ECEE 5623, Real-Time Systems:

Exercise #2 – Service Scheduling Feasibility

DUE: As Indicated on Canvas

Please thoroughly read Chapters 3 & 4 in RTECS with Linux and RTOS

Please see example code provided - Linux, FreeRTOS, VxWorks, Zephyr

This lab is written **to be completed with embedded Linux running on the R-Pi3b (order or borrow and use NOOB image). It is possible to also use a Jetson Nano (Getting started),** Jetson TK1/TX1/TX2 (JetPack install) or Beagle board(s). **The standard final project requires use of a camera, which is simple with embedded Linux using the UVC driver**.

Exercise #2 Requirements:

1) [5 points] If you're using embedded Linux, make yourself an account on your R-Pi3b, Jetson Nano, or Jetson TK1 system. To do this on Jetson TKI, use the reset button if the system is locked, use the well-known "ubuntu" password to login, and then use "sudo adduser", enter the well-known password, and enter user information as you see fit. Add your new user account as a "sudoer" using "visudo" right below root with the same privileges (if you need help with "vi", here's a quick reference or reference card– use arrows to position cursor, below root hit Esc, "i" for insert, type username and privileges as above, and when done, Esc, ":", "wq"). The old unix vi editor was one of the first full-screen visual editors – it still has the advantage of being found on virtually any Unix system in existence, but is otherwise cryptic – along with Emacs it is still widely used in IT, by developers and systems engineers, so it's good to know the basics. If you really don't like vi or Emacs, your next best bet is "nano" for Unix systems. Do a quick "sudo whoami" to demonstrate success. Logout of ubuntu and test your login, then logout. Use Alt+Print-Screen to capture your desktop and save as proof you set up your account. Note that you can always get a terminal with Ctrl+Alt+t key combination. If you don't like the desktop, you can try "GNOME Flashback" and please play around with customizing your account as you wish. Make sure you can access our class web page on Firefox (or default browser) and set your home page to http://ecee.colorado.edu/~ecen5623/index_summer.html . Overall, make sure you are comfortable with development, debug, compiler general native or cross-development tools and document and demonstrate that you know them.

2) [10 points] Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" [available on Canvas], by Gene Carlow and provide an explanation and critique of the frequency executive architecture. What advantages and disadvantages does the frequency

executive have compared to the real-time threading and tasking implementation methods for real-time software systems? Please be specific about the advantages and disadvantages and provide at least 3 advantages as well as 3 disadvantages.

3) [50 points] Download feasibility example code and build it on a Jetson or alternate system of your choice (or ECES Linux if you have not mastered the Jetson yet) and execute the code. Compare the tests provided to analysis using Cheddar for the first 4 examples tested by the code. Now, implement remaining examples [5 more] of your interest from those that we reviewed in class (found here). Complete analysis using Cheddar RM. In cases where RM fails, test EDF or LLF to see if it succeeds and if it does, explain why. Cheddar uses both service simulations over the LCM of the periods as well as feasibility analysis based on the RM LUB and scheduling-point/completion-test algorithms, referred to as "Worst Case Analysis". Does your modified Feasibility code agree with Cheddar analysis in all 5 additional cases? Why or why not?

4) [15 points] Provide 3 constraints that are made on the RM LUB derivation and 3 assumptions as documented in the Liu and Layland paper and in Chapter 3 of RTECS with Linux and RTOS. Finally, list 3 key derivation steps in the RM LUB derivation that you either do not understand or that you would consider "tricky" math. Attempt to describe the rationale for those steps as best you can do based upon reading in Chapter 3 of RTECS with Linux and RTOS.

[20 points] Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code and make sure your code is well commented, documented and follows coding style guidelines. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

Note: Linux manual pages can be found for all system calls (e.g. fork()) on the web at http://linux.die.net/man/ - e.g. http://linux.die.net/man/2/fork

In this class, you'll be expected to consult the Linux manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Canvas and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on an embedded Linux system (R-Pi 3b+ or Jetson). Please zip or tar.gz your solution with your first and last***

name embedded in the directory name and/or provide a GitHub public or private repository link. Note that I may ask you or SA graders may ask you to walk-through and explain your code. Any code that you present as your own that is "re-used" and not cited with the original source is plagiarism. So, be sure to cite code you did not author and be sure you can explain it in good detail if you do re-use, you must provide a proper citation and prove that you understand the code you are using.

Grading Rubric

[5 points] Create account on Jetson: ____

[10 points] Shuttle PASS paper review:

[3 points] Three advantages ____

[3 points] Three disadvantages____

[4 points] Overall understanding of paper and key point articulation_________

[50 points] Shared CPU system overload:

[5 pts] Example #1 code _______________________________________

[5 pts] Example #2 code _______________________________________

[5 pts] Example #3 code _______________________________________

[5 pts] Example #4 code _______________________________________

[5 pts] Example #1 Cheddar confirm_____________________________

[5 pts] Example #2 Cheddar confirm _____________________________

[5 pts] Example #3 Cheddar confirm _____________________________

[5 pts] Example #4 Cheddar confirm _____________________________

[5 pts] Example #5 Cheddar confirm _____________________________

[5 pts] Example #6 Cheddar confirm _____________________________

[15 points] Shared CPU system overload:

[3 pts] C#1, A#1 _______________________________________

[3 pts] C#2, A#2 _______________________________________

[3 pts] C#3, A#3 _______________________________________

[2 pts] key step #1 _______________________________________

[2 pts] key step #2 _______________________________________

[2 pts] key step #3_______________________________________

[20 points] Quality of reporting and code quality and originality:

> [10 pts] Professional quality of reporting, testing and analysis (0…6 is below average, 7 is average, 8 is good, 9 excellent, and 10 is best overall.)____
>
> [10 pts] Code quality including style, commenting, originality, proper citation for re-used code, modified code, etc.____

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

Note: Linux manual pages can be found for all system calls (e.g. fork()) on the web at http://linux.die.net/man/ - e.g. http://linux.die.net/man/2/fork

In this class, you'll be expected to consult the Linux manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Canvas and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on Ubuntu VB-Linux or a Jetson. Please zip or tar.gz your solution with your first and last name embedded in the directory name.***