

48450 Real-time Operating Systems

Spring 2005

Assignment 2

Due date: Friday, 28th October

Contribution to final mark: 20% (marked out of 20 marks also)

Mode: Individual completion and submission

Introduction

This assignment permits you to select from a collection of independent options based on your interests and motivation. The options are graded such that there are ceiling marks associated with each – this is used to reflect the level of depth and difficulty associated with the option. The topic descriptions are deliberately open-ended, affording you the opportunity to set the scope of work. A submission will be marked on its merits and will be awarded a mark which is less than the ceiling if it's of modest quality. You should select one of the options only, complete it *including a reflective self-assessment* in the conclusion and submit it by the due date above. Submission details will be made available at a later date.

Option A

Class of work: Research Investigation, Literature Survey

Ceiling: 14 marks out of 20

Choose one of the following topics and investigate it before writing and submitting a report. Use the descriptions as a starting point only. As a rough guide, something of the order of ten A4 pages of text and diagrams would be expected.

Topic 1: *The Common Object Request Brokering Architecture (CORBA)*

Find out what CORBA is and what its relationships are with respect to user level processes and an OS kernel. How is it implemented and what features does it possess that make it interesting? What advantages and disadvantages does its use and deployment entail? Compare it with competing proprietary technologies.

Topic 2: *Multithreaded programming under Windows and POSIX*

Examine the differences and make a comprehensive comparison between thread management under Windows (any version later than NT 4.0) and POSIX pthreads. Ensure you study the differences at the API level and at the behavioral level. Consider issues like porting between the two environments and performance on single and multiprocessor platforms.

Topic 3: *GUI systems*

Most modern operating systems support a GUI of some sort. Investigate *briefly* the history of their development (from XEROX PARC onwards) and compose a report which evaluates them from a **technical** perspective (instead of from an ergonomic or "User Friendliness" perspective). Consider issues such as the architecture of a GUI (various different models like client-server as opposed to integrated), their resource requirements compared to a CLI, interactive performance, I/O device communication and programming considerations as a starting point, and continue from there.

Option B

Class of work: Research Investigation, Programming Exercise

Ceiling: 17 marks out of 20

Choose one of the following topics, investigate it and develop some code that can be used to demonstrate the concept(s) before writing and submitting a report. As a rough guide, something of the order of fifteen A4 pages of text and diagrams, (or less, if a greater part of the submission is source code) would be expected. Use the descriptions as a starting point only.

Topic 1: Nachos

Nachos is an instructional operating system that is available to students to experiment with. It can be downloaded from <http://www.cs.washington.edu/homes/tom/nachos/> and installed on the user's machine. There is an overview of Nachos in Appendix C (page 764) of the textbook. The task here is to download and install Nachos (either on your own computer, or under your account on the coldfire server) before experimenting with it in order to gain familiarity with its capabilities. You should then develop and complete a sample project of your own under Nachos and complete your work by making an informed statement about Nachos' suitability as an operating systems teaching tool.

Topic 2: Process start-up times

This topic will involve some application program development using POSIX clocks to measure the time taken to create a new process. You are to construct a simple program which sets up a POSIX.4 `CLOCK_REALTIME` clock and get it to output the current time. Then it should make a call to `fork()` to generate a child process and measure and output the time again (from parent and child). Collect a sample of measurements and tabulate them – have a look at any apparent statistical variation and describe and rationalise your findings. Create another program that measures how long the system needs to create a thread – make another collection of measurements and see how this compares. You can complete this project entirely on the coldfire server.

Option C

Class of work: Research Investigation, Kernel Programming Exercise

Ceiling: 20 marks out of 20

A demanding programming exercise, not for the faint-hearted.

Topic 1: uClinux shell

This topic follows on from your programming task in Assignment 1 and involves the development of a shell which is to be developed under Linux on the coldfire server and then ported to uClinux. The uClinux default shell is "sash" which you've used in Assignment 1, but now you're asked to develop your own. This option is based in "Programming Project One: Developing a Shell" from page 153 of the textbook. In order to scale down the requirements a little, I suggest your shell should meet the following requirements (a subset of those listed on pages 153 and 154).

1. Internal commands: i, iii, iv, v, vi, viii, ix.
2. All other command line input is interpreted as program invocation, which should be carried out by the shell forking and executing the programs as its own child processes. The programs should be executed with an environment that contains the entry:

`parent=<pathname>/myshell` where `<pathname>/myshell` is described in 1.ix. above.

Ignore the other functional requirements and the submission details that are suggested in the text – you should supply a Makefile, source code and some reflective text as a summary. This project should only be attempted by those that enjoy a moderately complex programming challenge. It's recommended that a well-structured incremental development approach be taken.