16) (Change Log.)
17) (System version information.)
18) (Man/manual Page.)
Please enter your option: [1-18]? 
```

Figure (2): XTerm terminal loading the script.

A screenshot of a Gnome-terminal window. The title bar shows the user "root@matthew-Inspiron-M5040:" and the directory "/home/matthew/Desktop". The terminal content is identical to Figure 2, but the prompt "Please enter your option: [1-18]? " has been followed by the number "14".

```
root@matthew-Inspiron-M5040: /home/matthew/Desktop
Matt's System Administrative Tools For Linux V2.0
Sat Mar 14 01:55:22 GMT 2015.
1) Create a regular user.
2) Create a root/admin user.
3) Add a usergroup to a user.
4) Create a new usergroup.
5) Show the usergroups that a user is linked to.
6) Show all usergroups on the system.
7) Show all users on the system.
8) Remove a usergroup from a user.
9) Remove/delete usergroup from system.
10) Remove User and home directory or just the user account.
11) Make a file executable.
12) Add users to the system from a text file.
13) Delete users from the system using a text file.
14) [EXIT!]
15) (About.)
16) (Change Log.)
17) (System version information.)
18) (Man/manual Page.)
Please enter your option: [1-18]? 14
```

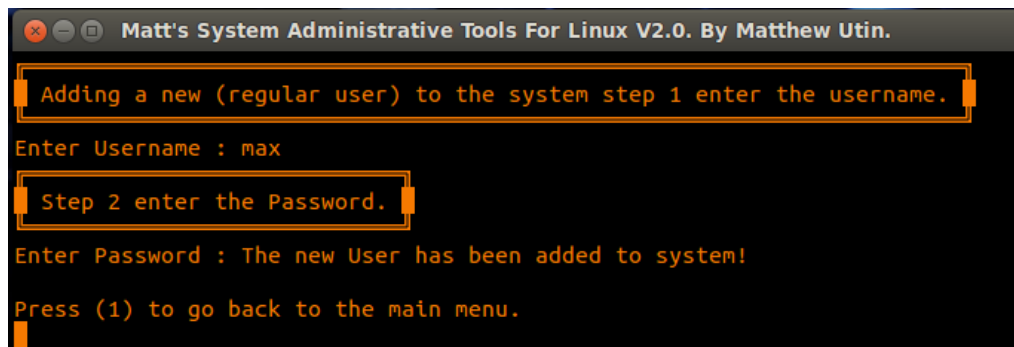
Figure (3): Gnome-terminal loading the script.

The next section will show some examples of the in-menu tasks and the types technologies used that are performed by the script, with the examples below:-

- Adding a new user to the system
- Adding a new user group
- Show all users on the system and home directories
- Removing a user from the system
- Showing the manual file
- Showing the about page
- Showing the change log of all changes made to the script
- Adding users from a file

8.1.2 Adding a new user to the system

When adding a new user to the system the script works, by invoking the system command “adduser”(Negus, C.N. 2012.) within Ubuntu Linux to add the user, but loads in the users typed data into the command line within the script using set variables. This will then be executed when the user has finished inputting data into the script and hits the enter key on the keyboard to execute that line of code inside the script.



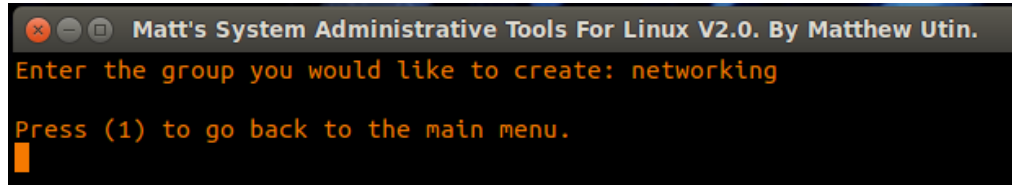
```
Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.
Adding a new (regular user) to the system step 1 enter the username.
Enter Username : max
Step 2 enter the Password.
Enter Password : The new User has been added to system!
Press (1) to go back to the main menu.
```

Figure (4): Adding a new user account to the system using the script.

8.1.3 Adding a new user group

When adding a new user group to the system the script will load in the system command “groupadd” (Arachnoid.com, 2014.) the script will then load in the users input, using a

set variable. This code is using the same terminology as the add user option within the script above.

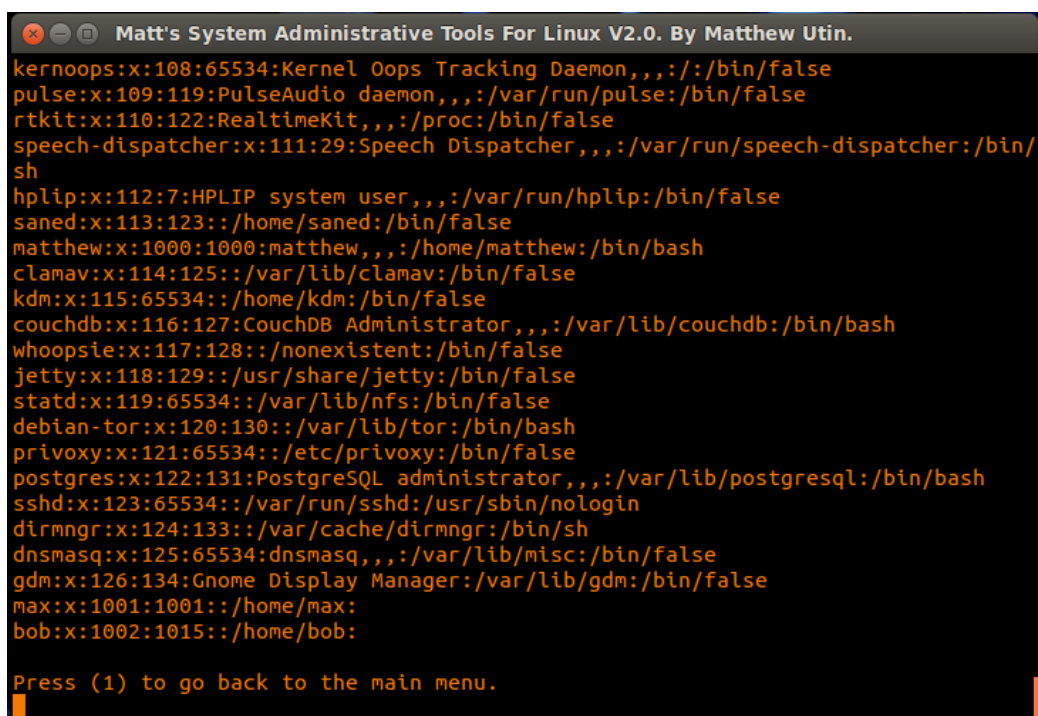


```
Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.  
Enter the group you would like to create: networking  
Press (1) to go back to the main menu.  
|
```

Figure (5): Adding a new user group to the system, using the script.

8.1.4 Show all users on the system and home directories

The user selects the menu option to show all users on the current system with their home directories. This is done by using “cat /etc/passwd” (Linfo.org, 2014; J. Emmons, 2006.) this Linux command displays the contents of the passwd file, this file holds the information of the users on the system and there directories. So the script launches a function this contains the Linux command line command, this is then executed and loaded into the terminal session.

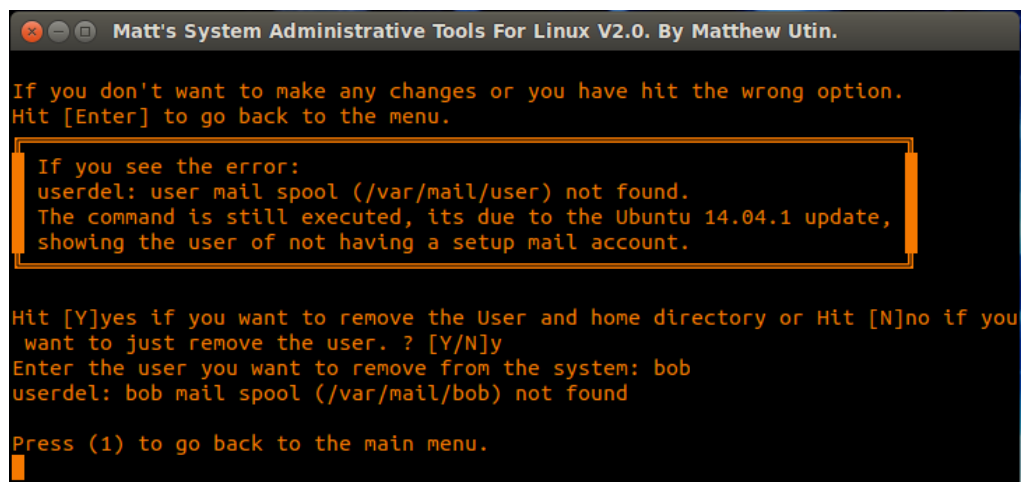


```
Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.  
kernoops:x:108:65534:Kernel Oops Tracking Daemon,,,:/bin/false  
pulse:x:109:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false  
rtkit:x:110:122:RealtimeKit,,,:/proc:/bin/false  
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh  
hplip:x:112:7:HPLIP system user,,,:/var/run/hplip:/bin/false  
saned:x:113:123:./home/saned:/bin/false  
matthew:x:1000:1000:matthew,,,:/home/matthew:/bin/bash  
clamav:x:114:125:./var/lib/clamav:/bin/false  
kdm:x:115:65534:./home/kdm:/bin/false  
couchdb:x:116:127:CouchDB Administrator,,,:/var/lib/couchdb:/bin/bash  
whoopsie:x:117:128:./nonexistent:/bin/false  
jetty:x:118:129:./usr/share/jetty:/bin/false  
statd:x:119:65534:./var/lib/nfs:/bin/false  
debian-tor:x:120:130:./var/lib/tor:/bin/bash  
privoxy:x:121:65534:./etc/privoxy:/bin/false  
postgres:x:122:131:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash  
sshd:x:123:65534:./var/run/sshd:/usr/sbin/nologin  
dirmngr:x:124:133:./var/cache/dirmngr:/bin/sh  
dnsmasq:x:125:65534:dnsmasq,,,:/var/lib/misc:/bin/false  
gdm:x:126:134:Gnome Display Manager:/var/lib/gdm:/bin/false  
max:x:1001:1001:./home/max:  
bob:x:1002:1015:./home/bob:  
  
Press (1) to go back to the main menu.  
|
```

Figure (6): Showing all users on the system with their home directory.

8.1.5 Removing a user from the system

When removing a user from the system using the script, the administrator will select the option from the main menu, this will load the function and execute the remove user command line interface. The first few lines displayed to the administrator will give information on how to exit the script option if the administrator had entered the remove user option accidentally. The next few lines will inform the administrator, if an error is seen that an email spool was not found, don't worry as this is just telling the user of the script that an email account was not created for that user account. The script will now display a message to enter Y for yes to remove the home directory and the user account, if the administrator enters N for no then the script will only remove the user account and not the home directory. After the selecting yes or no depending on the chosen option, a prompt will display for the administrator to enter the name of the user account that is going to be removed from the system. Then the keyboard key 1 is pressed to return to the main script menu. The Linux system command is "userdel" (Robbins, A R. 2010) this used within the scripts code function to perform the removal of the user from the system. From Figure (7), it shows the user bob being removed from the system.



```
Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.

If you don't want to make any changes or you have hit the wrong option.
Hit [Enter] to go back to the menu.

If you see the error:
userdel: user mail spool (/var/mail/user) not found.
The command is still executed, its due to the Ubuntu 14.04.1 update,
showing the user of not having a setup mail account.

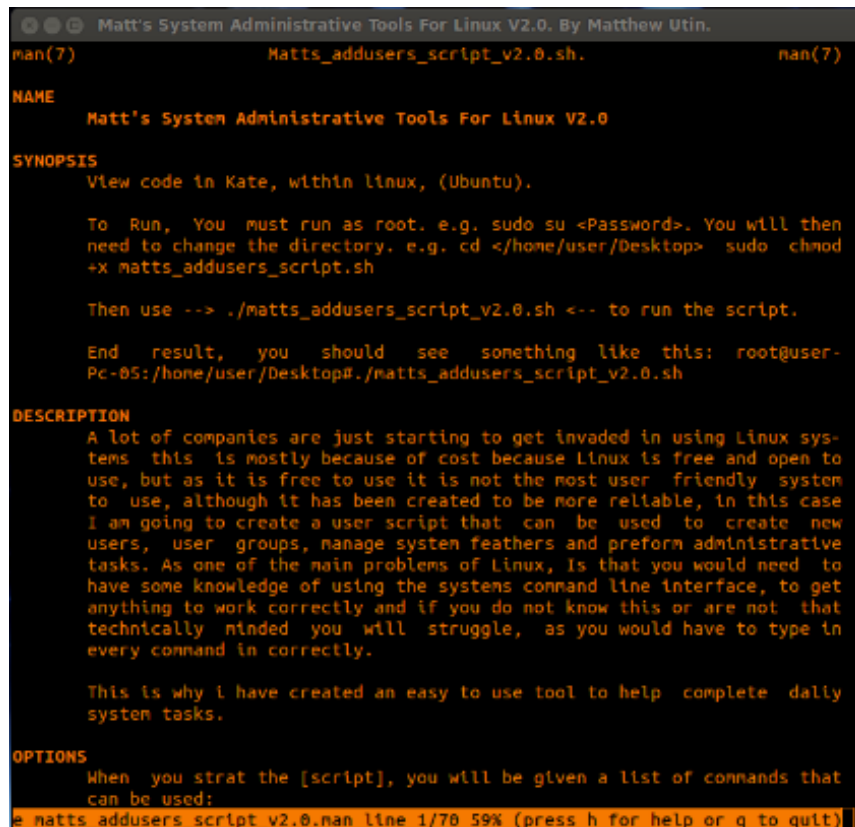
Hit [Y]yes if you want to remove the User and home directory or Hit [N]no if you
want to just remove the user. ? [Y/N]y
Enter the user you want to remove from the system: bob
userdel: bob mail spool (/var/mail/bob) not found

Press (1) to go back to the main menu.
```

Figure (7): Showing a user being removed from the system.

8.1.6 Showing the manual file

When the user of the script selects the option to open up the manual file. The file is loaded from the same working directory using the script code function, the code within the function will then be executed this is “man ./matts_addusers_script_v2.0.man” this loads the file. This can be viewed from Figure (8).



```

Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.
man(7)                                     matts_addusers_script_v2.0.sh.      nan(7)

NAME
    Matt's System Administrative Tools For Linux V2.0

SYNOPSIS
    View code in Kate, within linux, (Ubuntu).

    To Run, You must run as root. e.g. sudo su <Password>. You will then
    need to change the directory. e.g. cd </home/user/Desktop> sudo chmod
    +x matts_addusers_script.sh

    Then use --> ./matts_addusers_script_v2.0.sh <-- to run the script.

    End result, you should see something like this: root@user-
    Pc-05:/home/user/Desktop# ./matts_addusers_script_v2.0.sh

DESCRIPTION
    A lot of companies are just starting to get invaded in using Linux sys-
    tems this is mostly because of cost because Linux is free and open to
    use, but as it is free to use it is not the most user friendly system
    to use, although it has been created to be more reliable, in this case
    I am going to create a user script that can be used to create new
    users, user groups, manage system features and perform administrative
    tasks. As one of the main problems of Linux, is that you would need to
    have some knowledge of using the systems command line interface, to get
    anything to work correctly and if you do not know this or are not that
    technically minded you will struggle, as you would have to type in
    every command in correctly.

    This is why I have created an easy to use tool to help complete daily
    system tasks.

OPTIONS
    When you start the [script], you will be given a list of commands that
    can be used:
e matts_addusers_script_v2.0.man line 1/70 59% (press h for help or q to quit)
```

Figure (8): Showing the man file being loaded from the script.

8.1.7 Showing the about page

The about page is loaded using the script function selected using the main script menu, the code inside the function consists of echo statements that outputs text to the command line terminal session. Also some of the code inside this function is encapsulated using “eval `printf ‘ ‘ `” with slash notation hexadecimal code.

```

Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.
Sat Mar 14 02:37:40 GMT 2015
*/-----/
#####
## ## ## ## ## ## ##
## ## ## ## ## ## ##
## ## ##### ## ## ##
##### ## ## ## ## ##
## ## ## ## ## ## ##
## ## ##### ## ## ##
/-----/*

----->>[By: Matthew Utin]
----->>[Date: Sun Oct 26 2014]
----->>[Version: 2.0, Ubuntu Specific, Commands.]

Licence: LGPL, (c) Matthew Utin 2014-2
You can use this code in any tool but you must credit it.
There are going to be Detailed comments provided for use in this tool.

Press (1) to go back to the main menu.

```

Figure (9): The about page being loaded into the terminal.

8.1.8 Showing the change log of all changes made to the script

Selection of the main menus option, executes a function. Within this function there is whiptail code this is used to create a user-friendly command line interface the user interface will show the change log of the full projects life when the script has been in development.

```

Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.

Change Log: [Hit Enter To Exit.]

The script user can now make a file executable. Added on (Sat 27th Sep
2014).

The script user can now add users from a text file. Added on (Fri 26th Sep
2014).

The script user can now delete users from the system using a text file.
Added on (Fri 26th Sep 2014).

BugFixed on option 3, corrupt Hex values. Now deleted and changed on (Fri
14th mar 2015).

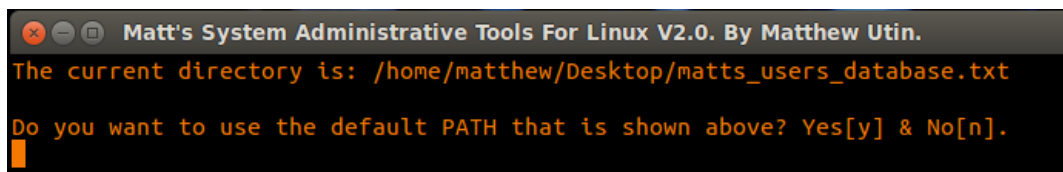
<OK>

```

Figure (10): Shows the script executing the whiptail command line interface.

8.1.9 Adding users from a file

When adding new users to the system the script user will need to select the menu option, this will in turn execute the code within the function. The main code when executed displays to the user a message prompting the script user, is the present working directory correct where the user account details file is stored. If the script user chooses a different path the code will then load that in as the working directory. Once the correct directory is chosen or the default path was used then code will read in the information from the file using place holders within the code to determine where to read the code. The code is set to read from the 4th line in the file as this is where the data is stored, also the place holders will load in 3 parts of data. This is the group, username and the password that the account is going to be using.



```
Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.
The current directory is: /home/matthew/Desktop/matts_users_database.txt
Do you want to use the default PATH that is shown above? Yes[y] & No[n].
█
```

Figure (11): Showing the user add code function from a file, within the script.

8.2.1 Project Progress and Pilot Study

The projects progression has been running at a steady pace, and completion was met on the planned schedule. There has also been some extra research preformed and included within this project. These are looking more into why BASH was used to create the script and not another language, for example Python or NASM. So to demine and embed the main point of using the BASH scripting language in this project. To create the administrative script. Some of the main findings that where identified are below:

8.2.2 NASM

NASM, (Nasm.us, 2015) is used in system development, for example creating core system code like hardware drivers. It is also a low level language. This means it communicates

to the CPU and system kernel when converted to byte code on the lowest level. This has the advantage of speed because NASM does not have to load in large software libraries and resources, for it to function correctly. But this means when you would like to program a simple task, you will have to write more code than using for example C++ (Cplusplus.com, 2015) and this will make it hard to understand where to start. Also for a new programmer the NASM code is very hard to read if you do not understand, how to work the computer systems memory storage. See (Appendix E) for an example adding a new group to a Linux system.

8.2.3 Python

Python is a high level language (Python.org, 2014) that has some great aspects, one of them is how the code is easy to learn and also easy to read, this makes it fast to implement a program on a system, it is also multi-platform being that it runs on Windows (windows.microsoft.com, 2015), Linux (Bhartiya, S. 2015) and Mac OS, (Apple, 2015). The only problem is if you want to use this language you will have to install all of the run files .e.g. libraries and resources for it to function correctly. There is also the problem of picking what version to use as there are updated versions but some Linux systems have the older versions, so you have to change the way you write your script to cater for what version the user is going to be using your script on.

Here is one example of different version code on how to take in user input: `raw_input()` Python 2.7.6, `input()` Python 3.

For an example of a user script created in python view: (Appendix F).

8.2.4 BASH

BASH is a Scripting language and is on most if not all Linux systems it is used to perform a number of automated tasks and does not need to be installed on to the system as it is normally the default scripting language on the Linux and UNIX systems, (Bhartiya, S. 2015). It is pretty fast to execute the code as you do not need to compile it like C++

(Cplusplus.com, 2015) for example. The only problem being is that it can be hard to understand at times as it uses somewhat of a C language, type syntax, unlike Python (Python.org, 2014) which can be easy to understand.

View: (Appendix G) to see the BETA - Test version of the BASH script.

Also the BETA test script was tested to see if there is any improvements needed before it is implemented into the final build. View: (Appendix H). To see the results from the script testing process.

So the project is going on the right path, the main Linux script is now finished and has been packed for install with the correct help file manuals to help the user use the script correctly.

See (Appendix I) to view the final finished script and see (Appendix J) to view the manual file for the completed script. To view the database file used to add the new users to the system view (Appendix K).

8.2.5 Project Planning

There has been no use of the risk plan that was created for this project as it has been on course. View: (Appendix B) to recap on the projects risks. There has been use of the contingency “timeslot 4” this is reflected within the Gantt chart, it was used because there was other course assessments that had to be handed in before starting the project review report. Apart from this the project has run smoothly. The Gantt chart was updated to show what has been completed within this project. View: (Appendix A) to see what has been completed in this project and how it is stuck to its schedule. Moreover the project planning level of achievement table (Appendix L) helped in with the overall planning metric when following what part of the script needed to be implemented in that week’s schedule.

8.2.6 Project Monitoring

There was no diversion from the project plan so far, as the project has not had any major problems. The only time that the contingency timeslot was used was when there was other course work to be completed and that was to set back starting the project review report and use the “contingency timeslot 4” within the Gantt chart, you can view this in (Appendix A). The methodology had no bad outcomes and impacts because SCRUM can be modified if there are any problems that present themselves mid though the project possess. You can view the methodology plan in (Appendix D).

8.2.6 Project Control

To start, there is no need for a change of the project plan, because it is running at its optimum level and has been working fine, there has also been no major problems within this project. The project metric that was used to keep track of the project was the Gantt chart and the Project planning level of achievement table (Appendix L), also sticking to the project aims and using the planned methodology.

This has been ensured and embedded using the contingency timeslots provided in the Gantt chart, also sticking to the risk plan if anything happens, within this project. To have the available backup plan to hand to keep the project performing at its maximum capacity.

The project log-file is also used to store any information that is going to be used within this project and backs up the amount of research preformed. View the available recourses to backup this information below: (Appendix A) Gantt chart, (Appendix B) risk table and (Appendix D) the methodology.

8.2.7 Conclusion

To conclude this section, it would be best to say that the project design was a success as, there were no issues with the final implementation of the user interface. For the project planning this was also a success because sticking to the correct project plan and

methodology as this has helped in the development of the script and its components, also managing the full project by exploring project risks (Appendix D) to create a logical backup plan if something unexpected happens. Moreover the project ran smoothly apart from the use of the contingency “timeslot 4” this is reflected within the Gantt chart, it was used because there were other course assessments that had to be handed in before starting the project review report.

9. Issues arising from implementation and Tests

9.1 Introduction

This section will explore the aspects of the implementation of the script, this will show if the script's main plan has been changed in anyway also showing how if the project plan has been followed. The Tests section will explain and break down, what the results were for the user survey for the final complete script giving an in-depth analysis.

9.1.1 Implementation

One element of this project was to identify what commands are used to add new users to the system. One problem was, that the commands used to add an admin user to the system. This is different in other Linux distributions for example, “SUSE Linux”, (Opensuse.org, 2015) uses the “ROOT” user group to add an administrator user to the system, but in Ubuntu to add and define an admin user, the “Admin” user group is used.

This has changed the way the project was set up, as now the script can only be run within the operating system Ubuntu Linux. This problem could have been overcome with a few (if statements) within the script code to check what system is being used and then only use the correct group for that system. Due to the time restrictions this was not implemented.

The other problem is the use of using different code editors. Why is this a problem? When developing the script and in code editing software the “char” e.g. character set

is sometimes saved in a different format and can corrupt the code within the script file. Most often this would happen when using the editing software called “G edit”, (Wiki.gnome.org, 2015) This was overcome by only using one main editor called “Kate”, (Kde.org, 2015) as this editor shows whitespace with unknown character sets. This will help to stop file corruption. This is not always the case as seen on the next section below, about the testing stage.

9.1.2 Tests

For the testing of the script, it was done by trialling the script for a week before fully publishing and implementing the script. The results were then included within a table in (APPENDIX H) the table shows all of the outcomes that were tested within the different parts of the script. There is an in-depth breakdown of the following testing outcomes below:-

A.1. Creating new a regular user

So the first part of the script to be reviewed and tested was creating a new regular user, the tests run on this part of the script were to try and create errors by pressing different keys on the keyboard also trying to submit a blank undefined data value or to crash the script. The only problem that was found from the test's, was that the code its self was not secure as it was in plain text and not encrypted.

A.2. Creating a root/admin user

The Create a root/admin user part of the script was the next section that was tested for any defects. The tests run on the Create a root/admin section is around the same lines of creating a regular user, but with the testing being a little more rigorous on the security aspects. For example testing to see if a non admin user can access the script and run it without ROOT access. Like the regular user section above, the only problem was that the code was in plain text and could be read by a non admin user to disassemble the code.

A.3. Adding a user-group to a user

When testing to see if the add a user-group to a user part was working within the script, the main tests that were used to test the code was, to try and intercept the code with random key press values. This code passed and did not need to be modified in any way.

A.4. Creating a new user-group

Testing the “creating a new user-group”, was not that difficult because it was very similar to testing the creating a new user as it was about the user inputting keys from the key board, so all that had to be done was to see if the input could crash the script or input blank data values. The creating a new user-group code did not fail any of the tests and passed, also the code did not need to be modified.

A.5. Showing user-groups that a user is linked to

Running the tests on “showing user-groups that a user is linked to” code section, was tested by looking to see if the main command used in Linux can be intercepted to output a different command by using different key presses from the keyboard, also the code was tested to see if it would crash. The code passed the tested on the code running correctly, but the user interface was a little unfriendly so a new user friendly interface needed to be implemented within the script for this section of code.

A.6. Showing all user-groups on the system

The tests that were used on this section of code was, just to test if the was running smoothly and the users interface was easy to use, as the base of the code used was a system command to show the users on the system. The code was also tested to see if it could be intercepted by the user using key presses. This code passed all of the tests.

A.7. Showing all users on the system

Like the “Showing all user-groups on the system” code above the tests were performed in the same way to test if the menu is user friendly and that the code

did not crash or could be intercepted using keys being, entered from the keyboard from the user. The code passed all of the tests with no problems.

A.8. About

The about code section within the script was tested to see if it would load correctly and does not crash the script. The about code passed the tests and did not need any changes implemented.

A.9. Remove a user-group from a system user

This code section was tested to on how well the user interface was when selecting the right options to remove a user-group from the system and to test that the user interface worked correctly when exiting the selected option early without any execution of the removal, this was tested because if the user just selected the menu by accident and wanted to end the process. The next stage was to test the code to see if it could try to remove blank unset values and to see if the code would crash doing so. The code passed all of the tests perfectly with nothing else needing to be implemented.

A.10. Remove a user-group from system

This code was tested in the same way as the “Remove a user-group from a system user” code was tested. The tests that were performed were to look into the user interface and main running of the code to see if it would crash. The code passed all of the tests, with no implications.

A.11. Remove User with the home directory or just the user account

With this code section, there were a number of tests performed to verify if the code is working correctly, this is because this code performs two different outcome depending on what the user has selected. The first part of the code to be tested is removing a user from the system without deleting the home directory. Key presses were tested and so was trying to delete the wrong directory with a blank value. This code section passed. The next part that was tested was removing a user with the home directory. The tests that were

performed was the same as the first part of the code, this also passed with flying colours.

A.12. Quit!

One of the most important parts to any script is to be able to exit the script when, all of the tasks performed by that user is completed. The tests performed on this part of code was to check if the script had exit correctly and that the Linux terminal clears the terminal window on exit. The code passed all of the tests with no problems.

A.13. The code element: options=("") script case opt selector

This was a code element that was tested, to see if it works correctly used as the selector when the user selects a main menu option. There was a problem when the selector code was run it would work when the users selected the option but when that option was finished it would go in an “infinite loop” and would not end the function. This has now been changed it now uses a different technique to select the functions within the script. Using a new function select technique with an exit loop, this now returns to the main menu now that the fix has been implemented.

9.1.3 Final testing

The full script was re-tested after being packaged, but one of the main code elements within the script that was coded in hexadecimal code had been corrupted, this could have been done by storing the script in a packaged gz.tar file or the file could have been re-saved in a text editor with a different character set. The code has now been fixed, by changing the corrupt hexadecimal code (Sanfoundry, 2013) to normal code, also the full script has now been repackaged. Here is the changed code extract below:-

Old code that has been corrupt.

```
add_a_usergroup_to_a_user() #opt 3
{
# Used the two commands to change the text strings to Hex. cat file.txt | hexdump -v -e "\\x" 1/1 "%02x"
""
# echo"sometext" | hexdump -v -e "\\x" 1/1 "%02x" ""
# eval `printf ""` Will be used to execute the hexadecimal code e.g. Base 16.
eval `printf "\x63\x6c\x65\x61\x72\x0a" #This is used to clear the command line screen above from past
commands to make the command prompt look more tidy. This makes it better/easier to read.
eval `printf
"\x65\x61\x64\x20\x2d\x70\x20\x22\x54\x68\x65\x20\x75\x73\x65\x72\x20\x67\x72\x6f\x75\x70\x20\x7
9\x6f\x75\x20\x77\x61\x6e\x74\x20\x74\x6f\x20\x61\x64\x64\x20\x74\x6f\x20\x74\x68\x65\x20\x75\x73
\x65\x72\x3a\x20\x22\x20\x67\x72\x6f\x75\x70\x0a" # read -p "The user group you want to add to the
user: " group
eval `printf
"\x65\x61\x64\x20\x2d\x70\x20\x22\x45\x6e\x74\x65\x72\x20\x74\x68\x65\x20\x75\x73\x65\x72\x3a\x2
0\x22\x20\x75\x73\x65\x72\x6e\x61\x6d\x65\x0a" # read -p "Enter the user: " username
eval `printf
"\x73\x65\x72\x6d\x6f\x64\x20\x2d\x61\x20\x2d\x47\x20\x24\x67\x72\x6f\x75\x70\x20\x24\x75\x73\x65
\x72\x6e\x61\x6d\x65\x0a" # usermod -a -G $group $username #code to add a usergroup to the user.
}
```

Extract (1). Showing the corrupt hexadecimal code.

New code that has replaced the corrupt hexadecimal code.

```
add_a_usergroup_to_a_user() #opt 3
{
# Used the two commands to change the text strings to Hex. cat file.txt | hexdump -v -e "\\x" 1/1 "%02x"
""
# echo"sometext" | hexdump -v -e "\\x" 1/1 "%02x" ""
# eval `printf ""` Will be used to execute the hexadecimal code e.g. Base 16.
eval `printf "\x63\x6c\x65\x61\x72\x0a" #This is used to clear the command line screen above from past
commands to make the command prompt look more tidy. This makes it better/easier to read.
read -p "The user group you want to add to the user: " group
read -p "Enter the user: " username
usermod -a -G $group $username #code to add a usergroup to the user.
}
```

Extract (2). Showing the new fixed code.

This change was included within the change log as: “BugFixed on option 3, corrupt hexadecimal values. Now deleted and changed on (Fri 14th mar 2015)”.

9.1.4 User survey results

A user survey was created to get a user's perspective of the Linux administrative BASH script, the results for this survey can be found in (Appendix M) the survey was conducted with a pool of ten participants and with a mixture of the participants being non-Linux users and some being experienced Linux users. The main reason to use a wide mixture of participants, was to get a better understanding of how user friendly the script is, also to get a non-Linux users perspective. So the results are not one-sided.

The overview of the Results

- The results that were collected were all in a positive manner ranging towards 4 to 5 on how user friendly the script is to use, there were also comments about how the manual “.man” file helped the fellow participants navigate around the script.
- 9 out of 10 participants said that the script was able to cut down the workload of performing daily server tasks.
- The survey results also gave some great information on how reliable the script has been to use. 9 participants ticked the box with the highest level of reliability and only one participant ticked the box ranking of 3, this being the middle range out of 5 being the highest ranking.
- None of the 10 participants came across any problems running and using the script.
- Out of 10 participants 8 were experienced Linux users and 2 had no experience within Linux.
- All 10 participants said they would promote the script with, some of the comments saying they would show the script to a Linux system administrator.

- Only one participant asked for an improvement to be made to the script and this was to be able to add custom user options to the script.
- Out of all of the participants there was one comment that stuck out, on the question of was BASH the correct scripting language to use. The comment was: you could next time try to use Ruby, C Shell or Python to create the script.
- There were 8 participants very satisfied with the script and 2 that were quite satisfied with the overall script.
- All of the last option comments of the survey have been very upbeat for example one comment was saying how “interesting the script was considering not being a Linux user.” One other comment said it was a “good handy script”.

9.1.5 User survey results conclusion

From the following results above, the script has gained some great input in this survey and shows that it was a good project in creating a Linux script that can cut the workload by automating daily server tasks, also that every participant wanted to promote the use of the script was a great sign that it has done well without a doubt.

9.1.6 Conclusion

To conclude after the tests were performed within this project the results were conclusive that there were not any underlying problems and even though there were minor problems, for example the user interface needed to be changed etc. It was very easy to overcome these problems with using the time planned for debugging, also using the allocated space within the SCRUM methodology to take control of any problems that could have been in the script, when it was being developed. One problem that came up was that Ubuntu uses a different type of administrator group, on its user accounts than most of the other Linux distributions that would normally use the ROOT group, this was discussed above but the current script has not had any cross platform implementation embedded within it. The last problem that was identified was that there were corrupt hexadecimal code inside the script when final testing was underway, this was quickly

fixed as the corrupt hexadecimal code was replaced with basic code. This then worked and there was a change log created to show the bug fix implementation. But other than that the testing was a complete success and showed it was done rigorously to get the best possibly result.

10. Evaluation and Conclusions

This project was created to look into creating an administrative BASH script for the ACME Company to explore the concept of solving the problem of cutting the company's daily tasks on their Linux system and in turn sped up the company's workflow. The administrative BASH script had one major bug when it was being developed, this was that some corrupt Hex values were placed in the "add a new user" section of the main script code, this was overcome with ease by just using the default code.

The survey results from how the public perceived the script had a great outcome in showing that the script was a success in performing sever tasks, also every one of the participants said they would recommend the use of the script.

All objectives within this project have been achieved

1. To research and identify the most useful commands, that are used within Linux, for system administration. (Completed) ✓
2. Find out how to implement those command line commands into a Linux BASH Script. (Completed) ✓
3. Design and create a Linux script and test/debug the script to see if it has any problems or if there could be any improvements. (Completed) ✓
4. Trial out a small sample for testing in a beta form to see if there are any unforeseen problems in the testing stage that can be fixed, also any improvements that could be made to the script. (Completed) ✓

5. After all of the script has had the fixes that are needed, this would be the time to finish the script off and package it for install, with the correct software licences and help files for download. (Completed) ✓

All aims were achieved and here is the end result, the fully packaged completed script:

<https://drive.google.com/file/d/0B-KNwipoXnOqQXh3QWRCNmJON0U/view?usp=sharing>

To conclude this project, it has been a very enjoyable experience because of the amount new knowledge gained and also being able to convey all of that knowledge within this project. The project went well without any possible hitches, but the end outcome for the project has changed a little because the script now only uses Ubuntu specific commands, this was shown in the section “Issues arising from implementation and Tests” because of the different command use of adding an admin user to the system.

11. Recommendations

The recommendations, which would be advised for this project, would be to implement the checking of the operating system, so that the script would know what commands to use if they were specifically for that operating system. Furthermore the full implementation of the script could use full code encryption to counter the users from reading the source code of the document. This was done to an extent in this project but only to show it can be implemented in the most unsecure areas of the script code, moreover the script could incorporate more error checking as the current final script within this project uses system command line error checking and very little scripted error checking. This should be implemented in a years' time in 2016 because it's a high priority aspect of the scripts functionality.

Future work

- The script could be used and implemented in a cloud server based environment, running virtualized operating systems, to manage new user creation hosted on the backend server of the provider. This would automate the process of administration on the cloud server.
- Identify and implement an automatic version of the script, ported to a fully-fledged programming language, which can add new users to the system in the background without the administrator having to insert parameters in manually and using the user account data from a database.

12. References and Bibliography

- [1]. SHAN, L., 2011. Exploration of education reform based on 32-bit assembly language programming. Computer Science & Education (ICCSE), 2011 6th International Conference on, 595-599.
- [2]. CHAVES, J.C., J. NEHRBASS, B. GUILFOOS, J. GARDINER, S. AHALT, A. KRISHNAMURTHY, J. UNPINGCO, A. CHALKER, A. WARNOCK and S. SAMSI, 2006. Octave and Python: High-Level Scripting Languages Productivity and Performance Evaluation. HPCMP Users Group Conference, 2006, 429-434.
- [3]. HE, K., X. XU and Q. YUE, 2008. A secure, lossless, and compressed Base62 encoding. Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on, 761-765.
- [4]. SINGH, G. and SUPRIYA, 2013. Modified Vigenere Encryption Algorithm and Its Hybrid Implementation with Base64 and AES. Advanced Computing, Networking and Security (ADCONS), 2013 2nd International Conference on, 232-237.
- [5]. LEE, G.L., D.H. AHN, B.R. DE SUPINSKI, J. GYLLENHAAL and P. MILLER, 2007. Pynamic: the Python Dynamic Benchmark. Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on, 101-106.
- [6]. JUN, L. and L. LING, 2010. Comparative research on Python speed optimization strategies. Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on, 57-59.
- [7]. BASEGMEZ, F., 2002. Extending a scientific application with scripting capabilities. Computing in Science & Engineering 4 (6), 52-59. [Journal].

- [8]. ORTIZ, A., 2007. An introduction to metaprogramming. Linux Journal 158(6), 15. [Journal].
- [9]. XIUMING, LUO XIAO REN YONG SHAN, 2004. PYTHON BASED MIXED-LANGUAGE PROGRAMMING AND ITS IMPLEMENTATION [J]. Computer Applications and Software 12, 007. [Journal].
- [10]. BEIKE, W.G.Z., 2010. RESEARCH ON EMBEDDED SCRIPT BASED ON PYTHON [J]. Computer Applications and Software 3, 033. [Journal].
- [11]. SCHMIDT, J.R. and S. BRYSON, 1997. Animation in the Unix environment: a primer, Using bash. Computers in Physics 11(6), 618-623. [Journal].
- [12]. ADAMS, P.D., R.W. GROSSE-KUNSTLEVE, L. HUNG, T.R. IOERGER, A.J. MCCOY, N.W. MORIARTY, R.J. READ, J.C. SACCHETTINI, N.K. SAUTER and T.C. TERWILLIGER, 2002. PHENIX: building new software for automated crystallographic structure determination, using python. Acta Crystallographica Section D: Biological Crystallography 58(11), 1948-1954. [Journal].
- [13]. Duntemann, J. and Duntemann, J. (2009). Assembly language step-by-step. 3rd ed. Indianapolis, IN: Wiley Pub., p.202. [Book].
- [14]. Kusswurm, D. (2015). Modern x86 assembly language programming. [S.l.]: Apress, p.310. [Book].
- [15]. Assembler, (2013). The Netwide Assembler. [online] SourceForge. Available at: <http://sourceforge.net/projects/nasm/> [Accessed 6 Jan. 2015].
- [16]. Nasm.us, (2015). The Netwide Assembler: NASM. [online] Available at: <http://www.nasm.us/> [Accessed 6 Jan. 2015].

- [17]. Nasm.us, (2015). NASM Manual. [online] Available at:
<http://www.nasm.us/doc/nasmdoc2.html> [Accessed 6 Jan. 2015].
- [18]. Kioskea, (2014). Compiling an assembly program with Nasm. [online] Available at:
<http://en.kioskea.net/faq/1559-compiling-an-assembly-program-with-nasm> [Accessed 6 Jan. 2015].
- [19]. NASM — The Netwide Assembler. (2012). 1st ed. [ebook] US: The NASM Development Team. Available at:
<http://www.nasm.us/pub/nasm/releasebuilds/2.10.04/doc/nasmdoc.pdf> [Accessed 6 Jan. 2015].
- [20]. Stackoverflow.com - mattlinux1, (2014). NASM code with user output, and then input into an argument. (code error). [online] Available at:
<http://stackoverflow.com/questions/27384874/nasm-code-with-user-output-and-then-input-into-an-argument-code-error> [Accessed 6 Jan. 2015].
- [21]. Stackoverflow.com - mattlinux1, (2014). Can you send a bash command to the Linux terminal using NASM. [online] Available at:
<http://stackoverflow.com/questions/27135443/can-you-send-a-bash-command-to-the-linux-terminal-using-nasm> [Accessed 6 Jan. 2015].
- [22]. Python.org, (2014). Welcome to Python.org. [online] Available at:
<https://www.python.org/> [Accessed 6 Jan. 2015].
- [23]. Wiki.python.org, (2012). WhileLoop - Python Wiki. [online] Available at:
<https://wiki.python.org/moin/WhileLoop> [Accessed 6 Jan. 2015].
- [24]. Python For Beginners, (2012). How to use Loops in Python. [online] Available at:
<http://www.pythonforbeginners.com/loops-2/for-while-and-nested-loops-in-python> [Accessed 6 Jan. 2015].

- [25]. Tutorialspoint.com, (2014). Python Variable Types. [online] Available at: http://www.tutorialspoint.com/python/python_variable_types.htm [Accessed 6 Jan. 2015].
- [26]. Stavros.io, (2006). Tutorial - Learn Python in 10 minutes - Stavros' Stuff. [online] Available at: <http://www.stavros.io/tutorials/python/> [Accessed 6 Jan. 2015].
- [27]. Trypython.org, (2014). Try Python: Interactive Python Tutorial in the Browser. [online] Available at: <http://www.trypython.org/> [Accessed 6 Jan. 2015].
- [28]. Lutz, M. (2011). Programming Python. Beijing: O'Reilly, pp.20-70.[Book].
- [29]. Shaw, Z. (2013). Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code. 3rd ed. U.S: Addison Wesley, pp.100-154. [Book].
- [30]. Drewsymo.com, (2013). Simple Base64 Encode & Decode on Mac OSX / Linux with OpenSSL | Drew Morris. [online] Available at: <http://drewsymo.com/2013/11/simple-base64-encode-decode-on-mac-osx-linux-with-openssl/> [Accessed 6 Jan. 2015].
- [31]. Fm4dd.com, (2014). The Base64 algorithm description. [online] Available at: http://fm4dd.com/programming/base64/base64_algorithm.htm [Accessed 6 Jan. 2015].
- [32]. Unix-manuals.com, (2001). ASCII Table - table of ASCII codes. [online] Available at: <http://www.unix-manuals.com/refs/misc/ascii-table.html> [Accessed 6 Jan. 2015].
- [33]. Docs.cs.up.ac.za, (2004). System Call Table. [online] Available at: http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html [Accessed 6 Jan. 2015].

[34]. Sanfoundry, (2013). 10+ Practical “hexdump” Command Usage Examples in Linux. [online] Available at: <http://www.sanfoundry.com/10-practical-hexdump-command-usage-examples-in-linux/> [Accessed 6 Jan. 2015].

[35]. Manpages.courier-mta.org, (2014). base64(1) — Linux manual pages. [online] Available at: <http://manpages.courier-mta.org/htmlman1/base64.1.html> [Accessed 6 Jan. 2015].

[36]. Cyberciti.biz, (2010). HowTo: Linux / UNIX Create a Manpage. [online] Available at: <http://www.cyberciti.biz/faq/linux-unix-creating-a-manpage/> [Accessed 6 Jan. 2015].

[37]. Vargas Mas, P. (2010). Creating Custom Man Pages | Linux Journal. [online] Linuxjournal.com. Available at: <http://www.linuxjournal.com/content/creating-custom-man-pages> [Accessed 6 Jan. 2015].

[38]. Man7.org, (2014). syscalls(2) - Linux manual page. [online] Available at: <http://man7.org/linux/man-pages/man2/syscalls.2.html> [Accessed 6 Jan. 2015].

[39]. Manpages.ubuntu.com, (2014). Ubuntu Manpage: syscalls - Linux system calls. [online] Available at: <http://manpages.ubuntu.com/manpages/hardy/man2/syscalls.2.html> [Accessed 6 Jan. 2015].

[40]. Kerrisk, M. (2014). The Linux man-pages project. [online] Kernel.org. Available at: <https://www.kernel.org/doc/man-pages/> [Accessed 6 Jan. 2015].

[41]. Beazley, D. and Jones, B. (2013). Python cookbook. Beijing: O'Reilly, p.40. [Book].

[42]. Linfo.org, (2014). Beginning Linux command line tutorial. [online] Available at: http://www.linfo.org/command_line_lesson_1.html [Accessed 4 May. 2014].

[43]. Oak.cs.ucla.edu, (2014). How to Write a Basic Shell Script. [online] Available at: <http://oak.cs.ucla.edu/cs144/projects/unix/shell.html> [Accessed 4 May. 2014].

[44]. William Shotts, J. (2014). Writing shell scripts - Lesson 1: Writing your first script and getting it to work. [online] Linuxcommand.org. Available at: <http://linuxcommand.org/wss0010.php> [Accessed 4 May. 2014].

[45]. NixCraft. (2014). Linux Shell script to add a user with a password to the system - nixCraft. [online] Available at: <http://www.cyberciti.biz/tips/howto-write-shell-script-to-add-user.html>. [Accessed online 13 February 2014].

[46]. Ask Ubuntu, (2014). How to remove a user from a group? - Ask Ubuntu. [ONLINE] Available at: <http://askubuntu.com/questions/80115/how-to-remove-a-user-from-a-group>. [Accessed online 13 February 2014].

[47]. Arachnoid.com, (2014). Bash Shell Programming in Linux . [online] Available at: http://www.arachnoid.com/linux/shell_programming.html. [Accessed online 13 February 2014].

[48]. Robbins, A R. (2010). bash Pocket Reference, Help for Power Users and sys Admins. 1st ed. USA: O'Reilly Media Inc.. [Book].

[49]. Negus, C.N. (2012). Linux Bible. 5th ed. USA: John Wiley & Sons, Inc.. [Book].

[50]. HowtoForge. (2014). How To Add Users To Linux OS From A Text file [ONLINE] Available at: <http://www.howtoforge.com/how-to-add-linux-system-users-from-a-text-file>. [Accessed 27 September 2014].

- [51]. Ezust, A. (2014). Bash: Using Colors. [online] Available at: <http://www.csc.uvic.ca/~sae/seng265/fall04/tips/s265s047-tips/bash-using-colors.html>. [Accessed 27 September 2014].
- [52]. Flozz, M. (2014). bash:tip_colors_and_formatting. [online] Available at: http://misc.flogisoft.com/bash/tip_colors_and_formatting. [Accessed 27 September 2014].
- [53]. Eisenberg, D. (2014). The chmod command. [online] Available at: http://catcode.com/teachmod/chmod_cmd.html. [Accessed 27 September 2014].
- [54]. Lam, S. and Herbert, S. (2014). whiptail(1): Linux man page. dialog boxes from shell scripts - Linux man page. [online] Available at: <http://linux.die.net/man/1/whiptail>. [Accessed 27 September 2014].
- [55]. Stack Overflow. (2014). how to read file from line x to the end of a file in bash. [online] Available at: <http://stackoverflow.com/questions/14110223/how-to-read-file-from-line-x-to-the-end-of-a-file-in-bash>. [Accessed 28 September 2014].
- [56]. Stunned. (2014). Shell Script to create user in Linux using text file data. [online] Available at: <http://forums.devshed.com/scripts-94/shell-script-create-user-linux-using-text-file-data-81336.html>. [Accessed 28 September 2014].
- [57]. SOLOMON, A., D. SANTAMARIA and R. LISTER, 2006. Automated Testing of Unix Command-line and Scripting Skills. Information Technology Based Higher Education and Training, 2006. ITHET '06. 7th International Conference, pp.120-125.
- [58]. MCNABB, A., J. LUND and K. SEPPI, 2012. Mrs: MapReduce for Scientific Computing in Python. High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion , pp.600-608.

- [59]. MOISE, D.L. and K. WONG, 2006. Extracting Facts from Perl Code. Reverse Engineering, 2006.WCRE '06.13th Working Conference, pp.243-252.
- [60]. MENDEL, C., 10th March 2014. Advanced Bash-Scripting Guide [online]. USA: The Linux Documentation Project [Viewed 18 Oct 2014]. Available From: <http://www.tldp.org/LDP/abs/absguide.pdf> [eBook].
- [61]. Gnu.org, (2014). GNU Lesser General Public License v3.0- GNU Project - Free Software Foundation. [online] Available at: <https://www.gnu.org/licenses/lgpl.html> [Accessed 4 May. 2014].
- [62]. Dickey, T. (2014). XTERM - Terminal emulator for the X Window System. [online] Invisible-island.net. Available at: <http://invisible-island.net/xterm/xterm.html> [Accessed 4 May. 2014].
- [63]. Help.gnome.org, (2014). Terminal. [online] Available at: <https://help.gnome.org/users/gnome-terminal/stable/> [Accessed 4 May. 2014].
- [64]. Ascii-table.com, (2015). ANSI Escape codes VT100 / VT52. [online] Available at: <http://ascii-table.com/ansi-escape-sequences-vt-100.php> [Accessed 4 May. 2014].
- [65]. Bhartiya, S. (2015). The Top 11 Best Linux Distros for 2015 | Linux.com. [online] Linux.com | The source for Linux Information. Available at: <https://www.linux.com/news/software/applications/810295-the-top-11-best-linux-distros-for-2015> [Accessed 5 May 2015].
- [67]. Kde.org, (2015). KDE - Kate - Advanced Text Editor. [online] Available at: <https://www.kde.org/applications/utilities/kate/> [Accessed 5 May 2015].
- [68]. Wiki.gnome.org, (2015). Apps/Gedit - GNOME Wiki!. [online] Available at: <https://wiki.gnome.org/Apps/Gedit> [Accessed 5 May 2015].

[69]. Cplusplus.com, (2015). A Brief Description - C++ Information. [online] Available at: <http://www.cplusplus.com/info/description/> [Accessed 5 May 2015].

[70]. windows.microsoft.com, (2015). Windows - Microsoft Windows. [online] Available at: <http://windows.microsoft.com/en-us/windows/home> [Accessed 5 May 2015].

[71]. Apple, (2015). Apple - OS X Yosemite - Overview. [online] Available at: <https://www.apple.com/uk/osx/> [Accessed 5 May 2015].

[72]. Opensuse.org, (2015). openSUSE.org. [online] Available at: <https://www.opensuse.org/en/> [Accessed 5 May 2015].

13. IEEE References

[73]. IEEE Computer Society, "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) System Administration - Part 2: Software Administration," IEEE Std 1387. 2-1995, pp. 0_3, 1996.

[74]. L. Yuqing, X. Hao and L. Xiaohui, "The Research of Performance Test Method for Linux Process Scheduling," Information Science and Engineering (ISISE), 2012 International Symposium on, pp. 216-219, 2012.

[75]. D. Turk and J. Bausch, "Virtual Linux servers under z/VM: Security, performance, and administration issues," IBM Syst J, vol. 44, pp. 341-351, 2005.

[76]. E. Leibovitch, "The business case for Linux," Software, IEEE, vol. 16, pp. 40-44, 1999.

- [77]. P. Hinds and S. Kiesler, "Essence of Distributed Work: The Case of the Linux Kernel," *Distributed Work*, pp. 381-404, 2002.
- [78]. Y. Tian, J. Lawall and D. Lo, "Identifying Linux bug fixing patches," *Software Engineering (ICSE), 2012 34th International Conference on*, pp. 386-396, 2012.
- [79] J. Emmons, *Easy Linux Commands : Working Examples of Linux Command Syntax*. Kittrell, NC: Rampant, 2006.
- [80]. X. Wang, J. Lin, Y. Zou and L. Zha, "A Login Shell for Computing Grid," *EScience, 2008. eScience '08. IEEE Fourth International Conference on*, pp. 762-769, 2008.
- [81]. S. Yan, "BASH Scripts Based Deployment and Management Framework for Multi-component Application in Unix Environment," *Jisuanji Xitong Yingyong - Computer Systems and Applications*, vol. 21, pp. 61, 2012.
- [82]. K. Mazurak, "ABASH:finding bugs in bash scripts," *Conference on Programming Language Design and Implementation: Proceedings of the 2007 Workshop on Programming Languages and Analysis for Security; 14-14 June 2007*, pp. 105, 2007.

14. Appendices

APPENDIX A: Gantt chart

The Gantt chart below displays the project schedule and the completion times of each component, within the project, also showing the project completion and finishing times.

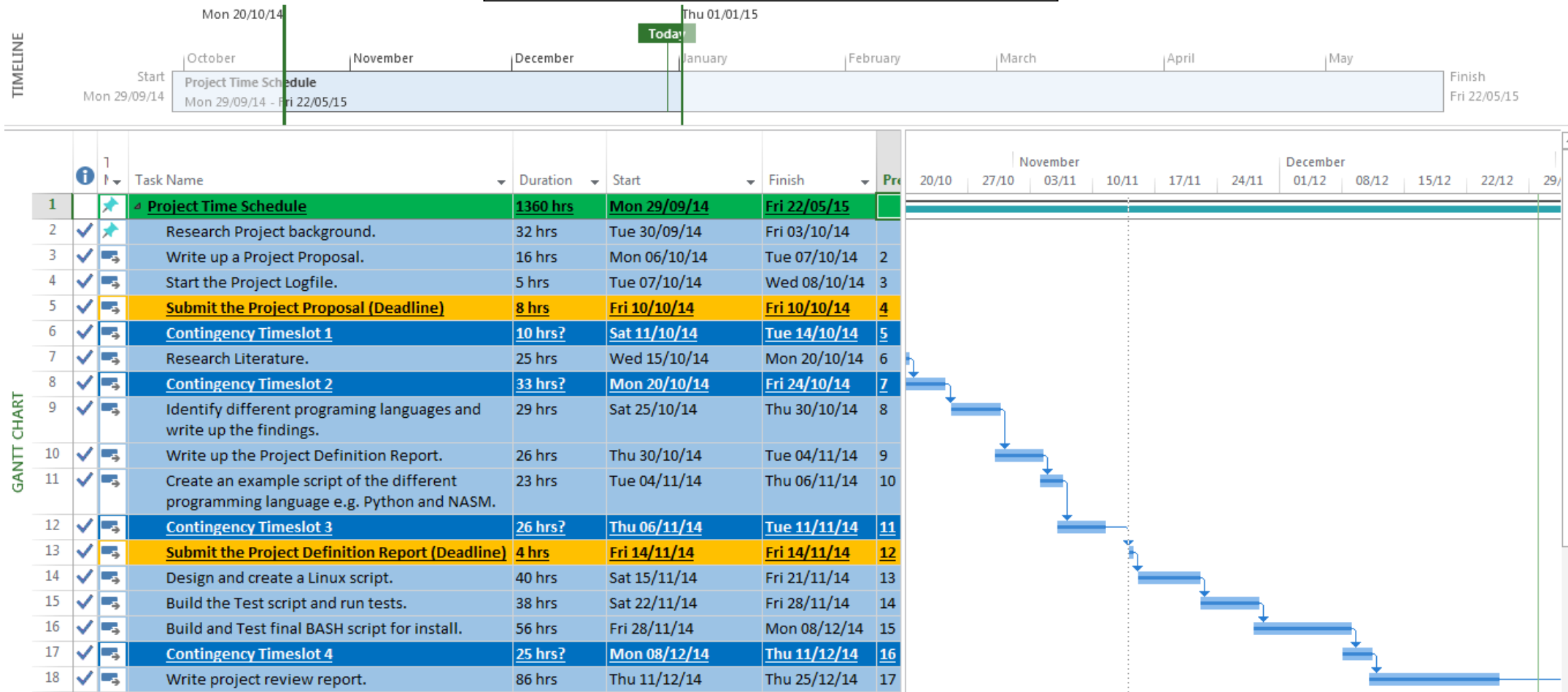


Figure (12): Start of the completed Gantt chart.

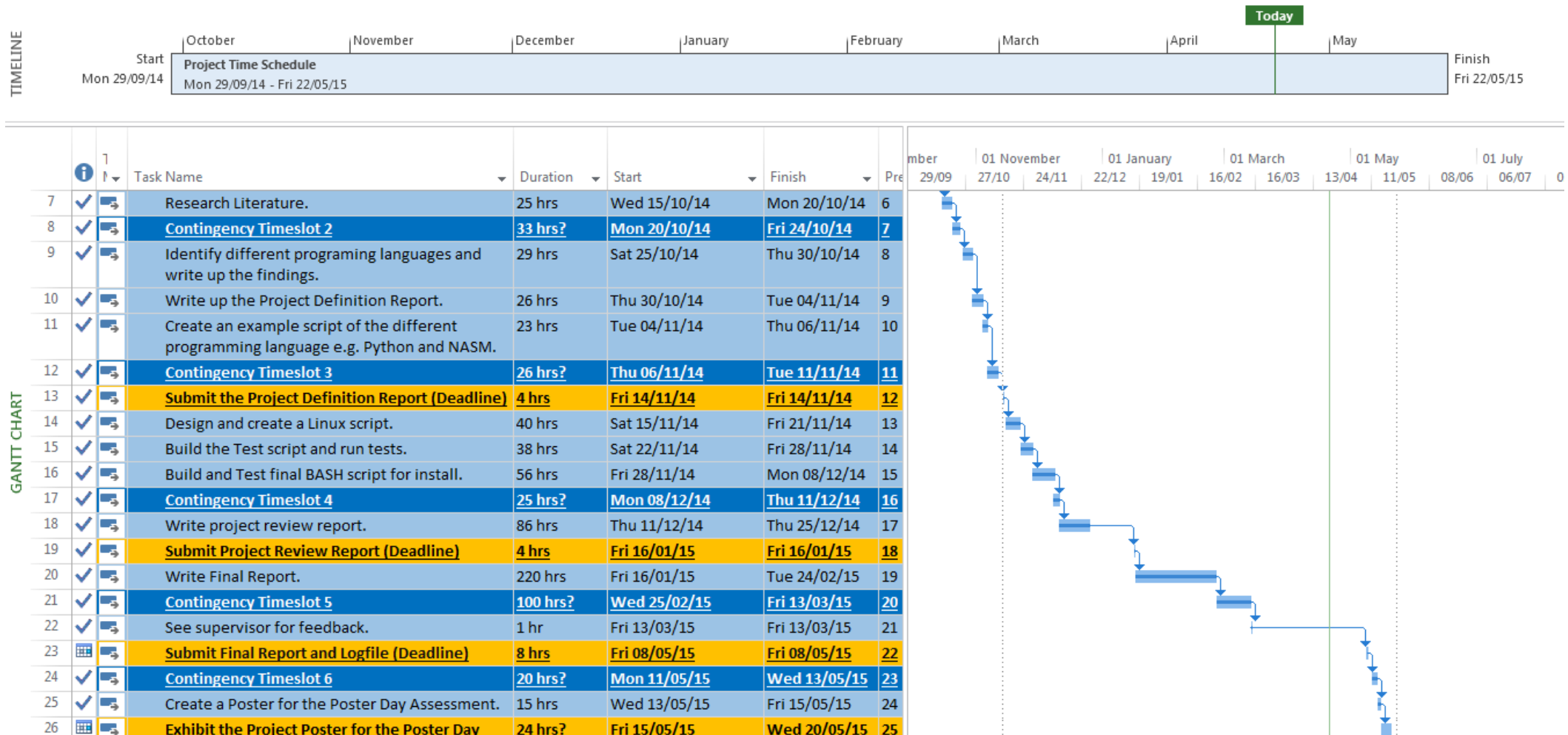


Figure (13): End of the completed Gantt chart.

APPENDIX B: Risk Table

<u>Risk</u>	<u>Likelihood</u>	<u>Impact</u>	<u>Impact description</u>	<u>Plan/Contingency</u>
Software goes down.	Medium	High	The software would not start up or it comes up with errors, upon execution.	To stop this from happening, also backing up copy of the software on a different computer system. To be safe.
Broken equipment.	Low	Medium	The computer or screen etc. is broken or in a non-working state.	A good way to counter act this would be by getting insurance for the equipment.
Linux script coding error.	Low	Low	The code does not start up, because the code is incorrect.	The best way to get past this would be by looking at the code and seeing if something is wrong also would be to create a test plan of the code beforehand.
The Linux scripting directory is not found or will not load.	Medium	Low	The directory is not found or is only accessed by an admin.	A good way to get around this would be, logging in as admin also programing on a load of different systems to know that the script would correctly.
Back up of files fails.	High	High	Lose of files this would mean that the system admin would have to start all over again.	If a RAID type system back up then this would make this less of a problem.

				Also save the data in different locations to minimize the risk.
A fire with no disaster recovery plan.	Low	High	This would create a loss in data and equipment also money loss in repairs.	The best way would be to have a backup location to work and save data. Also create a good disaster recovery plan.

Table (2): (Table showing the risk that could arise during the project).

APPENDIX C: Script Diagram and Pseudocode

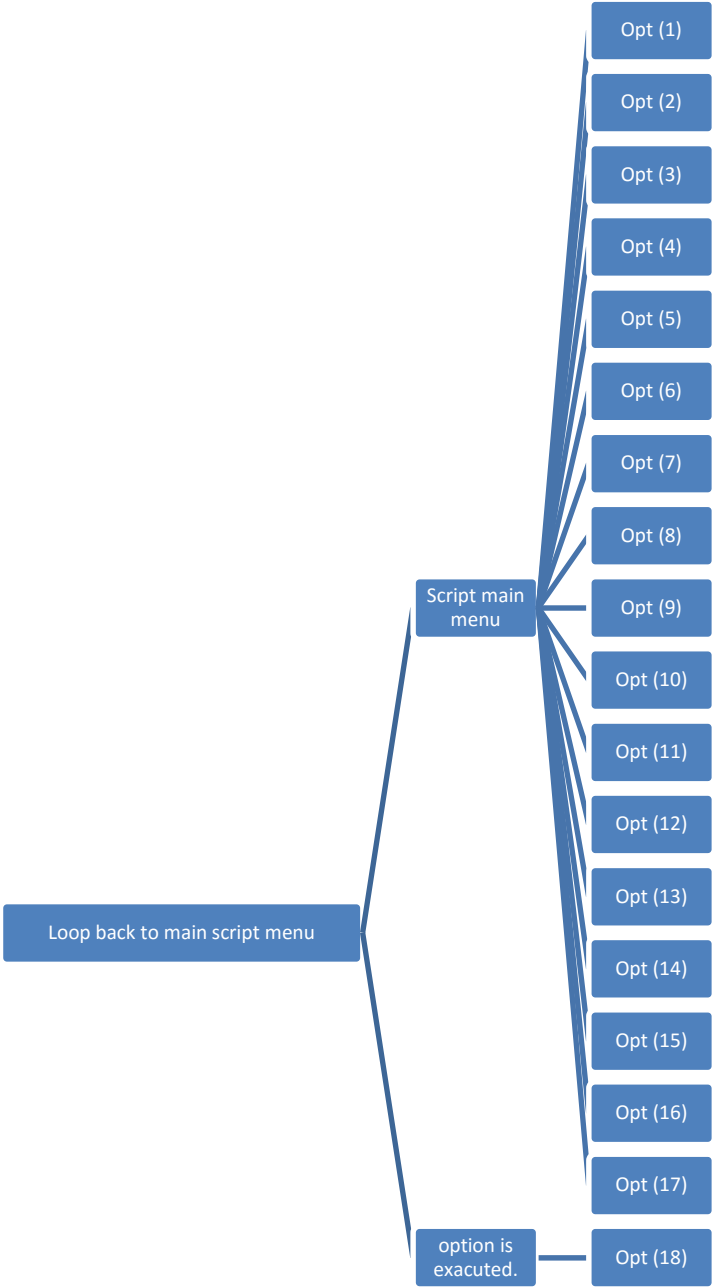


Figure (14): (Diagram showing, how the Script will perform).

This is a diagram of how the script will be implemented to function, when the user selects an option within the script. This will then execute the BASH loop back function. As you can see Above in Figure [14].

Here is the list of the Options that are going to be used above within this BASH script.
This will be shown below:-

- 1) Create a regular user.
- 2) Create a root/admin user.
- 3) Add a usergroup to a user.
- 4) Create a new usergroup.
- 5) Show the usergroups that a user is linked to.
- 6) Show all usergroups on the system.
- 7) Show all users on the system.
- 8) Remove a usergroup from a user.
- 9) Remove/delete usergroup from system.
- 10) Remove User and home directory or just the user account.
- 11) Make a file executable.
- 12) Add users to the system from a text file.
- 13) Delete users from the system using a text file.
- 14) [EXIT!]
- 15) (About.)
- 16) (Change Log.)
- 17) (System version information.)
- 18) (Man/manual Page.)

C.1. Pseudocode

This is going to be a brief example of how the script should work, when it is created, the script will work on a loop function, so that it will loop back to the menu when the user, has executed the element of the script that was chosen in the main script menu. For example here is the Pseudocode below:-

How the main menu will work.

User starts up the Script in the [main menu function](#).

IF the user **SELECTS** the option **1**

THEN the script will execute that function ((**1**) [Create a regular user.](#)).

Once the script has finished within function ((**1**) [Create a regular user.](#)) it will **RETURN** back, to the [main menu function](#).

Example of hard code that may be used e.g. (language BASH):

```
Matts_create _user() #The menu calls this function.
{
    #User add script would go here.
```

```

}

matts_menu ()
{
    opt=1 #options start
    while [ $opt -le 4 ] #Start of option loop.
    do
        echo -e " 1) Create a new user. " #The viewable options to pick.
        echo -e " 2) EXIT! "
        read -p "Select your option: (1-2)? " opt #Where it says opt it reads in what the
        user has entered.
        case $opt in #code option list below.
            1) Matts_create _user();;
            2) exit 0 ;;
            *) echo -e "You need to select a number within the list!" #If the user types in a
            wrong option number.
        esac
        echo -e "Type 1 to go back to the menu." #To go back to the main menu the user
        types 1.
        read opt #reads the input option if the user typed 1 for example.
    done
}
matts_menu #loads the menu function at start.
clear #Clears the screen above.
exit 0 #Exit code to end the script.

```

APPENDIX D: Scrum Methodology

<u>Type of Methodology</u>	<u>Description</u>
Scrum	Scrum is the most used methodology used within programming, because of the way it takes in many aspects of other methodologies for example spiral in the way it uses sprint terminology when splitting the roles in priority.
Waterfall	Waterfall is a downwards moving methodology that moves down when a task is completed. Is used in roles like construction for example.
Spiral model	Spiral, like it says moves round every time you complete a task, it tasks more of an objective role on managing a project, its used in design and managing risk.

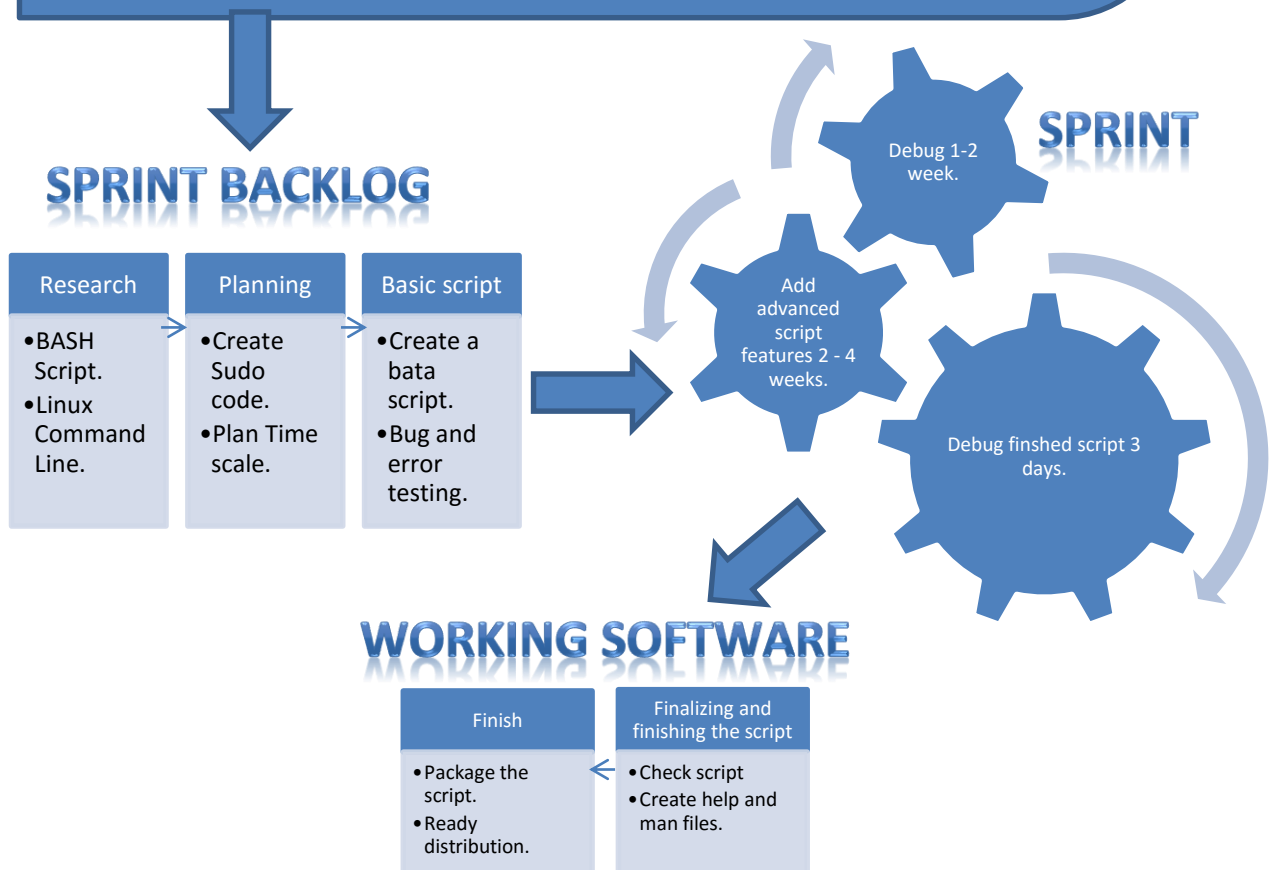
Table (3): (Methodology comparison).

As you can see from the above Table [3]. That the only option was to use Scrum as it is suited to this project in the correct way.

D.1 Scrum project plan

PRODUCT BACKLOG

1. Create a regular user script.
2. Create a root/admin user script.
3. Create a Add a usergroup to a user script.
4. Create a new usergroup script.
5. Create a Show the usergroups that a user is linked to script.
6. Create a Show all usergroups on the system script.
7. Create a Show all users on the system script.
8. Create a Remove a usergroup from a user script.
9. Create a Remove/delete usergroup from system script.
10. Create a Remove User and home directory or just the user account script.
11. Create a Make a file executable script.
12. Create a Add users to the system from a text file script.
13. Create a Delete users from the system using a text file script.
14. Create a About page script.
15. Create a Change Log page script.
16. Create a System version information page script.
17. Create a Man/manual Page script.



The chart (“Figure 15”) below shows the project working hours and what the predicted outcome of the projects workflow will be, when sticking to the Scrum Methodology. (Script completion).

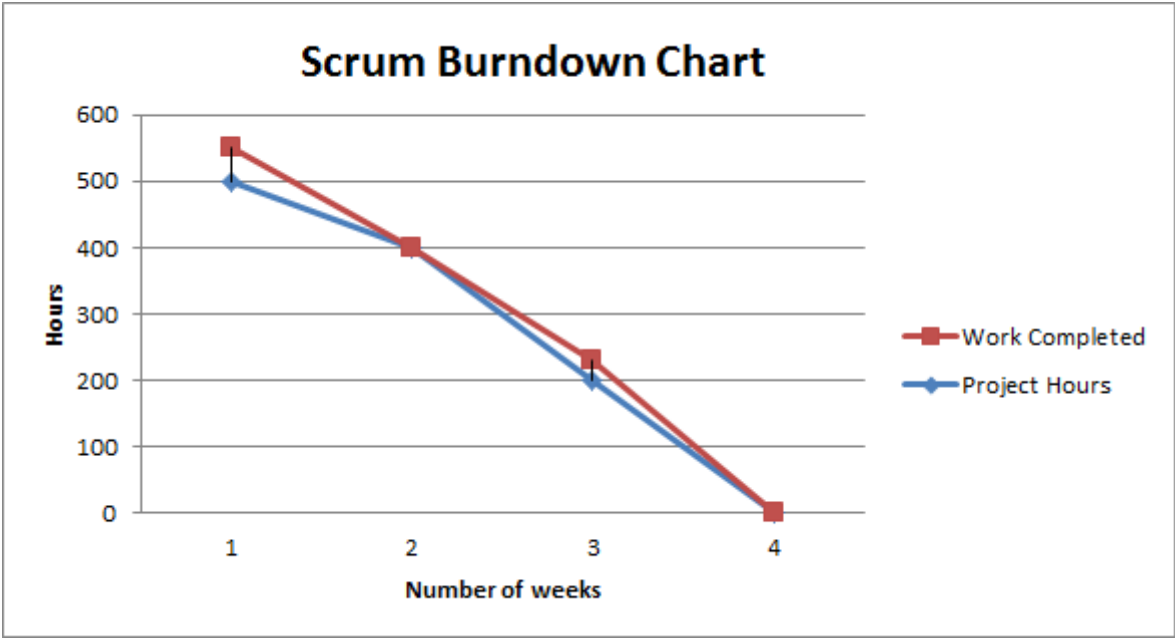


Figure (15): (Linux project Scrum methodology).

APPENDIX E: NASM CODE

Here is the code download link, to have a better view of the code: (Filename: myasm.asm)

<https://drive.google.com/file/d/0B-KNwipoXnOqdHFNQk5DdWdoX00/view?usp=sharing>

```
;-----NSAM--By-Matthew-Utin--Add-Group-----
section .data                ; Initialized data, can be variable's
userg:    db "Type in the usergroup to add to the system: ",10 ; 1st string, 10 is used
to drop down a line.
userg_L:   equ $-userg ; string length, automatically updates how long the string is, so
you don't have to type 10 etc. every time its changed
respsn:    db "You have added the system usergroup: ",10      ; 2nd string, 10 is used
to drop down a line. 0 is used to end the line.
respsn_L   equ $-respsn ; 2nd string length, automatically updates how long the string
is, so you don't have to type 10 etc. every time its changed
command:   db '/usr/sbin/groupadd', 0h ; Command to execute.
argu:      dd command ; Variable is passed to, then be executed.
          dd userg_V   ; Arguments passed to the command line.
          dd 0h        ; Struct.
enviro:     dd 0h      ; No arguments are needed for the environment
variables.

;=====
section .bss                ; Uninitialised data, also can be variable's
userg_V resb 255 ; Reserves 255 bytes of data.

;=====

section .text                ; Asm core code. any of the sections can be in any order.
global _start:              ; Makes _start the global function

_start:                     ; Makes _start the global function
    mov     eax, 0x04        ; Store the system call code = 4 e.g. = kernel function
(SYS_WRITE) opcode (04)
    mov     ebx, 0x01        ; 1 is the code to where to write the Asm out to. 1= The
terminal
    mov     ecx, userg       ; Contains the label of the string.
    mov     edx, userg_L     ; This is the length of the string.
    int     80h              ; Calls the Linux kernel

    mov     eax, 0x03        ; 3 is the code to read user input. e.g. kernel function
(SYS_READ) opcode (03)
    mov     ebx, 0x00        ; This is the error code memory block.
    mov     ecx, userg_V     ; Reserves 255 bytes of data name of string.
    mov     edx, 255         ; Reserves 255 bytes of data.
    int     80h              ; Calls the Linux kernel.
    push    eax              ; Save length of input
```

```

    mov     eax, 0x04      ; Store the system call code = 4 e.g. = kernel function
(SYS_WRITE) opcode (04)
    mov     ebx, 0x01      ; 1 is the code to where to write the Asm out to. 1= The
terminal
    mov     ecx, respns    ; Calls the variable
    mov     edx, respns_L  ; Calls the userg variable length
    int     80h            ; Calls the Linux kernel.

```

```

    mov     eax, 0x04      ; Store the system call code = 4 e.g. = kernel function
(SYS_WRITE) opcode (04)
    mov     ebx, 0x01      ; 1 is the code to where to write the Asm out to. 1= The
terminal
    mov     ecx, userg_V   ; The input text
    mov     edx, [esp]     ; This is the length of the input text
    int     80h            ; Calls the Linux kernel.
    jmp     adduser        ; Jumps and Calls the adduser function name below.

```

```

adduser:                                ; Start of adduser function below.
    pop     eax            ; Get back the input length
    mov     [eax + userg_V - 1], byte 0 ; Chop off the trailing newline to 0 bytes
    mov     edx, enviro    ; Environment variable's
    mov     ecx, argu      ; Arguments to pass to the command line
    mov     ebx, command   ; Address to execute
    mov     eax, 11        ; (SYS_EXECVE) kernel function opcode (11)
    int     80h            ; Calls the Linux kernel.
    call    exit           ; Calls the exit function.

```

-----ASM exit code below!-----

```

exit:
    mov     eax, 0x01      ; [EAX] is 1 this is the exit code! e.g. = kernel function
(SYS_EXIT) opcode (01)
    mov     ebx, 0x00      ; This is the error code memory block.
    int     80h            ; Calls the Linux kernel.

```

APPENDIX F: Python Script

Here is the download link, to have a better view of the code: (Filename: matts_python_script.py)
<https://drive.google.com/file/d/0B-KNwipoXnOqZWM0bGNGT21RZTA/view?usp=sharing>

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The coding utf is used so the == symbols can be used.

# By Matthew Utin , System Administrative Tools.
# python --version Python 2.7.6
import os #Imports the Python os module. So the OS function can be used.
os.system("clear") #Uses the Linux Clear command.
loop = 1 #Starts loop
choice = 0 #Sets the choice to 0 int.
#While loop below, set to 1 int.
while loop == 1:
    print " " #Print is used to print out text numbers or symbols
    print "
=====
"
    print "      ■ Matt's System Administrative Tools For Linux python, Alpha build. ■" #
    print "Information.
=====
"
    print " "
    print "Options below:" #Options listed, to the user.
    print " "
    print "1) About"
    print "2) Show all user groups on the system"
    print "3) Show all users on the system"
    print "4) Add a usergroup to the system"
    print "5) Add a new user to the system"
    print "6) Exit"
    print " "

    choice = raw_input("Please enter your option: [1-6]? ") #Sets the choice cursor line
    text. Also raw_input() is used and not input() as this script is using Python 2.7.6 so you
    can use input() as this is in Python 3.
    choice = int(choice) # selects the int.
    if choice == 1: # If statements
        os.system("clear")
        print
=====
"
```

```

    print "Matt's System Administrative Tools For Linux Alpha build" #
Information.
    print "This script has been ported from the BASH Script version."
    print
"||
=====
elif choice == 2: # elif statement
    os.system("cat /etc/group | cut -d: -f1") # Show all user groups on the system
elif choice == 3:
    os.system("cat /etc/passwd") # Show all users on the system
elif choice == 4:
    grp2 = raw_input("Add usergroup: ") # Adds a usergroup to the system
    os.system("groupadd " + grp2) # groupadd command the adds the usergroup
elif choice == 5:
    user = raw_input("Enter the username: ") # Adds a new user to the system
    os.system("useradd "+ user) # Gives the useradd the variable from the user
input then executes
    os.system("passwd "+ user) # Gives the userpw the variable from the user input
elif choice == 6: # End loop and exit.
    loop = 0
    print "Thank you for using Matt's System Administrative Tools." # message shown
when script exits.
else:
    print "Select an option from the menu." # If the user types in the wrong choice
e.g. 7, then it will show this message.

```

APPENDIX G: BASH “BETA - Test” SCRIPT v1

The script below is an early version of the fully finished script. Also the script below does not have any advanced features for example adding users from a file.

Here is the download link, to have a better view of the code: (Filename:

matts_addusers_script_v1.sh)

<https://drive.google.com/file/d/0B-KNwipoXnOqTnJNNG9CTUNrSTg/view?usp=sharing>

```
#!/bin/bash
# A Linux Script to add a new user to the system. Via using the command line By
Matthew Utin.
#----->>*START*<<-----
-----//
<<COMMENT1
View code in Kate, within linux.
To Run, You must run as root. e.g. cd /home/user/Desktop#
sudo chmod +x matts_addusers_script.sh

*////////////////////@
|[By: Matthew Utin]   |
|[Date: Wed Feb 12 2014]|
|[Version: 1.0]       |
@////////////////////*
```

References Below:

+-----+

[1]. Linux Shell script to add a user with a password to the system - nixCraft. 2014. Linux Shell script to add a user with a password to the system - nixCraft. [ONLINE] Available at: <http://www.cyberciti.biz/tips/howto-write-shell-script-to-add-user.html>. [Accessed online 13 February 2014].

[2]. How to remove a user from a group? - Ask Ubuntu. 2014. How to remove a user from a group? - Ask Ubuntu. [ONLINE] Available at: <http://askubuntu.com/questions/80115/how-to-remove-a-user-from-a-group>. [Accessed online 13 February 2014].

[3]. Bash Shell Programming in Linux . 2014. Bash Shell Programming in Linux . [ONLINE] Available at: http://www.arachnoid.com/linux/shell_programming.html. [Accessed online 13 February 2014].

[4].Robbins, A R, 2010. bash Pocket Reference, Help for Power Users and sys Admins. 1st ed. USA: O'Reilly Media Inc.. [Book].

[5].Negus, C.N, 2012. Linux Bible. 5th ed. USA: John Wiley & Sons, Inc.. [Book].

Licence:

+-----+

Licence: LGPL, (c) Matthew Utin 2014-2

You can use this code in any tool but you must credit it. There are going to be Detailed comments provided for use in my tool.

COMMENT1

```
echo
" |-----" #
The title of my linux tool
echo "■ Matt's System Administrative Tools For Linux V1.0 ■"
echo
" |-----"
echo ${tmp-`date`} # Shows date and time.
PS3='Hit [Enter] to re-show all options, Please enter your choice: ' # This is the start of
my bash select code to give the tool it's menus.
options=("Create a regular user." "Create a root/admin user." "Add a usergroup to a
user." "Create a new usergroup." "Show the usergroups that a user is linked to." "Show
all usergroups on the system." "Show all users on the system." "About." "Remove a
usergroup from a user." "Remove/delete usergroup from system." "Remove User and
home directory or just the user account." "Quit!") #This part of the code is the options
that are used to link the case select code-
# to the diffrent parts of the tool.
select opt in "${options[@]}" # selects the diffrent options that are in the code below
and exacutes them, in order. So if the user hit the key bored key[1] it would open and
exacute option 1, in my tool that would be the 1) Create a regular user. option.
do
    case $opt in
        "Create a regular user.") # 1) option.
            clear #This is used to clear the command line screen above from past commands to
            make the command prompt look more tidy. This makes it better to read the
            information.
            if [ $(id -u) -eq 0 ]; then
                echo
                " |-----"
                " |-----"
                echo "■ Adding a new (regular user) to the system step 1 enter the username. ■" # The
                title of command/information.
                echo
                " |-----"
                " |-----"
                read -p "Enter Username : " matts_code_username # This line of code is used to
                tell the user to type in some information. This will then be liked by a bash variable.
                e.g. $test for example.
                echo " |-----"
                echo "■ Step 2 enter the Password. ■" # The title of command/info.
                echo " |-----"
                read -s -p "Enter Password : " matts_code_password
```

```

    egrep "^$matts_code_username" /etc/passwd >/dev/null # modifies the file
and looks up usernames to see if they are there.
    if [ $? -eq 0 ]; then
        echo "$matts_code_username exists!" # If the username is there this
message will display.
        exit 1 # And exit the loop, it will also return to the case loop to pick an
option.
    else
        matts_code_pass=$(perl -e 'print crypt($ARGV[0],
"matts_code_password")' $matts_code_password) # This is some perl script that will
encrypt the users password.
        useradd -m -p $matts_code_pass $matts_code_username # After this I
then add the user but using the useradd command.
        [ $? -eq 0 ] && echo "The new User has been added to system!" || echo
"Sorry the system has failed to add the user!" #if the script works he first echo will
show if not the other will show.
    fi
else
    echo "Only the root user may add a new user to this system!" #If error starting
then script will show this message and exit.
    exit 2
fi
;;
"Create a root/admin user.") # 2) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
if [ $(id -u) -eq 0 ]; then
    echo
    "
=====
"
    echo "■ Adding a new (root/admin user) to the system step 1 enter the username. ■" #
The title of command/info.
    echo
    "
=====
"
    read -p "Enter Username : " matts_code_username # This line of code is used to
tell the user to type in some information. This will then be liked by a bash variable.
e.g. $test for example.
    echo "
=====
"
    echo "■ Step 2 enter the Password. ■" # The title of command/info.
    echo "
=====
"
    read -s -p "Enter Password : " matts_code_password
    egrep "^$matts_code_username" /etc/passwd >/dev/null # modifies the file
and looks up usernames to see if they are there.
    if [ $? -eq 0 ]; then
        echo "$matts_code_username exists!" # If the username is there this
message will display.
        exit 1 # And exit the loop, it will also return to the case loop to pick an
option.

```



```

else
    matts_code_pass=$(perl -e 'print crypt($ARGV[0],
"matts_code_password")' $matts_code_password)
    useradd -m -p $matts_code_pass $matts_code_username -G admin,root
# (Adds new user as admin and root in the usergroups.)
    [ $? -eq 0 ] && echo "The new User has been added to system!" || echo
"Sorry the system has failed to add the user!" #if the script works he first echo will
show if not the other will show.
fi
else
    echo "Only the root user may add a new user to this system!" #If error starting
then script will show this message and exit.
    exit 2
fi
;;
"Add a usergroup to a user.") # 3) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "The user group you want to add to the user: " group
read -p "Enter the user: " username
usermod -a -G $group $username #code to add a usergroup to the user.
;;
"Create a new usergroup.") # 4) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the group you would like to create: " newgroup
groupadd $newgroup #creates new usergroups.
;;
"Show the usergroups that a user is linked to.") # 5) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the username: " groups # shows the user groups that the user is in.
groups $groups
;;
"Show all usergroups on the system.") # 6) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
cat /etc/group |cut -d: -f1 # Using the cat command to look up usergroups in the
/etc/group file.
;;
"Show all users on the system.") # 7) option.
clear
cat /etc/passwd # Using the cat command to see all users on the system as it shows
the output of the file.
;;
>About.") # 8) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
echo ${tmp-`date`} # shows data and time.

```

```

echo "*/-----/"
echo "  ### #####  ##  ##  "
echo "  ## ##  ##  ##  ##  ##  " # About title.
echo "  ## ##  ##  ##  ##  ##  "
echo "  ##  #####  ##  ##  ##  "
echo "  #####  ##  ##  ##  ##  "
echo "  ##  ##  ##  ##  ##  ##  ### "
echo "  ##  #####  #####  ##  ### "
echo "*/-----/*"
echo " "
echo "----->[By: Matthew Utin]" # Information.
echo "----->[Date: Wed Feb 12 2014]"
echo "----->[Version: 1.0]"
echo " "
echo
"
"
"
"
echo "  Licence: LGPL, (c) Matthew Utin 2014-2" # Information.
echo "  You can use this code in any tool but you must credit it."
echo "  There are going to be Detailed comments provided for use in this tool."
echo
"
"
"
"
echo " "
;;
"Remove a usergroup from a user.") # 9) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the username: " username_matts_code
read -p "Enter the group you would like to remove from that user [e.g. root]: "
group_matts_code
gpsswd -d $username_matts_code $group_matts_code #Removes a usergroup from a
system user.
;;
"Remove/delete usergroup from system.") # 10) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the usergroup name you would like to remove from the system: "
matts_usergroup_del
groupdel $matts_usergroup_del # deletes a usergroup from the system.
;;
"Remove User and home directory or just the user account.") # 11) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
echo " "
echo "If you don't want to make any changes or you have hit the wrong option." # The
title of command/information.
echo "Hit [Enter] to go back to the menu."
echo " "

```

```

read -p "Hit [Y]yes if you want to remove the User and home directory or Hit [N]no if
you want to just remove the user. $foo? [Y/N]" answer #using a yes or no script.
if [[ $answer = y ]] ; then # if y then the below code is executed.
read -p "Enter the user you want to remove from the system: " users
userdel -r $users # removes the user and directory.
elif [[ $answer = n ]] ; then # if n then the below code is executed.
read -p "Enter the user you want to remove from the system: " user
userdel $user # Just removes the user.
fi # if [Enter] key is hit then back to menu.
;;
"Quit!") # 12) option.
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
    break # Exit code.
    ;;
    *) echo invalid option;;
esac
done #----->!END!<-----
-----//

```

APPENDIX H: BETA test script results

<u>Module tested in the script.</u>	<u>Results, tested on the 22/11/14.</u>	<u>Possible changes or fixes.</u>	<u>Script 2.0 implemented updates, 28/11/14 .e.g. Change list.</u>
Create a regular user.	Runs correctly Some changes needed	Could add some encryption so that the user add script cannot be views in plain text.	There is now, implemented script protection. In the form of BASE 64 encoding and hexadecimal / notation code.
Create a root/admin user.	Runs correctly Some changes needed	Like before needs some type of encryption so the script cannot be read in plain text, could use BASE 64 encoding in script 2.0.	The same as above has been added to the script to fix the vulnerability.
Add a usergroup to a user.	Runs correctly	No changes needed.	N/A
Create a new usergroup.	Runs correctly	No changes needed.	N/A
Show the usergroups that a user is linked to.	Runs correctly Some changes needed	Could have a more user friendly menu.	New updated user friendly interface, added.
Show all usergroups on the system.	Runs correctly	No changes needed.	N/A
Show all users on the system.	Runs correctly	No changes needed.	N/A
About.	Runs correctly	No changes needed.	N/A

Remove a usergroup from a user.	Runs correctly	No changes needed.	N/A
Remove/delete usergroup from system.	Runs correctly	No changes needed.	N/A
Remove User and home directory or just the user account.	Runs correctly	No changes needed.	N/A
Quit!	Runs correctly	No changes needed.	N/A
options=("") script case opt selector.	Needs to be changed	Tried to get individual functions to load and when finished, it should exit the function and go back to the main menu. But this did not happen. So to change this there should be some sort of select a function feature in script 2.0 final build.	Now uses a different technique to select functions, within the script. Using a function select with an exit loop and returns to the menu.

APPENDIX I: Final script build 2.0

The final script file: <https://drive.google.com/file/d/0B-KNwipoXnOqRUs5MzZUCtThtQ28/view?usp=sharing>

The full packed project with manual files: <https://drive.google.com/file/d/0B-KNwipoXnOqQXh3QWRCNmJ0N0U/view?usp=sharing>

```
#!/bin/bash
# A Linux Script to add a new user to the system. Via using the command line By
Matthew Utin.
#----->>*START*<<-----
-----//
<<COMMENT1
View code in Kate, within linux, (Ubuntu).
To Run, You must run as root. e.g. sudo su <Password>. You will then need to change
the directory. e.g. cd </home/user/Desktop>
sudo chmod +x matts_addusers_script.sh
Then use --> ./matts_addusers_script_v2.0.sh <-- to run the script.
End result, you should see something like this: root@user-Pc-
05:/home/user/Desktop#./matts_addusers_script_v2.0.sh
```

For best results when operating the script. Run in: GNOME Terminal, tested on GNOME Terminal version 3.6.2.

```
*////////////////////@
|[By: Matthew Utin]  |
|[Date: 28th Nov 2014]|
|[Version: 2.0]      |
@////////////////////*
```

References Below:

+-----+

[1]. Linux Shell script to add a user with a password to the system - nixCraft. 2014. Linux Shell script to add a user with a password to the system - nixCraft. [ONLINE] Available at: <http://www.cyberciti.biz/tips/howto-write-shell-script-to-add-user.html>. [Accessed online 13 February 2014].

[2]. How to remove a user from a group? - Ask Ubuntu. 2014. How to remove a user from a group? - Ask Ubuntu. [ONLINE] Available at: <http://askubuntu.com/questions/80115/how-to-remove-a-user-from-a-group>. [Accessed online 13 February 2014].

[3]. Bash Shell Programming in Linux . 2014. Bash Shell Programming in Linux . [ONLINE] Available at: http://www.arachnoid.com/linux/shell_programming.html. [Accessed online 13 February 2014].

[4]. Robbins, A R, 2010. bash Pocket Reference, Help for Power Users and sys Admins. 1st ed. USA: O'Reilly Media Inc.. [Book].

[5]. Negus, C.N, 2012. Linux Bible. 5th ed. USA: John Wiley & Sons, Inc.. [Book].

[6]. How To Add Users To Linux OS From A Text file | HowtoForge - Linux Howtos and Tutorials. 2014. How To Add Users To Linux OS From A Text file | HowtoForge - Linux Howtos and Tutorials. [ONLINE] Available at: <http://www.howtoforge.com/how-to-add-linux-system-users-from-a-text-file>. [Accessed 27 September 2014].

[7]. Bash: Using Colors. 2014. Bash: Using Colors. [ONLINE] Available at: <http://www.csc.uvic.ca/~sae/seng265/fall04/tips/s265s047-tips/bash-using-colors.html>. [Accessed 27 September 2014].

[8]. Bash:tip_colors_and_formatting - FLOZz' MISC . 2014. bash:tip_colors_and_formatting - FLOZz' MISC . [ONLINE] Available at: http://misc.flogisoft.com/bash/tip_colors_and_formatting. [Accessed 27 September 2014].

[9]. The chmod command. 2014. The chmod command. [ONLINE] Available at: http://catcode.com/teachmod/chmod_cmd.html. [Accessed 27 September 2014].

[10]. Whiptail(1): dialog boxes from shell scripts - Linux man page. 2014. whiptail(1): dialog boxes from shell scripts - Linux man page. [ONLINE] Available at: <http://linux.die.net/man/1/whiptail>. [Accessed 27 September 2014].

[11]. How to read file from line x to the end of a file in bash - Stack Overflow. 2014. how to read file from line x to the end of a file in bash - Stack Overflow. [ONLINE] Available at: <http://stackoverflow.com/questions/14110223/how-to-read-file-from-line-x-to-the-end-of-a-file-in-bash>. [Accessed 28 September 2014].

[12]. Shell Script to create user in Linux using text file data. 2014. Shell Script to create user in Linux using text file data. [ONLINE] Available at: <http://forums.devshed.com/scripts-94/shell-script-create-user-linux-using-text-file-data-81336.html>. [Accessed 28 September 2014].

Licence:

+-----+

Licence: LGPL, (c) Matthew Utin 2014-2

You can use this code in any tool but you must credit it. There are going to be detailed comments provided for use in my tool.

COMMENT1

```
Create_a_regular_user() #opt 1
{
```



```

        useradd -m -p $matts_code_pass $matts_code_username -G admin,root
# (Adds new user as admin and root in the usergroups.)
        [ $? -eq 0 ] && echo "The new User has been added to system!" || echo
"Sorry the system has failed to add the user!" #if the script works he first echo will
show if not the other will show.
    fi
else
    echo "Only the root user may add a new user to this system!" #If error starting
then script will show this message and exit.
    exit 2
fi
}

add_a_usergroup_to_a_user() #opt 3
{
# Used the two commands to change the text strings to Hex. cat file.txt | hexdump -v
-e "\\x" 1/1 "%02x" ""
# echo"sometext" | hexdump -v -e "\\x" 1/1 "%02x" ""
# eval `printf ""` Will be used to execute the hexadecimal code e.g. Base 16.
eval `printf "\x63\x6c\x65\x61\x72\x0a"` #This is used to clear the command line
screen above from past commands to make the command prompt look more tidy. This
makes it better/easier to read.
eval `printf
"\x65\x61\x64\x20\x2d\x70\x20\x22\x54\x68\x65\x20\x75\x73\x65\x72\x20\x67\x72\x6f\x
75\x70\x20\x79\x6f\x75\x20\x77\x61\x6e\x74\x20\x74\x6f\x20\x61\x64\x64\x20\x74\x6f\x
x20\x74\x68\x65\x20\x75\x73\x65\x72\x3a\x20\x22\x20\x67\x72\x6f\x75\x70\x0a"` #
read -p "The user group you want to add to the user: " group

eval `printf
"\x65\x61\x64\x20\x2d\x70\x20\x22\x45\x6e\x74\x65\x72\x20\x74\x68\x65\x20\x75\x73\x
65\x72\x3a\x20\x22\x20\x75\x73\x65\x72\x6e\x61\x6d\x65\x0a"` # read -p "Enter the
user: " username

eval `printf
"\x73\x65\x72\x6d\x6f\x64\x20\x2d\x61\x20\x2d\x47\x20\x24\x67\x72\x6f\x75\x70\x20\x
24\x75\x73\x65\x72\x6e\x61\x6d\x65\x0a"` # usermod -a -G $group $username #code to
add a usergroup to the user.

}

create_a_new_usergroup() #opt 4
{
# Used the two commands to change the text strings to Hex. cat file.txt | hexdump -v
-e "\\x" 1/1 "%02x" ""
# echo"sometext" | hexdump -v -e "\\x" 1/1 "%02x" ""
# eval `printf ""` Will be used to execute the hexadecimal code e.g. Base 16.
eval `printf "\x63\x6c\x65\x61\x72\x0a"` # clear #This is used to clear the command
line screen above from past commands to make the command prompt look more tidy.
This makes it better/easier to read.

```

```
eval `printf
"\x72\x65\x61\x64\x20\x2d\x70\x20\x22\x45\x6e\x74\x65\x72\x20\x74\x68\x65\x20\x67\x
72\x6f\x75\x70\x20\x79\x6f\x75\x20\x77\x6f\x75\x6c\x64\x20\x6c\x69\x6b\x65\x20\x74\
x6f\x20\x63\x72\x65\x61\x74\x65\x3a\x20\x22\x20\x6e\x65\x77\x67\x72\x6f\x75\x70\x0a
"` # read -p "Enter the group you would like to create: " newgroup
eval `printf
"\x67\x72\x6f\x75\x70\x61\x64\x64\x20\x24\x6e\x65\x77\x67\x72\x6f\x75\x70\x0a"` #
groupadd $newgroup #creates new usergroups.
}
```

```
show_the_usergroups_that_a_user_is_linked_to() #opt 5
{
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the username: " groups # shows the user groups that the user is in.
groups $groups
}
```

```
show_all_usergroups_on_the_system() #opt 6
{
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
cat /etc/group |cut -d: -f1 # Using the cat command to look up usergroups in the
/etc/group file.
}
```

```
show_all_users_on_the_system() #opt 7
{
clear
cat /etc/passwd # Using the cat command to see all users on the system as it shows
the output of the file.
}
```

```
remove_a_usergroup_from_a_user() #opt 8
{
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
read -p "Enter the username: " username_matts_code
read -p "Enter the group you would like to remove from that user [e.g. root]: "
group_matts_code
gpasswd -d $username_matts_code $group_matts_code #Removes a usergroup from a
system user.
}
```

```
remove_delete_usergroup_from_system() #opt 9
{
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
```

```
read -p "Enter the usergroup name you would like to remove from the system: "
matts_usergroup_del
groupdel $matts_usergroup_del # deletes a usergroup from the system.
}
```

```
remove_user_and_home_directory_or_just_the_user_account() #opt 10
{
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
echo " "
echo "If you don't want to make any changes or you have hit the wrong option." # The
title of command/information.
echo "Hit [Enter] to go back to the menu."
# The title of command/information. Within the: matts_decoder_base64
#
```

```
# If you see the error:
# userdel: user mail spool (/var/mail/user) not found.
# The command is still executed, its due to the Ubuntu 14.04.1 update,
# showing the user of not having a setup mail account.
#
```

[illegible]

```

}

make_a_file_executable() #opt 11
{
clear
read -p "Enter the path of the file (e.g. /home/user/Desktop): " matts_master_path
#Gets the path of the file.
cd $matts_master_path #Change the path to the files path.
ls #lists the files in the path.
read -p "Enter the file you want to make executable: " matts_file_name #Gets the file
name.
chmod +x $matts_file_name #This is the command used to make the file executable.
echo "Command Executed."
}

add_new_users_from_a_text_file() #opt 12
{
clear
    echo "The current directory is: `pwd` /matts_users_database.txt" #Shows the
current directory.
    echo
    echo -e "Do you want to use the default PATH that is shown above? Yes[y] &
No[n]." #Yes/no options.
    read matts_yes_no
    if [ $matts_yes_no == y ]; then
        Path=$(pwd)matts_users_database.txt #If yes then it will use the
current path.
    else
        echo -n "Enter the path of the file (e.g. /home/user/Desktop/filename.txt): " #If
no then the user will have to type the files path.
        read Path #Puts the information into the path to pass it on to the variable.
    fi
    if [ -e $Path ]; then #If the file exists.
        home_path_name="/home/" #Home directory
        #tail will start the file from a selected line, as you can see it's set on the 5th line.
        tail -n +5 ${Path} | \
        while read system_user users_password system_group #Reads the 3 areas within the
text file and as you can see it has put them into variables.
        do
            groupadd ${system_group} #Adds the usergroups
            useradd -g ${system_group} -p ${users_password} -m -d
            ${home_path_name}${system_user} ${system_user} #Adds the user to the system.
        done
    else #If the file cannot be found, the code below will be shown.
        echo -e "
        echo -e "■ Sorry the file was not found! ■"
        echo -e "
    fi

```

[illegible]

[illegible]

```

\x23\x23\x20\x20\x20\x20\x20\x20\x0a\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x23\x
23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23
\x20\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x20\x20\x20\x
20\x0a\x20\x20\x23\x23\x20\x20\x20\x23\x23\x20\x20\x23\x23\x20\x20\x20\x20\x20\x23
\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x
23\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x0a\x20\x23\x23\x20\x20
\x20\x20\x20\x23\x23\x20\x23\x23\x23\x23\x23\x23\x23\x23\x20\x20\x23\x23\x20\x20\x
20\x20\x20\x23\x23\x20\x23\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x20\x20\x23\x23
\x20\x20\x20\x20\x20\x20\x0a\x20\x23\x23\x23\x23\x23\x23\x23\x23\x23\x20\x23\x
23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23
\x20\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x20\x20\x20\x
20\x0a\x20\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23
\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x23\x20\x23\x23\x20\x20\x20\x20\x20\x23\x
23\x20\x20\x20\x20\x23\x23\x20\x20\x20\x20\x23\x23\x23\x23\x23\x23\x23\x23\x20\x20\x20\x23\x23
23\x23\x23\x23\x20\x20\x20\x23\x23\x23\x23\x23\x23\x23\x23\x20\x20\x20\x20\x20\x23\x23
\x20\x20\x20\x20\x23\x23\x23\x20\x0a\x2f\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x
2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d
d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d\x2d
\x2f\x2a\x0a"
echo ""
echo "----->>[By: Matthew Utin]" # Information.
echo "----->>[Date: 28th Nov 2014]"
echo "----->>[Version: 2.0, Ubuntu Specific, Commands.]"
echo ""
echo
"
"
"
"
echo "  Licence: LGPL, (c) Matthew Utin 2014-2" # Information.
echo "  You can use this code in any tool but you must credit it."
echo "  There are going to be Detailed comments provided for use in this tool."
echo
"
"
"
"
echo ""
}

change_log() #opt 16
{
#Shows what has been changed within the script, using the whiptail interface.
whiptail --title "Change Log: [Hit Enter To Exit.]" --msgbox "\n\nThe script user can now
make a file executable. Added on (Sat 27th Sep 2014).\n\nThe script user can now add
users from a text file. Added on (Fri 26th Sep 2014).\n\nThe script user can now delete
users from the system using a text file. Added on (Fri 26th Sep 2014).\n\n" 15 78

}

man_page() #opt 18
{

```



```

man ./matts_addusers_script_v2.0.man
}

matts_decoder_base64() #non opt
{
# The python code below will decode the base64 strings that are used within this
script.
echo "$1" | python -m base64 -d ; echo

}

matts_main_Script_menu() #opt men & exit 0 opt 14
{
MATTs_MASTER_ROOT=0    #If the user has a USERID=UID of 0 then they are ROOT, if
the user has the UID of 1 then they are not a ROOT user and cannot run the script.
if [ "$UID" -ne "$MATTs_MASTER_ROOT" ]; then
echo "You can only run this Script if you are a root user!"
exit
fi
opt=1 #options
while [ $opt -le 4 ]
do

clear
printf '\033[8;26;80t' # This code sets the Terminal window size at startup to 033 char
size and 26hx80w window size.
echo -e "\e[2;Matt's System Administrative Tools For Linux V2.0. By Matthew Utin.\a"
#Sets the Terminal window's title.
echo -e
"\e[1;37;47m"=====
=====
"\e[0m" # The title of my linux tool
echo -e "\e[1;37;47m" Matt's System Administrative Tools For Linux V2.0 "\e[0m" #\e
means execute the following font codes also this is shown with the -e on the echo.
echo -e
"\e[1;37;47m"=====
=====
"\e[0m" #[1 = bold font 37 = foreground white 47 = background gray. m=code
start \e[0m =End code.
echo -e "\e[1;4;5;34m${tmp-`date`}.\e[0m" # Shows date and time, 1=bold 4=underline
5=blink 34=blue.
echo -e "
1) Create a regular user.
2) Create a root/admin user.
3) Add a usergroup to a user.
4) Create a new usergroup.
5) Show the usergroups that a user is linked to.
6) Show all usergroups on the system.
7) Show all users on the system.
8) Remove a usergroup from a user.
9) Remove/delete usergroup from system.

```

```

10) Remove User and home directory or just the user account.
11) Make a file executable.
12) Add users to the system from a text file.
13) Delete users from the system using a text file.
14) \e[1;31m[EXIT!]\e[0m\n
15) \e[1;93m(About.)\e[0m
16) \e[1;93m(Change Log.)\e[0m
17) \e[1;93m(System version information.)\e[0m
18) \e[1;93m(Man/manual Page.)\e[0m\n"
read -p "Please enter your option: [1-18]? " opt #Main menu selection. EXIT 1=bold 31 =
Red, About 1=bold 93=Yellow. m=code start \e[0m =End code, \e means execute the
following font codes also this is shown with the -e on the echo. \n means space below
to the next line.
clear
case $opt in #Options code so that the script knows where the options are linked
within the script, by typing in the functions.
1) Create_a_regular_user ;;
2) Create_a_root_user ;;
3) add_a_usergroup_to_a_user ;;
4) create_a_new_usergroup ;;
5) show_the_usergroups_that_a_user_is_linked_to ;;
6) show_all_usergroups_on_the_system ;;
7) show_all_users_on_the_system ;;
8) remove_a_usergroup_from_a_user ;;
9) remove_delete_usergroup_from_system ;;
10) remove_user_and_home_directory_or_just_the_user_account ;;
11) make_a_file_executable ;;
12) add_new_users_from_a_text_file ;;
13) delete_users_listed_in_the_text_file ;;
14) exit 0 ;;
15) about ;;
16) change_log ;;
17) version_information ;;
18) man_page ;;
*) echo -e "You have input a bad option!" #This line is shown if the users of the script
has typed in the wrong key.
esac
echo -e "\nPress (1) to go back to the main menu." #This line shows the script user to
type 1, to get back to the main menu after the selected operations are finished.
read opt
done
}
matts_main_Script_menu
clear #This is used to clear the command line screen above from past commands to
make the command prompt look more tidy. This makes it better/easier to read.
exit 0
#----->>!END<<-----
-----//

```

APPENDIX J: Final script manual (MAN) file

The full packed project with manual files: <https://drive.google.com/file/d/0B-KNwipoXnOqQXh3QWRCNmJON0U/view?usp=sharing>

```
.\" Manpage for Matts_addusers_script_v2.0.sh
.\" Contact Outinm77@solent.ac.uk to correct errors.
.TH man 7 "26 Oct 2014" "2.0" "Matts_addusers_script_v2.0.sh."
.SH NAME
.B
Matt's System Administrative Tools For Linux V2.0
.SH SYNOPSIS
View code in Kate, within linux, (Ubuntu).
```

To Run, You must run as root. e.g. `sudo su <Password>`. You will then need to change the directory. e.g. `cd </home/user/Desktop>`
`sudo chmod +x matts_addusers_script.sh`

Then use `--> ./matts_addusers_script_v2.0.sh <--` to run the script.

End result, you should see something like this: `root@user-Pc-05:/home/user/Desktop#./matts_addusers_script_v2.0.sh`

```
.SH DESCRIPTION
A lot of companies are just starting to get
invaded in using Linux systems this is mostly
because of cost because Linux is free and
open to use, but as it is free to use it is not
the most user friendly system to use,
although it has been created to be more
reliable, in this case I am going to create a
user script that can be used to create new
users, user groups, manage system feathers
and preform administrative tasks. As one of
the main problems of Linux, Is that you
would need to have some knowledge of
using the systems command line interface,
to get anything to work correctly and if you
do not know this or are not that technically
minded you will struggle, as you would have
to type in every command in correctly.
```

This is why i have created an easy to use tool to help complete daliy system tasks.

```
.SH OPTIONS
When you strat the [script], you will be given a list
of commands that can be used:
```

(1) Create a regular user.

- (2) Create a root/admin user.
- (3) Add a usergroup to a user.
- (4) Create a new usergroup.
- (5) Show the usergroups that a user is linked to.
- (6) Show all usergroups on the system.
- (7) Show all users on the system.
- (8) Remove a usergroup from a user.
- (9) Remove/delete usergroup from system.
- (10) Remove User and home directory or just the user account.
- (11) Make a file executable.
- (12) Add users to the system from a text file.
- (13) Delete users from the system using a text file.

.SH BUGS

There are no known major bugs. There is one minor bug that is, the flashing blue time on the options page will only flash in Xterm. But for the best results when operating the script. Run in: GNOME Terminal, tested on GNOME Terminal version 3.6.2.

.SH COPYRIGHTS

Licence: LGPL, (c) Matthew Utin 2014-2

You can use this code in any tool but you must credit it.

There are going to be Detailed comments provided for use in this tool.

.SH AUTHOR

Matthew Utin (0utinm77@solent.ac.uk)

APPENDIX K: Script addusers database file

The full packed project with manual files: <https://drive.google.com/file/d/0B-KNwipoXnOqQXh3QWRCNmJON0U/view?usp=sharing>

File made by Matthew Utin, on the (28th Sep 2014). For use with the
matts_addusers_script_v2.0.sh

```
-----  
Username: Password: Group:  
===== [#!Do Not delete the 4 lines above or the script will not  
function correctly!#]  
matter tengers testers  
lack09 jack87 testers  
markT80 carboy53 themasters  
mrtong test01 admin  
usermyt68n6 pascomp7f6 testers
```

APPENDIX L: Project planning level of achievement table

<u>Project Task</u>	<u>Level of achievement</u>	<u>Scheduled Deadlines “Week Number”</u>
Research Project background.	(5)	Week 2
Write up a Project Proposal.	(5)	Predetermined, should be complete a week before the deadline.
Create a Project Log file.	(2)	Week 4
Research Literature.	(5)	Week 10
Identify different programing languages and write up the findings.	(5)	Week 17
Write up the Project Definition Report.	(4)	Predetermined, should be complete a week before the deadline.
Create an example script of the different programming language e.g. Python and NASM.	(5)	Week 27
Design and create a Linux script.	(5)	Week 28
Build the basic beta Test script and run tests.	(5)	Week 34
Build and Test final advance BASH script for install.	(5)	Week 49
Write project review report, about the project progression.	(5)	Predetermined, should be complete a week before the deadline.

Write Final Report.	(5)	Predetermined, should be complete a week before the deadline.
Create a Poster for the Poster Day Assessment.	(5)	Predetermined, should be complete a week before the deadline.

(Key 5 = being in the high range and 1 = being in the low range.)

Table (4): (This table shows the Project planning and level of achievement).

APPENDIX M: User Survey Results

M.1 Participant 1

By Matthew Utin

08/02/15

Participant: 1

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☒ 5. ☐

(Optional) If yes please, write a comment in the text box below:

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Very good, task's completed
Successfully.

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

The Manual file was a very good idea, helping me navigate the script.

M.2 Participant 2

By Matthew Utin

08/02/15

Participant: 2

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Excellent

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

Very User Friendly

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

Very Good

M.3 Participant 3

By Matthew Utin

08/02/15

Participant: 3

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Did all of the tasks correctly

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Very

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

N/A

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

To a Linux admin.

7. Are there any improvements that could be made to this script?

Yes: ☒

No: ☐

If yes please, write a comment in the text box below:

Could add custom, user options

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

N/A

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

N/A

M.4 Participant 4

By Matthew Utin

08/02/15

Participant:

4

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Very reliable

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☐

No: ☒

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒ Quite satisfied: ☐ Dissatisfied: ☐ Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

Interesting Script, considering
i'm not that experienced.

M.5 Participant 5

By Matthew Utin

08/02/15

Participant: 5

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Great Script, had no problems in completing tasks on a Linux system e.g. adding new users to a group.

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

The script Did not crash, also i did not see any Bugs.

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

N/A

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

People working with Linux.

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

N/A

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

But could use other languages, For example Cshell, Ruby or Python.

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

good and very handy script.

M.6 Participant 6

By Matthew Utin

08/02/15

Participant: **6**

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

Worked Great

5. Are you an experienced Linux user?

Yes: ☐

No: ☒

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

Any one styling Linux.

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

M.7 Participant 7

By Matthew Utin

08/02/15

Participant: 7

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Worked without any problems.

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Worked well

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☐

No: ☒

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☒

Quite satisfied: ☐

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

M.8 Participant 8

By Matthew Utin

08/02/15

Participant: 8

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

Very User Friendly.

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) If yes please, write a comment in the text box below:

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

No

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

Yes, to any one that needs to add new users to a Linux System.

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

It works well, in with the keeping of the Project, ~~the~~ using Scuiding.

9. Overall, are you satisfied with the finished script?

Very satisfied: ☐

Quite satisfied: ☒

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

M.9 Participant 9

By Matthew Utin

08/02/15

Participant:

9

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☒ 5. ☐

(Optional) ~~If yes please~~, write a comment in the text box below:

yes very user friendly

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☐

No: ☒

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☒ 5. ☐

(Optional) ~~If yes please~~, write a comment in the text box below:

It was ok, but I don't really use linux.

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☐

No: ☒

5. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

But to someone more technically minded

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☐

No: ☒

If No please, write a comment in the text box below:

I'm not really too familiar with 'bash'

9. Overall, are you satisfied with the finished script?

Very satisfied: ☐

Quite satisfied: ☒

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

I would need to use linux more, as I don't really use it.

M.10 Participant 10

By Matthew Utin

08/02/15

Participant: 10

Linux administration BASH Script user feedback survey.

This survey will use the rating scale of 1 to 5 in satisfaction, on how the script has performed. Tick the boxes that are applicable.

Please complete this survey in full, for the scripts feedback.

1. How user friendly is the script, when performing system tasks? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) ~~If yes please~~, write a comment in the text box below:

VERY USER FRIENDLY

2. Has the Script been, able to cut the workload of performing system tasks?

Yes: ☒

No: ☐

3. Give a rating of how reliable the script has been to use? "1 being low and 5 being high satisfaction."

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☒

(Optional) ~~If yes please~~, write a comment in the text box below:

VERY GOOD

4. Have you come across any problems?

Yes: ☐

No: ☒

(Optional) If yes please, write a comment in the text box below:

5. Are you an experienced Linux user?

Yes: ☒

No: ☐

6. Would you promote the use of this type of script to anyone else?

Yes: ☒

No: ☐

(Optional) If yes please, write a comment in the text box below:

Very Good Script

7. Are there any improvements that could be made to this script?

Yes: ☐

No: ☒

If yes please, write a comment in the text box below:

8. Do you think that the BASH language was the correct language, used in creating the automated administration script?

Yes: ☒

No: ☐

If No please, write a comment in the text box below:

9. Overall, are you satisfied with the finished script?

Very satisfied: ☐

Quite satisfied: ☒

Dissatisfied: ☐

Very dissatisfied: ☐

10. (Optional) If you have any other questions, leave them in the text box below:

Index

A

ACME, i, vii, 1, 2, 33
admin, iii, 2, 6, 25, 26, 34, 1, 2, 2, 3, 4, 1, 4, 5, 12, 2, 1
administration, 2, i, 3, 14, 33, 35, 45
administrative, i, 1, 6, 21, 31, 33, 1

B

Base 64, 9
BASH, 2, i, ii, iii, iv, vii, 1, 2, 3, 5, 7, 8, 9, 21, 22, 23, 31,
32, 33, 45, 1, 2, 2, 1

C

code, iv, vi, 1, 2, 3, 6, 8, 9, 10, 11, 14, 16, 17, 18, 19,
20, 21, 22, 25, 26, 27, 28, 29, 30, 32, 33, 35, 38, 1,
2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11,
12, 13, 1, 2
command line, vi, 2, 3, 5, 7, 8, 11, 16, 17, 18, 19, 20,
33, 35, 41, 1, 2, 1, 2, 3, 4, 5, 6, 1, 3, 4, 5, 6, 7, 10,
13, 1
commands, 3, 4, 5, 6, 7, 8, 11, 25, 33, 34, 35, 2, 3, 4,
5, 6, 3, 4, 5, 6, 7, 10, 13, 1
corruption, 26
Critical analysis, 11

D

development, vii, 6, 7, 10, 12, 13, 20, 21, 25
directory, iv, vi, 3, 7, 17, 18, 19, 21, 28, 1, 2, 2, 5, 6, 2,
1, 7, 8, 9, 10, 13, 1, 2

E

editor, 26, 29

F

files, 2, 3, 22, 34, 1, 8, 9, 1
future, i, 5, 6

G

G edit, 26
Gantt chart, iv, vi, 23, 24, 25, 47, 48

I

IEEE, iv, vii, 8, 9, 36, 44, 45
implemented, i, 2, 5, 8, 23, 25, 27, 28, 29, 35, 1, 1
implementing, 2, 26
introduction, 1, 37

K

Kate, 6, 26, 44, 1

L

language, 2, i, 1, 2, 4, 6, 7, 8, 9, 11, 21, 22, 32, 35, 36,
37, 2, 1
Linux, 2, i, vi, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 17,
18, 22, 23, 25, 27, 29, 31, 32, 33, 37, 38, 39, 40, 41,
42, 43, 45, 1, 3, 1, 2, 1, 2, 1, 2, 1, 2, 12, 1

M

methodology, iii, vi, 10, 11, 12, 13, 24, 25, 32, 1, 3

N

NASM, iii, iv, vii, 1, 9, 11, 21, 38, 1
next stage, 14, 28

O

operating system, 6, 35

P

Perl, 2, 8, 43
Perl Script, 2
problems, 2, 3, 24, 28, 29, 31, 32, 33, 1
Programming, 2, 39, 41, 45, 1

python, 2, 8, 22, 37, 38, 39, 1, 12
Python, iii, iv, 1, 2, 8, 21, 22, 23, 32, 36, 38, 39, 41, 43,
1

R

review, 23, 24, 25, 1
risk, vi, 23, 24, 2, 1
root, iii, vi, 2, 6, 14, 26, 2, 2, 1, 2, 3, 4, 5, 1, 4, 5, 6, 12,
13, 1, 2
run, vi, 2, 5, 6, 11, 14, 22, 23, 25, 26, 29, 1, 12, 1

S

script, 2, i, iii, iv, vi, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28,
29, 31, 32, 33, 34, 35, 41, 42, 1, 1, 2, 3, 2, 1, 2, 1, 3,
4, 6, 1, 2, 1, 2, 3, 4, 5, 7, 9, 11, 12, 13, 1, 2, 1
Scripting, 2, 4, 7, 8, 9, 22, 36, 43
scrum, 8, 11, 12, 13

solve, i, 2
SUSE, vii, 25
system, i, ii, iii, iv, vi, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14,
16, 17, 18, 21, 22, 23, 25, 27, 28, 31, 33, 34, 35, 40,
41, 42, 1, 2, 2, 1, 2, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3,
4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1, 2

T

technologies, 13, 16
Terminal, 4, 6, 10, 43, 1, 12, 2
terminal session, 14, 17
tests, 11, 26, 27, 28, 29, 32, 1

U

Ubuntu, 1, 6, 14, 16, 25, 32, 34, 40, 41, 1, 7, 9, 11, 1
Unix, 8, 9, 37, 39, 43, 45
user-group, iii, iv, 27, 28