Operating Systems: Assignment 2

Due on 7 March, 2019 at 8:30am

Prof. Fernando Pedone

Alessandro Romanelli

Question 1

System calls are used when a user program attempts to run a privileged system subroutine, because some of these routines are hazardous, only the kernel has the right to invoke them: a system call is thus a request to the OS to run a system routine on the user program's behalf.

Question 2

We can identify five classes of services that each OS provides in its on way:

1. File-system manipulation: allows the creation, allocation and deletion of files in the main memory. A user-level program can not be trusted to perform such actions as this could be exploited to manipulate files that a user shouldn't have access to.
2. I/O operations: reading and writing on I/O devices on a low level. I/O are guarded and operations on them are typically restricted; a user shouldn't be able to control it directly but the OS should handle that on the program's behalf.
3. Program execution: the system loads a program into memory and runs it. A user-program should not be allowed to allocate CPU run time.
4. Communication: the OS allows processes to exchange data by composing and broadcasting packets, which are processed and sent across the network and later reassembled by the destination operating system. A user-level application can not coordinate access to the network device and it might receive packets which were meant for other processes.
5. Error detection: the OS analyses the received data to check for possible corruptions (bit flips or data loss) to make sure that the received data has not been corrupted. A user program is not able to do this because it would need kernel privileges to handle independently all the possible errors (including halting the system).

Question 3

There are three possible ways of passing the necessary parameters to the OS:

1. They can be passed by using the registers;
2. They can be pushed on the stack by the program and later popped off by the OS;
3. They can be stored in memory and pass the OS a pointer to that specific memory address via register. (Especially when there are more parameters to be passed than registers available)

Question 4

There are two main ways of interprocess communication and they are the **message-passing model** and the **shared-memory model**.

Message-passing model: two processes exchange messages either directly or indirectly through a common mailbox. A lot of setup is required in order to open the connection between the two processes since they may run on two different machines. The implementation of such a system assumes a vast amount of information

Question 4 continued on next page. . .

to be known, but ideally it boils down to have a unique identifier to allow the two process to know where to send their messages to. This process requires considerable time and resources, thus resulting in a slower communication between the processes. It is however easier to implement and there are no conflicts to be avoided for smaller. bits of data.

Shared-memory model: two processes can exchange messages by accessing a memory region which is going to be accessible by both of them. Normally the OS would try to prevent this, but if both processes agree about the share of the memory region, then the OS will allow them to do proceed. Once both processes have access to the shared region, they will be able to exchange messages by writing and reading from it, although there will be no guarantee that they don't overwrite each other's messages. This model is the best in terms of maximum speed and convenience of communication: when two processes run on the same system, the speed of the communication is determined solely by the time it takes to read and write to memory. The biggest problem of this model as I've said above is that it requires a greater deal of synchronisation between the processes to avoid RAW and WAR conflicts (Read After Write, Write After Read).

Question 5

The *microkernel approach* means to strip the kernel of all the superfluous components, with minimal process and memory management. This makes extending the operating system much easier than with a regular kernel. The removed services have to be implemented at a user program level, therefore the OS can be ported easily from one system to another one having a completely different hardware.

User programs and system services interact indirectly by using the **message-passing model** described in un **Question 4**, by exchanging messages through the microkernel.

There are two main disadvantages to using the microkernel approach: in the first place that system calls can require quite a lot of changes and secondly that the performance may decrease due to the heavy use of the interprocess communication system.