UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

OPERATING SYSTEMS

Thursday 6 June 2002

14:00 to 15:30

Examiners:

Prof. K. Turner (External)

Prof. C. Stirling (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. (a) Illustrate what might go wrong if two separate processes use a shared variable without any synchronisation. How does the concept of a *critical section* help with such problems? [*4 marks*]

(b) How might the ability for a process to disable processor interrupts help to achieve mutual exclusion of processes from a critical section? Why might the designer of an operating system think this a poor idea? [*4 marks*]

(c) Lamport's 'Bakery' algorithm, which solves the mutual exclusion problem for multiple processes, uses two arrays of shared variables, `choosing[n]` and `token[n]`, each having one element per process and which are initialised to zero. The scheme for process `i` takes the form:

```
while (TRUE) {
   choosing[i] = 1;
   token[i] = max(token[0], token[1], ..., token[n-1]) + 1;
   choosing[i] = 0;
   for (proc=0; proc<n; proc++) {
      while (choosing[proc] == 1) { }
      while (token[proc] != 0 && (token[proc] < token[i]
            || (token[proc] == token[i] && proc < i))) { }
   }
   ....   // critical section i
   token[i] = 0;
}
```

Explain how and why this works. [*7 marks*]

(d) How can *deadlock* arise in an operating system? Discuss possible strategies for prevention and avoidance of deadlock. Why are many of the techniques often described impractical? What are the consequences of allowing deadlocks to occur but then detecting and recovering from them? As an operating system designer, how concerned do you have to be about deadlocks? [*10 marks*]

2. (a) Describe the difference in structure of operating systems which are described as *Monolithic* and those which are based on a *Microkernel*. What benefits and drawbacks do each have? [*6 marks*]

   (b) Distinguish between *processes* and *threads*. In what ways can threads be more efficient than processes? Give examples of situations where threads might beneficially be used. [*7 marks*]

   (c) Suppose you were given the job of designing the CPU scheduler of a new operating system intended to be used in both workstation and server contexts. What criteria would you take into account and what might the characteristics of a resulting system be? How well would it be able to cope in these contexts and how likely would it be to need manual tuning to optimise performance? [*12 marks*]

3. As an operating system designer, how might you seek to influence the design of a new computer architecture? What features are likely to be important in allowing you to implement an efficient kernel? In particular, discuss interrupt handling, context switching and process and thread organisation. [*15 marks*]

   What features of a virtual memory organisation do you regard as important from a user's perspective? How would the choice of virtual memory organisation affect the design of a kernel? [*10 marks*]