The University of Queensland
School of Information Technology and Electrical Engineering
Semester 1, 2007

COMP3301 COMP7308

ASSIGNMENT 1 - Scheduling

Due: **5pm Thursday 5th April**
Weighting: **20% (20 marks)**

Objective

As part of the assessment for this course, students are required to complete three assignments which will assess your understanding of the Minix operating system and ability to program at a low level in C. This particular assignment will assess your ability to understand the scheduling of processes in Minix and the implementation of a different scheduling algorithm for Minix.

The Assignment

The Minix scheduling algorithm will be explained in lectures briefly. Your task in this assignment is to replace the current algorithm with an implementation of lottery scheduling. This algorithm will not be discussed in lectures – it is up to you to find out the details of how it works and can be implemented.

Your implementation of this scheduling method should not break current system calls and programs packaged with Minix – for example nice(1).

You should base your version on that explained in the textbook (pages 107-108). You are required to alter the operating system source appropriately and generate a patch for a default kernel image. Your code should be appropriately commented to describe your code. In addition to this, you will be required to submit some user application(s) which will demonstrate the correct functionality of your code. This code should include a README file, describing how they demonstrate the scheduling algorithm.

Please note: This assignment does not necessitate the implementation of ticket transfers, ticket inflation, ticket currencies, compensation tickets, or any other functionality other than basic process scheduling – the task is to change the algorithm of scheduling processes only.

Marking

Your submission will include the following two files:

- s123456_ass1_final.patch (where 123456 is replaced with your student number)
- demo_program.tar (which will contain the code which demonstrates the operation of your scheduling algorithm, along with a suitable Makefile to build it and a README file explaining the procedure for operating the code)

In order to mark your assignment, we will use a default Minix image (which we will provide for your own testing purposes). The patch file will be applied in the /usr/src directory with the following command:

```
patch –p0 < s123456_ass1_final.patch
```

The kernel will then be built using "make install" and rebooted. Your programs will be tested by extracting them to /home and following the instructions in your README. Ensure that your submission will work correctly according to this procedure – particularly ensure that the patch file applies cleanly to the provided image.

Help sources

Help for this assignment is available from a number of sources. Help in understanding the Minix code is best obtained either from the textbook, or from the course newsgroup, uq.itee.comp3301. Help in understanding the algorithm is best obtained either from the internet or other textbooks. As a last resort, you can ask the tutor or lecturer for help.

Hints

It is recommended that tickets be allocated to processes based on their priority. This will help the implementation of lottery scheduling to avoid breaking current functionality.

Submission Details

The due date for the assignment is 5pm Thursday 5th April. The assignment must be submitted via the ITEE online submission system. Assignments may be resubmitted numerous times, but only the last submission will be marked.

Early Submission bonus

There will be a bonus for early completion: any assignment handed in 24 hours or more before the due date, which achieves at least 50% of available marks, will receive a 5% bonus. The maximum available for the assignment therefore is 105% of available marks.

Late Submission

Late submissions will **not be accepted except with a doctor's note and even then by permission of the course coordinator**.

Notification of Results

Students will be notified of their results via the course homepage. Students will be able to view their individual results only.

Allocation of Marks

Students will be marked on their ability to implement changes in the kernel code and the demonstration of how those changes can be seen in the operation of the kernel.

The following marking scheme will apply:

	Range applicable
Coding and testing: (14 marks)	
No significant changes to scheduler	0
Some attempt, but fails to boot or crashes with minimal testing	2-4
Boots, but fails some tests	5-8
Boots and passes all tests, algorithm incorrect	9-10
Boots and passes all tests, algorithm correct	11-14
Excellence: (2 marks)	
These marks will be allocated based on certain signs of excellence: • Efficient Code: Scheduling routine has been programmed to reduce time spent in algorithm, unnecessary code has been removed • Complete/Consistent Code: Existing code has been modified to be consistent with lottery scheduler	2
Documentation (4 marks)	
Nothing	0
Some, but not clear	1
Some, limited details	2
Reasonably clear	3
Very clear	4
Deductions	
Obscure code	2.5
Inefficient code (unnecessary computation)	1
Caps	
Inadequate tests	Capped at 14

Academic Merit, Plagiarism, Collusion and Other Misconduct

You should read and understand the statement on academic merit, plagiarism, collusion and other misconduct contained within the course profile and the document referenced in that course profile. You should note that this is an **individual assignment**.

Reference

Tanenbaum and Woodhull, Operating Systems: Design and Implementation (3rd Edition), Prentice Hall, 2006.