CS452 : Real-time Programming
Spring 2017
General Information

Bill Cowan
University of Waterloo

Term and Year of Offering:
Spring 2017.

Course Number and Title:
cs452/652, Real-time Programming.

Instructor:
Bill Cowan dc2111 x34527 wmcowan@cgl.uwaterloo.ca.

Teaching Assistants:
Ben Cassell.
David Choi.

Course Newsgroup:
defunct.

Course Email:
cs452@cgl.uwaterloo.ca (Email sent to this address goes to the instructor, the TAs and Fraser Gunn.)

Course URLs:
http://www.student.cs.uwaterloo.ca/~cs452/
http://www.cgl.uwaterloo.ca/~wmcowan/teaching/cs452

Lecture Times:
10.30 – 11.20 Monday, Wednesday, Friday.

Lecture Room:
mc4058.

Laboratory:
mc3018.

Programming Environment:
gcc, Ubuntu Linux.

Optional Text:
Sloss, Symes & Wright. *ARM System Developers Guide*.

1. Course Description

Tools and techniques of real-time programming, this course including microcomputer architecture, system support for real-time applications and calibrating interfaces to hardware, with an emphasis on modern high-capability embedded systems.

2. Course Objectives

At the end of the course successful students will be able to:

1. program low level system services in assembly language,
2. program hardware interfaces in C,
3. develop a multi-task application based on message passing, and
4. develop calibrated control of external hardware.

3. Important Dates

Assignment 0 due:
Wednesday, 10 May [Week 2]
Kernel 1 due:
Friday 26 May [Week 4]
Kernel 2 due:
Monday, 31 May [Week 5]
Kernal 3 due:
Monday, 5 June [Week 6]
Kernel 4 due:
Monday, 12 June [Week 7]
Train control 1 due:
Thursday, 29 June [Week 9]
Train control 2 due:
Thursday, 13 July [Week 11]
Project proposal due:
Wednesday, 5 July [Week 10]
Project demo:
TBD, about Wednesday, 26 July [Week 13]
Final examination:
TBD, during exam period

4. Important Skills Required for this Course

Reading.
Experimenting.
Programming.

5. Text, References, and Documentation

All the information you need to complete the course work is available on-line. The course web pages link to it.

The organizing knowledge and the skills you need are provided in the lectures. The instructor's lecture notes are on the course web pages.

More extended presentations of specific course material exist for some topics, also available in the course web pages.

The optional text provides a good introduction to ARM programming. In one respect, however, be cautious. The Small Operating System described in the book is quite different in philosophy and structure from the one you create in this course.

6. General overview of topics to be covered

Real-time Systems (1 hour)

Introduction to real-time systems.
Definition of a task, or process.

Concurrency and CPU Multiplexing (2 hours)

Review of concurrency and CPU multiplexing.
A simple application using cyclic execution for concurrency.

A Real-time Operating System (15 hours)

Multiple processes.
Inter-process communication and synchronization.
Interrupts.
Input/Output.

Process Structuring Techniques (6 hours)

Typical tasks: servers, clients, workers.
Task structuring for application programs.
Pathologies: deadlock, performance problems.
Debugging.

Control Applications (6 hours)

Control concepts such as feedback.
Calibration.
State machines.

Supplementary Topics, depending on instructor (6 hours)

Real-time, concurrent programming languages.
Type-safety.
Optimal control.
Multiprocessors, scheduling.
Analog/digital interfaces.
Other real-time programming paradigms.

7. Electronic Resources

Much of the course material is available on the course web pages, and there is much more relevant material on the web. Beware! Not every contributor to the web is a genius; many are not even competent; and the average contributor is less intelligent than you are. Also, it is too easy to copy code from web pages into your programming editor. When you do this it is essential to credit the source of the 'borrowed' code: in comments *and* in your documentation.

Subscribe to the course newsgroup, uw.cs.cs452, using your favourite news reader. Read the newsgroup daily if you want to be up to date on assignment clarifications, hardware availability and tips from your classmates. *Material posted on the newsgroup is expected to be known by every student.*

The teaching assistants and the instructor monitor the newsgroup and answer any relevant technical or policy questions. Often students provide faster and/or more precise answers.

8. Marking Algorithm

The final mark for the term will be computed according to the following weighting.

Assignment 0	5%
Kernel	30%
Project	35%
Final Exam	30%

You will notice that this weighting places 65% of the final mark on activities that are done in partnership with another student. You must choose your partner carefully, especially as the instructor does not have a marriage counselling licence. We reserve the right to request independent 'who did what' statements from each member of a partnership.

Please Note. The assignments in this course are cumulative. Therefore it is important that I manage the course so that almost all students succeed on almost every assignment. This suits me just fine: students learn a lot from doing an assignment on which they succeed or almost succeed. But this causes a problem: assignment and project marks don't vary a lot from student to student, even though it is quite obvious, to you and to me, that some students have achieved quite a bit more than other students.

My solution to this problem is to give a take-home examination with relatively open-ended questions, and to mark it in the European style, which uses the entire range of marks, so that a passing mark is about 10 out of 30, with marks spread out up to 30 based on students going beyond routine answers to the questions. After several terms of experience I have observed that this solution has two extremely important properties.

1. The marks it gives correlate well with my perceptions of how much students have learned in the course. This includes groups where I have perceived the contribution of the partners to have been unequal.
2. The mark range it gives corresponds well with faculty and school expectations for a fourth year specialist course.

9. Assignment Evaluation

For each assignment you hand in your source code and documentation. The documentation should describe the design decisions you made, and why you made them. Occasionally there are also questions to be answered.

The teaching assistants mark the assignments based on the contents and form of the documentation, the performance of the program and the quality of the code

If, on receiving a marked assignment, you have complaints or requests for clarification, they should be made to the teaching assistant who marked your work within one week of the return of the marked assignments. After one week, there is no mark alteration.

No Credit Will Be Given For Late Assignments.

10. Rules for group work

Assignment 0, the quizzes and the final examination are to be done individually: the kernel and project are done in groups of two.

How much may work be shared among individuals or groups that should be working independently? My rule of thumb is that students are in university to learn: learning from fellow students and teaching fellow students are equally encouraged. What is handed in, however, *must* be the student's own work, and *must* reflect what the student can do and has done.

If you are ever in any doubt if what you have done is okay, document where you got the information, with references. You are always in the clear when you do so.

11. Late and missed assignments

No late assignments will be accepted. Any assignment missed or submitted late will be assigned a mark of zero.

12. Submitting assignments and getting them back

Assignments are submitted in class during the first five minutes of the lecture on the day they are due. They are returned in class after they have been marked.

13. Examinations

The final exam will be scheduled by the Registrar. It will cover material from the entire course. The format of the final examination is decided by a vote of the students. Most terms the final examination is take-home.

Past final exams are available on the course web pages.

14. The Freedom of Information and Privacy Protection Act

The Freedom of Information and Privacy Protection Act (FIPPA) states that third parties should not be able to obtain information about a student without their consent. It is possible that in the course you will develop software that you want to leave for the benefit of future students.

Normally we attach the name of the author who, after all, holds the copyright, but if you wish to remain anonymous that is possible.

15. Questions

Questions about course material, due dates, or other course related matters may be directed to the instructor during class or to the course newsgroup. You are expected to attend class and to read the newsgroup: we assume you know any material provided there. Questions about assignments should be directed to the teaching assistant who is responsible for the assignment, during office hours or by e-mail. (Any answers that we deem to be of general interest will be discussed in class or on the newsgroup.

The teaching assistants will unless otherwise noted will spend their posted office hours in the trains lab. Please do not bother them at other times: they have commitments in addition to cs452!

The instructor provides office hours on demand whenever his door is open, which is about nine hours a day. His office is a short walk from the trains lab and your best strategy for seeing him is to stretch your legs and walk over. (You may even discover the answer to your question just by standing up and changing your context.)
Important. All e-mail from the instructor will go to your account on student.cs.uwaterloo.ca. You should read your mail on that account regularly.

16. University mandated information

16.a. Academic Integrity

In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility.[*]

16.b. Grievance

A student who believes that a decision respecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. Read Policy 70,Student Petitions and Grievances, Section 4.[†] When in doubt please contact the department's administrative assistant who will provide further assistance.

16.c. Discipline

A student is expected to know what constitutes academic integrity,[‡] to avoid committing an academic offences, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (e.g., plagiarism, cheating) or about 'rules' for group work/collaboration

[*] www.uwaterloo.ca/academicintegrity/
[†] www.adm.uwaterloo.ca/infosec/Policies/policy70.htm
[‡] www.uwaterloo.ca/academicintegrity/

should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean. For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline[*]. For typical penalties check Guidelines for the Assessment of Penalties[†].

16.d. Appeals

A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances), other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals)[‡].

16.e. Note for Students with Disabilities

The Office for persons with Disabilities (OPD), located in Needles Hall, Room 1132, collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the OPD at the beginning of each academic term.

* www.adm.uwaterloo.ca/infosec/Policies/policy71.htm
† www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm
‡ www.adm.uwaterloo.ca/infosec/Policies/policy72.htm

Laboratory Overview

No food or drink is allowed in the lab. That is, at no time when the instructor or any university official looks into the lab should there be any evidence of eating or drinking.

1. Computing Environment

Software development is based on the two box model. You edit your code on one box, and cross-compile it to run on a second box. You then download the executable to the second box and run it.

The first box is one of several PCs running Ubuntu Linux. The second box is an embedded processor based on the Cirrus EP9302 system on a chip, which contains the ARM 920T architecture. The cross-compiler is GCC.

You will probably want to use gmake to control compilation and link-editing.* You and your partner will probably also want to agree on a version control system, even if you only use it for revert.

The second box contains a version of the RedBoot boot loader. It downloads the executable over the campus network, and provides a standard environment in which execution begins. It also allows you to examine and change memory and to exercise devices by hand.

The second box is available in quantity one at about the price of an expensive textbook. If you get one to use at home that's fine, but it remains your responsibility to ensure that you code runs correctly in the lab, where it will be tested.

2. Lab Etiquette

Near due dates, the lab is busy. Seats in the lab are first come first served. You may have your favourite seat and may almost always use it, but only when you are the first to want it. You can't eject another student from it.

The lab is for the exclusive use of cs452 students. Do not give other students the combination. From time to time you will be happy that the couch is not occupied by a vagrant who isn't in the course.

3. Assignment Outlines

Assignment 0: An Introduction

You write a program for controlling a single train interactively using a polling loop. You get accustomed to the two-box programming environment and discover how to control multiple activities using polling loops.

* The instructor uses mk, the equivalent Plan 9 utility, because he likes to understand his makefiles.

Kernel 1: Tasks
: You create tasks and run them successfully under control of your scheduler, using a context switch, managing memory and designing your task descriptor.

Kernel 2: Message-passing for communication and synchronization.
: You add communication primitives to the kernel so that tasks can exchange data and synchronize their execution.

Kernel 3: Hardware Interrupts
: You use the overflow interrupt of a hardware counter to create a clock server that introduces time into the kernel.

Kernel 4: Serial I/O
: You provide the kernel with the ability to exchange data with external hardware by implementing a serial server using two UARTs, and you re-implement Assignment 0 based on your kernel.

Train control 1:
: You use sensor input and a calibration to know the exact position of a locomotive on the track.

Train control 2:
: You create a routing algorithm to route a locomotive to a given destination in the presence of other locomotives. You generalize the calibration you created in the first train control milestone, to keep track of several locomotives moving around the track simultaneously.

Project Proposal:
: The project proposal states the goal of the project, its technical challenges and the means by which you intend to surmount them.

Project:
: The project implements a scenario created and designed by your group. It should be challenging to implement, but not too challenging, and *must* be real-time, which means that it must have failure modes if trains arrive too late and if trains arrive too early. The instructor will provide, in class, examples of possible train projects as suggestions.

4. Ambiguities or Omissions in the Assignments

To emphasize their incompleteness we give out assignment descriptions, not specifications. The kernel API must be respected but many implementation details are up to you. Some are mentioned explicitly in the assignment description; others you will discover on your own.

If you are unclear with respect to what is acceptable the best thing to do is to post to the newsgroup long before the assignment is due. Then all students can take advantage of the clarification. If you are doing it in the long dark hours before the assignment is due specifically mention in your documentation what the question is, and why you chose the interpretation you chose.

5. Cheating

The programming assignments are designed to give practice on the topics presented in lectures. If you have any questions about what may or may not constitute cheating, please ask the instructor so you don't end up cheating 'by mistake'.

If you are thinking about cheating, think about cheating at driving school. Having your friend do the practice driving is not the best way of getting a licence.

If you cheat and we catch you you're lucky. You take another course, graduate a little later and life goes on. If you're unlucky we don't catch you. You then go on the job market, and because cs452 is a good credential you get a programming job. Your employer, of course, expects you to perform at the level of a cs452 graduate, but you can't. Hanging on by your fingernails, you use cheating tactics to get your work done, all the while scared that you might get caught. And if this cheating is successful you get a higher paid and more challenging job. Now things are even worse. You get the idea: cheating is its own punishment.

6. Previously Written Code, Use of Other People's Code

The code you submit should be code you wrote. For the project, it may make sense for you to use code written by someone else to assist you in your work. Further, you may wish to reuse code you wrote in previous terms. In both cases, you should check with the instructor before handing in the assignment, and credit the code (even if it was code you wrote in previous terms) in your documentation. However, you do not need to give the source of code that we provide for you, which isn't much. See the discussion of projects for more information on using code you previously wrote as part of your project.

7. Additional information

Make sure you don't change the specified default behaviour if your program has extra features. Let your extra features be optional and mention them in the README. You can lose marks for the things that are not explicitly stated in the assignment objectives. Submission of poorly written code has already been mentioned, for example. Failure to put the executable in the proper place is another example, and it will also result in deduction from the assignment mark. You can also lose marks for submission of a program with a poor user interface.