COMP1001

Computer Systems

Dr. Vasilios Kelefouras

Email: v.kelefouras@plymouth.ac.uk

Website:
https://www.plymouth.ac.uk/staff/vasilios-kelefouras

School of Computing
(University of Plymouth)

Introduction

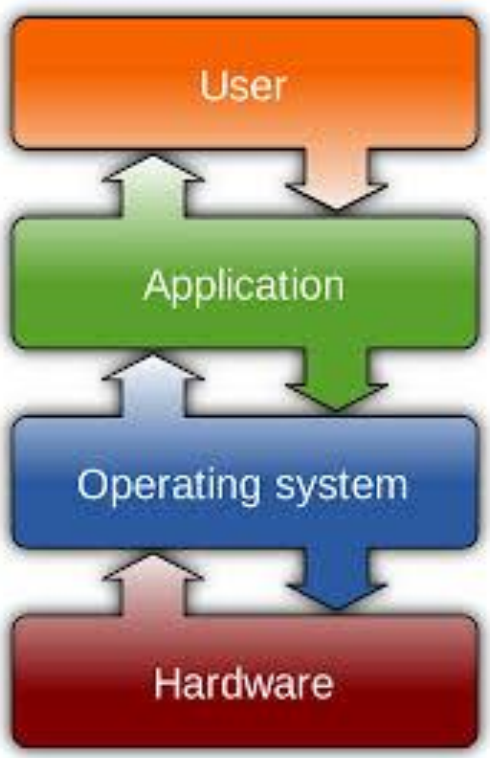
- Outline
  - What is an Operating System?
  - What is the role of the Operating System?
  - What are the parts of the Operating System?
  - Comparison of different Operating System kernels

What is an Operating System (OS) ?

- A program that acts as an intermediary between a user of a computer and the computer hardware – **It is a program that lets us run other programs**
- **Operating system goals:**
  - Execute user's programs
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient way
- OS is software that allows a user to run other applications on a computing device
  - While it is possible to write a software application without using an OS, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details
  - OS provides routines to read/write from disk, print to screen
  - OS manages I/O devices, such as keyboard, mouse, printer, screen, network devices etc

Computer System

- **Computer system can be divided into four components**
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

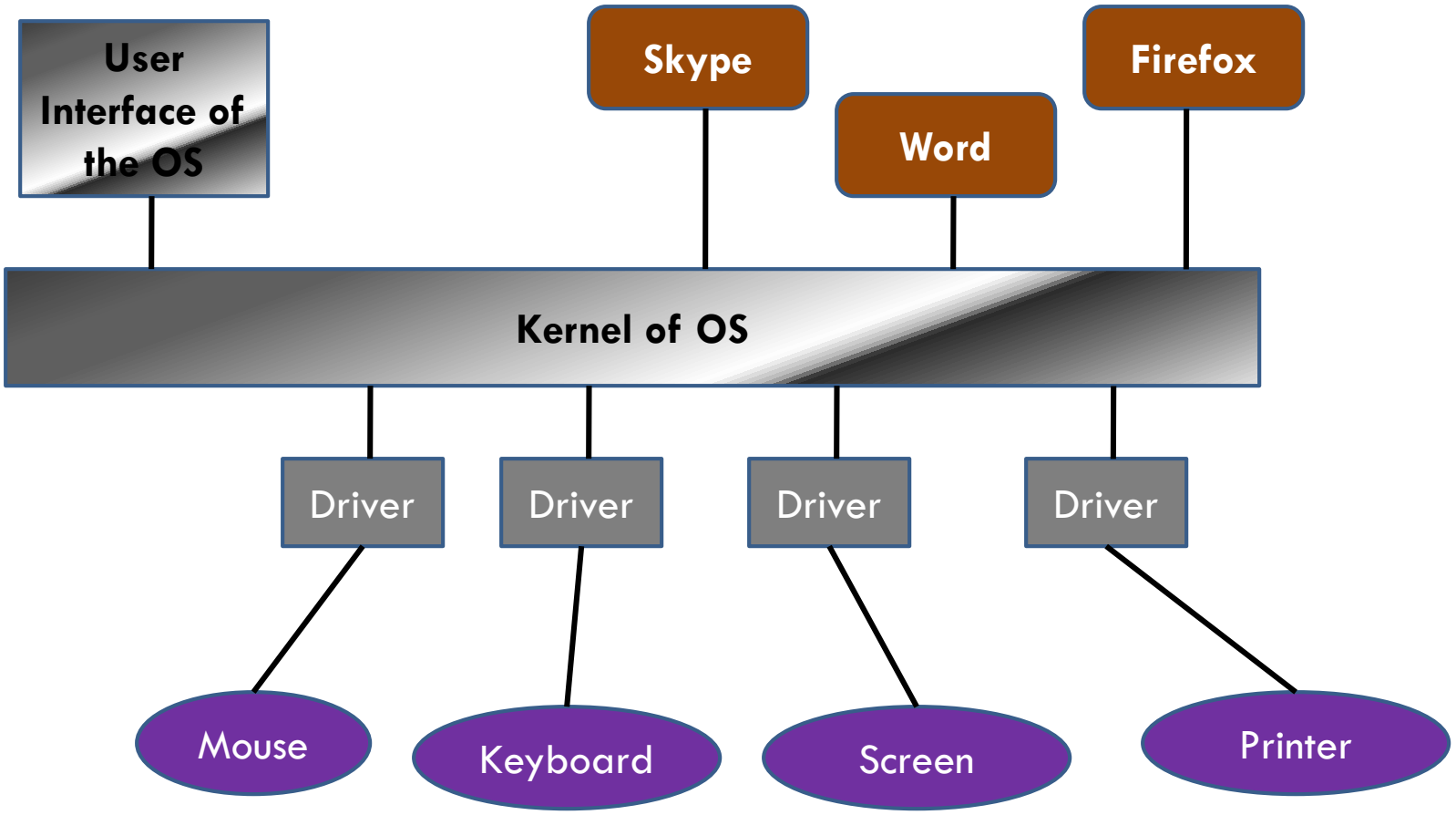


OS definition – there is no clear definition

- **OS is a resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- **OS is a control program**
  - Controls execution of programs to prevent errors and improper use of the computer

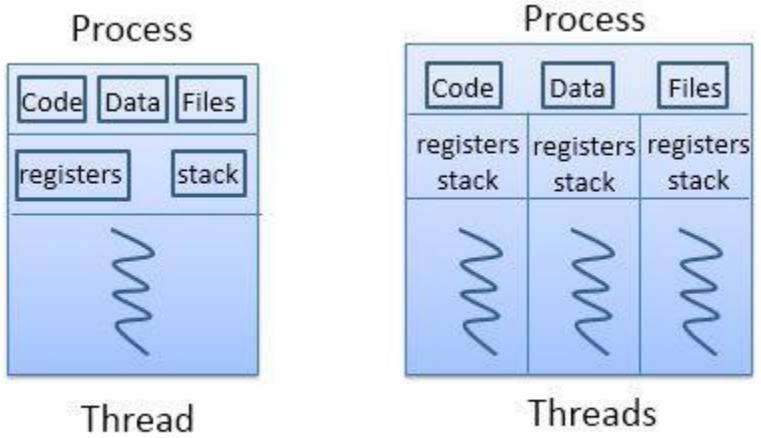
- **Main Objectives:**
  - Convenience (user interface)
  - Efficiency (manage hardware resources)
  - Hide the details of the hardware resources from users

The Operating System (grey color)



What is difference between thread, process and program?

- **Program:**
  - Program is an executable file containing the set of instructions written to perform a specific job on your computer
  - For example, *skype.exe* is an executable file containing the set of instructions which help us to run *skype*
- **Process:**
  - Process is an executing instance of a program
  - For example, when you double click on the *skype.exe* on your computer, a process is started that will run the skype program
- **Thread:**
  - Thread is the smallest executable unit of a process
  - For example, when you run skype program, OS creates a process and starts the execution of the main thread of that process
  - A process can have multiple threads
  - All threads of the same process share memory of that process



Multithreading

OS Classification

- **Multi-user OS:** Multiple users can run programs at the same time
- **Multiprocessing OS:** a program can run to more than one CPU cores
- **Multitasking OS:** run several tasks concurrently (firefox, excel, skype etc)
- **Multithreading OS:** run multiple threads of a process simultaneously
  - A thread is a basic execution unit which has its own program counter, set of the register but it shares the code, data, and file of the process to which it belongs
  - A process can have multiple threads simultaneously
- **Real time OS**: Responds to input instantly, e.g., in a car accident the airbag must open at the right time

What is the Role of the OS ? (1)

- **Resource allocation**
  - Manage the computer's resources and allow other programs to run and use these resources
  - These resources are the CPU, Memory and I/O devices

- **Process management:**
  - Create, load, execute, suspend, resume, and terminate processes, switch among multiple processes (scheduling), allocate/de-allocate resources properly etc
  - The OS must allocate resources to processes, enable processes to share and exchange information, protect the resources of each process from other processes and enable synchronization among processes

What is the Role of the OS ? (2)

- **Processor management**
  - Ensuring that each process and application receives enough of the processor's time to function properly
  - Using as many processor cycles as possible for real work

- **Memory management**
  - This makes sure that a program does not conflict with memory currently being used by another program
  - Map processes to memory locations
  - Allocate/deallocate memory space as requested/required

What is the Role of the OS ? (3)

- **I/O Device Management**
  - The kernel provides I/O to allow drivers to physically access their devices through some port or memory location
  - **The processes need access to the peripherals connected to the computer, which are controlled by the kernel through Device Drivers**
    - A device driver is a computer program that enables the operating system to interact with a hardware device
    - The device driver decides which process gets the device when and for how long, allocates/deallocates the device
- **File management**
  - File creation/deletion
  - Support for hierarchical file systems
  - Update/retrieval operations: read, write, append, seek
  - Mapping of files to secondary storage

What is the Role of the OS ? (4)

- **Provide a User Interface**
  - Graphical interface
  - Command line interface
- **Process Communication**
  - Kernel provides methods for Synchronization and Communication between processes, called Inter Process Communication (IPC)
- **Security**
  - Prevents un-authorized access to programs and data (authentication)

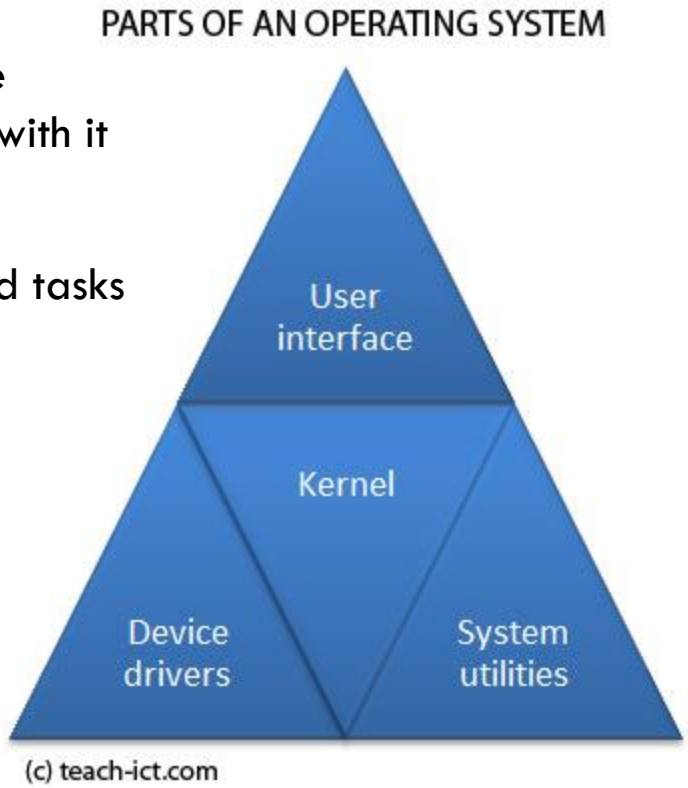
What is the Role of the OS ? (5)

- **Scheduling**
  - The kernel uses Scheduling Algorithms to determine which process is running when and how much time it will be given
  - The algorithm sets priorities among the processes
  - The kernel will assign a time slice to every program (normally, every program needs many time slices to be completed)
- **System Calls and Interrupt Handling**
  - A system call is a mechanism that is used by the application program to request a service from the operating system
  - The interface between the OS and the user programs is the set of system calls
  - System calls include close, open, read, wait and write

Parts of an Operating System

The operating system has four main parts:

- **The Kernel** (the 'brain' of the OS)
- **The Device Drivers**
  - Every piece of hardware in the computer has a device driver that allows the OS to control and communicate with it
- **The User Interface**
- **The System Utilities.** Programs responsible for specialized tasks
  - Windows Diagnostics
  - Windows Performance Monitor
  - Windows Event Viewer
  - Windows Registry Editor
  - Windows Task Manager



Taken from https://www.teach-ict.com/xml/submainlogin.php?fn=/2016/GCSE_Computing/AQA_8520/3_4_computer_systems/343_software_classification/operating_systems/miniweb/pg4.php

Booting Process

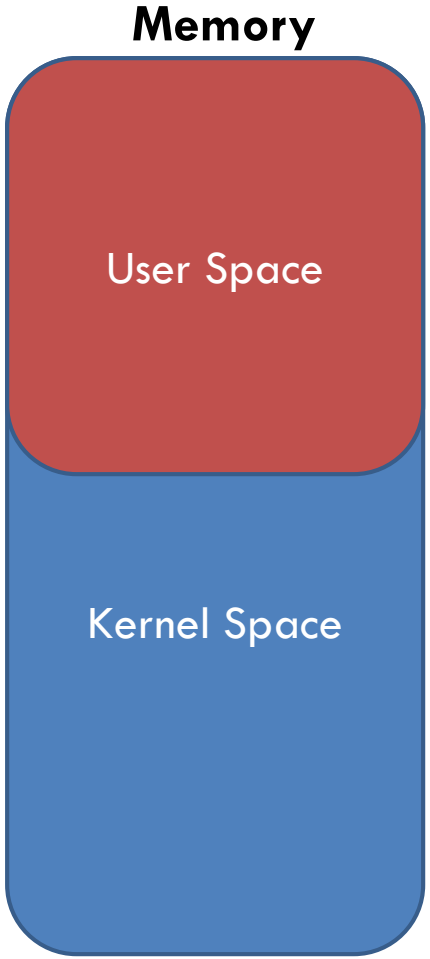
- **The bootstrap program is loaded at power-up or reboot**
  - Typically stored in ROM or EPROM, generally known as firmware
  - Initializes all aspects of system including the CPU registers, device controllers, and memory
  - Locates and loads operating system kernel and starts execution of the first process and waits for events to occur

The kernel

- **The kernel is the core component of an operating system**
- Think of a music player you are running in a Windows operating system
  - When you open an audio file in the music player, the music player needs Hardware like audio devices to perform your command
  - Kernel makes the communication between the music player & audio devices available
  - Kernel is a communication layer between Software & Hardware

User space and kernel space

- **Main Memory is divided into two distinct areas:**
  - **The user space,** which is a set of locations where normal user processes run (i.e everything other than the kernel)
  - **The kernel space,** which is the location where the code of the kernel is stored and runs
- Processes running under the user space have access only to a limited part of memory, whereas the kernel has access to all of the memory
- Processes running in user space also don't have access to the kernel space
- User space processes can only access a small part of the kernel via an interface exposed by the kernel - **the system calls**



Two main types of kernels – Monolithic and Mircrokernels

- **Monolithic kernel** is a single large process running entirely in a single address space
  - It is a single static binary file
  - All kernel services exist and execute in the kernel address space. The kernel can invoke functions directly
  - Examples of monolithic kernel based OSs: Unix, Linux
- In **Microkernels**, the kernel is broken down into separate processes, known as servers
  - Some of the servers run in kernel space and some run in user-space
  - All servers are kept separate and run in different address spaces
  - Servers invoke "services" from each other by sending messages via IPC (Inter Process Communication)
  - This separation has the advantage that if one server fails, other servers can still work efficiently. Examples of microkernel based OSs: Mac OS X and Windows NT

Monolithic kernel vs Mircrokernel

Monolithic kernel

- The entire OS is placed inside the kernel
- It runs as a single large process
- As all the services are placed inside the kernel, they have a single address space
- It is bigger in size
- It is easy to implement/code
- Performance is high (As kernel can invoke any function directly as everything is placed in the kernel)
- Less Secure (If one service fails, entire system crashes)

Microkernel

- Only bare minimum code is placed inside the kernel
- Here the kernel is broken down into processes called as servers
- It is smaller in size
- It is tough to implement/code
- Performance is low (As servers are separated, so to invoke services from other servers IPC (Inter Process Communication) is needed which requires kernel's permission and thus increases access time and lowers the performance)
- More Secure (Even if one service crashes, others can function properly because of separation)

Hybrid kernels

- The hybrid kernel design is something between the microkernel and the monolithic kernel
- This kernel approach combines the speed and simpler design of monolithic kernel with the modularity and execution safety of microkernel
- Examples include Windows 10

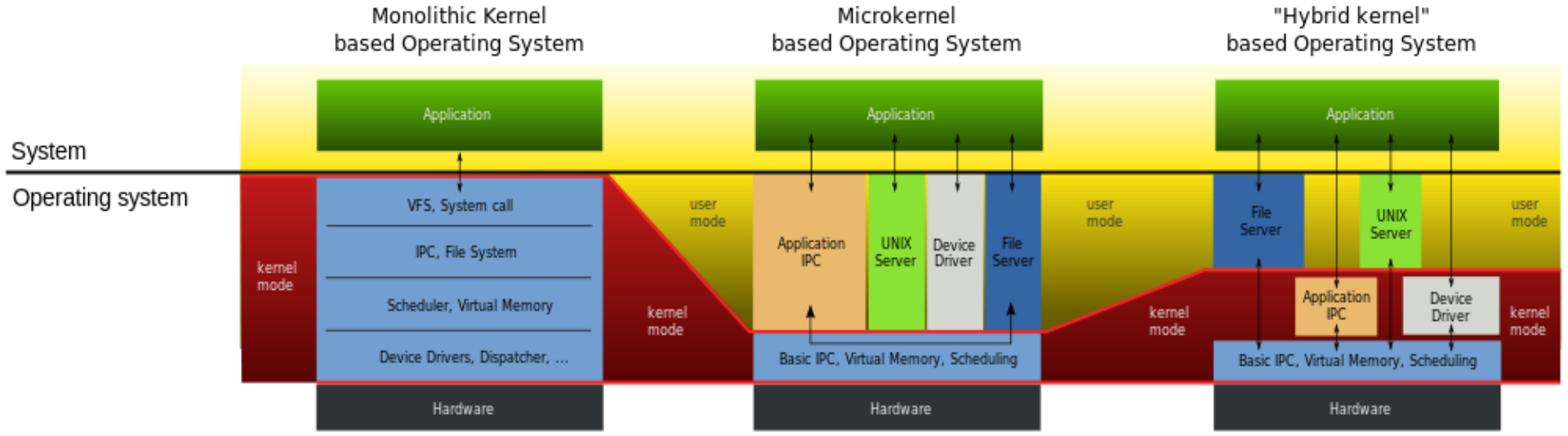


Image courtesy, wikipedia

Memory Management

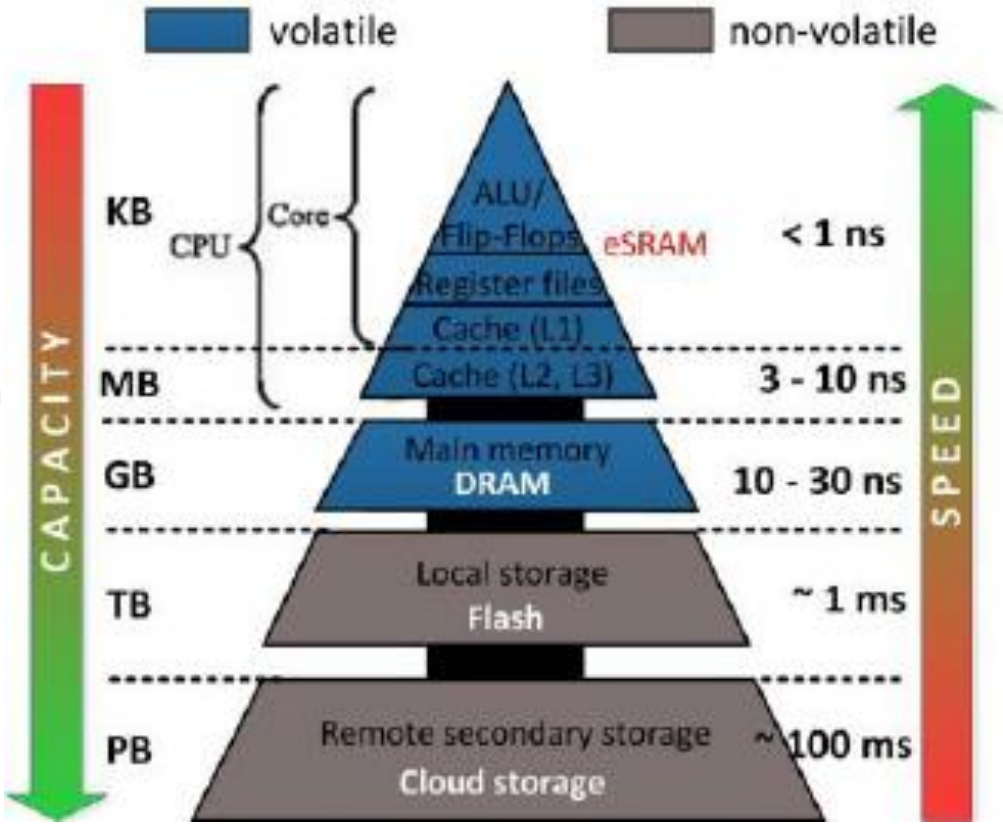
What is Memory Management ?

- **Is the task carried out by the OS and hardware to accommodate multiple processes in main memory**
  - This makes sure that a program does not conflict with memory currently being used by another program
    - Keep track of which parts of memory are currently being used & by what
  - Map processes to memory locations
  - Allocate/deallocate memory space as requested/required

- The size of main memory is limited
- In most schemes, the kernel occupies some fixed portion of main memory and the rest is shared by multiple processes

Swapping

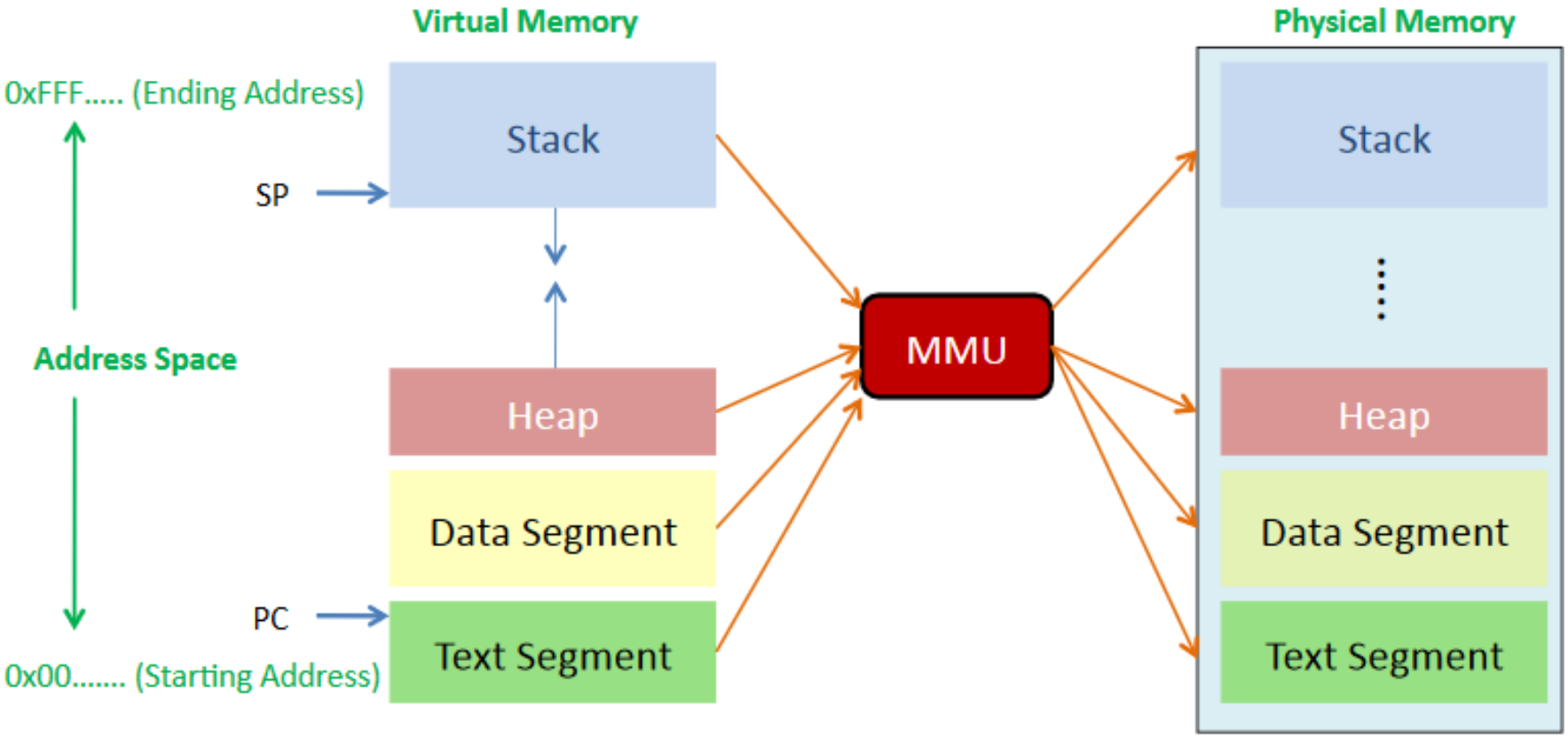
- **Swapping**
  - A process needs to be in main memory in order to be executed
  - But sometimes there is not enough main memory to hold all the currently active processes
  - So, process are kept on disk and brought in to run dynamically.
  - **Swapping is the process of bringing in each process in main memory, running it for a while and then putting it back to the disk.**



Virtual and Physical Memory

- Recall from assembly that in order to load/store we need to refer to a memory location but
  - Physical memory is limited, normally 4-16Gbytes
  - A computer must be able to address more memory (hard disc)
- This is why **programs use virtual memory addresses and not physical**
  - The ***OS*** and the ***Memory Management Unit (MMU)*** are responsible for translating a virtual address to a physical
  - Virtual memory is divided into pages, normally 4kbytes each
  - Moreover, using virtual addresses allows to have **memory protection**
    - Virtual memory allows the system to give every process its own memory space isolated from other processes - processes cannot easily interfere with each other

Virtual and Physical Memory



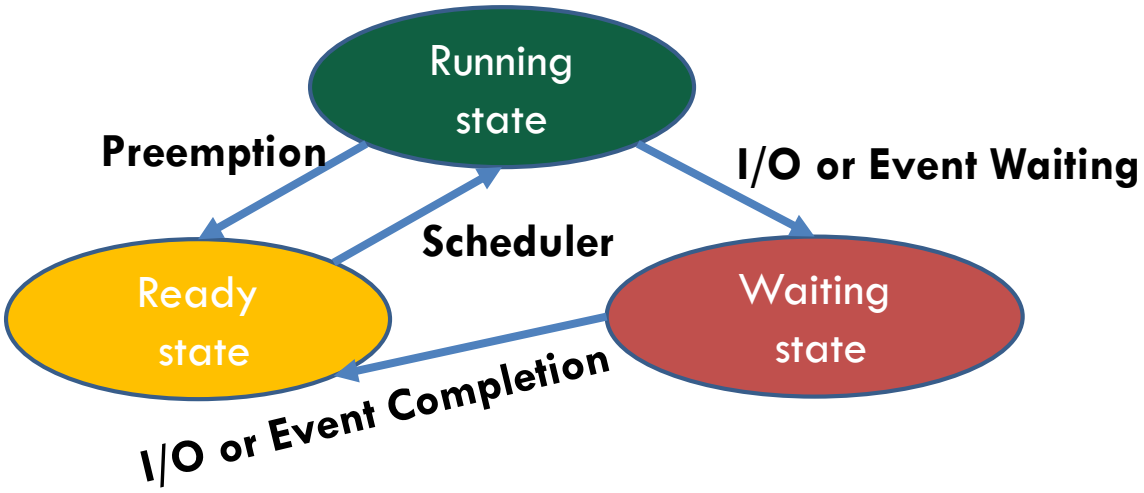
Virtual Memory

- Virtual memory enables programs to execute without requiring their entire address space reside in physical memory. This provides ***benefits***:
  - **Utilizes main memory size**
    - Many programs do not need all of their memory at once (or ever), so there is no need to allocate memory for it
  - **Flexibility**
    - Programs can loaded/evicted from/to memory in a dynamic way
  - **Security**
    - A process cannot access the memory of other processes (there are some exceptions)
- ***Challenges*** also arise such as the overhead of
  - context switch (page swapping is very slow as the slow disc is accessed)
  - Address translation from virtual to physical
  - **This is why hardware support is needed - MMU**

Process Management

Process States

- At any point in time, a **process** can be in one of **several states:**
  - **Running State:** The process is running using CPU and memory resources
  - **Ready State:** A process is ready to run, but it is not running
  - **Waiting State:** The process cannot run because it is waiting for some event to occur (or data)



Context Switch

- When the CPU switches to another process then
  - system must save the state of the old process
  - load the saved state for the new process
- Context-switch time is an overhead
  - system does no useful work while switching
- Time is dependent on hardware support

Interrupts

- **Interrupts are signals sent to the CPU by external devices, normally I/O devices**
- When an interrupt occurs, the CPU pre-empts the running process and services the interrupt
- Interrupts are classified into
  - *Hardware Interrupts*
    - Generated by hardware devices, e.g., transferring data from the hard disk
  - *Software Interrupts*
    - Generated by programs that they request a system call
  - *Traps*
    - Generated by the CPU itself to indicate that some error occurred, e.g., division by zero, or invalid memory access

Scheduling

Scheduling Objectives (1)

- Ensure fairness
  - Every process should get a fair share of CPU time
- Maximise throughput
  - Maximum no. of processes in any given period
- Minimise turnaround time
  - Minimise time between submission of a process & completion
- Maximise CPU utilisation
  - CPU as close to 100% use as possible
- Maximise resource allocation
  - Balancing CPU & I/O bound processes

Scheduling Objectives (2)

- Prevent starvation
  - Processes should not be allowed to starve (indefinite postponement).
- Minimise response time
  - Processes should complete as quickly as possible
- Provide consistent response time
  - Users expect long jobs to take more time than small jobs.
  - Users expect same job to take about the same amount of time each time run
- Immediate response time is important under some conditions:
  - Echoing keyboard instructions to a screen
  - Real-Time OS

Scheduling Algorithms

- A process scheduler schedules processes using a scheduling algorithm
  - **First Come, First Served** algorithm
  - **Shortest Job First** algorithm
  - **Priority Scheduling** algorithms
    - Each process is assigned a priority
    - Processes with high priority are executed first
  - **Shortest remaining time** algorithms
  - **Round Robin** algorithm
    - All processes are assigned to equal time slots and executed one after another
- The aforementioned algorithms can be either
  - **Pre-emptive:** A process might be pre-empted by a higher priority process
  - **Non Pre-emptive:** Once a process starts, it cannot stop, until it completes its allotted time

Virtualization

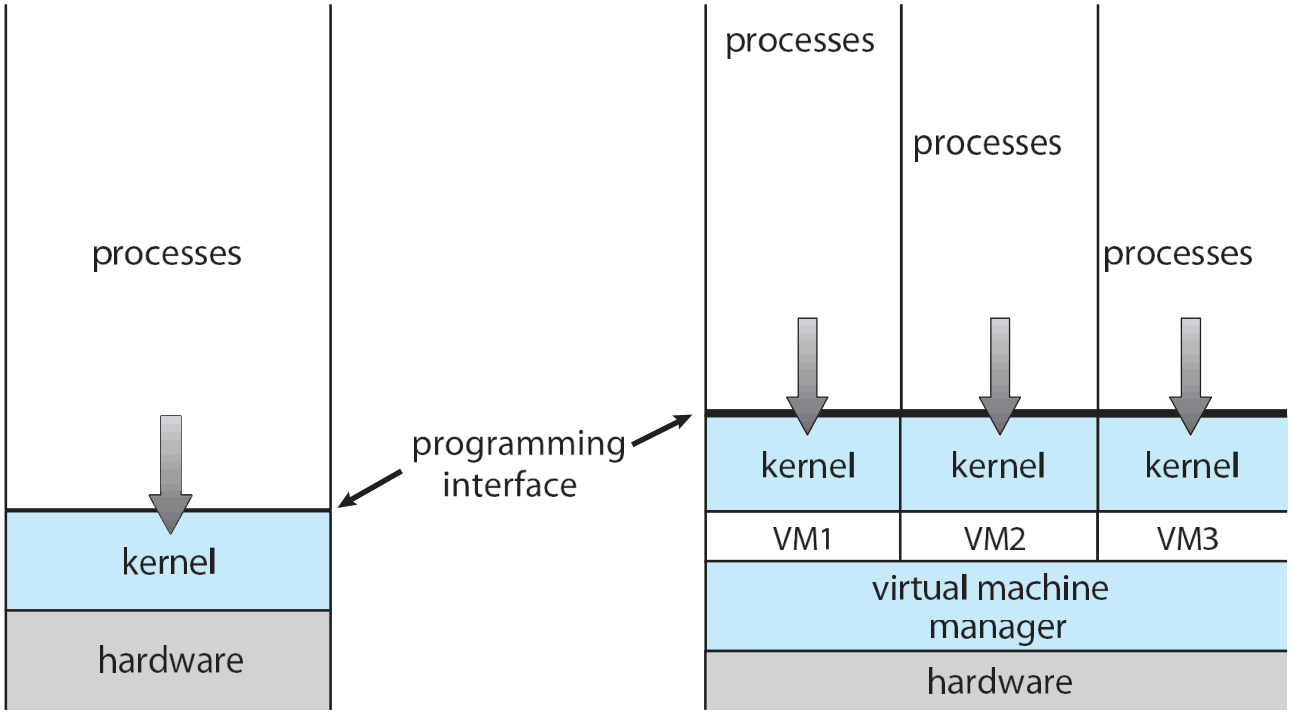
Virtualization (1)

- ***It is the process of running a virtual instance of a computer system in a layer abstracted from the actual hardware***
- Virtualization relies on software to simulate hardware functionality & create a virtual computer system
- Virtualization software makes computing environments independent of physical infrastructure
- Many virtual computers can be hosted on a single physical one
  - No need to deploy multiple servers
  - cost savings

Virtualization (2)

- Virtual system, known as a “virtual machine” (VM)
  - An OS & applications can run
  - multiple VMs on a single computer enables several OS & applications to run on just one physical server, or “host”
- **The Virtual Machine Manager (VMM)** or “hypervisor” software, decouples the VMs from the host
  - **creates & runs VMs** by providing interface that is identical to the host
  - dynamically allocates computing resources to each VM

Virtualization (3)



Non-virtual machine

Virtual machine

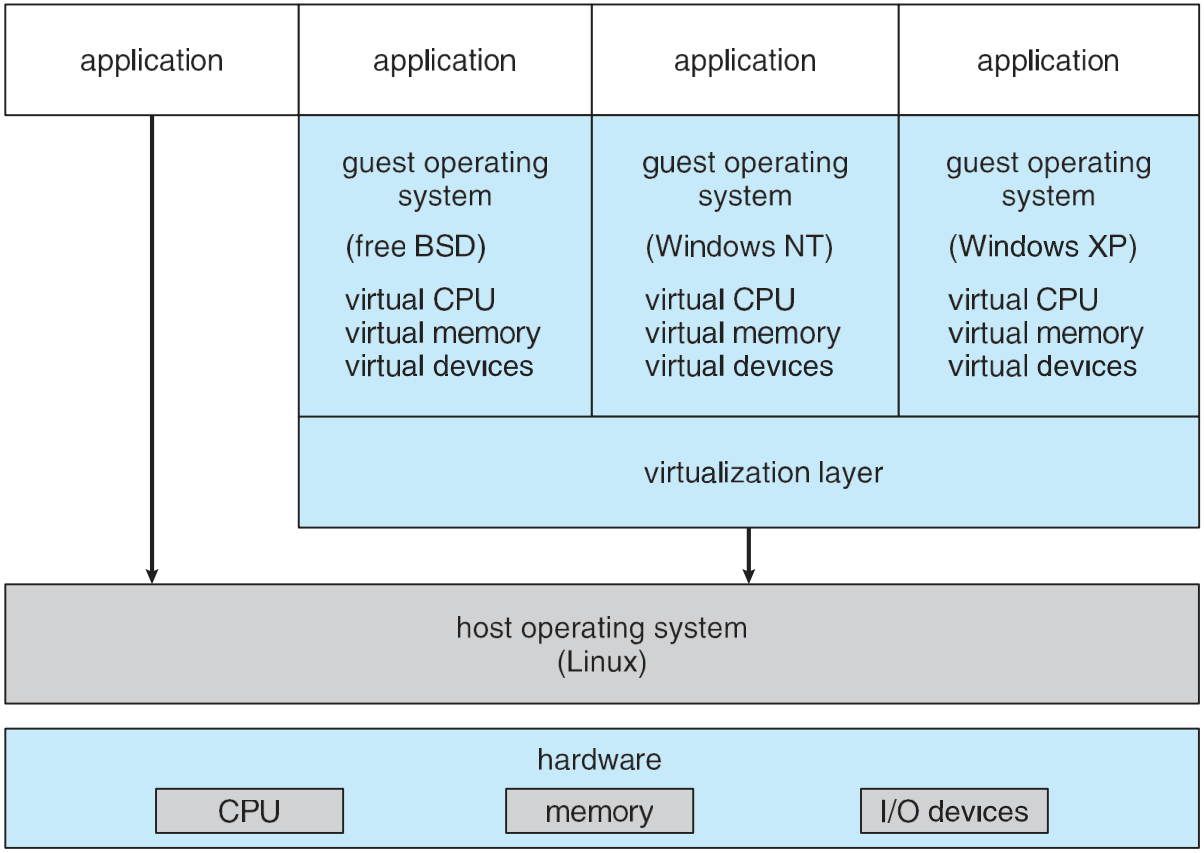
Benefits of Virtualization

- The host system is protected from the VMs – The VMs are protected from each other
  - Virus less likely to spread
- Freeze or suspend running a VM
  - Then can move or copy somewhere else & resume
  - Snapshot of a given state, able to restore back to that state
  - Clone by creating copy & running both original & copy
- Run multiple, different OSs on a single machine
- Can increase performance & availability of resources
  - Reduced capital & operating costs
  - Increased IT productivity, efficiency, agility & responsiveness
  - Simplified data centre management

Example – VMware (1)

- Virtualisation tools like VMware are now common
- You can host many virtual computers on a single physical one
  - each VM can run a different OS, inc Windows, Linux, etc
  - all runnable concurrently
- Virtualization layer abstracts the underlying hardware
  - providing guest with own virtual CPUs, memory, disk drives, network interfaces, etc

Example – VMware (2)



Activity – Task Manager

- See notes

Further Reading

- Operating Systems, Internals and Design Principles, available at https://dinus.ac.id/repository/docs/ajar/Operating_System.pdf

Thank you