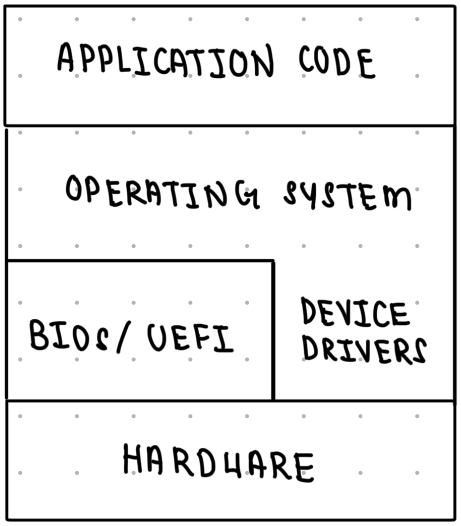
PASS CLASS 15/04/2022 (FIT 1047)



WHAT'S OPERATING SYSTEM?
↳ piece of software / collection routine
↳ that manages computer resource
↳ provide convenient interface between application

WHY ABSTRACTION USEFUL IN IT ?
↳ hide complexity
↳ provide clean, well-defined interface to functionality

OS core tasks provided by OS kernel:

- Managing multiple **processes** running in parallel. A process is a program that is currently being executed. – virtualising CPU
- Managing the **memory** that processes use. Virtualising memory
- Provide access **to file systems**, the **network** and other **I/O resources**. – kernel mode

What are the differences between a process and a program?

- Process is a running instance of a program, created by loading program code into the computer then executing it.
- Program is the code written, or result of compiling written code into machine code. Usually, a program is a file stored on a disk.

OS abstraction through virtualisation

OS provides illusion that each process

- has the CPU completely to itself
- has a large, contiguous memory just for itself
- has exclusive access to system resources

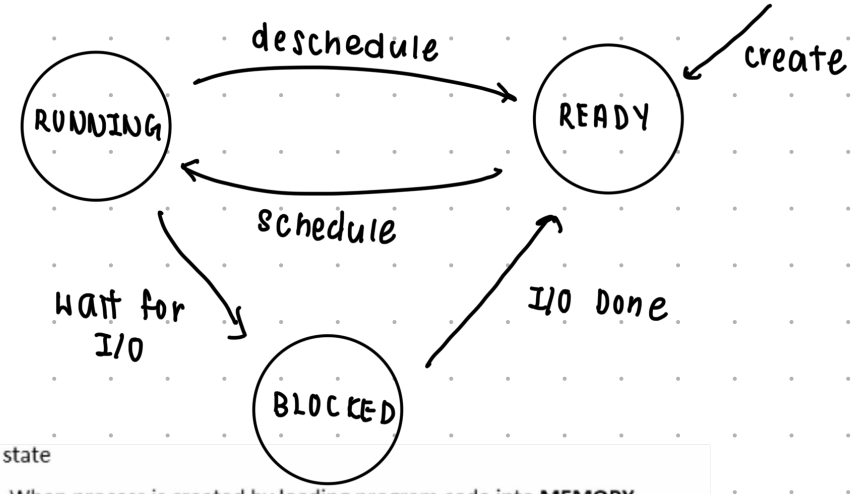
Virtualising the CPU

Goal?

- Run multiple processes at the same time, BUT in a way where we don't have to know that they are running in parallel (with other processes)
- OS create illusion that many processes are running in parallel BY **switching between processes quickly (multiple times per second)**

CHALLENGES :
1. process shouldn't notice that it's not running continuously
2. each process should get a fair share of CPU's overall time

Virtualising CPU = mechanism for OS to switch between processes & the policies used to decide when to switch between them ----- HOW to switch and WHEN to switch



- Ready state
  - When process is created by loading program code into **MEMORY**
  - Process is ready for execution, but not currently being executed yet
- Running state
  - OS decides process should be executed now – schedule it and put it into running state
  - The code is actually executed on the CPU
- Blocked state
  - Process request for IO
  - Since I/O may take some time – OS puts process into blocked state
  - As soon as I/o finishes, process is put back into ready state

User mode	Only a subset of all instructions are allowed to be executed.
Kernel mode	Code runs without any restriction, full access to I/O devices and memory.

Limited Direct Execution

- Performance
  - Virtualisation shouldn't be too taxing on the CPU
  - CPU time should be spent on actually running the process (not managing)
- Control
  - OS enable fair scheduling of processes
  - OS should be able to protect against buggy code.

System calls

- Special CPU instruction switching CPU from user mode into kernel mode.
- Process makes a system call with number n
  1. Save process context (registers) into memory
  2. Switch CPU to kernel mode
  3. Jump to handler n and execute it
  4. Restore process context
  5. Switch CPU to user mode
  6. Return to calling process

TIME-SHARING:

COORPERATIVE	vs	PRE-EMPTIVE
• X kill buggy / malicious program		• ✓ kill buggy / malicious program
• X timer interrupt		• ✓ timer interrupt

PROCESS SCHEDULING :

↳ Want :
- turnaround time to be as short as possible
- fairness (share CPU overall time equally)

↳ First-come-first-served
- poor turnaround time
- not fair

↳ Shortest job goes first
- good turnaround time
- not fair

↳ round-robin scheduling
- good turnaround time
- fair
- compromise performance
  ↳ spent more time managing than running the program
- time is spent switching between time slices
- split process into time slices

State the 3 main goals of virtualising memory

- To enable protection of a process's memory against access from other (malicious or buggy) processes.
- To make programming easier because a programmer does not need to know exactly how the memory on the target computer is organised (e.g. how much RAM is installed, and at which address the program will be loaded into RAM).
- To enable processes to use more memory than is physically installed as RAM in the computer, by using external storage (e.g. hard disks) as temporary memory.

QUESTIONS :

Question 22:

Briefly explain the concept of a *process*, including the states it can be in. (2%)

- A process is a program that is currently being executed. A process can be in the ready, running or blocked states.
- Running means that it is currently running on the CPU.
- Blocked means it is waiting for I/O to finish.
- Ready means it could be running, but the OS has not scheduled it yet.

Question 23:

Briefly explain how the hardware and operating system work together to control how programs access I/O devices. (3%)

- The hardware (the CPU) has a kernel mode and a user mode, and in user mode it restricts processes so that they cannot access I/O devices directly.
- The operating system provides a set of system calls, which are basically subroutines that implement I/O functionality and that user mode processes can call. That way, the OS can control what kind of I/O each process is allowed to perform.

Question 24:

Name and briefly explain the mechanism that operating systems use to provide each process with its own address space. (3%)

Virtual memory: when processes access a location in memory, the hardware maps that virtual location to a physical memory location, e.g. using a base register storing the physical address for the current process. The OS sets up the base register when it switches between processes.