Real-Time Embedded Systems

Exercise-2

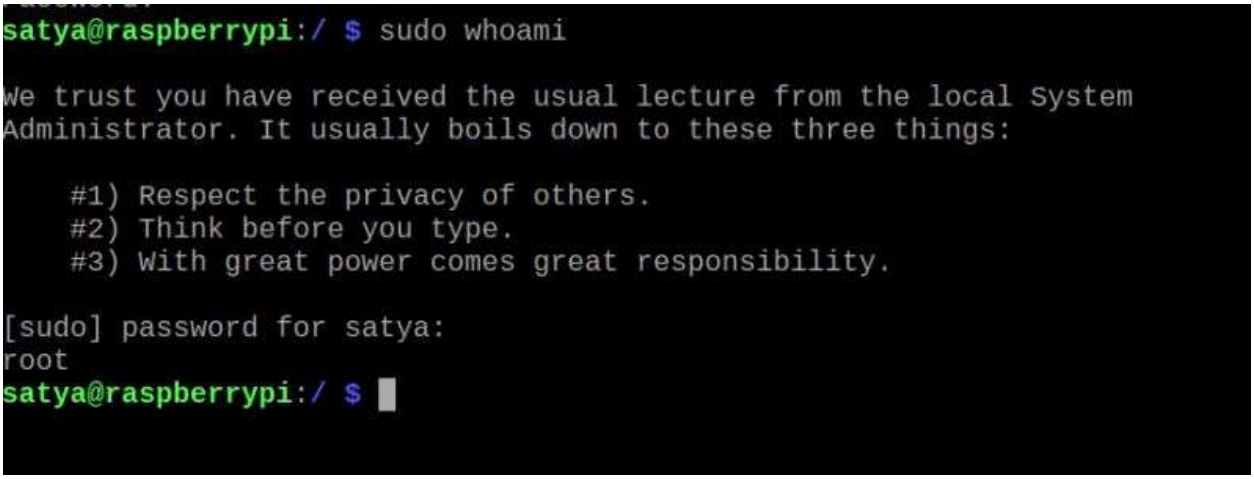
Satya Mehta, Souvik De, Sean Duffy

Date: - 02/21/2019

Boards used: - Q1) Raspberry Pi, Q4) Virtual Box

---

1) [5 points] make yourself an account on your Dev Kit. To do this, use the reset button if the system is locked, use your password to login, and then use "sudo adduser", enter a 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 Linux 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.




Satya Mehta
Shut Down
Log In


Satya Mehta
pi
Other...
Shut Down
Log In

2) [10 points] Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" [available on D2L], 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.

The Frequency Executive Architecture, also known as the Cyclic Executive Architecture, is a system architecture which schedules tasks by organizing function calls which execute those tasks in a central loop. Therefore, there is no generic scheduling software **which saves code space *(Advantage #1)*** and consumes no timing overhead to execute **such a generic scheduler *(Advantage #2)***. **This simplicity also leads to a very repeatable system with relatively few different possible significant scenarios *(Advantage #3).*** However, because the function calls which execute the tasks are all organized in a central loop, the timing between executions of these tasks is **dependent upon the execution time of other tasks** and the number of other tasks in the system ***(Disadvantage #1)***. This could lead to issues in the development process, particularly in more complex systems. Another issue with the Frequency Executive Architecture is that **the latency could be more variable** and, in worst cases**, worse than other architectures *(Disadvantage #2)*.** This is because the only opportunity for a task to begin responding to an event is when the main loop gets to the (or one of the) function call(s) which initiates that task. Finally, on systems with high utilization, one task (even a low priority one) overrunning its deadline could cause all of the tasks to miss their deadlines (unless the practice described in **"**Building Safety-Critical Real-Time Systems with Re-useable Cyclic Executives" is employed and a task **that overruns its "time-budget" or deadline then it is aborted) *(Disadvantage #3)*.**

3)[5 points] Read the paper "Building Safety-Critical Real-Time Systems with Reuseable Cyclic Executives", available from http://dx.doi.org/10.1016/S0967-0661(97)00088-9. In other embedded systems classes you built ISR (Interrupt Service Routine) processing software and polling/control loops to control for example stepper motors – describe the concept of the Cyclic Executive and how this compares to the Linux POSIX RT threading and RTOS approaches we have discussed.

Cyclic Executive, also known as Frequency Executive. is described above in Question 2. Cyclic Executive is quite distinct from a system that utilizes Linux POSIX RT threading and other RTOSs because the scheduling of tasks is built into the organization of the source code of itself whereas typically other schedulers are separate pieces of software. Therefore, there are no equivalents to the POSIX functions like "pthread_create" or "pthread_join".

In addition to the tradeoffs discussed above, this could also make a system that utilizes a cyclic executive more difficult to port to another operating system while preserving the performance. Furthermore, using

a cyclic executive, as outlined in the paper, can lead to a very rigid architecture which forces tasks to fit into a limited number of formats which may not be natural given the nature or purpose of the task.

However, with regards to safety-critical applications, systems utilizing a cyclic executive architecture (particularly in projects where the requirements are well defined at the beginning of the project and do not change significantly during development) are much simpler to test thoroughly because of the limited number of code paths. Similarly, they are much more predictable (and controllable). If the tasks of the system are well-suited, they can also be simpler (and cleaner) to develop because of the limited number of places where the task software can be preempted than in preemptive RTOSs.

4) 50 points] Download Feasibility example code and build it on a Jetson, DE1-SoC or TIVA or Virtual Box and execute the code. Compare the tests provided to analysis using Cheddar for the first 4 examples. Now, implement the remaining examples [5 more] that we reviewed in class (found here). Complete analysis for all three policies using Cheddar (RM, EDF, LLF). In cases where RM fails, but EDF or LLF succeeds, 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?

Feasibility code example executed on Virtual Box with all the examples: -

```
satya@satya:~/Satya/Spring19/RTES/EX-2$ ./feasibility_tests
******** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE


******** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
satya@satya:~/Satya/Spring19/RTES/EX-2$
```

EDF and LLF Output: -

```
********************EDF Test*****************************

Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): LCM of periods of ex0_period:- 30 FEASIBLE
Ex-1 U=0.99 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): LCM of periods of ex1_period:- 70 FEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex0_period:- 910 FEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): LCM of periods of ex3_period:- 15 FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): LCM of periods of ex4_period:- 16 FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): LCM of periods of ex5_period:- 10 FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex6_period:- 910 FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): LCM of periods of ex7_period:- 15 FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex8_period:- 910 FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): LCM of periods of ex9_period:- 24 FEASIBLE


*********LLF Test*****************

Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): LCM of periods of ex0_period:- 30 FEASIBLE
Ex-1 U=0.99 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): LCM of periods of ex1_period:- 70 FEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex0_period:- 910 FEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): LCM of periods of ex3_period:- 15 FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): LCM of periods of ex4_period:- 16 FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): LCM of periods of ex5_period:- 10 FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex6_period:- 910 FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): LCM of periods of ex7_period:- 15 FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): LCM of periods of ex8_period:- 910 FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): LCM of periods of ex9_period:- 24 FEASIBLE
satya@satya:~/Satya/Spring19/RTES/EX-2/EDF$
```

Analysis of all the services using Cheddar: - (Screenshots in the zip)

Example 0: - The analysis using Cheddar matches the output of the test code provided.

Example 1: - The analysis using Cheddar matches that of output of test code provided. RM protocol has failed to give us any information about the feasibility of the scheduled set, since utilization factor of said set equals 0.84 which is greater than the RM Least upper bound of 0.78. In this same test EDF and LLF have succeeded in verifying the feasibility. This is because these two protocols are based on using dynamic priorities. RM LUB using fixed priority is unable to optimize tasks based on whose deadline is nearest, or "earliest". Also, the feasibility test for this dynamic priority-based protocol is U <=1 which holds true for this task sets.

As a side note Cheddar 3.1 has a prospective bug. It can plot the schedule for this task set correctly but mentions in the 'Scheduling feasibility' section that the task set cannot be schedule. Cheddar 2.1 has no such discrepancy. Screenshots of both versions have been provided.

Example 2: - The analysis using Cheddar matches that of output of test code provided. RM protocol has failed to give us any information about the feasibility of the scheduled set, since utilization factor of said set equals 1.0 which is greater than the RM Least upper bound of 0.78. In this same test EDF and LLF have succeeded in verifying the feasibility. This is because these two protocols are based on using dynamic priorities. RM LUB using fixed priority is unable to optimize tasks based on whose deadline is nearest, or "earliest". Also, the feasibility test for this dynamic priority-based protocol is U <=1 which holds true for this task sets.

As a side note Cheddar 3.1 has a prospective bug. It can plot the schedule for this task set correctly but mentions in the 'Scheduling feasibility' section that the task set cannot be schedule. Cheddar 2.1 has no such discrepancy. Screenshots of both versions have been provided.

Example 3: - The analysis using Cheddar matches the output of the test code provided.

Example 4: - Test agree with the analysis of Cheddar, but there is another possible bug in Cheddar 3.1. Although utilization for the task set equals 1, RM declares the service sets as feasible. Because its deriving an RM LUB of 1 instead of 0.78.

Example 5: - The analysis using Cheddar matches the output of the test code provided.

Example 6: - It is quite similar to example 2, only difference being deadlines have been provided which are different from the periods. Due to this, Cheddar analysis says RM analysis cannot be computed since the above is the requirement of the RM. Deadline monotonic also fails in Cheddar but EDF and LLF are scheduled successfully.

Example 7: - The analysis using Cheddar matches the output of the test code provided.

Example 8: - Example 8th is same as 2nd so the analysis remains same.

5) [30 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 the text. 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 the text.

In order to derive the RM LUB of U ≤ m(2^(1/m) – 1), Liu and Layland made the following assumptions which led to the resulting LUB being constrained to a subset of problems:

Assumption	Constraint
Requests for all tasks are periodic	Only systems in which requests for tasks are made at a predefined period can use the RM LUB safely. Even using the worst-case (shortest) period as part of the RM LUB calculation does not guarantee accurate results.
Deadlines for tasks are equal to the period of the requests	Only systems in which it is not necessary to complete a task until the next request for that task is made can use the RM LUB safely.
Tasks are completely independent from one another	Only systems in which tasks do not depend on one another or restrict access to certain shared resources (other than the processor) can use the RM LUB safely.

Each of these constraints limit the utility of the RM LUB by eliminating systems with some common characteristics. The constraint that all tasks be periodic eliminates systems in which requests come in randomly, such as those that come as a result of some human input. However, this constraint could be

overcome in the LUB calculations if the indication of this request is only checked and serviced at some fixed period in the system and the addition of that period as latency is acceptable from a performance perspective. The constraint that all deadlines be equal to the period of the requests eliminates a large number of systems in which requests, perhaps those that come at a fairly low frequency, need to be serviced very quickly. Finally, requiring task independence creates issues for systems which has services which require exclusive access to resources for a period of time and those in which tasks must execute in a specific order for the system to function properly.

Furthermore, these assumption-constraint pairs do not capture all of the assumptions or constraints of the LUB, some other assumptions are: All tasks always require constant execution time, the time used by the scheduler and thread switching is negligible, and All tasks are "preempt-able" at any point in their execution.

Do not understand/"tricky math":

It is unclear to me why in Theorem 4 there is a constraint that "the ratio between any two request periods is less than 2", but at the end of the proof for Theorem 4, it is mentioned that "the restriction that the largest ratio between request period less than 2 in Theorem 4 can actually be removed". Theorem 5 then apparently illustrates why those with the largest ratio between the request period less than 2 are the only cases that need to be considered. It seems this is done to simplify the proof of Theorem 4 by splitting it from the proof that some assumption made in the Theorem 4 proof is valid (which is done in Theorem 5). In the textbook, Figures 3.6 and 3.8 seem to illustrate that the timing behavior is somewhat cyclical (but not exactly) and that the behavior when the ratio is < 2 covers the full range and both extremes of the timing behavior for all other ratios, so considering this range effectively considers all other possible values.

In Theorem 4, the proof that Cm-1 = Tm - T(m-1) for tasks that fully utilize the processor and minimize the processor utilization factor seems to use some "tricky" math to set up the equations that will eventually be combined and used to determine the minimum (or LUB) through the taking of the derivative. The use of the delta to identify the 2 cases and express them mathematically was astute, but tricky. This was done because it illustrates the behavior of the utilization factor in the two cases (one were the last requested instance of S1 can fit in the critical instant and one where it cannot).

At first, the starting point of the proof for Theorem 4 is confusing because it defines a set of tasks for which the processor is fully utilized, but the processor utilization factor is minimized, but from reading the just of the proof and the textbook it seems the "fully utilize the processor" simply refers to a set of tasks in which the increase of any computation time or the decrease of any period would cause the system to fail through a task failing to meet its deadline. These are the sets of problems which are interesting to analyze in the proof/derivation of the LUB.

REFERENCES: -

- Architecture of the space shuttle primary avionics software system. Link: - https://dl.acm.org/citation.cfm?id=358258
- Building Safety-Critical Real-Time Systems with Reusable Cyclic Executives. http://dx.doi.org/10.1016/S0967-0661%2897%2900088-9
- Feasibility Code examples: - http://mercury.pr.erau.edu/~siewerts/cec450/code/Feasibility/