University of Waterloo

cs452 – Final Examination

Spring 2011

Student Name:	_____
Student ID Number:	_____
Unix Userid:	_____

Course Abbreviation:	cs452
Course Title:	Real-time Programming

Time and Date of Examination:	09.00, 12 August, 2011 to 11.30, 13 August, 2011.
Duration of Examination:	24 hours.
Number of Pages:	7.

Rules of the Examination.

1. You must work independently.
2. You may use any source of information you want on this examination. Any information from sources you consult MUST be referenced. (Your memory, course notes and lectures are the only exceptions.)
3. I prefer answers in PDF format (whatever.pdf). If PDF is inconvenient then I accept plain text (whatever.txt) but you will have to stretch a little to make diagrams. Two pages (~800 words, or less if there are diagrams) is as long an answer as you need for any question; some questions require less. Put your name, student number and userid on every page.
4. Your answers should be submitted by e-mailing them to me at `wmcowan@cgl.uwaterloo.ca`.
5. A strategy that works well for me is to read the exam twice, then do something else for a couple of hours, then plan my answers, rest again, and finish by writing them. The total writing time should be about three hours.
6. Please remember that the questions are open-ended: you get most of your marks from going beyond simple answers; explicit instructions are intended as prompts to get you started in the right direction. You gain marks for the thoughts that you contribute to your answers. Write other people's thoughts, including mine, only to the extent that I need them to understand your answer.
7. When the examination says 'your kernel' it means the kernel you actually created, not an ideal kernel or the kernel you wish you had created. When the examination says 'your OS' it means your kernel plus the other tasks (couriers, notifiers, servers) on top of which applications run. When the examination says 'your train application' it means the application you tried to create in what you would consider to be its final form.
8. When asked to estimate something you should respond using an almost logarithmic, which contains only the values …, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, … . All measurements and estimates must have units.
9. There may be places in the examination where you can say something clever by making assumptions. I encourage you to do so. Be certain that your explain your assumptions and how they are related to what you are saying.
10. Read each question carefully, and more than once. More marks are lost because of misunderstood questions than from any other single cause. To show that you read this far, for one mark put the phrase 'nolens volens' at the top of your first page. The advice to read at least twice is self referential.
11. The cover page exists only to fulfil the registrar's regulations.

Do Questions 1 & 2.

Question 1. Performance Analysis of your Kernel.

This question concerns two aspects of your kernel, Send/Receive/Reply and AwaitEvent.

1.a Send/Receive/Reply (SRR). In the second part of the kernel you measured the time taken by SRR for small and large messages with optimization turned off.

- **(i)** What were your times? During a system call three things occur, context switch(es), copying memory, and manipulating kernel data structures.
- **(ii)** Using the logarithmic scale introduced in the introduction, estimate parameters A and B in the equation

$$SRRTime = Am + B,$$

where m is the amount of memory copied in bytes. Answers should have units. Give your reasoning.

1.b AwaitEvent. Assume that an interrupt occurs while execution is in the kernel while it is processing the AwaitEvent call. Between a call to AwaitEvent and its return the three things mentioned above, (i), may occur, context switch(es), copying memory, and manipulating kernel data structures. If you measured the minimum call/return time for AwaitEvent, give the result, otherwise estimate it. Estimate the amount of time spent in switching context, copying memory and manipulating kernel data structures.

1.c Context Switches. Counting the number of context switches in **1.a** & **1.b**, you can estimate the time for a context switch. Not counting turning on the caches and compiler optimization, what did you do after the third kernel assignment to speed up your context switch? Explain your answer. Estimate the speed up.

1.d Caches. Assume that both instruction and data caches of the ARM are lockable. Explain how you could use cache locking to improve the time taken by your context switch. Estimate the execution speed-ups when you turn on the caches, both with and without locking. Give as many concrete details as you can, such as how many cache lines to lock, what to put in the locked cache lines, the effect of a smaller cache on performance of user code, and so on.

Question 2. Global Variables

2.a Admonitions. Several times in class I told you not to use global variables, yet many students used them. Explain why global variables are thought to be bad. (A global variable is a variable that can be read and/or written by any task.)

2.b Actual Usage. If you used global variables what did you use them for? How were they implemented? Why did you use them? Answer for uses in the kernel and in your project, as appropriate.

If you did not use global variables say so, and answer the question above, explaining where, how and why you think other students used them.

2.c Simon Says. 'that there is only one safe way to use a global variable: in the exact order given,

- **(i)** one, and only one, task writes the global variable;
- **(ii)** there is a synchronization barrier that all tasks cross at the same logical time; and
- **(iii)** after the barrier any task may read the global variable, but no task may write it.'

Describe the class of critical races that Simon's recommendation avoids. Explain, using a real or hypothetical example, the symptoms of this type of critical race, and what you would have to do to debug it.

2.d The Track Graph. A data structure that can be used in accord with Simon's recommendations is the track graph.

- **(i)** Draw a simple task diagram showing the tasks that accessed (or would have accessed) a global variable containing the track graph, with a short description of how each task used the graph.
- **(ii)** How would each of these tasks deal with the problem of modifying the track graph to account for a turn-out that is usable in one direction, but unswitchable?
- **(iii)** What problem might occur with your answer to (ii)? Explain.

2.e Track Data. A data structure that contains the current state of the track cannot be used in accord with Simon's recommendations.

- **(i)** Explain why.
- **(ii)** Are there general principles you can draw from the contrast between the track graph and track data? Describe them and justify your answer.

Do Two (2) of the Remaining Three (3) Questions.

(All questions expect answers of the same size and difficulty.)

Question 3. Calibration.

3.a Control. 'The details of calibration always depend on the controlled system, including the application task it performs.' Explain this statement, giving concrete examples from your train application, and how it differs from other possible applications.

3.b Static versus Dynamic Calibrations. Explain the benefits and drawbacks of static and dynamic calibrations, giving concrete examples from your train application. Which of the following did you actually implement:

- **(i)** static calibration,
- **(ii)** dynamic calibration, or
- **(iii)** a mixture of static and dynamic calibration?

Giving concrete examples explain which you implemented and why. Would you do things differently if you were starting your project again? How? Why?

3.c Calibration Performance. You are expected to have enough familiarity with the train(s) you used that you can provide estimates for

- **(i)** train speeds,
- **(ii)** time lags in giving commands to trains and switches,
- **(iii)** times and distances for stopping trains,
- **(iv)** and so on.

The train control and reservation systems in your train application had some difficult cases. Describe two such cases, giving all of the

- **(i)** concrete details and estimated numbers with units,
- **(ii)** performance requirements,
- **(iii)** technical details, and
- **(iv)** so on

that you consider relevant.

Question 4. QNX

QNX (`www.qnx.com`), recently acquired by RIM, is a company based on a real-time operating system kernel created by Dan Dodge and his partner in an early offering of CS452. But you may be puzzled if you look at the API of QNX: it seems to have every communication/synchronization primitive you have ever encountered.

I know that at least one group implemented locks as a way of controlling access to resources. There are two ways of implementing locks, as a kernel primitive and as a lock server. There is a third way of doing the task that a lock performs, putting the resource itself in a server which accepts requests using Send/Receive/Reply and performs the requested operations.

4.a Software Management. In my view, 'mixing primitives from several synchronization models is a problem looking for a place to happen, and it has found the place: your code.'

- **(i)** If you agree with the above statement explain what the problem is, and what happens when it occurs. If you disagree with the above statement, please explain why.
- **(ii)** As the manager of a programming team developing using QNX how would you avoid the problem?

4.b Lock Server. Let's reject locks as a kernel primitive compatible with a micro-kernel.

- **(i)** Explain the incompatibility, or disagree if you choose. (Even if you disagree you must do the remainder of the question.)
- **(ii)** Describe the API you would provide for user code accessing a lock server, explaining why each part of it is necessary. Give a schematic description of how you would implement the API. (Your API should take into account the possibility that the set of lockable resources may not be known at compile time.)
- **(iii)** How would you exclude a rogue task from using the resource without acquiring the lock?

4.c Resource Proprietor. One reason often given for preferring locks to servers is execution efficiency. This part of the question makes an elementary comparison between the efficiency of accessing a data structure controlled by a lock server or a proprietor. Each requires

1. synchronization, acquiring and relinquishing the lock (lock server) or receiving and responding to the request (proprietor),
2. manipulating the data structure by the user (lock server) or the proprietor, and
3. transferring data.

- **(i)** In terms of efficiency of manipulating the data structure, (2.), is a wash, but synchronization and data transfer are not. Describe how the two solutions differ in those two cases.
- **(ii)** Estimate, using the timings you have for your kernel how much time each consumes if n bytes of data are transferred.
- **(iii)** At what value of n do you change from preferring one solution to preferring the other?

4.d Priorities. Comment on the following statement. 'A resource proprietor uses the resource at a priority determined by the importance of the resource, while a lock server allows a resource to be used at the priority of the user.'

Question 5. Caches in Multi-core Systems.

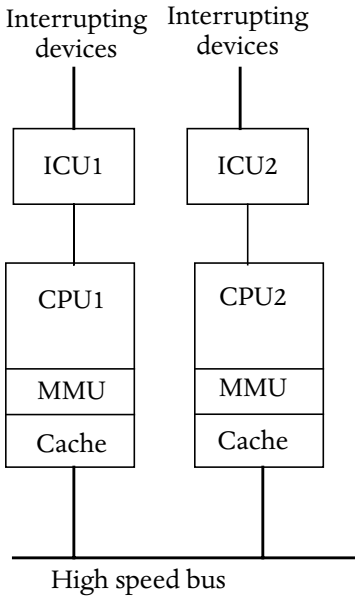
Several times during the lectures I claimed that it's easy to generalize your kernel to run on multiple processors. This question asks how you would solve some of the problems of doing so.

What you're doing in this question is very topical. You might be interested in looking at a few articles from the blog of James Hamilton, a Vice President at Amazon, which you can find at

`http://perspectives.mvdirona.com/default,date,2009-09-07.aspx`
`http://perspectives.mvdirona.com/default,date,2009-09-16.aspx.`

They address the costs of running a data centre doing transaction processing and particularly the heat generated, concluding that a multi-core ARM processor is likely to be a competitive solution. (Most data centres, such as Amazon's, are loaded primarily with transactions on data bases: customers submit requests; the requests are translated into SQL queries; the SQL queries are parallelized at small granularity; the fragments are scattered over several processors; and the results are later gathered into a response to the request, which is replied to the client.)

Substantially oversimplifying, base your answer on a multicore system with the hardware architecture described in the diagram to the right. An important aspect of this architecture is shared memory: the two CPUs have the same physical address space and different virtual address spaces, because each CPU has its own MMU. Note also that interrupts from the various devices in the system are hardware routed to either one CPU or the other, never to both.



5.a Overall Architecture. Give a rough idea of how you would spread your kernel across the two core system shown to the right. You should address which parts of the kernel would execute on each CPU, what kernel data structures they would access, how tasks on different CPUs communicate, how interrupts get to the tasks waiting on them, and so on. Remember that there is not one single correct answer to this question: you should respond as a designer, not as a student writing a calculus examination.

5.b Task Granularity. In the course I pushed the idea that task structure is a useful way of structuring programs, and that it is both possible and desirable to have many tasks, each of which does something simple well, and that the application as a whole is well-structured as a collection of cooperating tasks. Hamilton, above, is suggesting that SQL queries can be split up into parts that are provided to tasks that specialize with respect to algorithms or data. For example, tasks on a specific CPU might use parts of the data base that can be maintained in cache. You should see immediately that the design shown has a caching problem. What is it? What sort of hardware is needed to solve the problem? How fast does this hardware need to be in order to provide a good solution.

5.c MMUs. Caching within the MMUs is extremely important to obtain adequate memory performance, and you should remember TLBs from CS251. Each CPU its own set of TLBs: how does this affect cache usage? Explain your answer. (Hint. You may be interested in reading the MMU section of the ARM processor documentation.)

5.d Other Caches. Where else in the system are there caches that would matter? How would you manage them?