SOLUTION NOTES

Operating Systems 2002 Paper 1 Question 12 (TLH)

This question covers the entire course, with the four sections needing specific reference to the syllabus items ’Processes and scheduling’, ’I/O subsystem’, ’Memory management’ and ’File management’. Details from the UNIX and Windows 2000 case studies are not mandated, but are of course reasonable to give as illustration.

i. Processing by the CPU

- Protection is required to enable preemption of the CPU.

- Before scheduling a process the OS sets a programmable timer that will expire, generating an interrupt, when the scheduler wishes to regain control -- e.g. after 100ms if that duration is being used as a fixed scheduling quantum.

- The interrupt handler will re-enter the scheduler, allowing it to make a fresh decision over which process to execute.

- This relies on the protection provided by a dual-mode OS to prevent (i) the application cancelling the programmable timer or (ii) changing the interrupt vector so that it is ignored.

- The interface provided by the CPU is the straightforward execution of a stream of instructions, including both user-mode and supervisor-mode operations.

- The abstraction provided by the scheduler is of a per-process ‘virtual CPU’ supporting the user-mode operations and the system calls provided by the OS.

ii. Access to a device, such as a serial or parallel data port

- Without protection multiple processes could try to interact with the device at the same time -- perhaps when trying to send data concurrently, or one process reading data that was not intended for it.

- Protection can be enforced because I/O operations are typically only permitted in supervisor mode or -- if the device is accessed through a memory-mapped interface -- the memory locations through which it is accessed can made inaccessible to user-mode.

- The interface exported by the device will usually be through read and write operations to a number of I/O locations -- for example, on a serial device, there may be one location to control settings (such as the conventions to use for handling the data) and another location to hold a single byte for transfer over the interface. (In contrast, the interface to a network device will usually use DMA for transferring data -- in that case the device will be accessed explicitly to indicate the physical memory ranges to transfer.)

- For both devices the interface accessed by an application will be a character stream, typically with concurrency control enforced by the OS so that only one process may be using the device at once.

iii. Storage of data and code in memory

- Protection is required to prevent one process from accessing (and, specifically, often updating) memory that has been allocated to some other process or to the operating system

- This protection is provided under the control of the processor's Memory Management Unit (MMU)

- Dual-mode operation prevents an application from reconfiguring the MMU -- e.g. it may require privileged operations that can only be performed in supervisor mode.

- The physical memory provides storage in a (usually contiguous) stretch of pages.

- The abstraction provided to an application is typically of a much larger virtual address space, some of which will be associated with physical memory. Traditionally this would be three contiguous regions holding process code, data and the stack. Virtual addresses may be specified as an offset within an identified segment, or merely as a 'flat' address within a 32-bit or 64-bit wide address space.

iv. Storage of files

- As opposed to the previous examples, protection here is usually performed at a per-user level (or something more flexible) rather a per-process level.

- Protection is enforced by interposing checking within the OS on every filesystem access. User-level processes must be prevented from making direct access to the disk holding the filesystem --

otherwise they could bypass the checks by reading disk blocks directly. As before dual-mode operation is needed to enforce this.

- The interface exported by the hardware will typically be a block-store holding fixed size chunks of data -- perhaps 512 bytes. The disk controller will allow blocks to be transferred to/from memory, typically using DMA.

- The usual kind of interface provided to an application will provide directory services for organizing files, access control with per-file and per-directory settings and arbitrary read/write operations within variable-length files.