Operating Systems and Concurrency

Lecture 1 introduction

Module goals:

- Introduce the fundamental concepts and principles of an Operating System and Concurrency.
- Help to understand how programs rely on the OS.
- Understand the basic writing of concurrent(also known as parallel) code.

What you should know by now(Labs):

- Fundamentals of OS concepts
- How to use operating system APIs and how OS schedulers work(HUGE CW we had)
- Concurrency

The exam

- It will be 120 minutes focused on:
  - Knowledge
  - Understanding
  - Application
- The exam will be 3 of 4 questions accounting for 75% of the module
- Sample questions and exams are on moodle

Past exams

- Exam 2009-2010
- Exam 2010-2011
- Exam 2011-2012
- Exam 2012-2013

Module content

This table is confusing but he gave it out like that

Subject	Lectures	Lecturer
Introduction to OS	1-2	GDM
Processes, Thread	3-4	GDM
Concurrency and Deadlocks	5-6	GDM
Memory management,Swapping and virtual memory	4-5	IT
File Systems	3-4	IT

Subject	Lectures	Lecturer
Visualization	1	IT
Revision	2	IT

Definitions

You should know these by now but let me refresh your memory

- File Systems: Physical location of a file(on disk)
- Abstraction: Covering up complex process with nice high level functions and methods
- Concurrency: Allowing programs to run at the same time(without them causing conflicts)
- Security: I can't be bothered to explain this one come on guys

When an OS comes in handy?

An OS is basically a massive layer of abstraction between code and the actual Hardware and this comes in handy on a series of occasions because as programmers we can forget about hardware limitations(sometimes) and focus on algorithms.

Examples of OS abstraction:

- OSs will find a way of dealing with data when memory(RAM) is full, of course this won't be super fast and efficient but it will make the program work(this is called Swap memory on some operating systems)
- OSs will optimize the usage of memory when only certain data in an array is needed instead of all of it
- OSs will handle processes so a computer can "Multi task" making you resource intensive program bearable to use while doing other tasks.

For these and many more reasons you love your OS because it allows you to be lazy and not have to worry about the hardware most of the time you write code.

Quote this next line on the exam for some ez marks:

> *"All problems in computer science can be solved by another level of indirection"* Dr David Wheeler PhD in computer science 1951

Concurrency

One of the most important features of modern Operating Systems is their implementations of Concurrency. These allow programmers to implements code that will run asynchronously in multiple CPU cores theoretically speeding up

your program to up to twice its normal speed. Modern CPUs are what we call multi-core or multi-threaded this means that one single CPU can do multiple operations at the same time, as long of course they don't rely on the result of another core computation.

Kernel mode

To achieve concurrency and what is called Multi-programming(concurrent code) most modern OSs work with multiple modes notably Kernel mode and User mode. The OS will transition between both modes in a controlled manner usually having a mode bit which will distinguish between both operation modes.

- Kernel mode is a more complete version of the lower level OS allowing all instructions available for the CPU
- User mode is one level of abstraction above Kernel mode and allows only a subset of instructions which can and will access Kernel mode when necessary

One way of visualizing this is User mode occurs on applications and programs where kernel mode occurs on the operating system level.

Lecture conclusions

The OS sits on top of hardware and has full access to its features while providing abstraction for the User/Programmer. It also controls the user by switiching between different modes with different levels of access to the hardware.

Different OSs have different purposes and implementations but most of the time the focus on the following: Memory management, CPU scheduling, multi-programming, file system, communication, memory management, interrupt handling, GUI, Browser