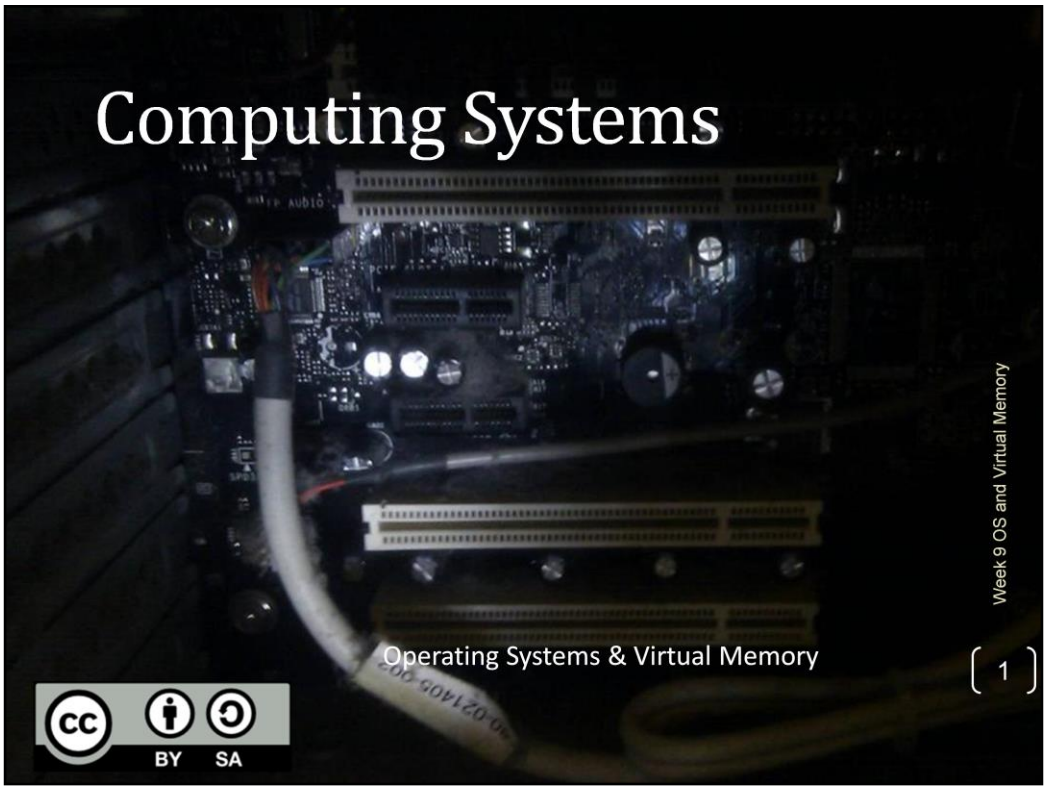
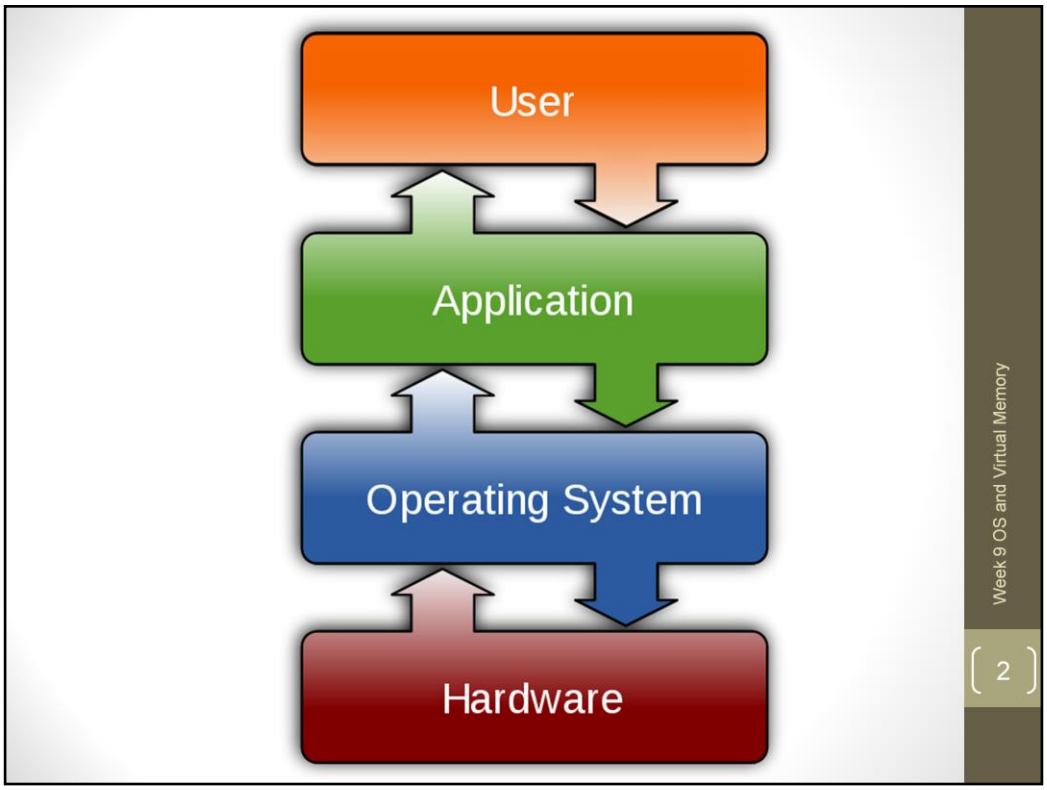

Computing Systems
Operating Systems & Virtual Memory
Week 9 OS and Virtual Memory
1
BY SA

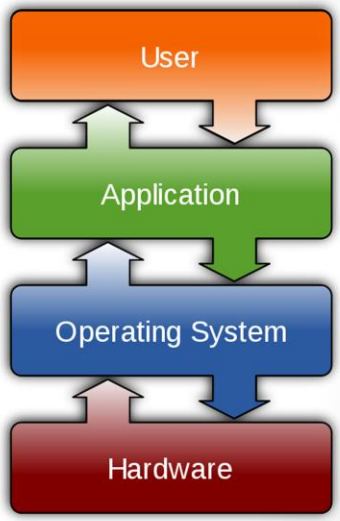

User
Application
Operating System
Hardware
Week 9 OS and Virtual Memory
2

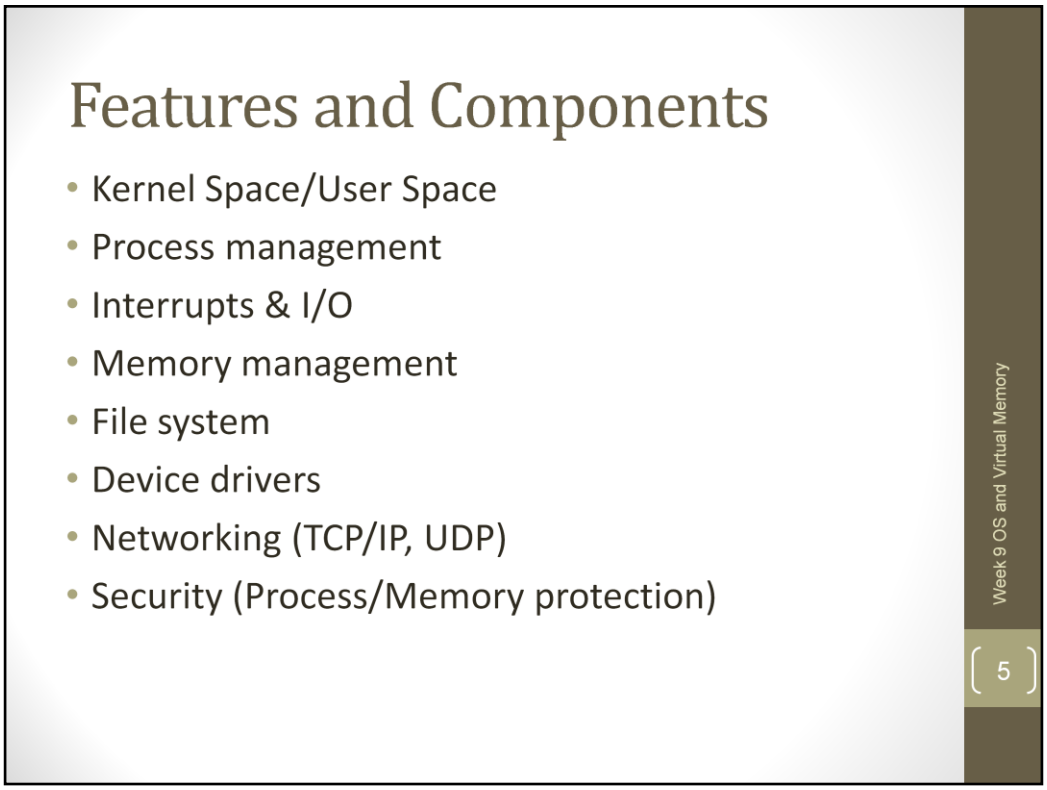
Operating Systems (OS)

- OS is a collection of software that manages computer hardware resources and provides common services for computer programs.
- The operating system is an essential component of the system software in a computer system.
- **Application** programs usually require an **operating system** to function.

Operating Systems

Software that sits between applications (**e.g. office applications or games**) and the hardware itself

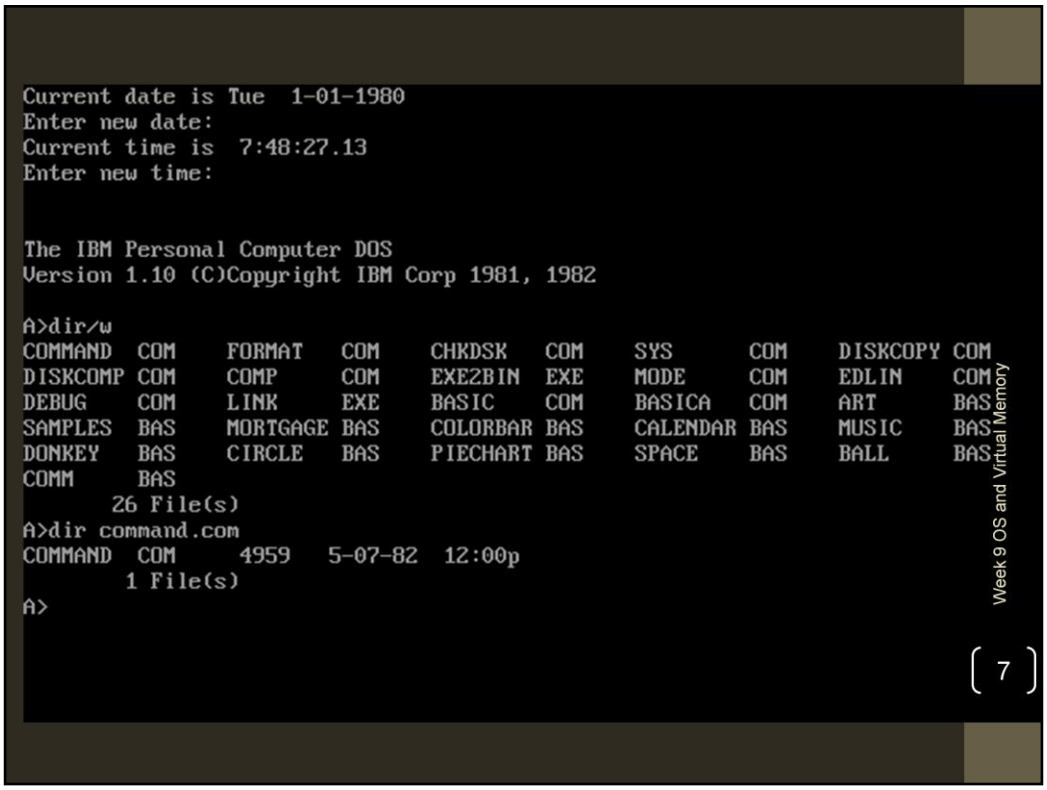




List taken from
http://en.wikipedia.org/wiki/Operating_system

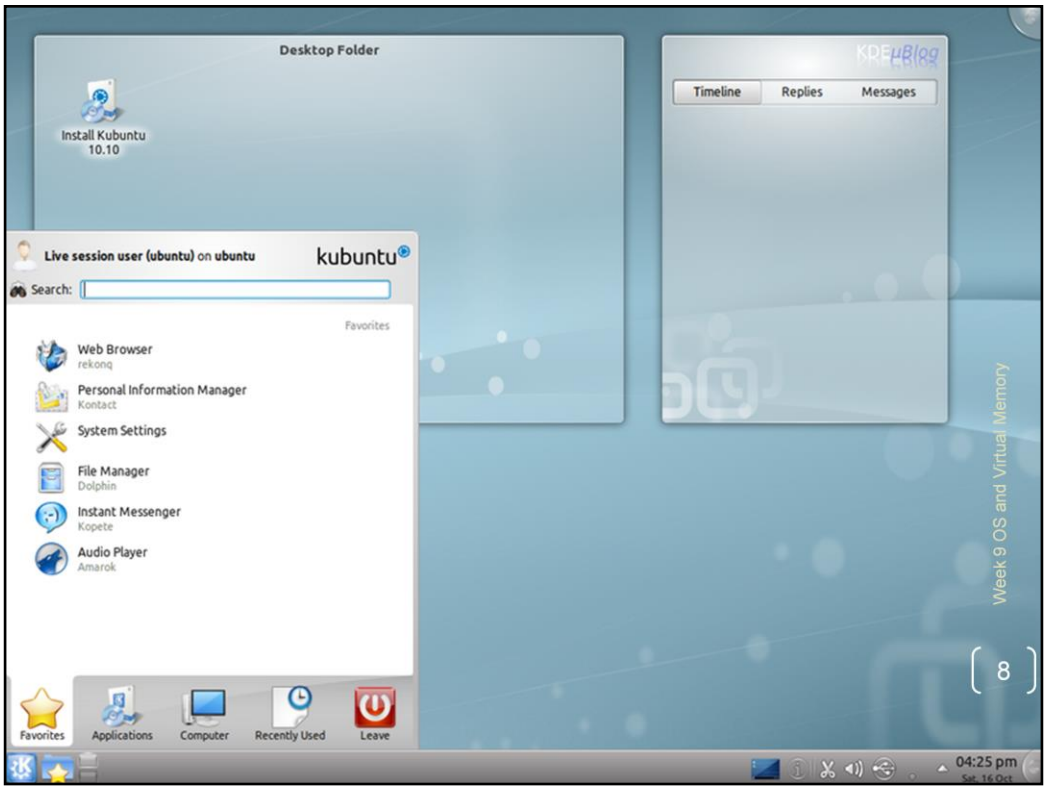
Popular Operating Systems

- Windows
- *nix (**Unix** family of operating systems – including **Mac OS X**, **Linux**, **iOS**, Android and many others)
- **DOS** (pre-Windows Operating Systems for PCs)
- Also: **Symbian**, **WebOS**, **BlackBerry** OS, ...

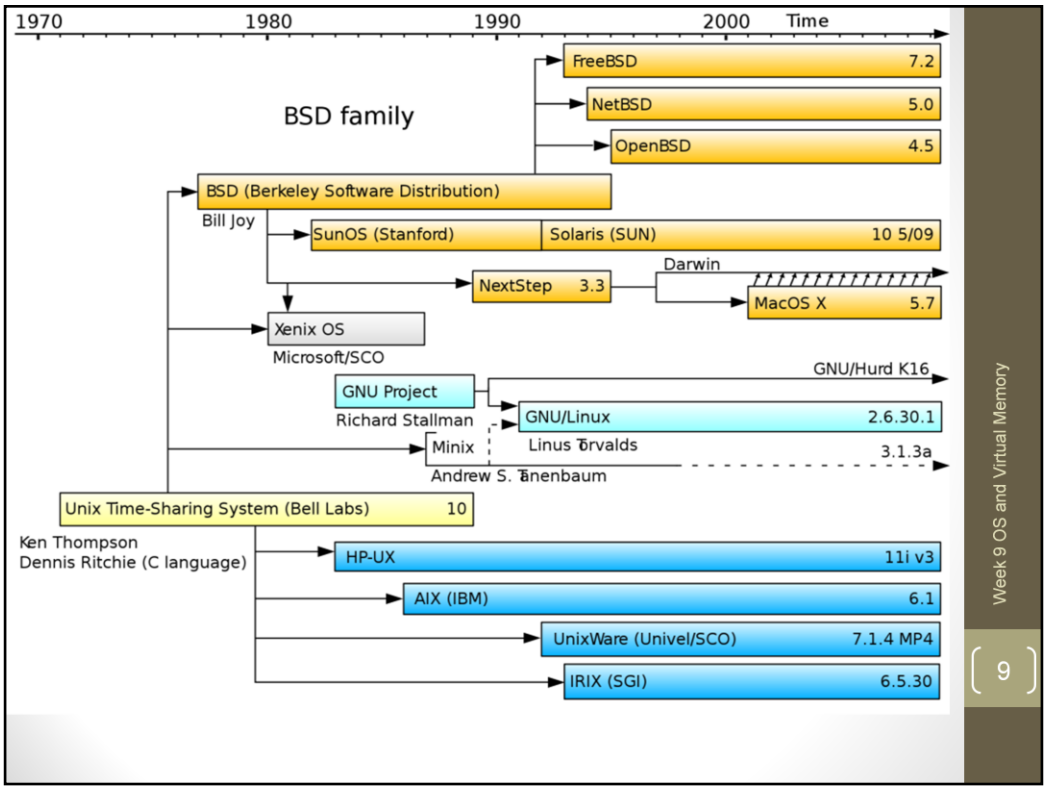


This is PC-Dos – the IBM version of MS-DOS, the same DOS as sold by Microsoft, but with the IBM name instead.

Other companies sold compatible DOS systems for PCs, but Microsoft's licensing agreement with IBM allowed Microsoft to become the dominant software company for Operating Systems – and to exploit this to grow into other areas of computing.



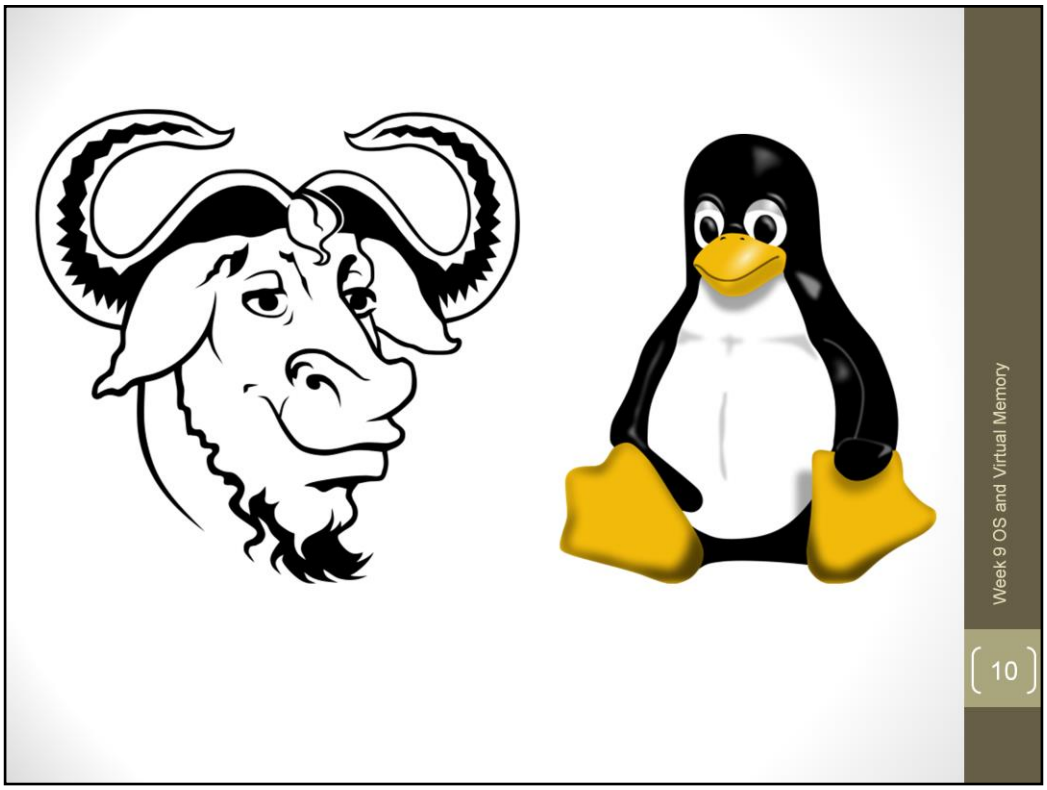
This image shows an Ubuntu KDE desktop (Kubuntu). Ubuntu is a popular Linux distribution. Linux is open source, and the license allows companies or individuals to create their own distributions or versions. The Linux OS can work with (or without!) a GUI component. The X Windows System (aka X11) provides the GUI API, but two distinct front-ends desktop environments with their own sets of applications have emerged: KDE and Gnome. (Prior to KDE and Gnome, *nix applications had generally inconsistent GUIs and often looked visually poor in comparison to e.g. Windows or Mac OS)



The UNIX family has a long and convoluted history.

A multi-user OS underdevelopment by Bell Labs, MIT and GE 'Multics' was innovative but problematic. Bell Labs tried again. A team including Ken Thompson, Dennis Ritchie and Brian Kernighan developed *Unics* – then renamed Unix. Along the way, Ritchie and Kernighan created the C programming language to write the new OS.

Overtime many versions of UNIX were developed, with many major IT companies having their own version – including Microsoft.

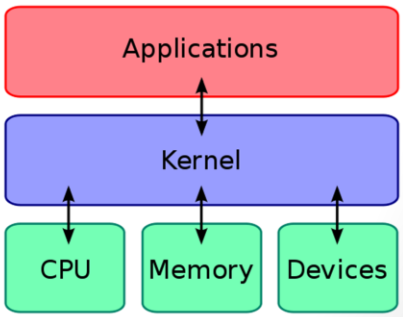


GNU (GNU is Not Unix), Tux the Penguin (Linux mascot)

The GNU project started in 1983 by Richard Stallman with the goal of creating a complete free and Open Source version of the UNIX operating system. Over the following years the GNU project was able to make substantial progress with the new OS, creating much of the software required – compilers, editors, libraries and utilities. But progress was slow in developing the OS Kernel – the core of the new OS. In 1991 Linus Torvalds wrote and released Linux – an OS core that could be used with GNU to create a complete and usable OS.

The Kernel

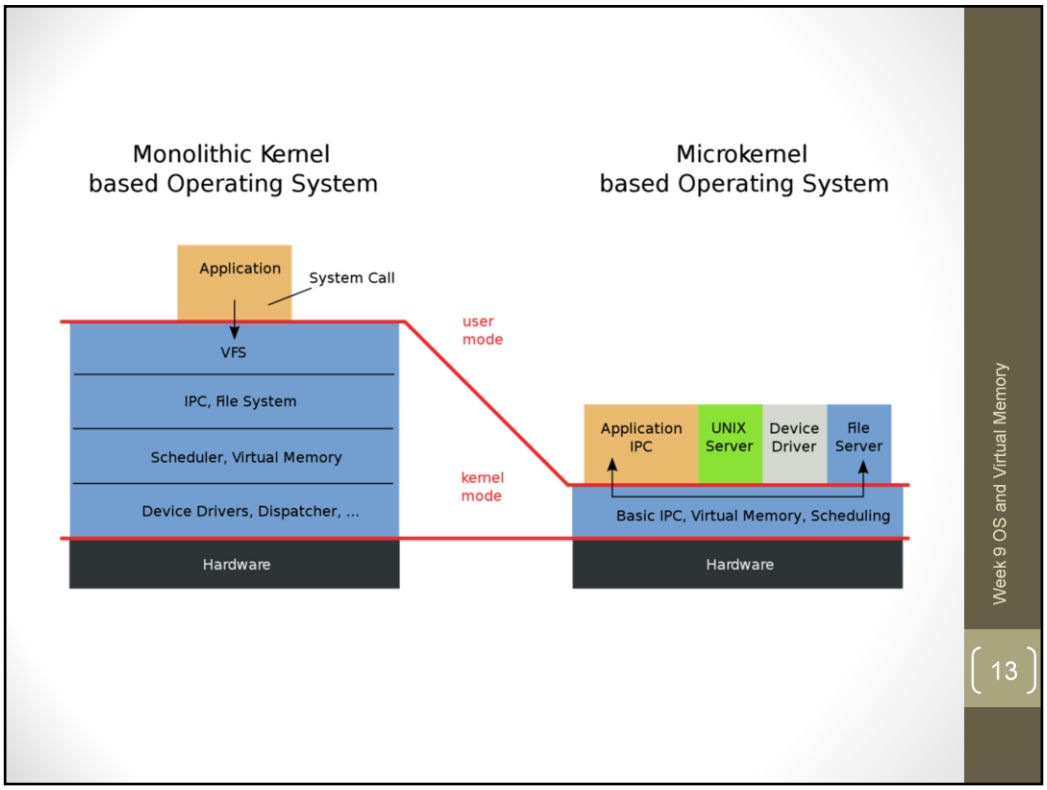
- Most basic set of OS functionality exists in the **OS Kernel**
- **Kernel** has some key tasks:
  - **Manage different running processes**
  - **Allocate memory**
  - **Process I/O**
- Kernel may also perform **file management**, run **device drivers**, and carry out a range of other tasks



Linux actually refers to the Kernel – GUI and many other elements are not part of the core of Linux. A Linux distribution is made up (typically) of the Linux Kernel, a windowing system, desktop environment and a range of applications. The kernel itself is much more limited... Which is why some Linux systems look very different to one another.

Kernel

- The **kernel** is a computer program that ***manages*** **input/output** requests from **software** and ***translates*** them into **data processing instructions** for the **central processing unit** and other electronic **components** of a computer.
- It ***connects*** the **application software** to the **hardware** of a computer.
- It ***manages memory*** access for programs in the **RAM**, it determines which **programs** get ***access*** to which **hardware** resources, it ***sets*** up or ***resets*** the **CPU's operating states** for optimal operation at all times, and it ***organizes*** the **data** for **long-term non-volatile** storage with **file systems** on such **media** as **disks**, **tapes**, **flash** memory, etc.

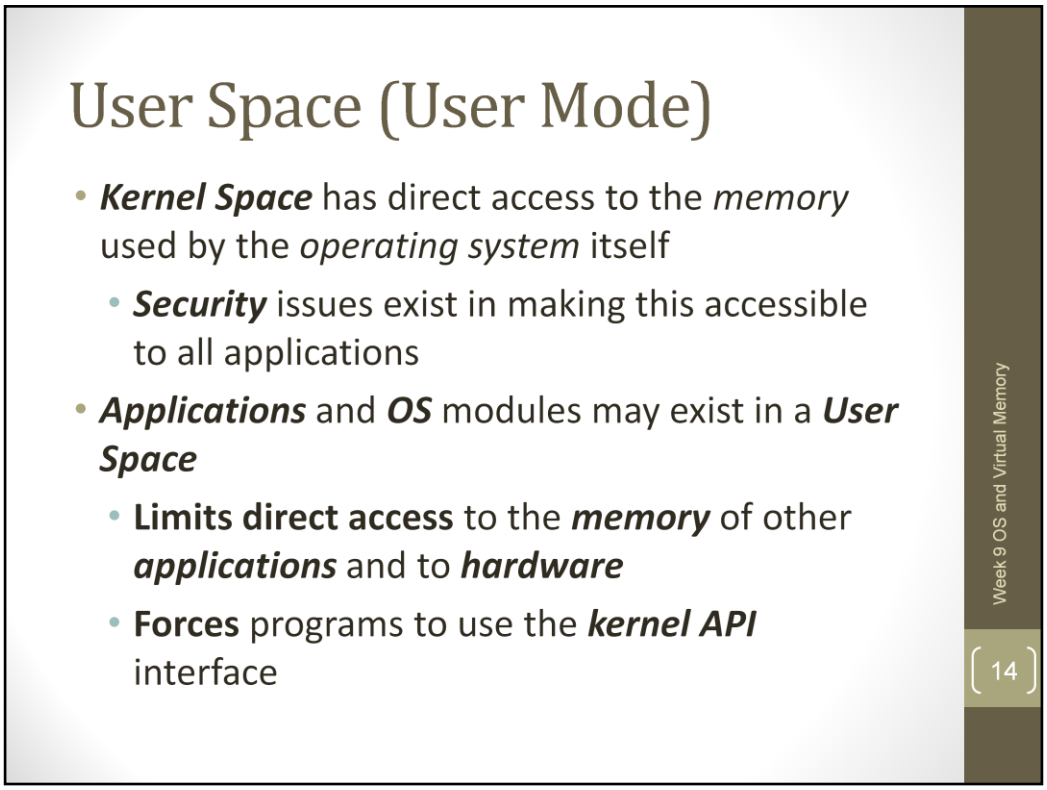


Windows and **Linux** both use a fairly **monolithic** kernel design.

Micro-kernels move parts of the **operating system** into services that run **outside** of the **kernel**. This can mean that some failures (in e.g. The file server or with device drivers) are less likely to cause crashes, and the system may be more secure. The cost is that there may be a performance hit.

IPC = "Inter Process Communication"

VFS = "Virtual File System" – more later...



User space provides some protection to the system from malicious or poorly written programs that might crash or damage the system.

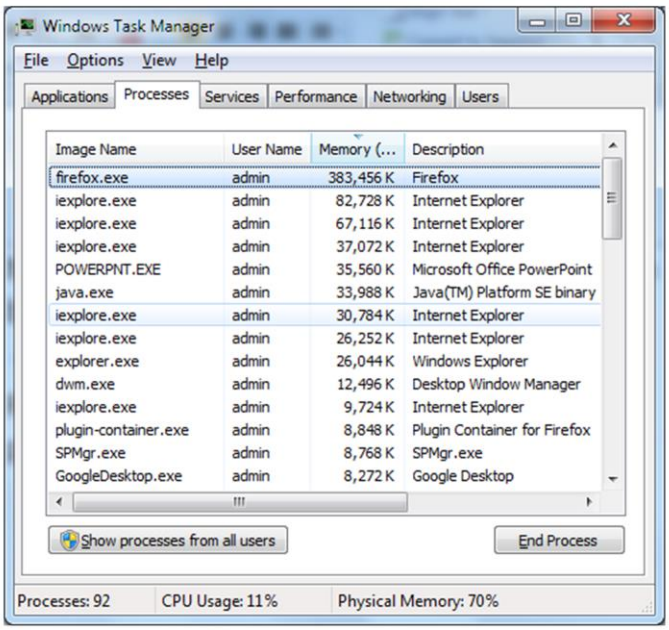
A little more on this later!

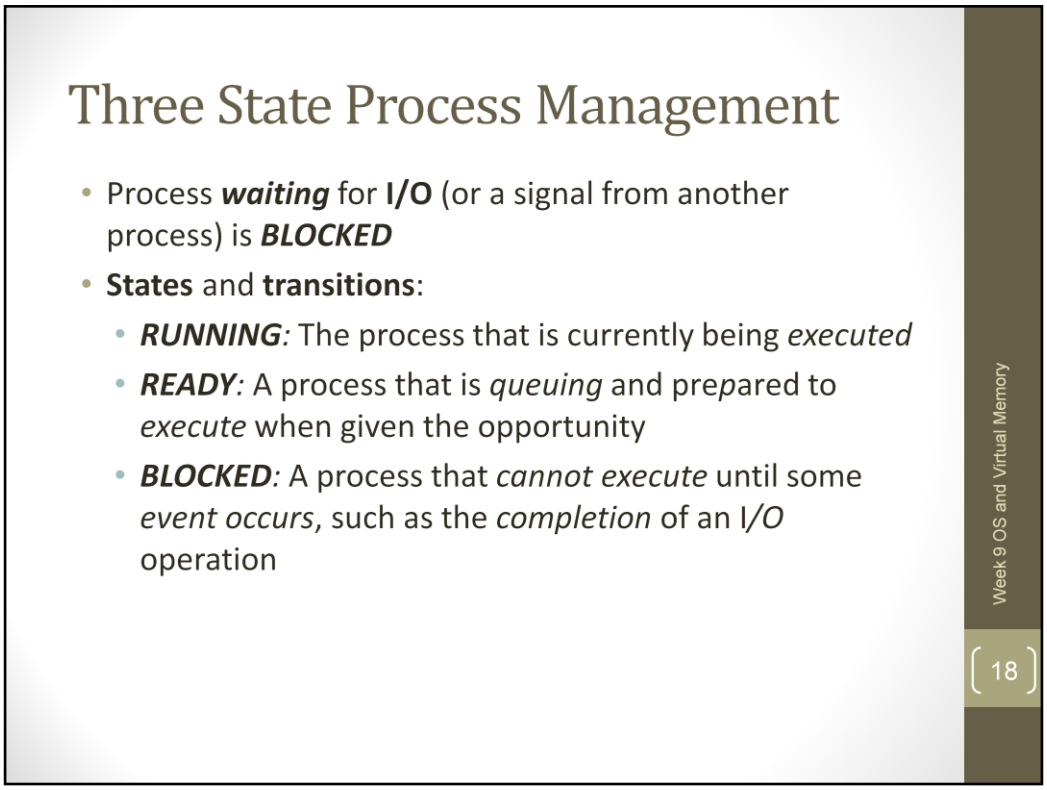
Process Management

- Modern **OS** allow *multiple programs* to be run by a *computer* at *one* time
  - Even if there is only *one processor*
- Requires **OS** to switch between *running programs/processes*
  - If one process is *waiting* for user input, *pause* it and run a *different* process
- Some process may need to interact
  - **OS** manages ***Inter-Process Communication (IPC)***

Multi-Processing & Multi-Tasking

- ***Multi-Processing*** uses **multiple processors** (or **processor cores**) to run **multiple tasks** (programs)
- ***Multi-Tasking*** allows **multiple tasks** to run on **one** or **more processors**
  - May have **multiple tasks** per **processor**
  - **Rapid switching** between **tasks**, performing a little bit of each, allows computer to give the appearance of running all the tasks at the same time


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Image Name User Name Memory (... Description
firefox.exe admin 383,456 K Firefox
iexplore.exe admin 82,728 K Internet Explorer
iexplore.exe admin 67,116 K Internet Explorer
iexplore.exe admin 37,072 K Internet Explorer
POWERPNT.EXE admin 35,560 K Microsoft Office PowerPoint
java.exe admin 33,988 K Java(TM) Platform SE binary
iexplore.exe admin 30,784 K Internet Explorer
iexplore.exe admin 26,252 K Internet Explorer
explorer.exe admin 26,044 K Windows Explorer
dwm.exe admin 12,496 K Desktop Window Manager
iexplore.exe admin 9,724 K Internet Explorer
plugin-container.exe admin 8,848 K Plugin Container for Firefox
SPMgr.exe admin 8,768 K SPMgr.exe
GoogleDesktop.exe admin 8,272 K Google Desktop
Show processes from all users
End Process
Processes: 92 CPU Usage: 11% Physical Memory: 70%



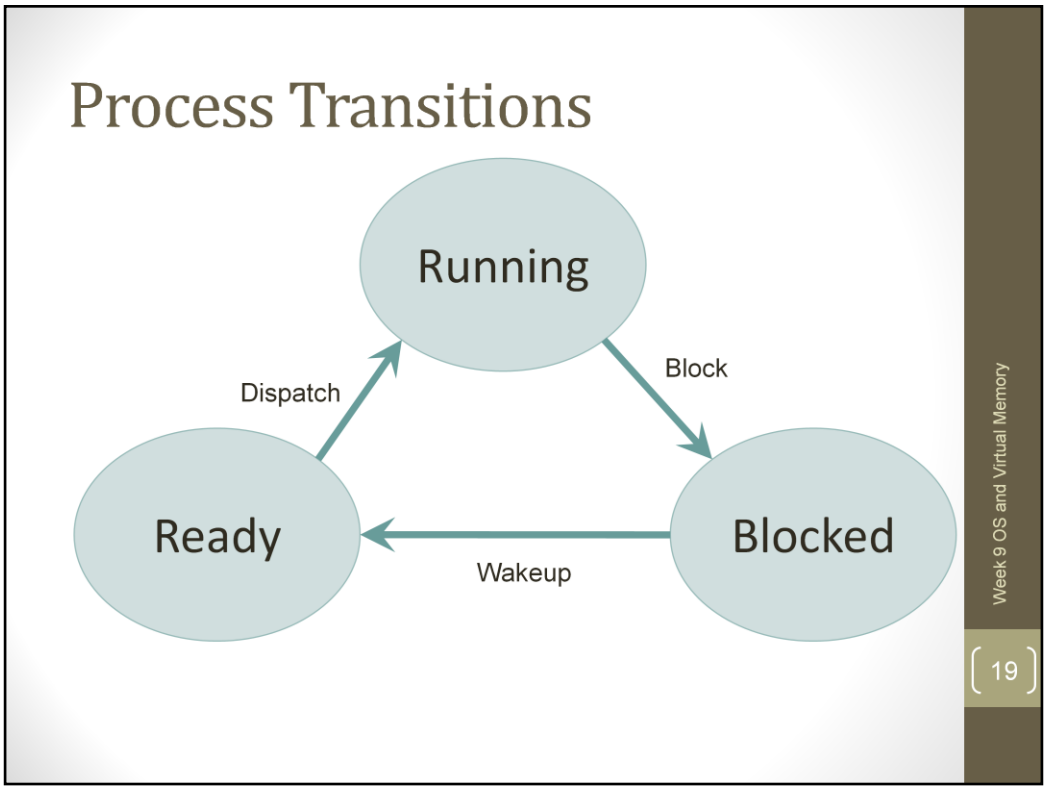
States and transitions text from http://en.wikipedia.org/wiki/Process_management_(computing) (CC-BY-SA)

Transitions (again, text from Wikipedia, as above):

Processes entering the system must go initially into the *READY* state, processes can only enter the *RUNNING* state via the *READY* state. Processes normally leave the system from the *RUNNING* state. For each of the three states, the process occupies space in main memory. While the reason for most transitions from one state to another might be obvious, some may not be so clear.

RUNNING → READY The most common reason for this transition is that the running process has reached the maximum allowable time for uninterrupted execution; i.e. time-out occurs. Other reasons can be the imposition of priority levels as determined by the scheduling policy used for the Low Level Scheduler, and the arrival of a higher priority process into the READY state.

RUNNING → BLOCKED A process is put into the *BLOCKED* state if it requests something for which it must wait. A request to the OS is usually in the form of a system call, (i.e. a call from the running process to a function that is part of the OS code). For example, requesting a file from disk or a saving a section of code or data from memory to a file on disk.



With **pre-emptive** multitasking, the OS will regularly pause or block running processes to allow other processes to run. **Co-operative** multitasking requires process to voluntarily suspend or block themselves. These is also a five state variant of this diagram. From wikipedia http://en.wikipedia.org/wiki/Process_management_(computing) (CC-BY-SA) :

Five-state process management model

While the three state model is sufficient to describe the behaviour of processes with the given events, we have to extend the model to allow for other possible events, and for more sophisticated design. In particular, the use of a portion of the hard disk to emulate main memory (so called virtual memory) requires additional states to describe the state of processes which are suspended from main memory, and placed in virtual memory (on disk). Of course, such processes can, at a future time, be resumed by being transferred back into main memory. The Medium Level Scheduler controls these events. A process can be suspended from the *RUNNING*, *READY* or *BLOCKED* state, giving rise to two other states, namely, *READY SUSPEND* and *BLOCKED SUSPEND*. A *RUNNING*

process that is suspended becomes *READY SUSPEND*, and a *BLOCKED* process that is suspended becomes *BLOCKED SUSPEND*. A process can be suspended for a number of reasons; the most significant of which arises from the process being swapped out of memory by the memory management system in order to free memory for other processes. Other common reasons for a process being suspended are when one suspends execution while debugging a program, or when the system is monitoring processes. For the five-state process management model, consider the following transitions described in the next sections.

BLOCKED → BLOCKED SUSPEND If a process in the *RUNNING* state requires more memory, then at least one *BLOCKED* process can be swapped out of memory onto disk. The transition can also be made for the *BLOCKED* process if there are *READY* processes available, and the OS determines that the *READY* process that it would like to dispatch requires more main memory to maintain adequate performance.

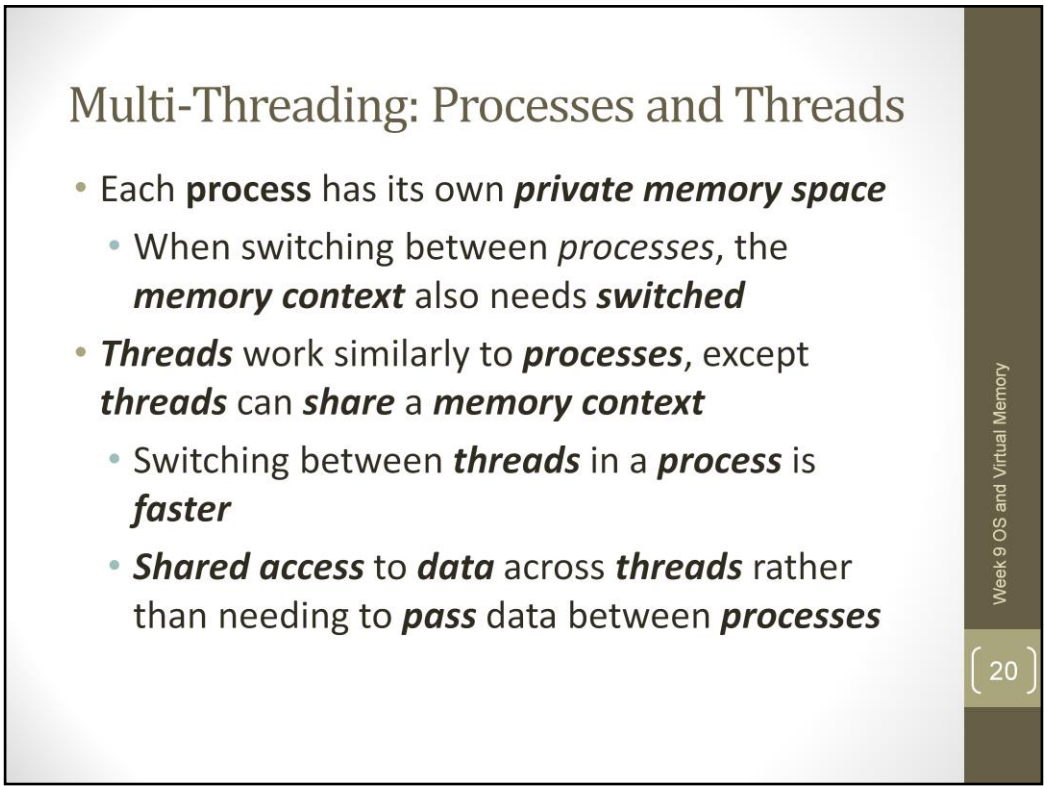
BLOCKED SUSPEND → READY SUSPEND A process in the *BLOCKED SUSPEND* state is moved to the *READY SUSPEND* state when the event for

which it has been waiting occurs. Note that this requires that the state information concerning suspended processes be accessible to the OS.

READY SUSPEND → READY When there are no *READY* processes in main memory, the OS will need to bring one in to continue execution. In addition, it might be the case that a process in the *READY SUSPEND* state has higher priority than any of the processes in the *READY* state. In that case, the OS designer may dictate that it is more important to get at the higher priority process than to minimise swapping.

READY → READY SUSPEND Normally, the OS would be designed so that the preference would be to suspend a *BLOCKED* process rather than a *READY* one. This is because the *READY* process can be executed as soon as the CPU becomes available for it, whereas the *BLOCKED* process is taking up main memory space and cannot be executed since it is waiting on some other event to occur. However, it may be necessary to suspend a *READY* process if that is the only way to free a sufficiently large block of main memory. Finally, the OS may choose to suspend a lower-priority *READY* process rather than a higher-priority *BLOCKED* process if it believes that the

BLOCKED process will be ready soon.



Multi-Threading: The ability of an operating system to **execute** different **parts** of a **program**, called **threads**, **simultaneously**. The programmer must carefully design the program in such a way that all the threads can run at the same time without interfering with each other.

Multiprocessing: Refers to a computer system's ability to support more than one **process** (**program**) at the same time. Multiprocessing operating systems enable **several programs** to *run concurrently*. **UNIX** is one of the most widely used multiprocessing systems, but there are many others, including OS/2 for high-end PCs. Multiprocessing systems are much more complicated than single-process systems because the operating system must allocate resources to competing processes in a reasonable manner.

I/O and Interrupts

- Already looked at **interrupts** and **I/O**
- **Operating System** responsible for taking **inputs** and passing these onto user applications where necessary
  - E.g. Passing mouse click or key press to the current application

Input/Output (I/O)

- I/O is the communication between an information processing system (such as a computer) and the outside world, possibly a human or another information processing system.
- **Inputs** are the **signals** or **data *received*** by the **system**, and **outputs** are the **signals** or **data *sent* from it**.

Interrupt

- An **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- An **interrupt** alerts the processor to a **high-priority** condition requiring the ***interruption*** of the current **code** the processor is **executing** (the current *thread*).

Interrupt

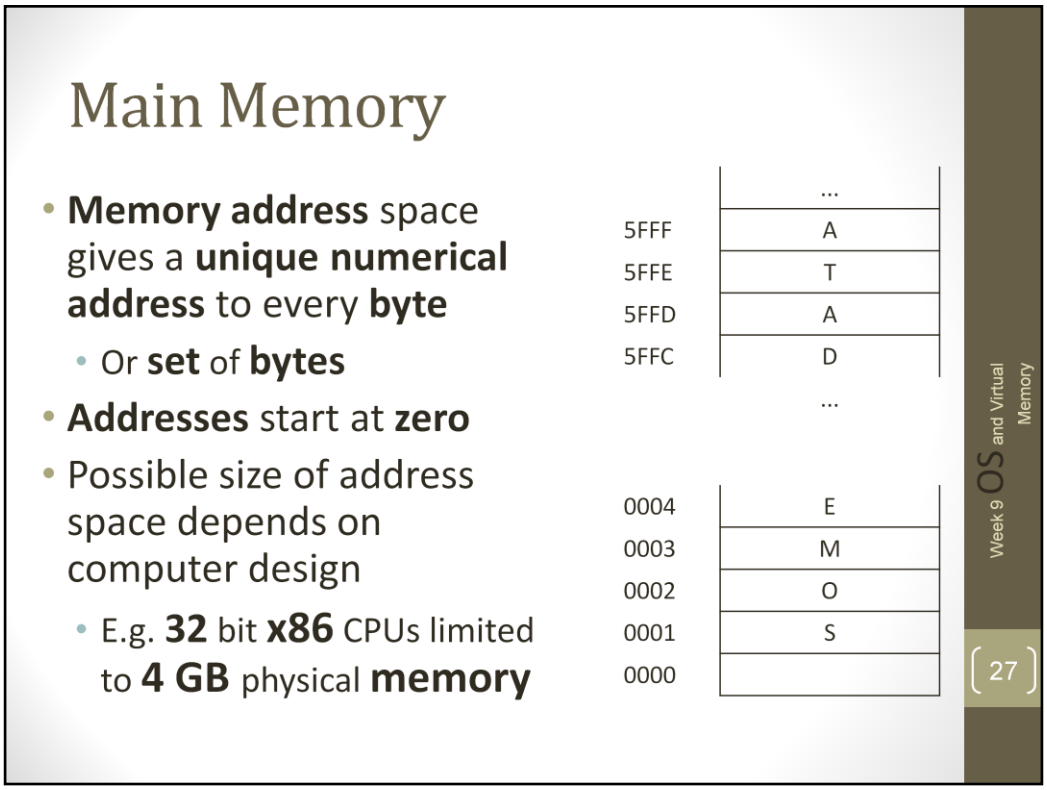
- The processor responds by suspending its current activities, saving its state, and executing a small program called an ***interrupt handler*** (or **interrupt service routine, ISR**) to deal with the event.
- This *interruption* is *temporary*, and after the interrupt handler *finishes*, the processor *resumes execution* of the *previous thread*.

Device Drivers

- **Programs** written to work with specific hardware systems – and to provide a simpler, abstract, **interface** to the **OS** and **applications**
  - **Kernel space** and **user space drivers**
- **OS** and application developers should **not need** to **know** the specific **low-level details** of all the different **hardware** systems that might be used
  - Cannot know about **future devices**

Memory Management

- **Memory management** is the act of managing computer memory.
- The essential requirement of memory management is to provide ways to **dynamically allocate portions** of **memory** to **programs** at their request, and free it for reuse when no longer needed.



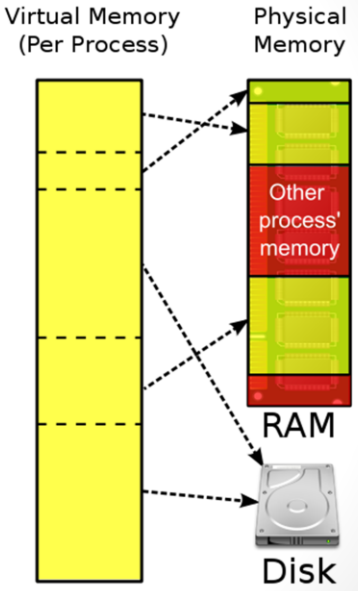
The first computers and first home computers had minimal OS memory management. Only one program could be run at a time, and it would have access to all of the computers main memory. In the 1970's and 80's this was often used (and abused) by expert programmers to squeeze the maximum performance possible from the computers of the time. Parts of main memory were also given over to system I/O – to the extent that by writing data values to specific addresses of main memory, output could be drawn to the screen, or locations in memory directly read to obtain input.

Memory Pages & Segments

- Instead of using physical address of each byte, can use a system where a memory address is stored as a reference to a ***page*** or ***segment*** of memory
  - Plus ***offset*** to data in the block
- Early architectures had very tight limits on the address space available to individual programs (***64K*** on early ***PCs***)
  - ***Segment***: Offset addressing increased total available memory

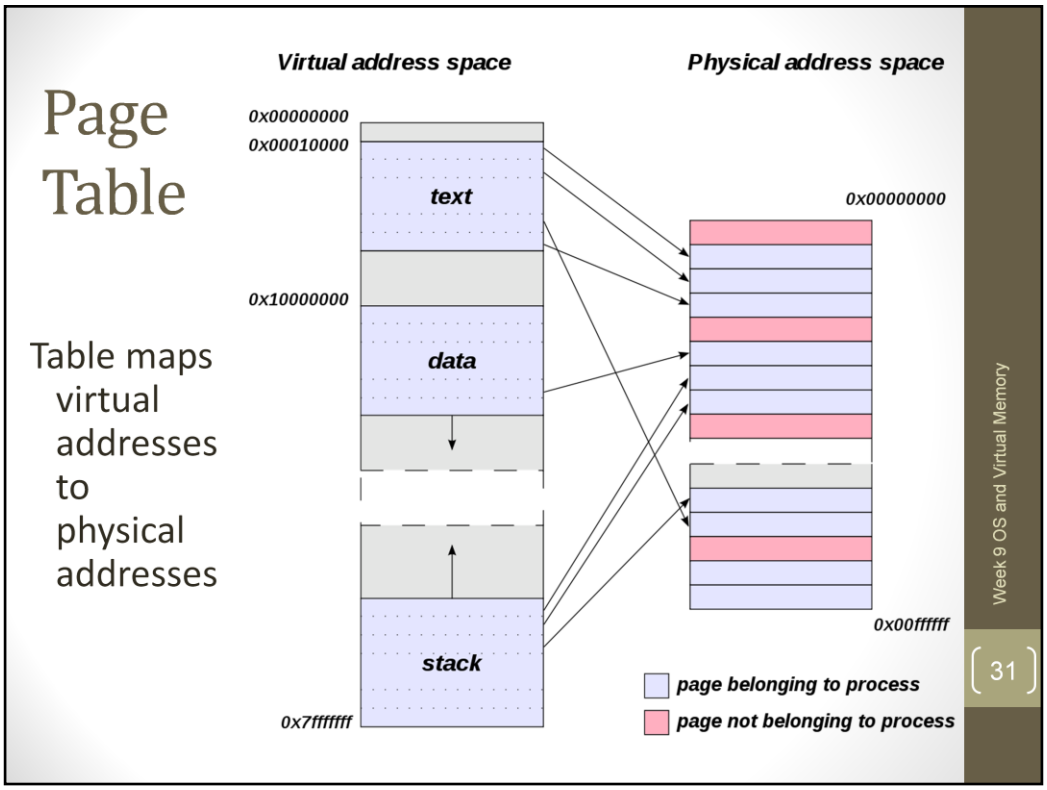
Virtual Memory

- With limited **physical memory**, ***multi-process OS*** could struggle to fit all programs into memory
- **Virtual Memory** allows the use of secondary **storage** to give the appearance of more memory than actually exists



Virtual Memory

- On disk *swap files* or *swap partitions* used to add *virtual memory* to *physical memory*
- Programs may have *partial residence*
  - Partly in physical **memory**, partly in **VM**
- *Scatter loading* allows different programs to exist on pages scattered through the physical and virtual memory
- **Scatter loading** and **partial residence** is invisible to the applications themselves



From: http://en.wikipedia.org/wiki/Page_table

Relationship between pages addressed by virtual addresses and the frames in physical memory, within a simple address space scheme. Physical memory can contain pages belonging to many processes. Pages can be swapped to disk if used infrequently, or if physical memory is full. Not all pages are in physical memory in the above diagram.

Virtual Memory Operation

- Application attempts to access a **VM address**
- **OS** checks virtual memory address on page table
  - If page currently in memory (a page '**hit**'), the **physical address** is **returned**
  - Else (a page '**miss**') the page needs **loaded** into **physical memory**, perhaps replacing an existing page which gets copied out to disk (***swap***)
- ***Thrashing*** can occur when low physical memory requires constant disk swapping

Virtual Memory & User Space

- Applications running in *user space* have their own *virtual memory address spaces*
- When they attempt to *read/write* memory, virtual addresses are *translated* to *physical ones* by the **OS** & the **hardware memory management unit**
- Attempts to access discs, etc., similarly mediated by OS
- This is all largely *invisible* to the *applications*

File System

- A **file system** is used to **control** how **data** is ***stored*** and ***retrieved***.
- Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins.
- By separating the data into *individual* pieces, and giving each piece a *name*, the information is easily *separated* and *identified*.
- Taking its name from the way paper-based information systems are named, each piece of *data* is called a "*file*".
- The *structure* and *logic* rules used to manage the *groups* of *information* and their *names* is called a "***file system***".

File System

- Modern file systems allow creation of folders, subfolders, and long file names.
  - *nix file systems allow spaces in file names
  - – Windows did not until Windows 95.

Networking

- OS will typically include a subsystem to support **networking**.
- This is part of **kernel** in **Windows**.
- In computer **networks**, **networked computing devices** pass ***data*** to each other along ***data connections***.
- The connections (***network links***) between **nodes** are established using either ***cable*** **media** or ***wireless*** **media**.

Networking

- **Nodes** can include **hosts** such as **personal computers**, phones, **servers** as well as **networking hardware**.
- **Two** such **devices** are said to be **networked together** when one **device** is able to ***exchange*** **information** with the **other device**, whether or ***not*** they have a **direct connection** to each other.

Security & Reliability

- Applied to both computers and computer networks.
  - The field covers all the **processes** and **mechanisms** by which computer-based ***equipment***, ***information*** and ***services*** are **protected** from **unintended** or **unauthorized access**, **change** or **destruction**.
- Additional OS support to limit the potential damage caused by **malicious** or **poorly written applications** and **drivers**.

Quick quiz

- Join the ‘Socrative’ app ‘Room 642124’ and try the quick quiz.

Further Reading

- Wikipedia:
  - Operating Systems, memory management, paging, virtual memory, process management, Unix, Linux, etc.
- PCH
  - Chapter 13
- Modern Operating Systems by Andrew Tanenbaum

Required Reading For Next Week

- HCW:
  - How the Processor Uses Registers (p66-67)
  - How Multi-core processors work (p72-73)
- Online Microprocessor tutorial at:
  - http://www.eastaughs.fsnet.co.uk/cpu/index.htm
  - Read: Introduction, CPU Structure (all), Instruction Execution (Introduction, Instruction Sets, Execution Cycle)
- Additional material/links may be posted on Blackboard: Please check!

Image Credits:

Title Image – CC-BY-SA Daniel Livingstone, 2011

Operating System placement diagram CC-BY-SA Golftheman, http://en.wikipedia.org/wiki/File:Operating_system_placement.svg

Kubuntu screenshot is free software; you can redistribute it and/or modify it under the terms of the **GNU General Public License** as published by the Free Software Foundation; either version 2 of the License, or any later version. This work is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of **merchantability** or **fitness for a particular purpose**. See version 2 and version 3 of the GNU General Public License for more details

Simplified UNIX family tree, Public Domain, from http://en.wikipedia.org/wiki/File:Unix_history.svg

GNU Gnu by Aurelio Heckert. Copyleft: This work of art is free; you can redistribute it and/or modify it according to terms of the **Free Art License**. You will find a specimen of this license on the Copyleft Attitude site as well as on other sites

Tux Penguin: The original Tux, the official Linux mascot created by Larry Ewing [1] in 1996: *Permission to use and/or modify this image is granted provided you acknowledge me lewing@isc.tamu.edu and The GIMP if someone asks.*

Kernel schematic, CC-BY-SA Bobbo, http://en.wikipedia.org/wiki/File:Kernel_Layout.svg

Monolithic vs Micro Kernel schematic, Public Domain by Wooptoo http://en.wikipedia.org/wiki/File:OS-structure.svg

Windows task manager is (c) Microsoft. User here under fair use provisions to illustrate range of processes on a modern Windows PC. Screenshot taken by Daniel Livingstone.

Virtual Memory schematic CC-BY-SA Ehamberg, http://en.wikipedia.org/wiki/File:Virtual_memory.svg

Page Table diagram, Copyright © 2011 en:User:Dysprosia: Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of en:User:Dysprosia nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. http://en.wikipedia.org/wiki/File:Virtual_address_space_and_physical_address_space_relationship.svg