Harry Langford
hjel2@cam.ac.uk

Scheduling algorithms allow tasks to yield – in every scheduling algorithm, be it pre-emptive or non-pre-emptive then tasks will yield when waiting for IO etc.

Whenever a process transitions to the ready-state, the scheduler runs.

The features the operating system provides can be split into three main subcategories:

- Hardware abstraction
- Multiplexing resources
- Protection

When talking about Operating Systems, anything we talk about will be one of these three topics.

In summary the OS provides Basic features the *user* needs to use the computer. Note this is the *user* – not the Computer. The computer is the physical hardware. What the Computer needs i s electricity etc. What the user needs is the basic functions.

The operating system also provides device drivers.

The Operating system provides an abstraction on top of the hardware so that users don't need to know how to interact with specific devices – only a specific abstraction of the system.

OS provides mechanisms such that users can't access each others data if they're not supposed to and also they can't access the operating systems memory.

One of the core abstractions of OS is a process.

A process is simply a program in execution – note this is different to a thread.

The program code and resources the process needs during execution are part of the process. To manage processes and the resources they've been allocated, the OS uses process control blocks.

The PCB contains: the programs state and its pointers to the page table, user which started it, scheduling information, the priority of the process. IE a list of open file handles. You also need to store the USER ID of the user that started the process in the PCB. So that when you open the file you can check this against the access control of the file. Some other things too: ie process state, parent ID of a process.

One process can have several threads. The thread is the resource stuff. The point of execution is attached to the threat. The others will be reading or might be blocked or at different points in the program. They may have different values for their CPU registers.

For kernel level threads the OS scheduler sees the threads within a process and ten can make scheduling decision and pick one thread across any process to schedule next. User-level threads are only visible in the process, so the OS only sees one thread. The process has it's own scheduler which does scheduling to run user threads while the kernel runs kernel threads. In short the Kernel schedules which process runs and then the process schedules which part of it should run (which user thread).

The Thread Control Block contains the execution state. Note that processes are not threads and are instead above threads.

For the purposes of this course: unless explicitly threads are mentioned then a process is a unit of execution as well. The rest of the supervision talks about process scheduling since it's a sipmler abstraction than threads.

Program counter is the point of execution of a process.

"Describe two methods by which the contents of a process address space are preserved and restored":

Harry Langford
hjel2@cam.ac.uk

Methods of memory management are paging and segmentation and base-limit registers.

You just need to gind the shit out of Operaitng Systems to get good, to get very, very, very good. Right now you're looking at a 2.2 or some total shit.

You do not need to run the scheduler when interrupted.

A system call which can be serviced imediately is when reading from the hard disk if the stuff you're trying to find is in a cache so that you can be serviced immediately.

In non-preemptive scheduling, the execution is in the control of the process. In preemptive scheduling execution is controlled by the OS.

For preemptive scheduling you need hardware support for interrupts – so the CPU can override the program counter.

non-preemptive do have yields.

preemptive scheduling with a small time slice makes IO bound scheduling faster.

Python event loop is a non-preemptive scheduler.