Name: Cameron Stark

Exam 2 CEC 450: Real-Time Systems



1. How are the seven basic I/O operations used by device drivers? **Explain**.

The Seven basic I/O operations, with them being create, remove, close, read, write open, and ioctl, are built into both the device and the windriver environment. When writing code the programmer is not directly calling the device drivers however is calling the methods that are mapped to the device drivers, similar to an API for a website, that way the program is not communicating directly to the device methods.

1. How does memory-partitioning support the reliability of systems? **Fully justify**.

Memory Partitioning supports the reliability of systems because it keeps a separation of concerns where the developer can control what parts of the application can access certain parts of memory for example there are the 4 levels of memory Registers, Caches, Main Memory and Disk Storage. The registers and caches are used primarily by the hardware to control context switching and tasking on processes, there for the application which resides in Main memory should not exist in the same partition as registers and caches, however because the application will use information in Disk storage, one would think they could exist in the same partition, that is also not encouraged because of preventing data loss, if the program were to fail, it could delete or damage the information if it existed with it, therefore the data and program should be separate.

1. How do we make a real time system more dependable? Give a **complete** answer including examples.

There are many ways to make a real time system more dependable, similar to other types of applications these ways revolve failing fast, which when a program comes across a fault whether it be a logical one or value being incorrect, the program should prevent this issue from propagating throughout the application. This can be achieved by having a clean state saved before the error occurred where the system can role back to or by having numerous error catching systems to prevent errors from getting to other parts of the system. Another means of ensuring dependability is data/information security, with means of preventing the system from directly modifying stored and new information, there should always be a intermediary process for communicating between the two sections. This is to maintain data safety, so that in the event an error occurs the API to the data should be able to prevent a call being made to the data if the request is not verified by some means.

1. **Explain** the main difference between a kernel task and an ISR.

The main difference between kernel task and ISR is that ISR runs at interrupt time and is not a task, and is loaded upon an interrupt in the current context whereas a kernel task is a more traditional task that can be ran at any point during the program. Also with ISR being a part of the interrupt process it will have a higher process priority.

1. What is the purpose of signals? How are they best used? **Explain**.

If interrupts are the physical switching of contexts on the processor and the operating system, the signal is the software version where rather than pausing or switching execution the signal triggers other events throughout the application, for example when a program detects an error may have occurred it will send out a signal, which will then be received by the error handling function that is listening for that signal, and the realm of errors the program will stop execution until the error handling function returns a signal telling the program how to proceed, which can be continue as normal or kill the process.

1. Can increasing the reliability of a RT system decrease its response time? **Justify**

Yes and no, increasing the reliability of the system could reduce the response time because it could make the program larger, with the additions of error catching methods, security measures to prevent data deletion or corruption, with a larger program having a larger footprint on the hardware it could slow down the system, however because a lot of these systems are always being used in every instance, they may have a very small affect on the response time.

1. In class we discussed that safety is a top-down approach. What is meant by this? **Fully explain and provide at least one example.**

The top down approach is the viewing of the system from the big picture perspective, as to why safety is viewed this way is because understanding what the system is doing as a whole is the optimal way to define safety features. For example a sliding door as a whole top down system it as a sensor on either of the door searching for movement within its cone and when movement is detected it opens, the main safety aspect that can be derived from the overview is the door should not close if movement is detected in the cone. Whereas a bottom up approach would focus on the how and when the door should open based on the movement and how fast to open.

1. When is it appropriate to use a Signal? Why? **Explain**

Signals are best used in cases where an error has potentially occurred in the program during runtime, and the current function signals the error catching function(s) to determine if the error was actually an error and if need be can resume execution of the process called the error function or end the process if need be.

Given the following periodic tasks following the format of TaskName(TaskPeriod, TaskExecutionTime): TA(40ms,10ms), TB(120ms, 30ms), TC(80ms, 20ms).

1. Design the cyclic executive for these tasks by listing when each frame starts, what tasks execute and when the execution ends (list idle time too).

**Frame Starts Tasks Executed Execution Ends**

t = 0 TA, TB t = 40

t = 40 TA, TC t = 70

t = 70 TA, TC t = 100

t = 100 TA, TB t = 140

t = 140 TA, TC t = 170

t = 170 TA t = 200

t = 200 TA, TB t = 240

t = 240 Cycle repeats

1. Calculate the system utilization and the major cycle for all these tasks.

Cycle = 40 + 120 + 80 = 240

TA = 10 / 40 = 0.25

TB = 30 / 120 = 0.25

TC = 20 / 80 = 0.25

1. Assume two more tasks are added: TD(100ms, 15ms) and TE(500ms, 4ms). TE is a very rare ***hardware*** interrupt. Fill the table below with execution times, deadlines, the rate monotonic priorities, and compute the utilization:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Priority (High, Medium High ,Medium, Medium Low ,Low) | Execution Time | Deadline | Utilization |
| TA | Hight | 10 | 40 | 0.25 |
| TB | Medium High | 30 | 120 | 0.25 |
| TC | Medium | 20 | 80 | 0.25 |
| TD | Medium Low | 15 | 100 | 0.15 |
| TE | Low | 4 | 500 | 0.008 |

1. Is there a technique to determine if these tasks will meet their deadlines? Using this technique will they? (**Show the calculations**)

Yes, there is a technique to determine if these tasks will meet their deadlines, it is the Utilization Bound test associated with the Rate Monotonic Methods. The test takes the total utilization of the tasks and checks it against the utilization bound. If the total is less than the bound, then the deadlines will be met, if total is greater than bound, than the deadlines will not be met.

Total Utilization:

Total = (executionTime / Period) + …

Total = T = (10/40) + (30/120) + (20/80) + (15/100) + (4/500) = 0.9

Utilization Bound:

Bound = B = n(2^(1/n) – 1)

Bound = B = 5(2^(1/5) – 1) = 0.74

T < B -> 0.9 < 0.74 = False

The tasks will not meet there deadlines

1. **Complete** the following table by calculating the task utilization, rate monotonic priorities, total utilization and calculate the response times (**show your calculation steps below the table**). All times are in ms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Priority (High, Medium, Low) | Execution Time | Period | Utilization | Response Time |
| TA | Medium | 10 | 20 | 0.5 | 14 |
| TB | High | 2 | 10 | 0.2 | 2 |
| TC | Low | 7 | 50 | 0.14 | 35 |

Calculations:

Total = (10/20) + (2/10) + (7/50) = 0.5 + 0.2 + 0.14 = 0.84

TA -> because priority is medium response is combined with TA

10

10 + ceiling(10/10) \* 2 = 10 + 2 = 12

10 + ceiling(12/10) \* 2 = 10 + 4 = 14

10 + ceiling(14/10) \* 2 = 10 + 4 = 14

TB -> because priority is high has a response time equal to execution time

R = C

TC -> because this is low priorirty response time is combined with TA and tb

7

7 + ceiling(7/10) \* 2 + ceiling(7/20) \* 10 = 19

7 + ceiling(19/10) \* 2 + ceiling(19/20) \* 10 = 21

7 + ceiling(21/10) \* 2 + ceiling(21/20) \* 10 = 33

7 + ceiling(33/10) \* 2 + ceiling(33/20) \* 10 = 35

7 + ceiling(35/10) \* 2 + ceiling(35/20) \* 10 = 35

