ECEE 5623, Real-Time Systems:

Exercise #5 – Real-Time Continuous Media Systems

(Start of Extended Lab Individual or Group Effort)

DUE: As Indicated on Canvas

Please thoroughly read Chapters 9, 10 & 11 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**. Linux examples can be found here - http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/code/

This exercise asks you to explore some of the real-time software engineering errors made in historical systems and how they might have been avoided. In researching and reviewing the case, be sure to identify (or imagine) what the real-time requirements were and how failure to meet them resulted in failures – often facts will be unavailable, so do your best to reconstruct.

Exercise #5 Requirements:

1) [15 points] Research, identify and briefly describe the 3 worst software and hardware computer engineering design flaws and/or software defects of all time [product lack of success can be considered a failure such as Blackberry Storm, Windows Genuine Advantage, Windows 8, Windows ME, Apple Lisa, Pentium FPU bug, as well as mission failures such NORAD false alarms, Mars Express Beagle 2, Challenger and Columbia Shuttle Loss]. State why the 3 you have selected are the worst in terms of negative impact, negligence, and bad decisions made, but why the failure was not really a real-time or interactive systems design flaw. Rank them from worst to least worst.

2) [15 points] Research, identify and briefly describe the 3 worst real-time mission critical designs errors (and/or implementation errors) of all time [some candidates are Three Mile Island, Mars Observer, Ariane 5-501, Cluster spacecraft, Mars Climate Orbiter, ATT 4ESS Upgrade, Therac-25, Toyota ABS Software]. Note that Apollo 11 and Mars Pathfinder had anomalies while on mission, but quick thinking and good design helped save those missions]. State why the systems failed in terms of real-time requirements (deterministic or predictable response requirements) and if a real-time design error can be considered the root cause.

3) [20 points] The USS Yorktown is reported to have had a real-time interactive system failure after an upgrade to use a distributed Windows NT mission operations support system. Papers on the incident that left the USS Yorktown disabled at sea have been uploaded to Canvas for your review, but please also do your own research on the topic.
   a) [5 pts] Provide a summary of key findings by Gregory Slabodkin as reported in GCN.
   b) [5 pts] Can you find any papers written by other authors that disagree with the key findings of Gregory Slabodkin as reported in GCN? If so, what are the alternate key findings?
   c) [5 pts] Based on your understanding, describe what you believe to be the root cause of the fatal and near fatal accidents involving this machine and whether the root cause at all involves the operator interface.
   d) [5 pts] Do you believe that upgrade of the Aegis systems to RedHawk Linux or another variety of real-time Linux would help reduce operational anomalies and defects compared to adaptation of Windows or use of a traditional RTOS or Cyclic Executive? Please give at least 2 reasons why or why you would or would not recommend Linux.

4) [50 points total] ***Form a team of 1, 2, or at most 3 students and propose a final Real-Time prototype, experiment***, or design exercise to do as a team. For the summer RT-Systems class, all groups are asked to complete at time-lapse monitoring design as outlined in requirements with more details provided in this RFP. Based upon requirements and your understanding of the UVC driver and camera systems with embedded Linux (or alternatives such as FreeRTOS or Zephyr), evaluate the services you expect to run in real-time with Cheddar, with your own analysis, and then test with tracing and profiling to determine how well your design should work for predictable response. You will complete this effort in Lab Exercise #6, making this an extended lab exercise, and ultimately you will be graded overall according to this rubric and you must submit a final report with Lab Exercise #6 as outlined here.

   a) [30 pts] Your Group should submit a proposal that outlines your response to the requirements and RFP provided. This should include some research with citations (at least 3) or papers read, key methods to be used (from book references), and what you read and consulted.

   b) [20 pts] Each individual should turn in a paragraph on their role in the project and an outline of what they intend to contribute (design, documentation, testing, analysis, coding, drivers, debugging, etc.). ***For groups of 2 or 3, it is paramount that you specify individual roles and contributions***.

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

[15 points] 3 worst software failures:

[4 points] #1 _____

[4 points] #2_____

[4 points] #3_____

[3 point] ranking_____

[15 points] 3 worst real-time system failures (design and implementation flaws):

[4 points] #1 _____

[4 points] #2_____

[4 points] #3_____

[3 point] ranking_____

[20 points] USS Yorktown investigation:

a) [5 pts] Key findings _____
b) [5 pts] Alternate key findings_____
c) [5 pts] Root cause and culpability_____
d) [5 pts] RT Linux ok, why or why not?
   _____

[50 points] Final exercise proposal:

[30 points] Group proposal _____

[20 points] Personal contribution_____