Dept. Computer Science	**Operating Systems**
Vrije Universiteit	**31.08.2005**

1a A process is often informally defined as "a program in execution." What does "in execution" actually mean? *5pt*

This means that the operating system maintains an administration and facilities to support the execution of the program by the CPU, but also to temporarily suspend the program on behalf of other processes. This administration consists of data structures for managing registers, memory, files, etc.

1b When a process does a system call, it switches context to the kernel. What does this mean? *5pt*

The key issue here is that the kernel is shared by different processes so that most of its data needs to be protected against malicious use. To this end, a part of the OS is treated as a special program that cannot be directly accessed by processes. However, to carry out services offered by the OS, system calls establish a protected and controlled means to execute kernel routines. Executing these routines is referred to as running in kernel context.

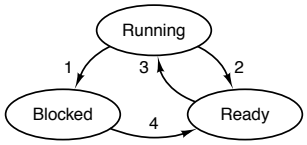
1c In MINIX, all processes communicate through synchronous message passing. What does this mean? *5pt*

It means that when a process A needs to communicate with process B, A will send a message to B and wait until B accepts that message. As long as B does not accept the message, A will be blocked by the kernel in order to hand over the CPU to another process.

1d If all communication is by message passing, does this also mean that MINIX can be easily implemented on a network of computers? Motivate your answer. *5pt*

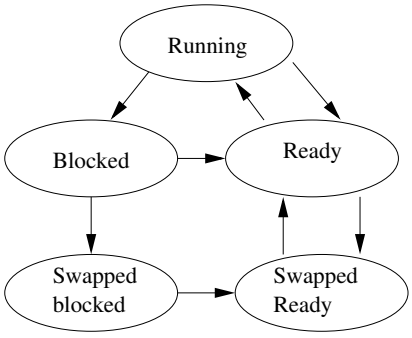
Yes and no. The problem is always porting the kernel data structures that are shared by all processes to a network. In addition, the actual implementation of message passing needs to be considered. In MINIX, for example, messages are copied between address spaces. Something similar will need to be done for a network. Fortunately, MINIX is already split into separate units that operate as independent processes. This alone makes it much easier to turn the OS into a full-fledged network operating system.

2a Consider the following state-transition diagram. Explain the cause for each transition. *5pt*



1: Process blocks for input; 2: Scheduler picks another process; 3: Scheduler picks this process; 4: Input becomes available.

2b Expand the state-transition diagram so that it also captures the situation that a process can be temporarily kept on secondary storage (i.e., swapped out). *5pt*



2c In order to swap processes in and out, a separate scheduler can be used. Assume this scheduler is implemented as a process. Does the CPU scheduler need to be aware of this process? Explain your answer. *5pt*

No. The process can be scheduled as any other process, at which time it will just do what it needs to do. The only thing that we need to take care of is that the process itself is not swapped out.

3a Describe the operations down and up of a semaphore. *5pt*

A semaphore has an associated integer value. The operation down decrements its value each time it is called until the value reaches 0, at which point the calling process is blocked. The up operation increments its value unless there is a blocking process, at which point that process is unblocked and allowed to proceed.

3b Using semaphores, give a solution to the producer-consumer problem. *10pt*

```
process producer() {
  int item;
  while (1) {
    produce_item(item);
    down(empty);
    down(mutex);
    enter_item(item);
    up(mutex);
    up(full);
  }
}
```

```
process consumer() {
  int item;
  while (1) {
    down(full);
    down(mutex);
    remove_item(item);
    up(mutex);
    up(empty);
    consume_item(item);
  }
}
```

4a Explain the difference between memory-mapped I/O and I/O-mapped I/O. *5pt*

With memory-mapped I/O, the registers of a device controller (which are needed to communicate with the device and to control it) are mapped into addressable memory. In this way, the CPU can simply access the device by reading/writing values from/to specific memory locatons. With I/O-mapped I/O, all I/O takes place through special I/O instructions.

4b After a device generates an interrupt, a series of actions are taken, in part by the hardware, in part by the operating system. Describe those actions up to the point that the interrupt has been completely handled. *5pt*

(1) The interrupt controller interrupts the CPU and passes a device ID (or interrupt vector) to the CPU. (2) The ID is used as an index for the interrupt table, where the hardware locates the address of the OS interrupt handler after pushing essential register values onto the current stack (e.g., program counter and program status word). (3) The address is loaded into the program counter. (4) The interrupt handler pushes register values related to the interrupted process onto the current stack, and starts handling the interrupt.

4c Handling I/O is generally organized in a number of layers. Describe these layers. *5pt*

(1) Device controller (hardware), which is responsible for low-level accesses to a device, along with generating interrupts. (2) Interrupt handler. (3) Device driver, which is a specialized piece of software for a single class of devices. (4) Device-independent software, consisting of caches, naming routines, etc. (5) User-oriented I/O, generally consisting of programming libraries that simplify access to I/O devices from a programmer's perspective.

5a What are the four necessary conditions for deadlock to occur? *5pt*

(1) A resource is allocated to at most one process at a time. (2) Processes can request a resource while one has already been allocated. (3) A resource cannot be taken away from the process to which it has been allocated. (4) Two or more processes must be waiting for a resource that is held by another process (i.e., a circulair wait should be possible).

5b Consider the following allocation of resources R1, R2, R3, and R4. Show that this is a safe state. *5pt*

Process	R1	R2	R3	R4
A	4	1	0	1
B	0	2	0	0
C	1	0	1	0
D	1	0	0	1
E	0	0	1	0

Allocated resources

Process	R1	R2	R3	R4
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	1
E	2	1	1	0

Resources still needed

R1	R2	R3	R4
6	3	2	2

Originally available

There was a mistake in this question: the "originally available" matrix should have been (7 4 2 2). *The state is safe, as we can complete the allocation in the order A - C - D - B - E.*

6a What is the purpose of the block cache in MINIX? *5pt*

The block is designed to hold copies of those disk blocks that are expected to be accessed soon by the current pool of processes. In this way, the need for actual data transfer between main memory and disks can be minimized.

6b When a buffer is returned to the block cache, it can either be prepended or appended to the free list. Explain under which conditions which choice is made. Motivate your answer. *5pt*

When a block is expected to be used again soon, it is appended to the list, otherwise it is prepended. The reason is simple. If another process needs a buffer (because the block it was looking for was contained in the cache), it will search for a free buffer by starting at the head of the free list. Efficiency of the block cache degrades if the head of the list contains blocks that are expected to be accessed soon again, as these will then need to be read from disk upon their next access.

6c Assume MINIX wants to read block *B*. Outline the search for a buffer containing block *B* in the block cache. *5pt*

The OS will first lookup B through the block cache's hash table. If it doesn't find the block, this means that it's not contained in the cache, for which reason it will have to evict another block. It does so by inspecting the free list, starting at the head. If it finds a buffer, it checks whether the contained block had been modified. If so, it flushes the buffer to disk, and then assigns it to read in the requested block. If you also mention that all dirty blocks for the same device are written to disk as well, you'll be almost sure of the full 5 points.

Grading: *The final grade is calculated by accumulating the scores per question (maximum: 90 points), and adding 10 bonus points. The maximum total is therefore 100 points.*