CPT264 – UNIX (Linux) Systems Administration and Programming (USAP), Study Period 3, 2019.

Assignment 2

Due:

Part A: 22nd October, 9 pm

Part B: 12th November, 9 pm

Please remember that all times specified in this assignment specification are specified in Melbourne local time. No extensions shall be given for misreading of deadlines or being in a different time zone.

Part A – Kernel Compilation (20%)

This component of the assignment requires you to compile and install a custom kernel. The instructions for doing this are contained in separate documents available here: https://docs.google.com/document/d/183dyIIUMDA9KyfQGrWR5gO8DZBatEy3DlWmJrJsEdtc/edit?usp=sharing and https://medium.com/@jagan.dream/cross-compile-flash-kernel-for-raspberry-pi-3-29699275abc6

You must make two changes to the compiled kernel:

- append your student number to the kernel version so that it displays when you enter the command "uname -r".
- turn off all audio support

You are to submit the compiled kernel and device drivers in a tar.xz file so we can mark them. They must be submitted using the following form, and you can upload your submission as many times as you like: https://docs.google.com/forms/d/e/1FAIpQLSct8MyE9NVqzedflnDPoltp7L3BldgUeWwWnln_3qI0sfiMGw/viewform

Please note that this submission will be BIG (500 mb compressed, 1.7gb uncompressed). If you submit multiple attempts, I will delete your previous attempts as the form only supports a maximum of 1 terabyte of space. Also note you will need to be logged into google drive with your RMIT credentials to submit your kernel.

You'll need to install a package called pxz for this - it's a parallel implementation of xz which we have previously seen in the course. Compress the kernel using the following command:

```
tar cvf - linux | pxz > studentno-linux.tar.xz
```

Where studentno is your actual rmit student number and linux is the directory containing the compiled kernel and linux source tree (this itself will take quite a while).

For this component of the assessment, you will get ten marks for successfully building a kernel that boots, and 5 marks for each modification that is required.

General Requirements

The general requirements for this assignment are marks that are allocated to part B of the assignment.

General Requirement 1 – Code Quality (10%)

An important part of the management of Linux installations is to use shell scripts in combination with the right choice of system programs. Many of the skills you will learn in this assignment are using programs you would use for system maintenance. You should use standard Linux utilities to fulfil each requirement.

Your program will be required to report on aspects of the state of the computer system via a menu-driven interface. Your scripts must be written using the bash shell scripting language with the addition of utilities such as grep, awk and sed. In particular you should not use python or perl scripting languages (I'm mentioning this as some students had some confusion about this in the past). If you are unsure whether a tool you are using is acceptable, please ask on the discussion board.

All scripts written for this assignment should be written with the appropriate preamble (the shebang/hash-bang). All your scripts should have appropriate permissions to be runnable by a normal user as ./scriptname where "scriptname" is the name you have given the script. All paths to executables need to be specified by constants at the beginning of your script, you should quote variables to avoid certain kinds of issues with the shell, variables should have meaningful names, should be local if they are not required for the sharing of information. Also, implement all options in functions if they are more than a line or two.

You will also be required to have your scripts be validated by shellcheck. In the lab exercises we will provide instructions for how to install this. Do not use the web version of shellcheck as it is not consistent with the installed binary which is what we will use in marking.

General Requirement 2 - Git Log(5%)

You are required to maintain a **private** git repository for this assignment. You may have this on the provided UNIX servers, your raspberry pi, or you may use a service like github, bitbucket or gitlab. You are required to commit on a regular basis. You must ensure that when setting up git you modify the global config so that your full name and your student email address are added to each commit.

You may export your git log for submission as follows:

```
git log --pretty >gitlog
```

which will create a text file that you may submit which has your commit history.

General Requirement 3 – Project Documentation (15%)

This requirement is broken into two parts – comments and user documentation. You are expected to provide comments sufficient so that a maintainer of your code will be able to understand what you are trying to do solely from the comments. There should be file level, function level and also comments in the code itself which clearly show your intention. I don't want you to go overboard with this but it does need to be sufficient so that a maintainer can understand your intentions. You should also identify yourself in all files.

You are also expected to provide user documentation in part B which explain to a user how to use your program. Your user documentation should follow the standard manpage format (one man page for each script). See https://www.tldp.org/HOWTO/Man-Page/q3.html but feel free to leave out a section if it is not relevant.

You must provide man page sources in groff format and a script called "build-documentation" which will create the documentation and output to text files.

Scripting Requirements

These are the actual tasks you need to complete for the assignment. These need to be implemented with the general requirements in mind but they will be marked separately. You will need to do some research about what commands / programs will give you the required information.

Do not share the names of those programs on the discussion board but rather perhaps hint on where you looked to find your answer. With this in mind, you may use the wiki on canvas to share resources you have found that help you with the assignment. The address for this wiki is: https://rmit.instructure.com/courses/64226/pages/further-reading

Please note that all menus and prompts are expected to be presented in a professional manner. Use headings at the start of each requirement (underlined) and give clear output, your scripts should be robust so that even if a user does something wrong they behave in a robust manner, giving clear error messages. All unexpected inputs must be handled in a robust manner.

Part B Requirements

Part B Scripting Requirement 1 – basic information (10 %)

Write a script which, based on the argument passed in (a script that takes user input here will get a fail mark) will display one of the following:

- the amount of free and occupied memory on the system in human readable format (that is 500 mb rather than 500000000 bytes).
- the amount of disk space occupied and free on the system in human readable format.
- the connection information for each network connection on the system (as a minimum, the ip and mac addresses for all connections on the system should be displayed).
- The system load average.

Please note that for this requirement you are not expected to do any post-processing of the output from these functions but just to call a program to get you this information. If your solution does involve doing post-processing that is fine but realise that you are not required to do that for this first requirement and won't get any marks for it here.

Part B Scripting Requirement 2 – basic information with post processing (10%)

This requirement is to write a bash script using getopts to receive the arguments that provides the ability to do the following:

- the number of cpu cores on the system
- the current process' priority (nice number)
- the total number of processes running under the current user
- the number of open file descriptors to regular files owned by the current user
- the maximum system stack size.

The scripting requirements below (requirements 3 and 4) are much more difficult than the earlier ones. That's my intention. I would only expect a hd-level student to complete all of these.

Part B Scripting Requirement 3 - Find script (10%)

You are to write a script that gets information from the user to run a query on the system using the `find` command.

You need to get from the user:

- the starting point (directory) where the search should start.
- the thing to search for. This can be one of path, type, group, fstype
- The value that is applied to the thing to search for. this might contain wildcards so you'll need to ensure that what is entered by the user is escaped.
- the maximum depth of the search (if desired) and whether or not to follow symbolic links.
- The action to be taken on the search results - this may be one of delete, print, print0 or custom which will allow the user to enter a scriptlet that should be applied to each search result.

Note: for some inputs you need to protect their expansion by the shell or they will not do the correct thing.

Part B Scripting Requirement 4 - Basic Profiler (10%)

For this requirement you are to prompt the user for the name (not the path) to a program that they want to do a basic profile of. If there is a partial match to two programs currently running then you will display a menu asking them which they want to run (for example they type chromium and both chromium-browser and chromium-bsu are running). For this purpose I have found that both `top` and `ps` can be useful. You will need to process that output with programs like `grep`, `sed` and `awk`.

Next, you will ask the user whether they want to monitor memory or cpu utilization. Next, you will run a background script that you have written. The background script will flicker the red led on your raspberry pi in proportion to the memory or cpu utilization of the process. That is, if the process utilization is 20% of the specified resource then the led should be on for one fifth of every second. This means that every second your background script will poll the specified resource

This script must be run in the background because when the user presses [enter] in the foreground process, you must terminate this background script gracefully.

Part B Scripting Requirement 5 - Menu System (10%)

Finally you need to create a script that provide a menu system by which the user can select from the menu the various functionality that you have implemented. At the start of this script, you must disable ctrl-c and each level of menu must have a way to exit gracefully to the previous menu level and the whole script should only exit from the top level. Your menu must be well-presented and must be well-behaved; all inputs must be validated and should behave as a reasonable user might expect. You may wish to clear the screen at each new menu but I'm not concerned either way about that.

What to Submit?

For the scripting submission, submit a .tar.gz file. This file should contain your script files, your user documentation and your git log. Please note that submissions of any other format will receive a 10% penalty.

Late Submission Policy and Extensions

You may request extensions no less than 24 hours before a component is due but please note that your request must comply with university policy as outlined here: https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/adjustments-to-assessment

In the absence of extensions, you will lose 10% of the marks available for each day late, up to five days and then no marks can be awarded for submissions that are later than this.