CS 1MD3 - An Intro To Operating Systems

What is an operating system?

An operating system is a **layer of software** which takes care of technical aspects of a computer's operation. It shields the user of the machine from the low-level details of the machine's operation and provides frequently needed facilities. Beyond this definition, the ways in which the operating system works is arbitrary and there is no prescribed way for the OS to behave.

Despite this fact however, trends in development and research have produced many similarities across platforms. For this tutorial, we will be looking at the Windows OS. Windows is a good example as it is not as apparent as say Linux, the ways in which the OS behaves.

Before we can delve further into the Windows OS, we need to define some terms first.

THE PROCESS

A basic definition - A process is the **unit of work** in a system.

In More Detail

A process is the fundamental building block of the OS. Essentially all software on the computer is organized into a number of sequential processes. The operating system delegates and manages all the running processes on the computer. Quite frequently, you'll find many processes running that you didn't even know about.

- Press **CTRL-ALT-DEL** on your keyboard.
- Select Task Manager - this brings up the **Windows Task Manager**
- Select the **Processes** tab.

You'll notice that there is quite a large list of processes running even though you've barely done anything. Here you can find information about each process like what % of the CPU power is it using, memory usage etc If you're

reading this document in a web browser you'll probably see IEXPLORE.EXE in the process list. You'll even see TASKMGR.EXE representing the window you're looking at right now. Every program in the OS can be broken down into its processes.

Take special notice of the column **PID**. This stands for the **Process ID**. As programmers, you can ask the operating system to provide you with a new process and reference it by its ID.

There are many more pieces of information about a process which is available. Use the View -¿ Select Columns menu to explore these. This allows one to see all about the memory usage, disk use, resource use and so on associated with a process.

THE SCHEDULER

A computer is not just a box which adds numbers together. It has resources like the keyboard and the screen, the disk drives and the memory. In a multi-tasking system there may be several programs which need to receive input or write output simultaneously and thus the operating system may have to share these resources between several running programs.

Most multi-tasking systems have only a single central processor unit and yet this is the most precious resource a computer has. A multi-tasking operating system must therefore share cpu-time between programs. That is, it must work for a time on one program, then work a while on the next program, and so on. If the first program was left unfinished, it must then return to work more on that, in a systematic way. The way an OS decides to share its time between different tasks is called scheduling.

The scheduler is responsible one (or combinations of) the following:

Queueing. This is appropriate for serial or batch jobs like print spooling and requests from a server. There are two main ways of giving priority to the jobs in a queue. One is a first-come first-served (FCFS) basis, also referred to as first-in first-out (FIFO); the other is to process the shortest job first (SJF).

OR

Round-robin. This is the time-sharing approach in which several tasks can coexist. The scheduler gives a short time-slice to each job, before moving on to the next job, polling each task round and round. This way, all the tasks advance, little by little, on a controlled basis.

CACHING

Caching is a technique used to **speed up** communication with slow devices. Usually the CPU can read data much faster from memory than it can from a disk or network connection, so it would like to keep an up-to-date copy of frequently used information in memory. The **memory area** used to do this is called a **cache**. You can think of the whole of the primary memory as being a cache for the secondary memory (disk).

- Press CTRL-ALT-DEL on your keyboard.
- Select Task Manager - this brings up the Windows Task Manager
- Select the **Performance** tab.

Find the number of processes you're running as well as the System Cache. What is the Cache measured in?

VIRTUAL MEMORY

Virtual memory is the use of space on a **hard disk** to simulate additional main memory (also referred to as primary memory or just memory). In order to free up space in memory, the operating system transfers data which is not immediately needed from memory to the hard disk, and when that data is needed again, it copied back into memory. This "swap space" is divided into segments called **pages**.

Return to the **Performance** tab again.

How much of the memory is paged? How much is not paged?

FAT

Not in "stop eating cake" sense. This stands for **File Allocation Table** which is an area on a hard disk or floppy disk where information is stored about the physical location of each piece of every file on the disk and about the location of unusable areas of the disk.

THE SYSTEM CALL

Essentially it is an instruction that **interrupts the program being executed and passes control to the supervisor**. For the sake of this course, you can think of the supervisor as the operating system performing a task that you're not allowed to do yourself. Since the operating system manages the many resources of your computer, it need to manage who gets what, when. For example, if your video game say Half Life 2 requests a screen update from the OS and at the same time MSN wants to pop up a message, both will need to perform a system call and let the OS continue from there.

THREADS

A part of a program that can execute independently of other parts. Operating systems that support multithreading enable programmers to design programs whose threaded parts can **execute concurrently**. All threads can be grouped according to their parent process. You can think of a thread as a single line of execution running under your process.

With your experience in working with binary trees, what would happen if two threads tried to perform an insert at the same node in your tree at the same time? This brings up the issues of **Critical Sections** of code the require atomicity.

This topic deserves a far more in depth discussion.

Controlling the OS

There are mnay available configuration items, some well known, some well hidden for the OS. Of course, there is the control panel, the simplest way to access information. Here is a non-standard way to bring it up:

- Bring up a DOS window (Start – All Programs – Accessories – Command Prompt).
- Type: chdir
  windows
  system32
- Type: control

There are other interesting views. From the control panel, bring up the 'System Properties' dialog, Hardware tab. Press on the 'Device Manager' button to see a lot of information about the underlying hardware of your machine.

Also from the Control panel, you can go to 'Administrative Tools', and explore the sub-tools 'Computer Management' and 'Services' to get an in-depth view of what is going on inside the OS.

Another entirely different view can be had by looking at the (deep) settings maintained by the computer. The easiest way to get to this is via Start Menu -¿ Run, and type in "regedit".

Finally, if you want to see information about your network configuration, you can open a DOS window and type in "ipconfig /all".

SELF TEST QUESTIONS

1. Go to **www.google.ca** or your favourite search engine and lookup:
   preemptive
   non preemptive
   As they relate the operating system.

   How do they relate to scheduling?

2. Make an estimate of how much Cache memory you think would be needed for a laptop. Go to **www.dell.ca** and try and find the amount of cache available on one of their Inspiron Notebooks. Was the amount more or less than you guessed?

3. Look up Defrag (as in a hard disk) and with you knowledge of what it does, how would it affect the FAT?

4. What is supervisory mode? What is user mode?

5. What is a Kernel Thread vs a User Thread? Why do we need say, a mutex to keep threads from 'running over one another'?

Bonus Question

Why is it that windows consistently needs to be restarted after system changes (or application installs or sneezing for that matter), but Linux does not? Hint: Look up the computer's kernel on the net and how it is loaded into memory.

Suggested Related Topics for Self Study:

Spooling, Interrupts, Traps, Exceptions, Distributed Systems, Dispatcher, Pipelining, Stack vs Register architecture, and especially Threads.