Data Provided: None

The University Of Sheffield.

DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

Autumn Semester 2013-14 (2.0 hours)

EEE6032 Operating Systems

Answer **THREE** questions. **No marks will be awarded for solutions to a fourth question.** Solutions will be considered in the order that they are presented in the answer book. Trial answers will be ignored if they are clearly crossed out. **The numbers given after each section of a question indicate the relative weighting of that section.**

1. **a.** What is the minimal set of functionality which needs to be provided in a microkernel? Justify the inclusion of each component with reference to the microkernel philosophy. **(6)**

Explain why a microkernel might be a good architecture for a real-time operating system targeted at embedded systems. **(2)**

b. What modification needs to be made to the central-processing unit (CPU) of a computer to support pre-emptive multitasking? Explain how this additional feature is used by the OS to implement pre-emptive multitasking. **(3)**

c. Describe the low-level implementation of simple round-robin scheduling. How is the sequencing of processes managed by the OS? **(3)**

d. Operating systems make extensive use of interrupts. Many operating systems split interrupt handlers into a *top half* and a *bottom half*. The top half executes in response to an interrupt, spawns the bottom half as a process (or kernel thread), and then returns from the interrupt. Suggest why this top half/bottom half arrangement is used. What complications might arise? **(6)**

2. **a.** The following C code fragment is executed on a UNIX system. How many processes result? Explain your reasoning.

```
int main()
{
fork();
fork();
fork();
return 0;
}
```

(5)

The `clone()` function has largely replaced `fork()`. Explain the advantages of `clone()` over `fork()`. **(5)**

b. In the context of process synchronisation, what is a *critical section*? For what purpose would a critical section be used? For what purpose would a critical section be inappropriate? **(5)**

c. Describe the *completely fair scheduling* (CFS) algorithm used in Linux. What are the major advantages of CFS? How does CFS behave as the number of processes tends to infinity? How is this behaviour dealt with in practice? **(5)**

3. **a.** What is the principal difference between statically-linked and dynamically-linked programs? Compare the advantages and disadvantages of each approach. **(5)**

b. In the software creation process, a linker typically performs two phases: symbol resolution and relocation. Briefly explain what symbol resolution and relocation achieve. **(5)**

c. In the context of a graphical user interface, what is a message loop? Explain the operation of a message loop and how it facilitates user interaction. **(5)**

d. In most operating systems, the kernel's code and data are mapped into the upper half of a user process's address space. Explain why? **(5)**

4. **a.** Explain why many operating systems use a multi-level page table to implement paged memory. Sketch a typical multi-level page table arrangement taking care to show how the logical address of the CPU is translated into a physical address. **(5)**

What is the main disadvantage of a multi-level page table? How can this disadvantage be reduced? **(3)**

b. In systems which do not natively support threads, it is necessary to implement multi-threading in user space. In such user-space thread libraries, it is common to use a special system call to determine if another system call, for example, a keyboard read operation, will block if called. (In this example, if the keyboard buffer is empty, a read operation would block waiting for the user to press a key on the keyboard.) If the system call would block, the thread executes a yield to return to the thread scheduler. What is the principal advantage of this arrangement of allowing a pre-call to determine if a system call would block? **(3)**

For user-space threading libraries, why is it necessary for all the threads in a process to periodically execute a `thread_yield()` (or equivalent) instruction? **(2)**

c. What is the main advantage of converting a single thread process to a multi-threaded one? What particular issue needs to be considered in converting a program to multi-threading? **(3)**

d. Describe how sequentially-accessed files are indexed using a file allocation table (FAT). What is the principal advantage of this method of file organisation, and what is the principal disadvantage? **(4)**

PIR