Operating Systems 2

**Functions of an Operating System**

As mentioned in a previous workbook, there are a number of functions that all operating system, regardless of their ‘type’, will perform.

The following is a formalised list of the main functions of an operating system:

* Interface
* Memory Management (paging, segmentation and virtual memory)
* Peripheral Management (Device Drivers)
* Multi-Tasking (interrupts, the interrupt service routine and scheduling)

We will now look at each of these functions in more detail.

**Memory Management**

One of the jobs of the OS is to manage the system’s memory. As each program is opened by the user, it is loaded onto the RAM and an important role of the OS is to ensure that no two programs can access each other’s data as this could allow for malicious data attacks.

Another role is to ensure that the free space on RAM is being used efficiently.

For example, if 3 programs of different sizes were loaded onto the RAM…

…and 1 closed…

…and another opened (4)…

…it’s the OS’s job to ensure that the program can be loaded onto the RAM, even if there are no contiguous spaces available.

***Paging and Segmentation***

There are two different methods that operating systems use to split programs up so that the RAM can be utilised efficiently.

***Paging***

Paging is where the operating system will split a program up into small, equally sized blocks of data (called pages, a few kb in size), which can then be assigned to non-contiguous memory locations in the RAM. This system requires a paging table, which is a record of where the different ‘pages’ (blocks of data) have been stored, so that the necessary data can be accessed when needed.

***Segmentation***

Segmentation is similar to the above but instead of a program being split into equally sized data blocks, the program is split logically, i.e. into modules or functions.

**Virtual Memory**



Sometimes, no matter how well the OS has used paging or segmentation to manage the free space in memory, the RAM can get used up.

Another role of the operating system is to manage this situation. It does this by making use of something called virtual memory. Virtual memory is where secondary storage (usually the hard disk) is used as RAM. Programs that haven’t been used (but are still open) are moved to virtual memory, freeing up space in the RAM, so that new programs can be opened.

The issue with virtual memory is that secondary storage devices are far slower at communicating with the CPU, so the user sees a notable dip in performance, when using programs in virtual memory.

**Interrupts / Interrupt Service Routine**

The CPU is an excellent multi-tasker. It can carryout the processes of several programs in quick succession resulting in the appearance of processing simultaneously. However, there are times when the CPU will have to momentarily stop processing instructions for running programs and deal with new requests (for example, an input signal from the user).

These requests are known as interrupts – the CPU is interrupted and needs to deal with a request before continuing with its processing of running programs.

When an interrupt is received, the CPU will carry out an Interrupt Service Routine (ISR). The ISR will evaluate how important the interrupt is and will either add it to a process queue (low importance interrupt) to be dealt with later or be handed to the CPU for immediate processing (interrupt with high importance).

**Multi-Tasking**

When multiple programs are running on a system, it is the job of an operating system to ensure that all programs are able to run simultaneously.

The operating system does this by managing the processes that are given to the CPU. Operating systems attempt to ‘fairly’ share out the jobs required by several programs, using something called a ‘scheduler’.

A scheduler uses an algorithm to work out how to share out the processor amongst the different programs.

There are several different scheduling algorithms:

* Round Robin
* First Come, First Serve
* Multi-Level Feedback Queue
* Shortest Job First
* Shortest Remaining Time

***Scheduling Algorithms***

Round Robin

A ‘Round Robin’ scheduling algorithm is very simple in the way it works. It gives each process an equal, set amount of time to complete. If it fails to complete, it goes to the back of the queue making way for the next process.

First Come, First Serve

This scheduling algorithm does as the name suggests. The first process to request the services of the CPU is served, then the next process and the next etc. All processes are queued up in the order they arrive and dealt with according to their position.

Shortest Job First

Unsurprisingly, this algorithm will prioritise the jobs which can be executed the quickest, but clearly knowledge of the execution time of queued jobs needs to be calculated. Imagine you are wanting to carry out some maintenance jobs on your computer. You might want to move files into folders, delete unwanted documents, uninstall unused programs before running a disk defragmentation job which may take a long time to complete.

Shortest Remaining Time

Out of all the jobs queued up at anyone time, this algorithm will prioritise those which have the shortest remaining time to process. Consider a chef cooking several meals at once, he/she may chose to serve those that have the shortest time left to cook, regardless of the time the orders came in.

Multi-Level Feedback Queue

This algorithm makes use of a range of queues, with each queue given a different priority. For example there may be a queue for an input/output requests, one for the shortest jobs but whatever the queue categories, the algorithm can schedule jobs between the queues to ensure the CPU works as efficiently as it can, maximising the quality of the experience for the user.

**Peripheral Management**

An important job of an operating system is to management peripheral devices. In particular, its role is to provide an interface between software and hardware as one cannot talk to the other.

For example, in the case of some audio playing software, the operating system will receive a request from the program to pass on data to the speaker. The operating system will the communicate with the speaker in order to pass this data on, so that it will output the desired sound.

***Buffers and Spooling***

Because all other devices are so much slower than the CPU, the operating system needs to manage the requests of peripheral devices to use the CPU, as efficiently as possible.

For example, if a request to print several documents was made by a user, it would be inefficient for the CPU to be actively waiting for each printing job to complete before dealing with the next printing job.

To overcome this, the operating system makes use of a small piece of secondary storage and creates with it, something known as a buffer. This buffer enables spooling to occur.

Spooling comes from the term ‘Simultaneous Peripheral Operations Online’. Spooling is where peripheral device requests are added to a queue / buffer, with each request then being seen to in turn – this allows the CPU to be freed up to deal with millions of other requests, between the slower peripheral device jobs.

***Device Drivers***

Clearly it is impractical for programmers of an operating system to be able to program an OS to be able to communicate with every possible peripheral device as there are so many different types and makes.

To get around this, each peripheral device will have a small piece of software known as a device driver. This file has the information that an operating system needs to be able to communicate with the device.

**The BIOS**

So far we have discussed the functions of operating systems. The BIOS is not a function of an OS, nor is it an OS. But without one, the computer will be unable to run the OS.

BIOS stands for Basic/Input Output System.

When you first switch on your computer, the BIOS will run and its job is to first check that all the components required for a computer to function, are all present and working. This is known as the power-on self-test (POST). After running the POST, the BIOS will load the operating system’s kernel from the boot loader, allowing the operating system to run and load on the computer.

**Virtual Machines**

A virtual machine is software. It is a program which takes on the function of a physical machine. It can run an operating system for example. Virtual machines are becoming more and more common as it means that with just one computer, this software can create the illusion that there are several different computers, each running their own operating systems independently, but at massively reduced cost. The components of just one computer being utilised to produce the experience of several computer systems.

Virtual machines are also really advantageous when it comes to the portability of programs. Currently, we know that a program needs to be fully translated into machine code in order for the CPU to run the program. The alternative is to run the source code with an interpreter but this reveals the code and makes the program run much slower.

Virtual machines provide another option. They can execute something called intermediate code which is neither source code nor machine code. It is a far more efficiently packaged version of source code called Byte Code and because it cannot be read like source code, the programming code is secure. Usually, programs come fully compiled and it will be compiled to be run on only certain types of CPU, but because bytecode is only partially translated, before the code is run, the virtual machine can first check the computer’s hardware before deciding how to translate it, so that it can be executed by that computer’s hardware. Bytecode is therefore highly portable.

**Questions**

**‘General Understanding’ Questions**

1. What are the functions of an OS? [4]
2. Explain the difference between segmentation and paging and discuss their relative efficiencies when managing memory. [7]
3. Explain what an Interrupt is and the consequential Interrupt Service Routine. [5]
4. Describe the various ways a CPU may carry out multi-tasking. [10]
5. Explain what ‘device drivers’ are and why they are needed. [3]
6. Explain what a virtual machine is. [5]