Creative Technologies

Module name:	UNIX
Tutor:	Robert Campbell
Assignment Title:	A UNIX administration facility.
Issue Date:	1st November 2016
Submission Deadline:	9th December 2016

This coursework constitutes 50% of the grade for the module and covers the following learning outcomes:

LO:2 correctly perform advanced UNIX system administration tasks and configure a system to an appropriate level of security.

LO:3 design and develop shell scripts for specified administration purposes.

Overview

It is normal for an organisation to adopt a multi-tiered approach to systems support. The first line of support is generally a service desk staffed by people who have a broad overview of the IT infrastructure. In other situations, the first line of support might even be one of the users, this is particularly common if the users are programmers or database administrators who although technically competent, are unfamiliar with the UNIX operating system.

The first line of support (whoever they are) will then seek to deal quickly with the more trivial issues such as a systems needing closed down for maintenance, a file that needs deleted or someone who has forgotten their password. If there is a more serious issue, first line support will then pass the incident onto the problem management team where UNIX experts can be found.

There is an obvious problem here, first line support are not competent in using the UNIX command prompt but they need to perform routine simple tasks. To facilitate this, simple to operate programs that interact with the system on the operators behalf are common. Such programs are often menu driven and enable first line support staff to perform simple functions like adding new users, changing user passwords, closing the system down for maintenance, or generating a report on system health that can be forwarded to a specialist team. In this assignment you will write such a programme. In the first instance, you should develop a menu driven script that an operator can run, that is capable of performing various tasks on their behalf. Once it is developed, the concept can be advanced as you see fit. What your script covers is completely up to you, however for guidance purposes you may want to consider the following functionalities:

1. System reboots
    - Reboot the system the immediately.
    - Reboot the system on the condition that nobody is currently logged in.
    - Reboot the system on the next occasion that no one is logged in.
    - Schedule a reboot for a later time and date.
    - Report on when the last reboot occurred.

2. User management
    - Add a new user.
    - Delete an existing user.
    - Change a password.
    - Confirm the existence of a user.
    - Report on users who have not logged in for a given amount of time.
    - Delete all users who have not logged in for a given time amount of time.
    - Report on when each user last logged in.
    - Report on which users have entered a given command

3. Network management
    - Show the arp table
    - Ping a remote machine
    - Add an IP address
    - Delete an IP address

4. File management
    - List the files in a directory
    - Confirm the existence (or absence) of a file
    - Add or delete a directory
    - Delete a file
    - Find a file
    - Archive a directory’s contents.

5. Health reporting

Generate a report on system health that can be e-mailed to a specialist team for analysis.
(Hint - you might find some of the following commands useful: top, vmstat, who –b, free, df, du users, ps –eaf, chkconfig –list, ifconfig, ping, stat, arp)

Operation

At the outset you should create a menu driven interface capable of performing a series of functions on the operator's behalf. Once your menu driven script is working you might like to advance its implementation by, for example:

- Setting up an operator login which allows someone to log in as 'operator', once logged in the operator would not be provided with a shell but the menu that you have created.
- Adapting your script so that it creates a log of what tasks have been actioned by which operators when.
- Providing a single option that causes an extensive written report on the system's health to be written that could then be sent to the second line support team
- None interactive flags could be introduced. For example the script (call it 'oppscript') could be set so that if it was run with parameters "-u rhc1" (i.e. "oppscript –u rhc1") it would add a user called rhc1.

Guidance on grades

Grade D (40 to 49%)
A basic menu driven script that has four functioning options each of which presents some useful information. For a grade D this information need not be formatted but may simply be the output of four relevant commands.

Grade C (50 to 59%)
As above but a reasonable range of functions should be available to the user (perhaps twelve options). Output and menu structure should be formatted and tidy as should the script. For a higher C grade the techniques used within the menu options to should include at least a couple of conditional and/or iteration statements.

Grades A and B (60 to 100%)
Getting the higher grades will depend on your ability to use advanced UNIX features and to overcome systems administration obstacles. When you think something might be hard to resolve – why not give it a go?

Deliverables

Your submission should be sufficient to inform 'the markers' of all your achievements. Consider the following.

- A listing of your script(s), configuration files and a description of how they are implemented
- A series of screen shots and outputs that provide an explanation of everything that your script(s) is capable of.