Murdoch
UNIVERSITY

Topic 5: Systems Software

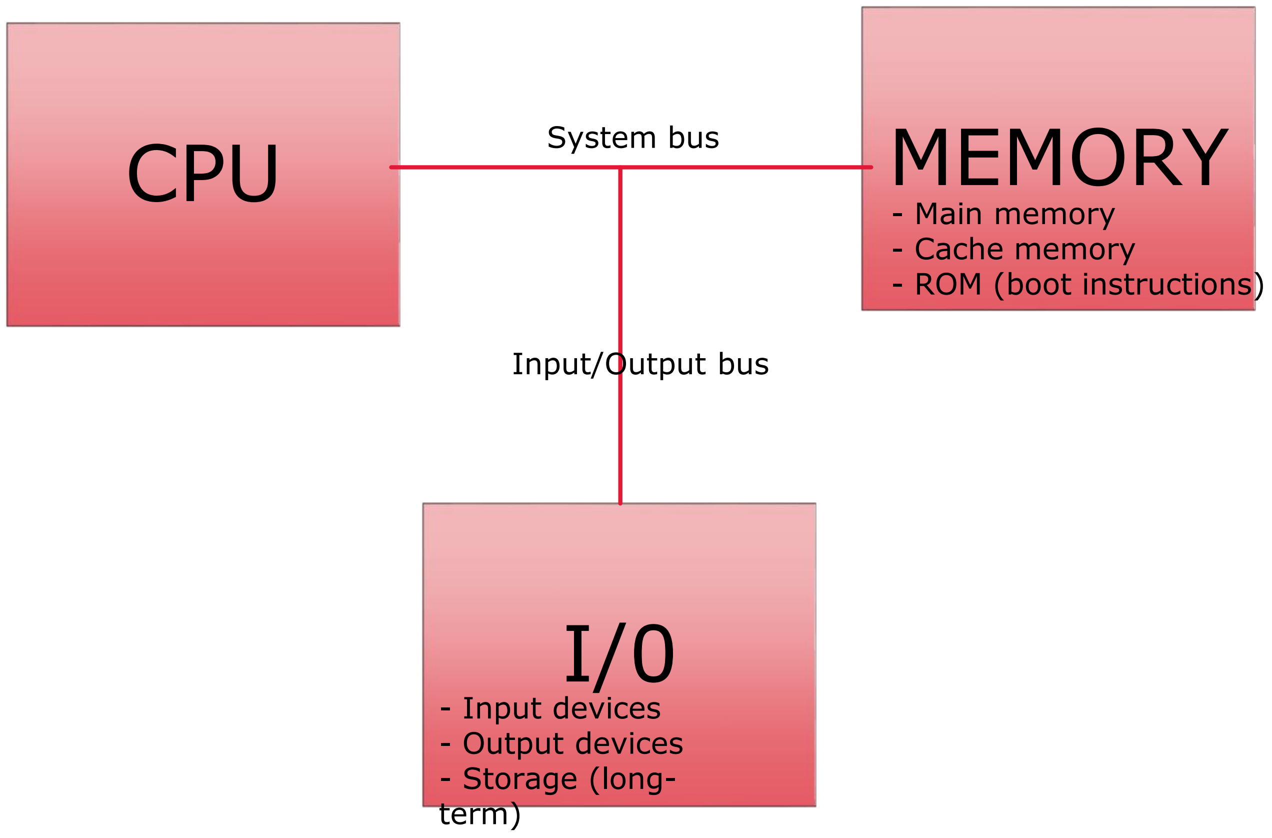
ICT170: Foundations of Computer Systems

Recap: Last Week

Topics:

- Memory
- Primary Memory
- Secondary Memory
- Secondary Memory Storage Technologies
- Input
- Output
- Buses

Recap: Up to now!



System Software

Overview

- The Operating System Level
- Operating System Services
- File System Fundamentals
- File System Types
- Process Management
- CPU Scheduling
- Memory Management
- Example Operating Systems

Objectives

In order to achieve the unit learning objectives, on successful completion of this topic, you should be able to:

Describe how computing systems are constructed of layers upon layers, based on separation of concerns, with well-defined interfaces, hiding details of low layers from the higher layers. This can be motivated by real-world systems, like how a car works, or libraries.

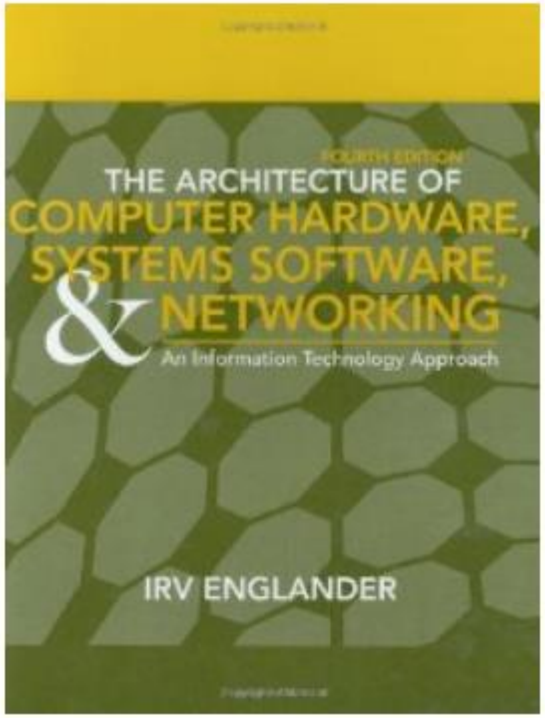
Recognize that hardware, VM, OS, application are additional layers of interpretation/processing.

Describe the mechanisms of how errors are detected, signalled back, and handled through the layers.

Reading

The required reading for this topic is in 'Topic 5' of the reader. The resource is:

Title: **The Architecture Of Computer Hardware, Systems Software, & Networking: An Information Technology Approach (4th Edition)**
Authors: Irv Englander
Publisher: Wiley
Keywords: networking, software, systems, computer, hardware, architecture
Pages: 704
Published: 2009-05-04
Language: English
ISBN-10: 0471715425 ISBN-13: 9780471715429
Binding: Hardcover (4)
Reading: Chapter 15 "Operating Systems: An Overview"

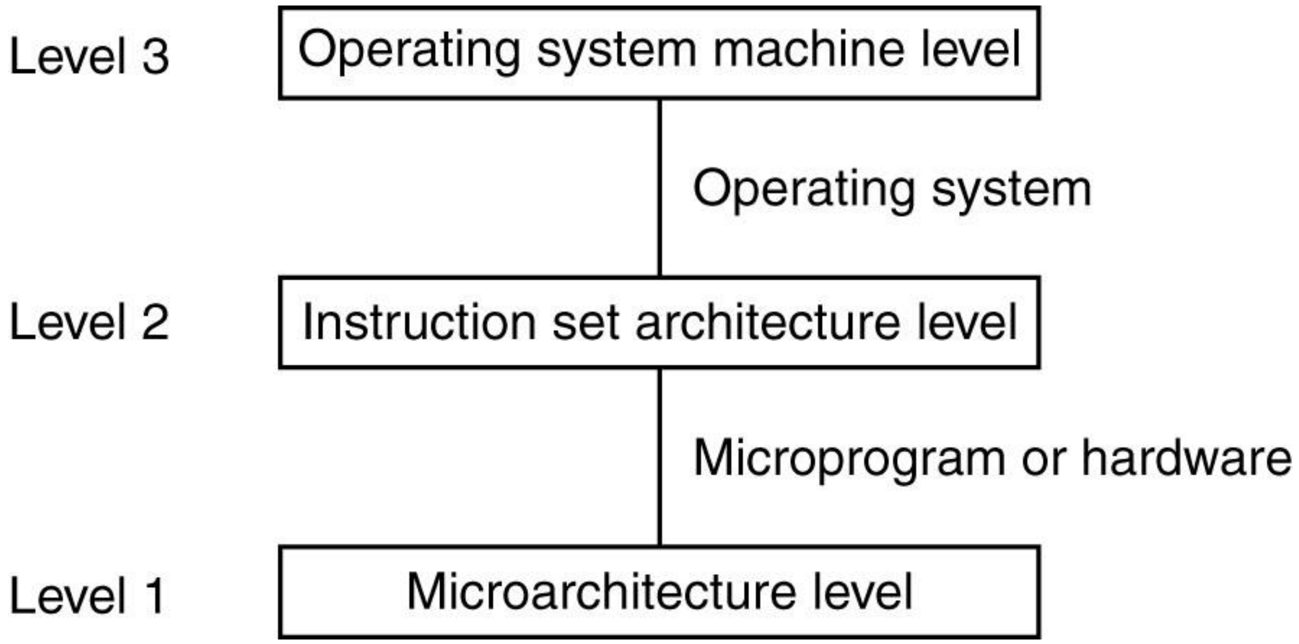

FOURTH EDITION
THE ARCHITECTURE OF
COMPUTER HARDWARE, SYSTEMS SOFTWARE, & NETWORKING
An Information Technology Approach
IRV ENGLANDER

The Operating System Level

The Operating System Level

- We’ve seen the digital logic level – that forms the basis of the computer
- We’ve seen the microarchitecture level which provides access to the digital logic level
- We’ve seen the Instruction Set Architecture level which provides a software/hardware interface which the compiler writes to
- Now we will move up another level to the *Operating System Level*

Operating System Machine



Positioning of the operating system machine level.

Systems Software

Systems software

- Consists of all the programs that enable the computer and its peripheral devices to function smoothly
- Divided into two main categories:
  - The Operating System
  - System utilities (utility programs)

Operating system (OS)

▪Set of programs that coordinate:

- Interactions of hardware components to each other
- Interaction between application software & computer hardware
- Mostly stored on a hard disk or memory chip in handheld PCs
- Usually designed to work with a specific hardware e.g., PC, iPhone

The Operating System Level

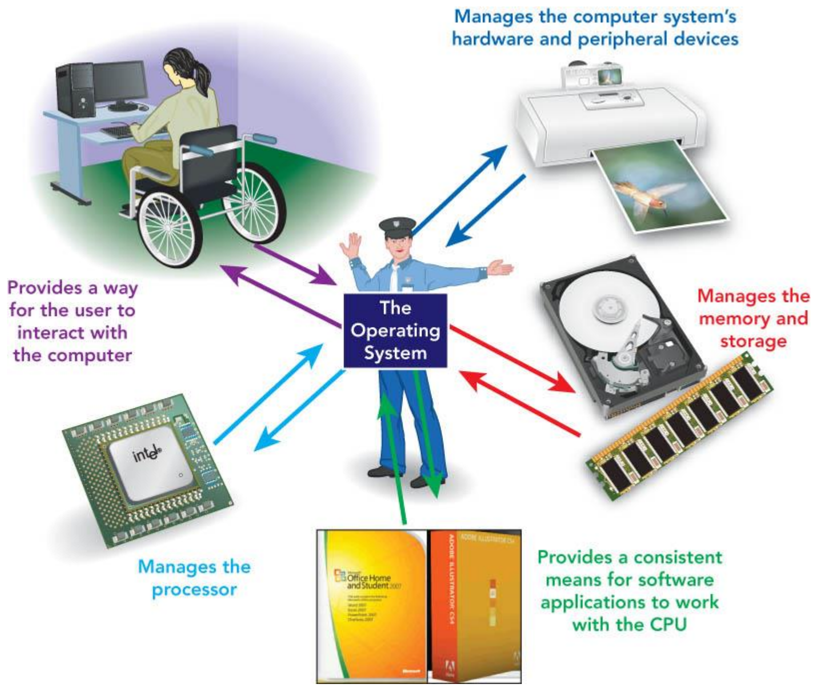
Definition

An Operating System (OS) consists of a set of sophisticated, coordinated and cooperative programs which makes the hardware useful by providing an ***interface*** to the user and other programs which require the use of the computer's resources.

The Operating System

Basic functions

- Starts the computer
- Handles input and output device messages
- Manages applications
- Manages memory
- Provides a user interface for communication

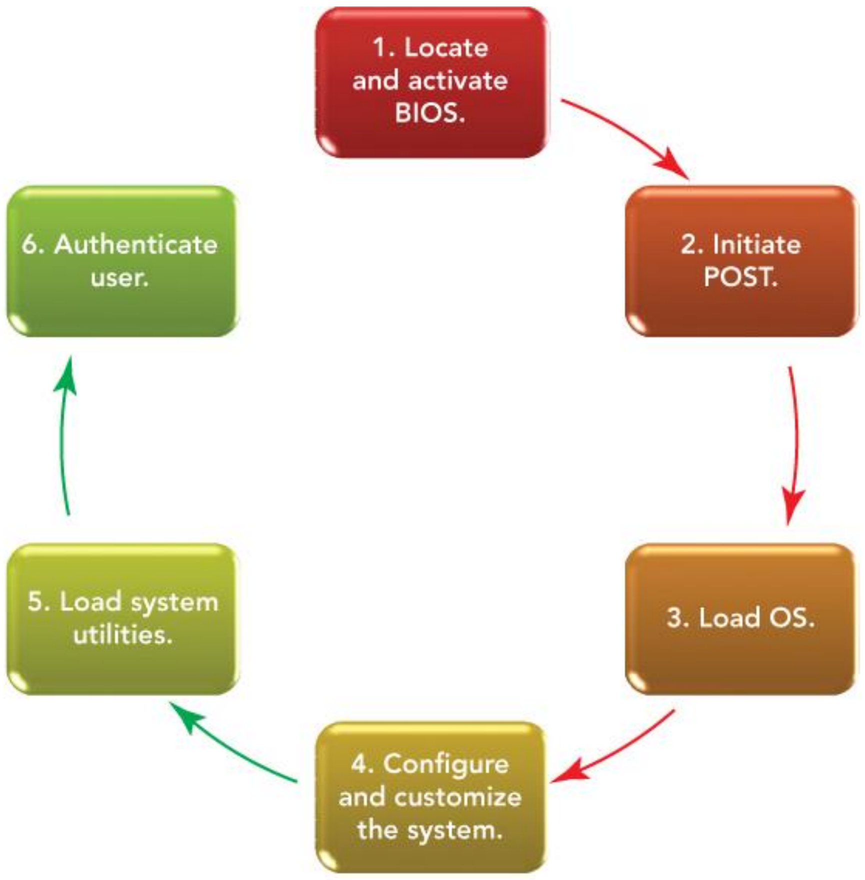

Manages the computer system's hardware and peripheral devices
Provides a way for the user to interact with the computer
The Operating System
Manages the memory and storage
Manages the processor
Provides a consistent means for software applications to work with the CPU

The Operating System

System Startup

BIOS = Basic Input/Output System

POST = Power-on Self-Test


1. Locate and activate BIOS.
2. Initiate POST.
3. Load OS.
4. Configure and customize the system.
5. Load system utilities.
6. Authenticate user.

OS required?

- Not all processors need an OS.
- For example: dedicated embedded applications.
- In such applications, a program in Non-volatile memory controls the system.
- Examples:
  - a car's engine management system, climate control.
  - All microprocessor controlled home electronic devices
  - Bank ATM’s
- Embedded systems *are not designed to do* general purpose computing!

Operating System Services

Operating System Services

The services provided by an operating system (OS) can be broken down into the following basic and fundamental functions:

- User interface and User Management
- I/O Management
- File System Management
- Process Management
- Memory Management
- CPU Management

Operating System Services

(1) User interface and User Management

- GUI (Quartz/X11/MS windows) or CLI (terminal).
- login/logout
- accounting info
- access control
- account maintenance

Operating System Services

(2) I/O Management

- I/O services
- Device management
- “Every day common services”
- Typical examples are
  - Keyboard
  - Mouse
  - Window manager I/O support

Operating System Services

(3) File System management

- Providing an abstract view of secondary storage to the user.
- Functions include
    - device maintenance,
    - volume maintenance,
    - creation/deletion,
    - backup/recovery

Operating System Services

(4) Process Management

- The running of programs (note: the OS itself is a program).
- Process creation/deletion
- Process loading
- Inter-process communication

(5) Memory Management

- Very important task of OS
- Even the most simple OS has to perform memory management.
- Involves memory allocation, swapping, address translation.

Operating System Services

(6) CPU Management

- Interrupt handling
- CPU scheduling

File System Fundamentals

Disk Basics

- A single disk has two sides (surfaces).
- Each side is divided into **tracks**.
- Tracks are numbered: 0 (outermost), 1(the next one), 2, and so on.
- A track is divided into **sectors**.
- Each disk within a hard drive is called a platter.

Accessing a Disk

- Sectors on a disc are accessed by specifying platter, side, track and sector within track.
- However, it is easier to deal with *logical* sectors instead: all sectors on a drive are aggregated into a single numbering system: 0, 1, 2, 3....
- A sector is the smallest block that can be read/written.
- However most OS can read/write larger blocks.

The Boot Sector

- All bootable disks have a boot sector.
- It is expected to be found in sector 0.
- It contains a data section and a boot loader program.
- The data stored includes information about the sector and cluster sizes of the disk.
- The boot loader loads the operating system.
- File systems ‘know’ that the boot sector cannot be used.

Files

- Files are an abstraction provided to the user by an OS.
- A file is a named collection of bytes on secondary storage.
- They are composed of *data clusters.*
- The data clusters may not be contiguous.
- The bytes are translated by a program to do what they are meant to do.
- For example, a text file can be converted to readable text using a program – emacs, notepad etc.– that can translate the ASCII to screen characters.
- A binary file (non ASCII) will require the appropriate program to translate it to something relevant to the program.
- The file extension gives the OS information about what program to use.

File System Definition

- Concerns with:
  - storage,
  - hierarchical organisation,
  - manipulation,
  - navigation,
  - access,
  - and retrieval of files.
- The file system is the part of the OS which manages files stored on secondary memory devices.
- It provides device *independence.*

File System File Functions (1/2)

- Creating a file
  - find space on the device
  - add an entry into the directory
- Writing
  - use a system call to specify the file and the data to be written
  - find space on the device
  - maintain a write pointer into the file
- Reading
  - use a system call to specify the file and the data to be read
  - maintain a read pointer into the file

File System File Functions (2/2)

- Deleting
  - search the directory index for the file
  - remove the entry
  - update the free space information
- Truncating (to shorten, or lengthen)
  - search the directory index for the file
  - change the entry
  - update the free space information
- Seeking (within a file)
  - maintain a seek pointer within the file

File System Types

- OS dependent file system types:
  - FATnn (File Allocation Table): MS-DOS and early Windows
  - NTFS: Windows NT onwards
  - MFS: Early MAC file system
  - HFS: High speed MAC file system
  - EXTn: Linux file systems

- Hardware dependent file systems:
  - ISO9660: International Standards Organisation standard for CD-ROM
  - ISO/IEC 13346: ISO standard for optical disks
  - JFFS2/YAFFS: Flash memory file systems

Variations Between File Systems

File systems vary in terms of:

- maximum volume (disk) size
- maximum file size
- maximum file name size
- compression/encryption availability
- access checking
- directory contents (file information is stored)
- directory structure (how it is stored)
- file allocation
- free space tracking

File Information

- All file systems must store data about the files.
- This *metadata* includes:
  - name;
  - descriptor (unique ID);
  - location(s) on disk (addresses);
  - size;
  - attributes (e.g. read only);
  - permissions (some FS only);
  - date and time file created, last modified etc;
  - compression/encryption information.
  - system file type: directory, link, data etc (some FS only).

Directory Implementation

- There are many ways in which file information can be stored within the directory file.
- Three common methods are:
  - Table (FAT)
  - B+ Trees (NTFS) (Can have many children per node)
  - B* Trees (EXT) (includes child redistribution)
- Tree is an important data structure and will be covered in other units.

File Allocation Types (1/4)

- There are three main types of disk file allocation:
  - Contiguous (IBM VM/CMS)
  - Linked (FAT)
  - Indexed (EXT2 and NTFS)

File Allocation Types (2/4)

(1) Contiguous File Allocation

- Early method.
- The simplest method.
- Each file placed in contiguous clusters.
- This means whole file needs to be moved if it becomes too big for the space.
- A large enough space has to be found for the whole file.
- This has a performance impact.

File Allocation Types (3/4)

(2) Linked File Allocation

- Each file block links to the next file block.
- Sequential file access is still fast.
- Non-sequential becomes slow.
- Files become fragmented which slows down access.
- To improve access time, the list of links can be stored in a table: hence the FAT (File Allocation Table) disk system.

File Allocation Types (4/4)

(3) Indexed File Allocation

- Files do not have to be contiguous.
- A link block keeps track of where the file parts are stored.
- Multiple link blocks can be used if the file is large.

Managing Free Space

The are four main methods by which free space is tracked:

1. Bit Vector (ext2)
   - Free space on the disk is tracked by a bit-vector
2. Linked List (FAT)
   - Inverse of the linked allocation method
   - in this case we have a linked list of the free space
3. Grouping
   - Inverse of the index allocation method
   - The first free block holds the addresses of subsequent free blocks.
4. Counting (NTFS)
   - Keeps a list of the contiguous blocks of free space
     - e.g Block 20, 3000: means that starting from block 20, there are 3000 blocks of free space
   - Leverages the fact that space is generally assigned or freed in large blocks

Disk Problems

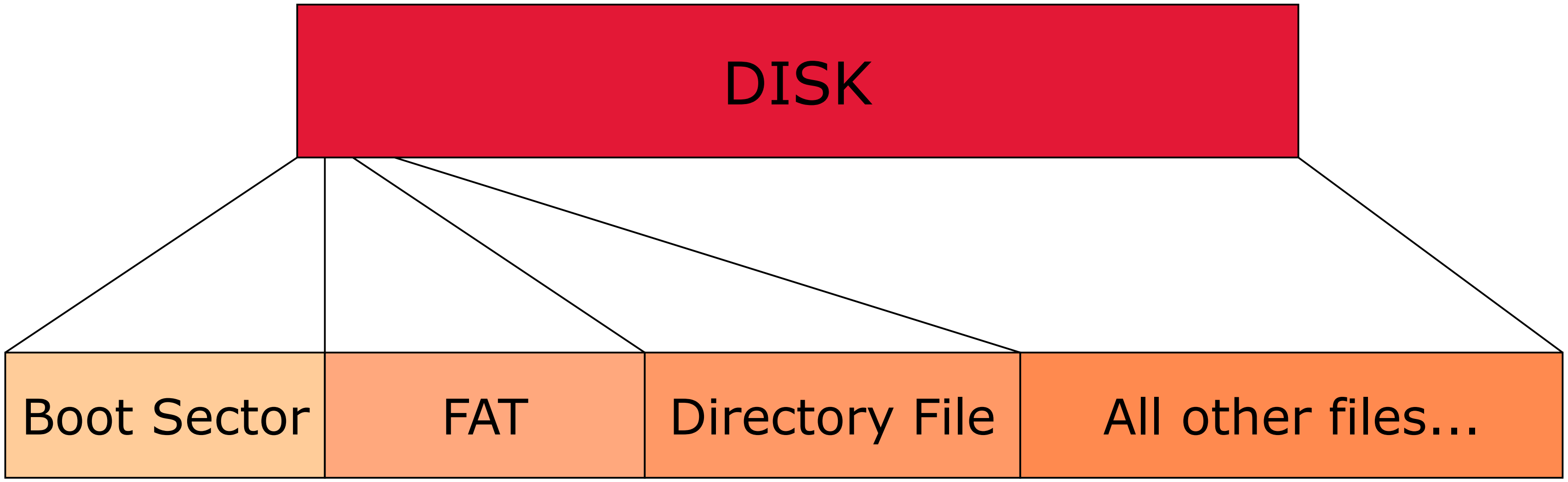
- There is no guarantee that a file is correct or that the directory structure matches that stored in the directory information.
- Program/computer crashes and bad sectors cause problems.
- Tools such as fsck/chkdisk helps to check the consistency of the claimed structure with the actual structure.
- These can fix the directory information, but cannot recover lost information.

File System Types

The FAT File System

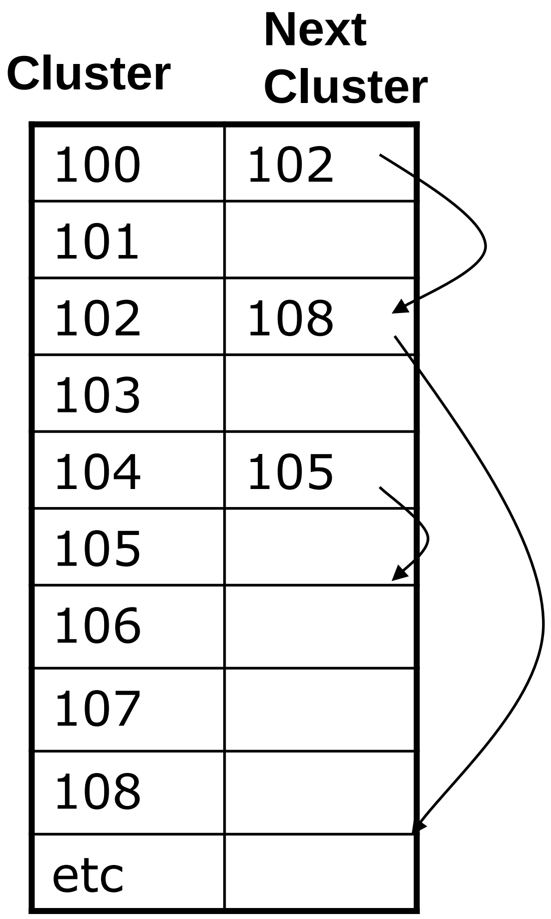
- Used within MS-DOS and early Windows
- There are two parts to the FAT file system information:
  - The FAT file itself
  - The directory index file
- These are found directly after the boot sector on the disk.

The FAT File System



The FAT

- The File Allocation Table mimics the disk structure.
- Originally designed for floppy disks.
- It has a fixed size (64 K entries).
- Therefore the number of clusters within the structure must remain the same.
- This means that the larger the disk, the larger the size of the file cluster.
- This leads to inefficiencies.
- The different clusters used for a file are linked using ‘pointers’.

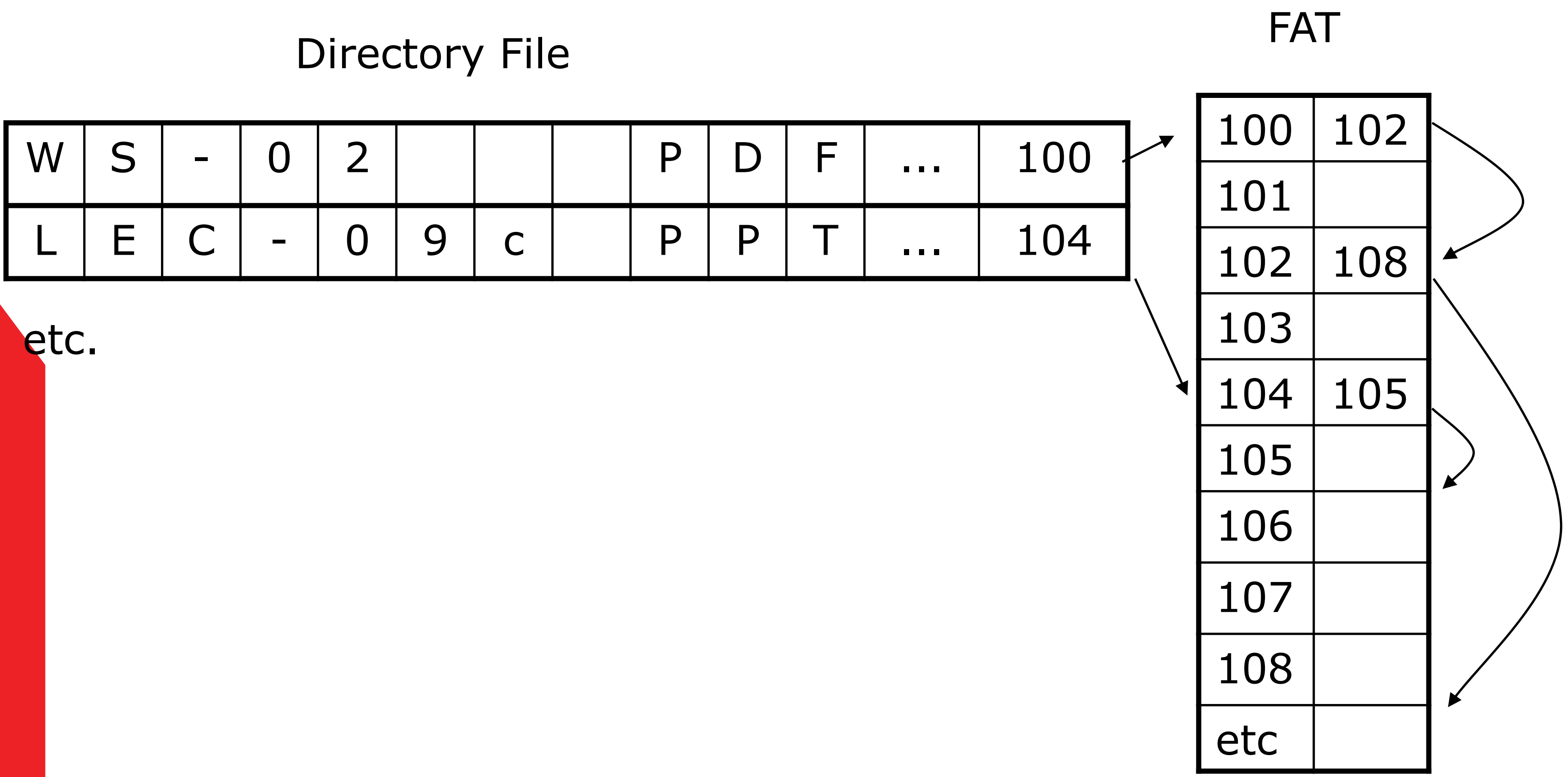


Cluster	Next Cluster
100	102
101	
102	108
103	
104	105
105	
106	
107	
108	
etc	

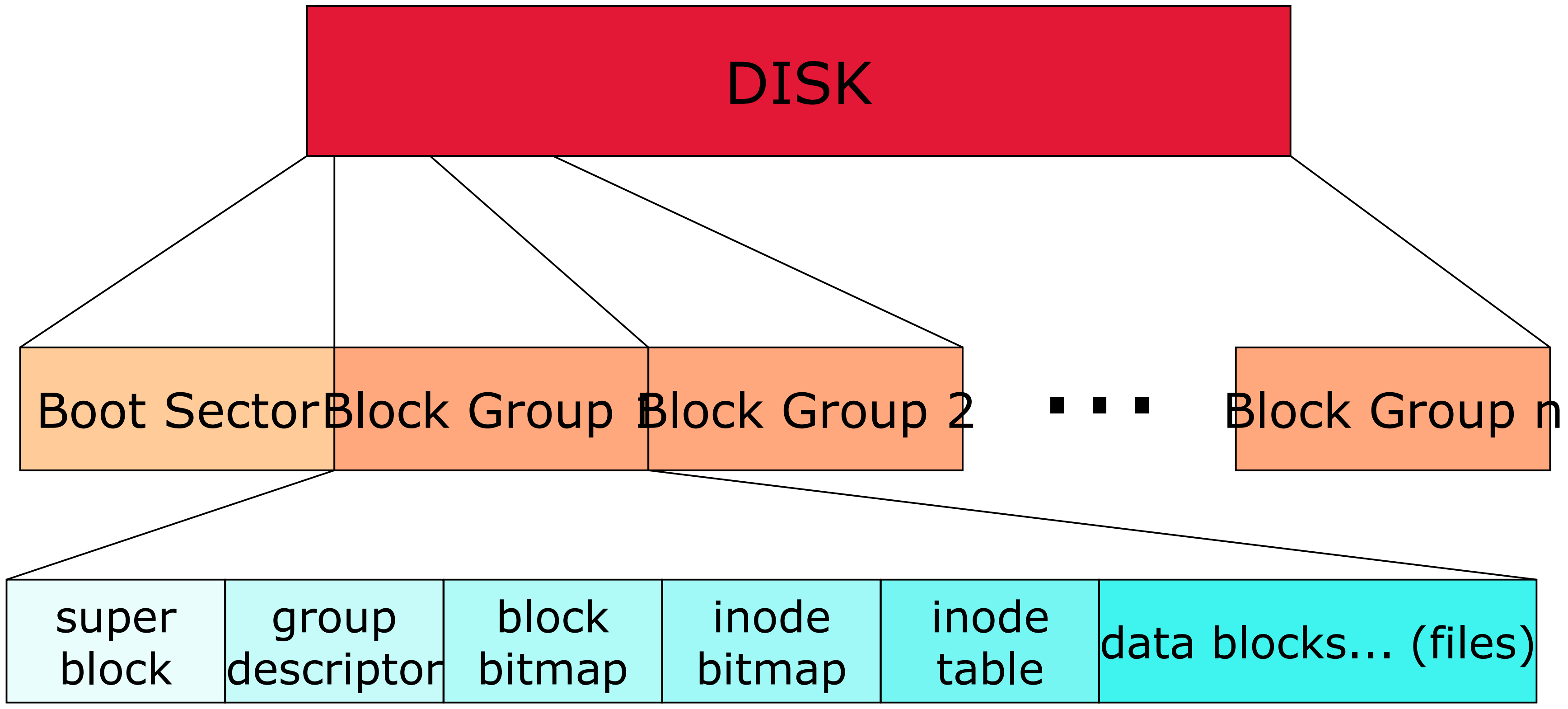
The FAT Directory Structure

- The directory information is stored in a simple file.
- The file is a simple table.
- This makes it slow to search, add to, delete from etc.
- Each file has one entry in the directory table.
- Each entry is 32 bytes:
  - 0x00..0x07: Filename (8 chars/bytes)
  - 0x08..0x0a: Extension (3 chars/bytes)
  - 0x0b..0x10: Attributes: read-only, hidden, system file, subdirectory, archive, dates, etc (19 bytes)
  - 0x1a..0x1b: First cluster: an entry point into the FAT (2 bytes).

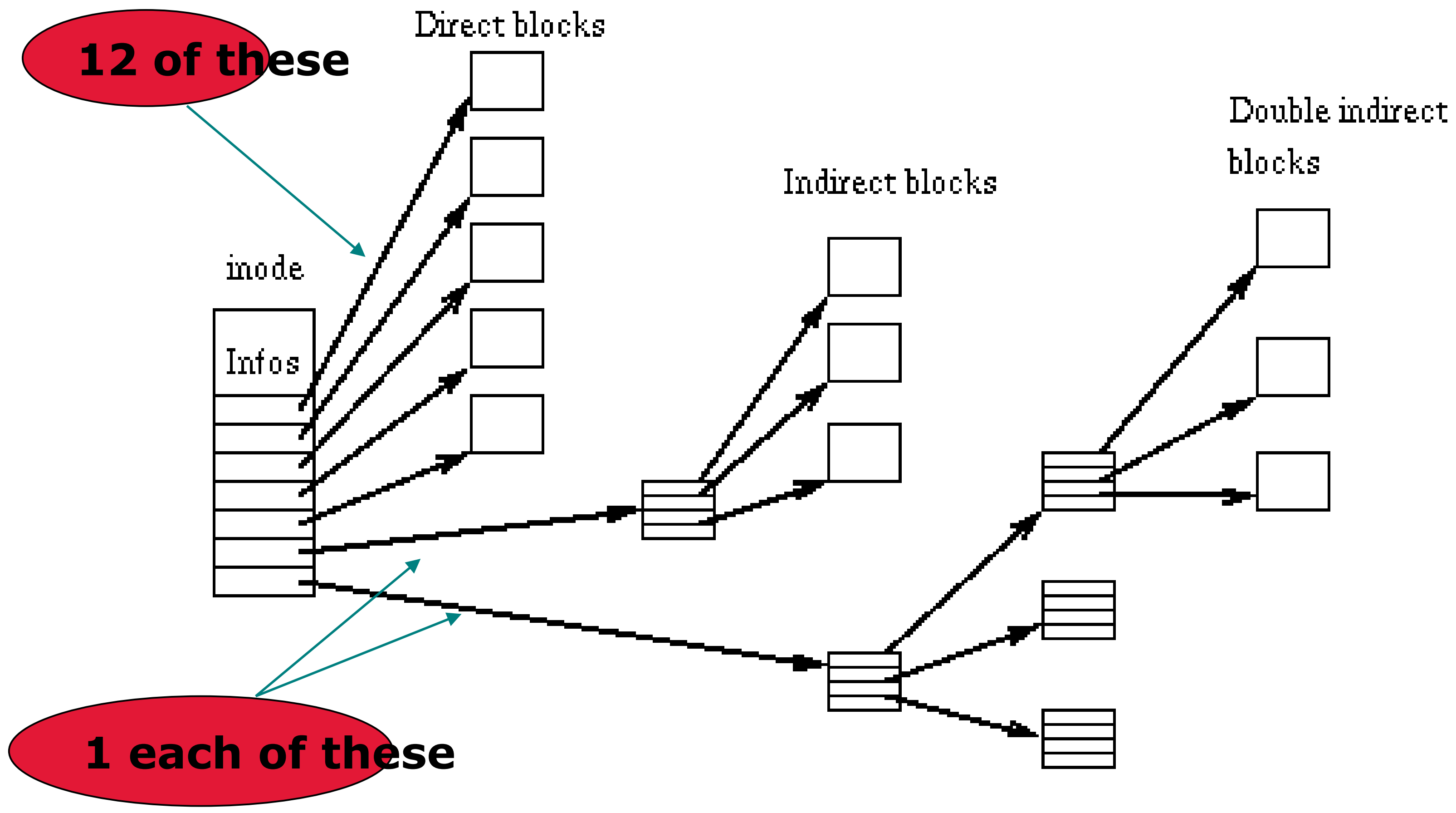
The FAT Directory Structure



The ext2 File System



inode links
(http://e2fsprogs.sourceforge.net/ext2intro.html)



Ext3, third extended filesystem

- The same as ext2, but with the addition of *journalling.*
- All writes are preceded and followed with an entry into a log.
- In the event of a disk crash, any incomplete writes can be rolled back.

NTFS

- Windows NT, Windows XP, Windows Servers 2003, 2008, Vista and Windows 7, Released 1993
- Similar to ext3, but with
  - file system encryption
  - file system compression
  - access control lists
- Unlike FAT it allows hard and symbolic links, however Explorer does not handle these!
- Unlike FAT and ext2/3 it uses a B+ tree to store the directory information: this allows faster searching.

Flash File Systems

- Flash drives do not use the same file systems as hard drives because:
  - they suffer from *wear;*
  - the seek time is fast compared to a hard drive;
  - the erase time is slow compared to a hard drive.
- Each segment of a flash drive can only be used/erased a limited number of times.
- Therefore a file system is required that does *wear levelling*.
- One of the most common methods is the log-structured file system.
  - The storage is considered a circular log.
  - Writes always occur at one end of the log.
  - Free space is then reclaimed at the end of the log.

Murdoch
UNIVERSITY

Process Management

Multi-Tasking Rationale

- Why do we want to run multiple programs concurrently?
  - To make as efficient use of the computer resources (typically the CPU) as possible. The CPU should not be kept running idle waiting for something to happen.
  - User programs may not be running all the time. Such processes could be waiting for I/O - waiting for the user to press a key.
  - Think of a word processor. Almost all of the word-processor’s time is spent waiting for a key to be pressed.

Multi-Tasking Rationale

Does the CPU actually stop running?

- No: the CPU goes through the fetch/execute cycle (FEC) from power up – i.e. it runs something all the time.
- If there are no user programs to run, then maintenance tasks such as indexing may be running.
- If you look at the Task Manager in Windows, and sort under CPU, then you will see that the process using most of the CPU is usually the "System Idle Process"!

Task Manager: Processes

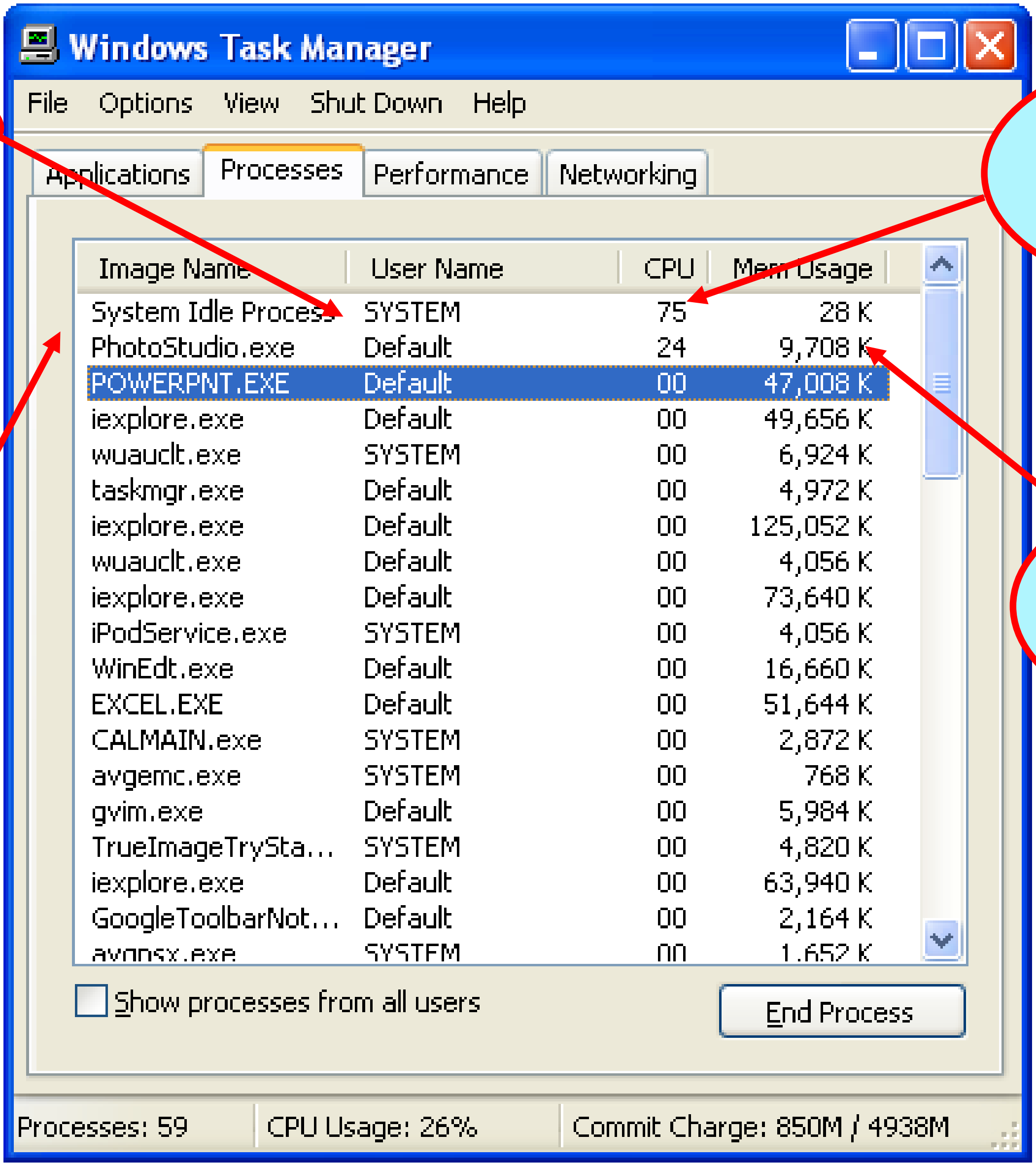
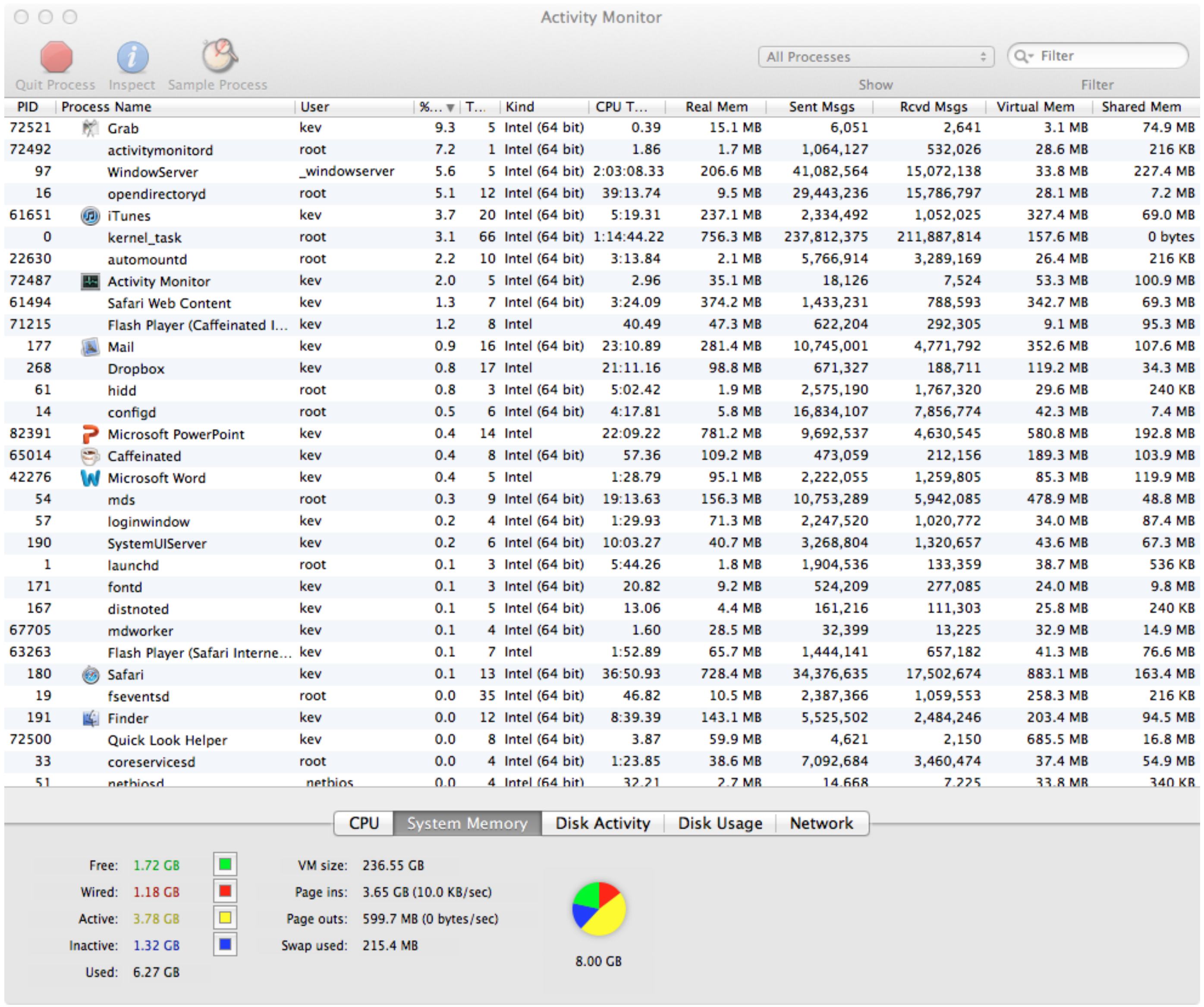

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking
Show processes from all users
End Process
The user running that process
The current amount of CPU usage for that process
The process name
The current amount of RAM being used by that process

Image Name	User Name	CPU	Mem Usage
System Idle Process	SYSTEM	75	28 K
PhotoStudio.exe	Default	24	9,708 K
POWERPNT.EXE	Default	00	47,008 K
iexplore.exe	Default	00	49,656 K
wuauclt.exe	SYSTEM	00	6,924 K
taskmgr.exe	Default	00	4,972 K
iexplore.exe	Default	00	125,052 K
wuauclt.exe	Default	00	4,056 K
iexplore.exe	Default	00	73,640 K
iPodService.exe	SYSTEM	00	4,056 K
WinEdt.exe	Default	00	16,660 K
EXCEL.EXE	Default	00	51,644 K
CALMAIN.exe	SYSTEM	00	2,872 K
avgemc.exe	SYSTEM	00	768 K
gvim.exe	Default	00	5,984 K
TrueImageTrySta...	SYSTEM	00	4,820 K
iexplore.exe	Default	00	63,940 K
GoogleToolbarNot...	Default	00	2,164 K
avgnsx.exe	SYSTEM	00	1,652 K

Processes: 59 | CPU Usage: 26% | Commit Charge: 850M / 4938M

Murdoch UNIVERSITY

Activity Monitor



PID	Process Name	User	%...	T...	Kind	CPU T...	Real Mem	Sent Msgs	Rcvd Msgs	Virtual Mem	Shared Mem
72521	Grab	kev	9.3	5	Intel (64 bit)	0.39	15.1 MB	6,051	2,641	3.1 MB	74.9 MB
72492	activitymonitord	root	7.2	1	Intel (64 bit)	1.86	1.7 MB	1,064,127	532,026	28.6 MB	216 KB
97	WindowServer	_windowserver	5.6	5	Intel (64 bit)	2:03:08.33	206.6 MB	41,082,564	15,072,138	33.8 MB	227.4 MB
16	opendirectoryd	root	5.1	12	Intel (64 bit)	39:13.74	9.5 MB	29,443,236	15,786,797	28.1 MB	7.2 MB
61651	iTunes	kev	3.7	20	Intel (64 bit)	5:19.31	237.1 MB	2,334,492	1,052,025	327.4 MB	69.0 MB
0	kernel_task	root	3.1	66	Intel (64 bit)	1:14:44.22	756.3 MB	237,812,375	211,887,814	157.6 MB	0 bytes
22630	automountd	root	2.2	10	Intel (64 bit)	3:13.84	2.1 MB	5,766,914	3,289,169	26.4 MB	216 KB
72487	Activity Monitor	kev	2.0	5	Intel (64 bit)	2.96	35.1 MB	18,126	7,524	53.3 MB	100.9 MB
61494	Safari Web Content	kev	1.3	7	Intel (64 bit)	3:24.09	374.2 MB	1,433,231	788,593	342.7 MB	69.3 MB
71215	Flash Player (Caffeinated I...	kev	1.2	8	Intel	40.49	47.3 MB	622,204	292,305	9.1 MB	95.3 MB
177	Mail	kev	0.9	16	Intel (64 bit)	23:10.89	281.4 MB	10,745,001	4,771,792	352.6 MB	107.6 MB
268	Dropbox	kev	0.8	17	Intel	21:11.16	98.8 MB	671,327	188,711	119.2 MB	34.3 MB
61	hidd	root	0.8	3	Intel (64 bit)	5:02.42	1.9 MB	2,575,190	1,767,320	29.6 MB	240 KB
14	configd	root	0.5	6	Intel (64 bit)	4:17.81	5.8 MB	16,834,107	7,856,774	42.3 MB	7.4 MB
82391	Microsoft PowerPoint	kev	0.4	14	Intel	22:09.22	781.2 MB	9,692,537	4,630,545	580.8 MB	192.8 MB
65014	Caffeinated	kev	0.4	8	Intel (64 bit)	57.36	109.2 MB	473,059	212,156	189.3 MB	103.9 MB
42276	Microsoft Word	kev	0.4	5	Intel	1:28.79	95.1 MB	2,222,055	1,259,805	85.3 MB	119.9 MB
54	mds	root	0.3	9	Intel (64 bit)	19:13.63	156.3 MB	10,753,289	5,942,085	478.9 MB	48.8 MB
57	loginwindow	kev	0.2	4	Intel (64 bit)	1:29.93	71.3 MB	2,247,520	1,020,772	34.0 MB	87.4 MB
190	SystemUIServer	kev	0.2	6	Intel (64 bit)	10:03.27	40.7 MB	3,268,804	1,320,657	43.6 MB	67.3 MB
1	launchd	root	0.1	3	Intel (64 bit)	5:44.26	1.8 MB	1,904,536	133,359	38.7 MB	536 KB
171	fontd	kev	0.1	3	Intel (64 bit)	20.82	9.2 MB	524,209	277,085	24.0 MB	9.8 MB
167	distnoted	kev	0.1	5	Intel (64 bit)	13.06	4.4 MB	161,216	111,303	25.8 MB	240 KB
67705	mdworker	kev	0.1	4	Intel (64 bit)	1.60	28.5 MB	32,399	13,225	32.9 MB	14.9 MB
63263	Flash Player (Safari Interne...	kev	0.1	7	Intel	1:52.89	65.7 MB	1,444,141	657,182	41.3 MB	76.6 MB
180	Safari	kev	0.1	13	Intel (64 bit)	36:50.93	728.4 MB	34,376,635	17,502,674	883.1 MB	163.4 MB
19	fseventsd	root	0.0	35	Intel (64 bit)	46.82	10.5 MB	2,387,366	1,059,553	258.3 MB	216 KB
191	Finder	kev	0.0	12	Intel (64 bit)	8:39.39	143.1 MB	5,525,502	2,484,246	203.4 MB	94.5 MB
72500	Quick Look Helper	kev	0.0	8	Intel (64 bit)	3.87	59.9 MB	4,621	2,150	685.5 MB	16.8 MB
33	coreservicesd	root	0.0	4	Intel (64 bit)	1:23.85	38.6 MB	7,092,684	3,460,474	37.4 MB	54.9 MB
51	netbiosd	_netbios	0.0	4	Intel (64 bit)	32.21	2.7 MB	14,668	7,225	33.8 MB	340 KB

CPU | System Memory | Disk Activity | Disk Usage | Network

Free:	1.72 GB	🟩	VM size:	236.55 GB
Wired:	1.18 GB	🟥	Page ins:	3.65 GB (10.0 KB/sec)
Active:	3.78 GB	🟨	Page outs:	599.7 MB (0 bytes/sec)
Inactive:	1.32 GB	🟦	Swap used:	215.4 MB
Used:	6.27 GB			

8.00 GB

Murdoch UNIVERSITY

Process Management

- It is useful to have several user programs/processes alternating in execution, e.g., background printing, clock, network popup messages etc.
- The full list of processes will include many things that are not applications that you have started explicitly started. For example drivers, virus protection etc.
- This is known as multi-tasking - it gives the appearance that a computer is running several programs simultaneously
- The CPU is still only executing one program at any one time - The CPU switches between the tasks
- The Task Manager indicates which processes are using the CPU, and what percentage of the CPU time they are using.

The Number of Processes

- There is no limit on the number of processes that can be in RAM at the same time.
- However some of the processes may slow down the running of your computer.
- Others may slow down the booting up of your computer.
- Still others may be malware!

Types of Multi-tasking

- MSDOS used a TSR (terminate and stay resident) method to emulate multi-tasking.
- Windows uses co-operative multi-tasking.
- Unix uses time-sharing.
- Of course in a dual core CPU you do actually have two processors working simultaneously and hence true multi-tasking!
  - A quad has four processors (etc.)

Processes (tasks)

- Fundamental to the concept of multi-tasking.
- An *instance* of execution of a program - each instance is a process.
- Executing Code + Data constitutes a Process
- In unix (including cygwin) typing “ps” will list the current processes.
- In Windows, the Task Manager (ctrl-alt-del), has a tab showing the currently running processes.

Process States

- Active (running): a process is said to be active if it currently has the CPU
- Ready: a process is said to be in the ready state if it could use a CPU if one was available
- Blocked: a process is said to be in blocked state if it is waiting for some event to happen (e.g., I/O transfer) before it can proceed

Round Robin Scheduling

- The simplest scheduling policy.
- The CPU time is divided into time-slices.
- A real-time clock generates pulses at regular and frequent intervals (typically 1/100 second).
- At each pulse, the OS performs the following steps:
  - suspends the current process
  - searches the process list for a ready process
  - runs that process for the rest of the current time-slice

Memory Management

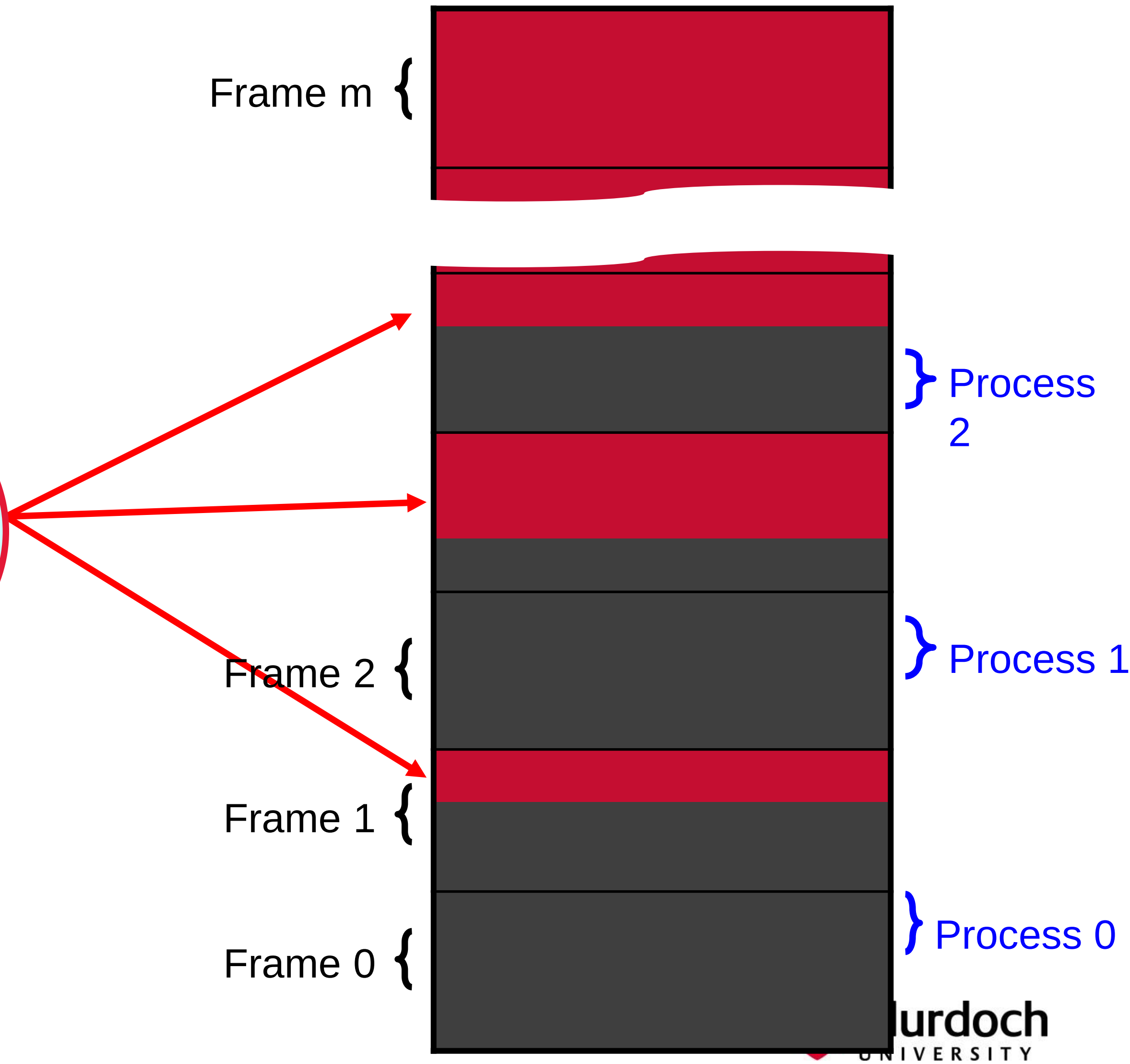
Memory Management

- Memory Management (MM) is required for the effective use of a computer even for a single tasking OS (eg. to hold the OS and user program)
- For a multi-tasking OS, MM is absolutely essential.
- Memory must be divided/shared by the resident part of the operating system.
- It is the function of the OS to perform this task of Memory Management.
- There are different ways to manage memory – as follows…

MM Using Frames

- The physical memory in RAM is divided into equal sized pages (page frames).
- When processes (running programs) are loaded into memory, they use up 1 or more frames.
- In other words the program is divided up into equal sized ‘chunks’.
- Unfortunately it is extremely unlikely that any process will have a size that is an exact multiple of the page size.
- Therefore for any process there will be a page that is partly empty.
- This is called **internal fragmentation**.

Internal Fragmentation of RAM due to Paging

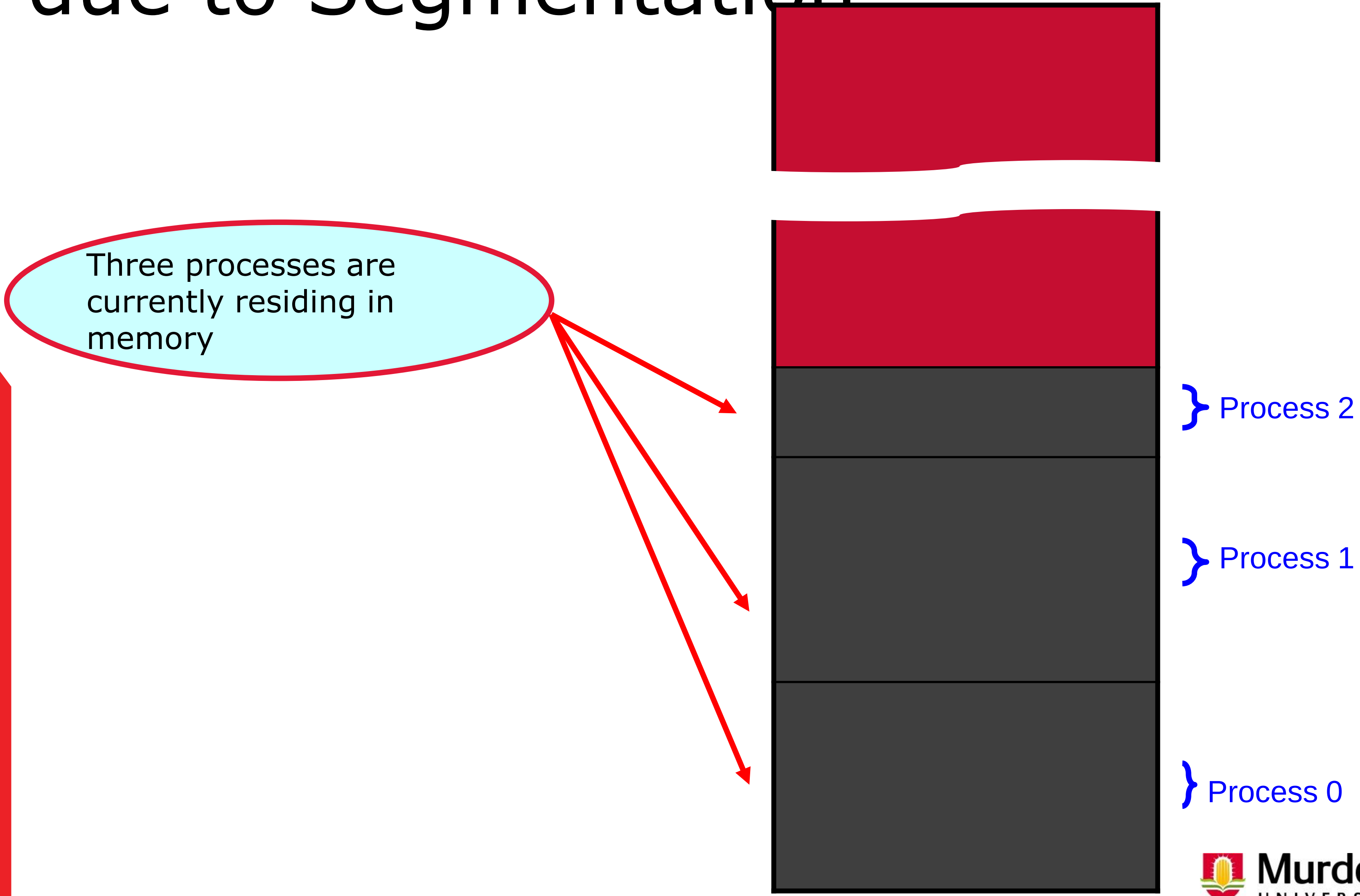


These portions of memory are lost until the process using that particular frame is completed.

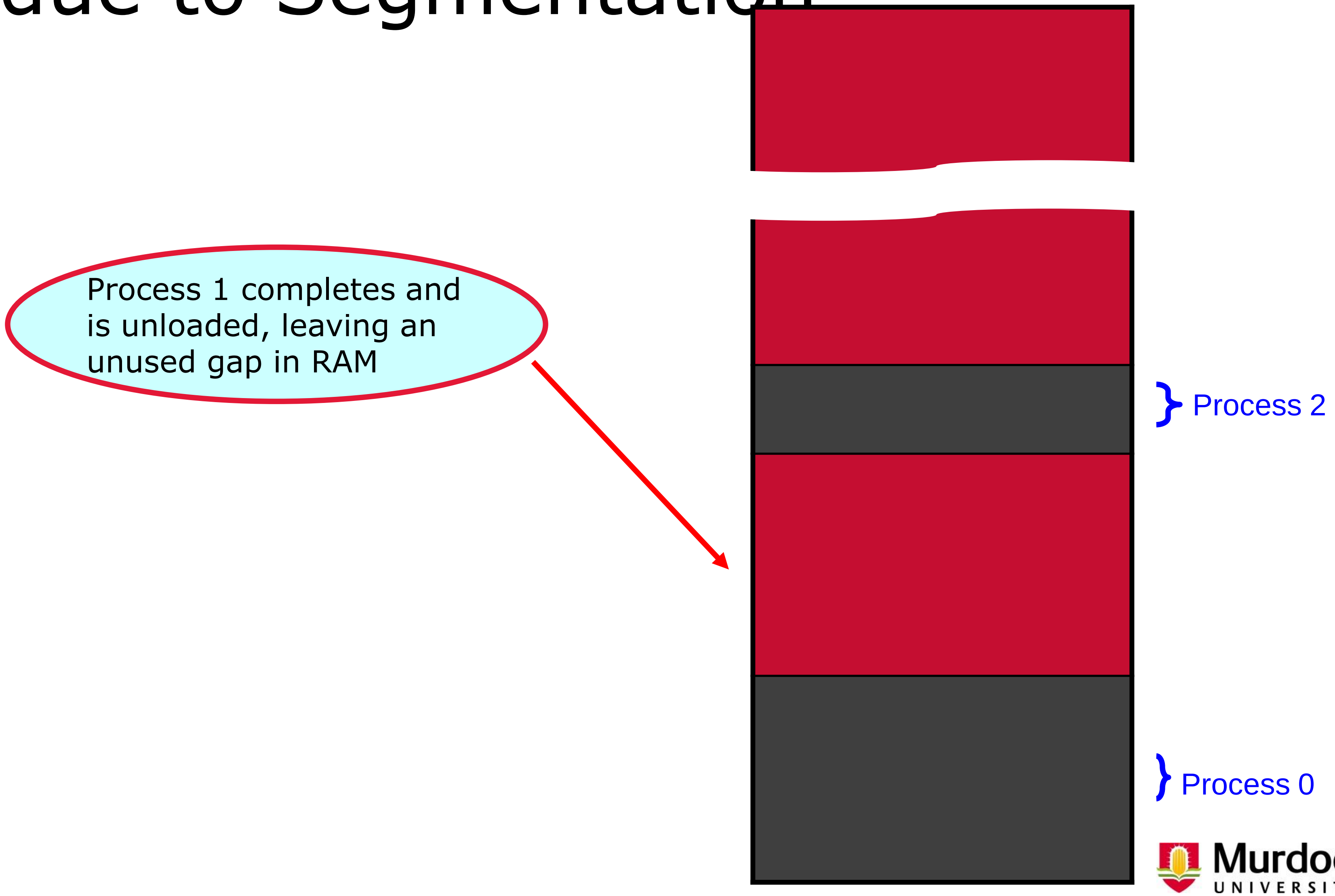
MM Using Segmentation

- The physical memory in RAM is divided into variable sized segments.
- When processes (running programs) are loaded into memory, they are divided up into variable sized *logical* segments of memory: usually for code, data, the program stack and extra data.
- This use of variable sized pieces of memory avoids internal fragmentation.
- However segmentation leads to a different problem.
- Since the space required by the last process is unlikely to be exactly the same size as the space required by the next process, unused ‘gaps’ start being left in memory.
- This is called **external fragmentation**.

External Fragmentation of RAM due to Segmentation

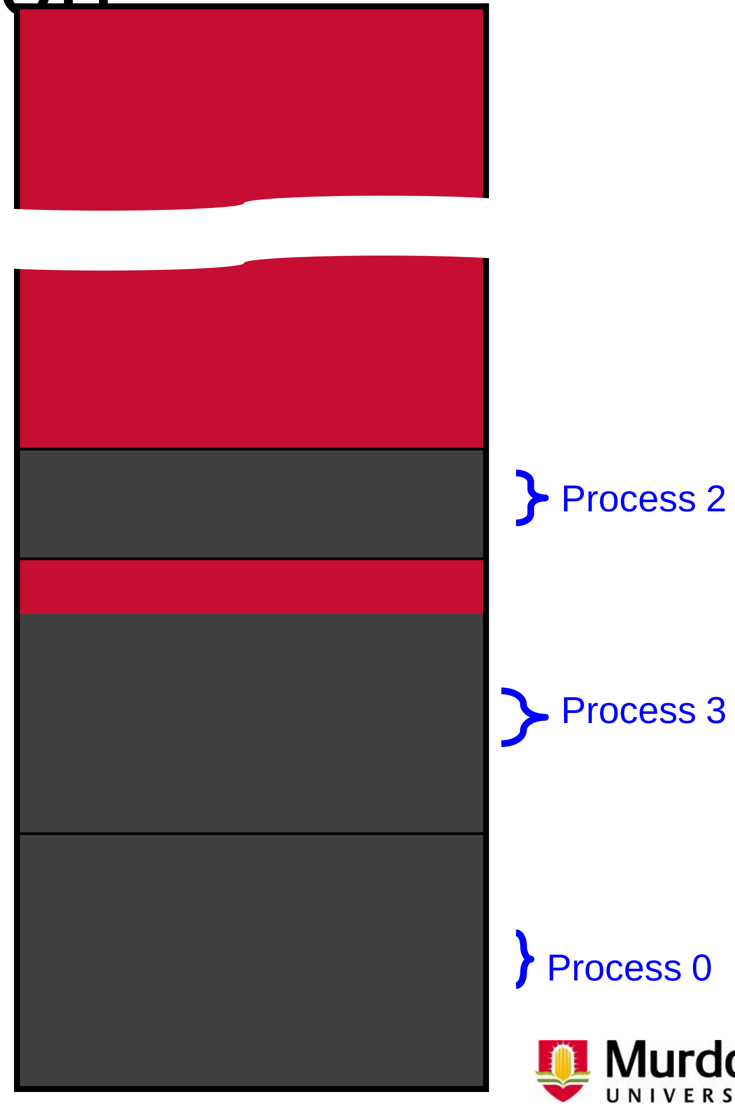


External Fragmentation of RAM due to Segmentation


Process 1 completes and is unloaded, leaving an unused gap in RAM
Process 2
Process 0

External Fragmentation of RAM due to Segmentation

Process 3 is loaded, but is smaller than Process 1 was, leaving wasted space



Virtual Memory

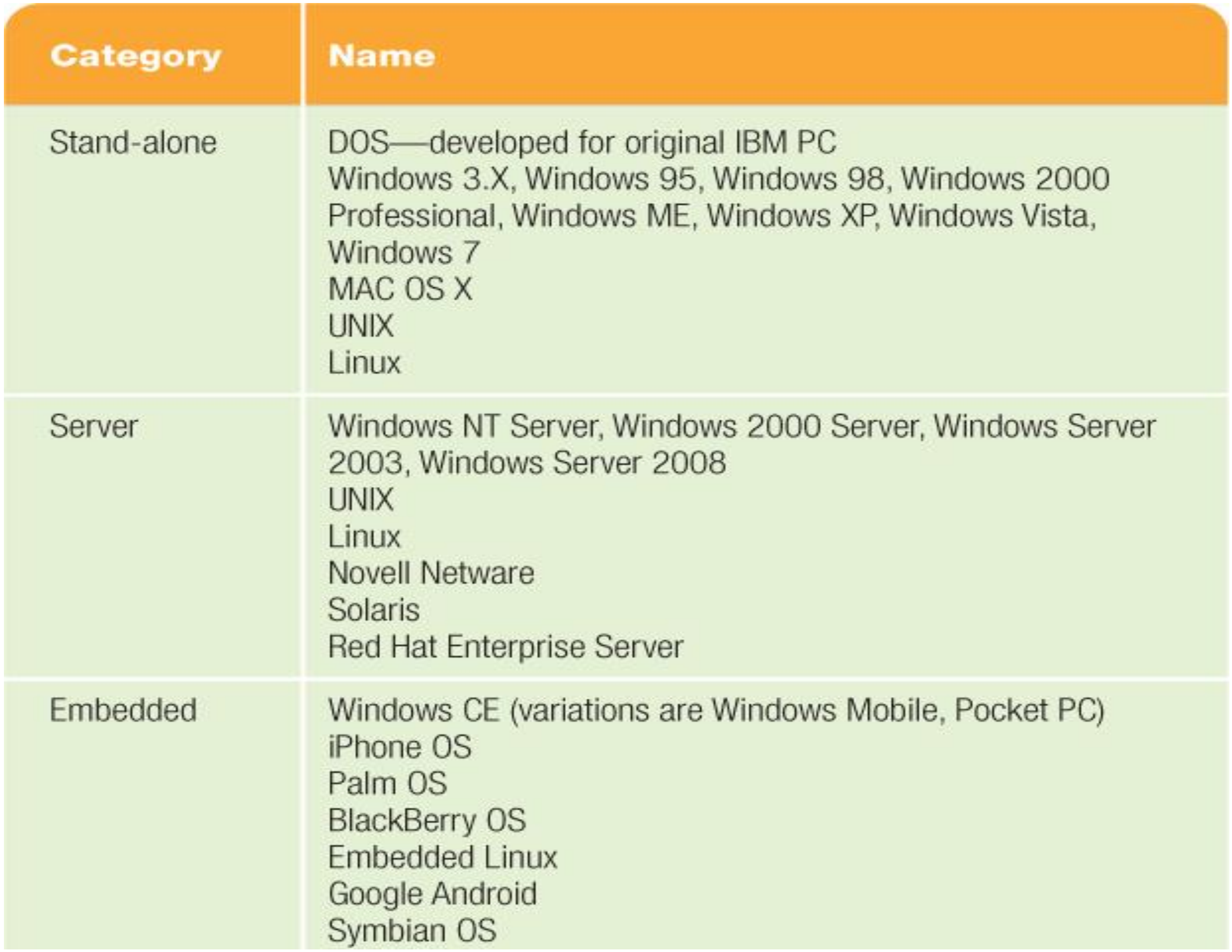
- On my desktop I generally have open around 20 programs.
- They cannot all actually fit in RAM at once.
- To enable a computer to handle more processes at once than can fit in memory, pages are created on disk that are the same size as those in RAM.
- At a specific point in time, a page (frame) may either be in RAM or on disk, they are **swapped** between the two as needed.
- A page table keeps track of which frames are currently in RAM and which are on disk.
- The memory on disk is known as **virtual memory** and resides in the **disk swap area** of the hard drive.
- The use of virtual memory should be transparent to the user.

Example Operating Systems

Exploring Popular Operating Systems

- **Three major categories of operating systems**
- **Client: Stand-alone operating systems**—used by single users – does not need to be connected to any other system to run
- **Server: Server operating systems**—used in client/server network environments
- **Embedded operating systems**—found on ROM chips in portable or dedicated devices
- Examples on next page

Exploring Popular Operating Systems



Category	Name
Stand-alone	DOS—developed for original IBM PC Windows 3.X, Windows 95, Windows 98, Windows 2000 Professional, Windows ME, Windows XP, Windows Vista, Windows 7 MAC OS X UNIX Linux
Server	Windows NT Server, Windows 2000 Server, Windows Server 2003, Windows Server 2008 UNIX Linux Novell Netware Solaris Red Hat Enterprise Server
Embedded	Windows CE (variations are Windows Mobile, Pocket PC) iPhone OS Palm OS BlackBerry OS Embedded Linux Google Android Symbian OS

Murdoch UNIVERSITY

Additional Material for Self-Learning Exercises

Assembly language: FLAGS

For signed integer comparisons, there are three *FLAGS* that are important. These are the zero (ZF) flag, the overflow (OF) flag and the sign (SF) flag

The overflow flag is set if the result of an operation overflows (or underflows)

If vleft == vright, the ZF is set (just as for unsigned integers).
If vleft > vright, ZF is unset and SF = OF
If vleft < vright, ZF is unset and SF ≠ OF.

The corresponding conditional branch instructions are as follows:

JZ	*branches only if ZF is set*
JNZ	*branches only if ZF is unset*
JO	*branches only is OF is set*
JNO	*branches only is OF is unset*
JS	*branches only is SF is set*
JNS	*branches only is SF is unset*

Assembly language: LOOP

x86 Assembly language also supports for-style loops. These are as follows, and will loop to the operand code label.

LOOP Decrements ECX, if $ECX \neq 0$, branches to label

LOOPE, LOOPZ Decrements ECX (FLAGS register is not modified), if $ECX \neq 0$ and $ZF = 1$, branches

LOOPNE, LOOPNZ Decrements ECX (FLAGS unchanged), if $ECX \neq 0$ and $ZF = 0$, branches

These are extremely powerful and very efficient. For example, the following code:

```
    sum = 0;
    for( i=10; i >0; i−− )
        sum += i;
```

Can be translated as the following assembly:

```
        mov     eax, 0          ; eax is sum
        mov     ecx, 10         ;ecx is i
loop_start:
```

Murdoch
UNIVERSITY

Summary

Summary

- The Operating System Level
- Operating System Services
- File System Fundamentals
- File System Types
- Process Management
- CPU Scheduling
- Memory Management
- Example Operating Systems

Murdoch
UNIVERSITY