UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

OPERATING SYSTEMS

Friday 19th August 2011

09:30 to 11:30

Year 3 Courses

Convener: K. Kalorkoti
External Examiners: K. Eder, A. Frisch

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

1. (a) Give **brief** (one sentence) definitions of the following key terms in memory management: *virtual memory*, *page table*, *segment*. [*6 marks*]

   (b) In the terminology used in the course, what is the difference between a *process* and a *thread*? [*2 marks*]

In addition to the traditional Unix `fork()` system call for creating new processes, Linux has a more general system call `clone()`, which allows the user more freedom to control what does and does not get shared between parent and child processes. As with `fork()`, the parent and child 'processes' both continue from the point of the `clone()` call.

One use of `clone()` is to implement thread creation: if `clone()` is called with the flag `CLONE_VM`, then the child 'process' runs in the same address space as the parent.

   (c) Recall that programs use the *stack* for temporary variables, function arguments etc. In most Linux systems, the stack is set up to grow downwards from a particular point in the process' memory space. What immediate problem does this present for thread creation via `clone()`? [*3 marks*]

   (d) The problem of the previous part is easy to solve in the operating system on an architecture that uses segments. How? [*2 marks*]

   (e) Unfortunately, Linux does not use segments, since it runs on some architectures that do not support segmentation. Suggest how the problem of part (c) might be avoided. [*3 marks*]

Another flag that can be passed to `clone()` is `CLONE_PARENT`. If this is given, the parent *process* of the cloned process will not be the cloning process, but the parent of the cloning process – that is, `clone()` produces a sibling rather than a child.

   (f) Does the possibility of specifying `CLONE_PARENT` together with `CLONE_VM` raise any new problem with your answer to part (e)? If so, describe the problem; if not, give a short justification. [*3 marks*]

   (g) There are several other flags which can be passed to `clone()` to make other process attributes be shared between the cloning and cloned process. Discuss what other attributes should be shared to provide a thread implemention. [*6 marks*]

2. (a) What are the main OS operations on a process during its life cycle? [*6 marks*]

   (b) What criteria need to be considered when designing the task scheduler of an operating system? [*5 marks*]

   (c) How does the Linux scheduler work? How well does it address the criteria you described in the previous part? [*5 marks*]

   (d) You are asked to customize a Linux system to be used to control autonomous automatically driven vehicles – that is, motor vehicles that are to drive on public roads without human attention. Discuss the impact of the scheduling policy on the performance of such a system. Do you think that the current Linux scheduler could handle the task? If so, justify your answer; if not, discuss how you would modify or even redesign the task scheduler.

   What other aspects, apart from task scheduling, are likely to have a significant impact on the performance of such a customized system? [*9 marks*]

3. (a) Briefly describe the *indexed sequential* format for file organization, and its advantages and disadvantages. [*5 marks*]

(b) In some operating systems, complex access methods such as indexed sequential are implemented within kernel routines. In others, the OS provides only basic file access, and user applications must provide their own complex access routines.

Discuss the possible reasoning behind these different approaches, and suggest scenarios which might favour one or the other. [*7 marks*]

(c) A question sometimes encountered from new users of multi-user Unix systems is, "I have write permission to the file `bar/foo`, but I can't delete it! How come?"

Using your knowledge of the structure of the Unix file system, explain how this situation can arise. [*5 marks*]

(d) Describe how the data in a Unix file is stored on disk. [*4 marks*]

(e) Unix supports direct access to the data in a file. Referring to your previous answer, explain why, even using direct access, it might be quicker to access data at the start of a large file than at the end. Would you expect this effect to be observable in practice (give reasons)? [*4 marks*]