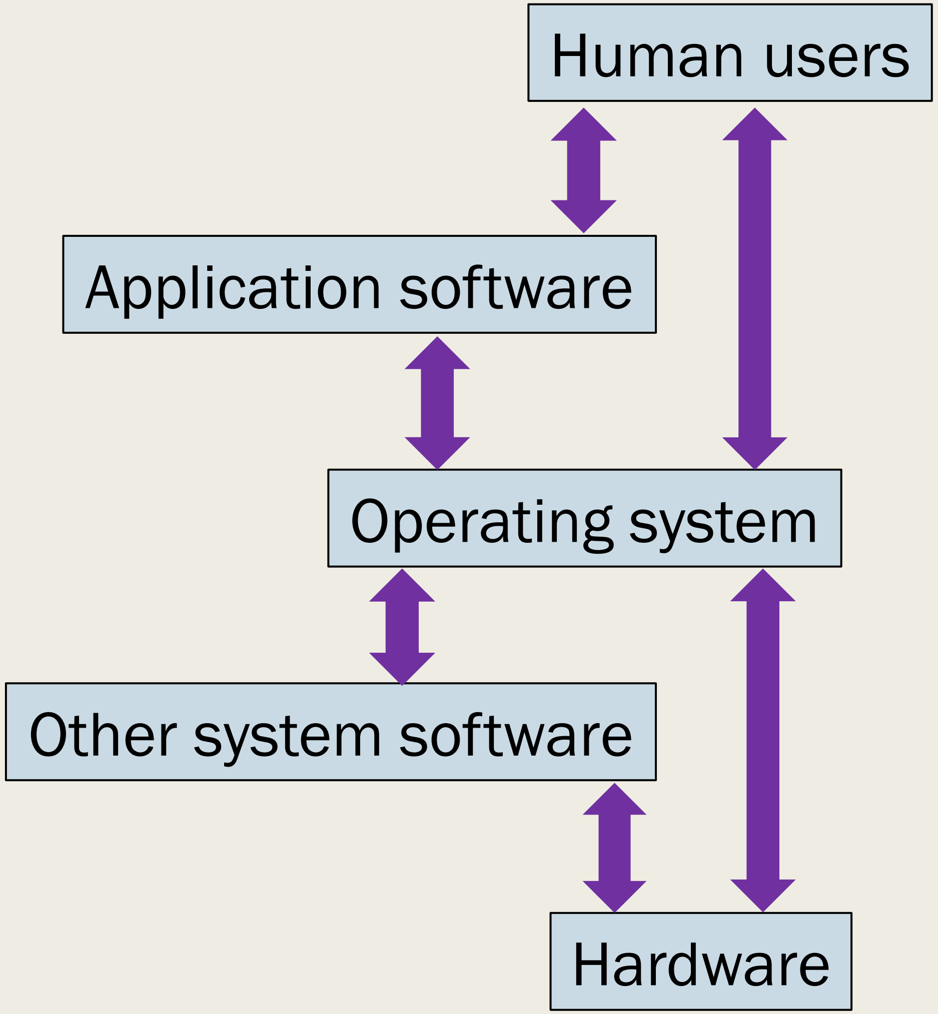
COMPUTER SYSTEMS FUNDAMENTALS
(4COSC004W)

Operating systems overview:

- Operation of Hardware is controlled by software.
  - *Operating system*
    - Every computer must have
    - Human role: King, Emperor, Director
- Different types for different purposes
- Functions
  1. *File Management*
  2. *Memory Management*
  3. *Process Management*
  4. *Input/output functionality*
  5. *General purpose functions – system information*
- File systems

Operating System interactions



This week:

- Operating Systems
  - *Types*
  - *Operations*
- Process Management
  - *Process lifecycle*
  - *Process scheduling*
- Memory Management
  - *Logical & Physical addressing*
  - *Memory management methods*
  - *Virtual memory*

TYPES OF OS

Types of OS

By the end of this unit, you will:

- Gain a brief appreciation of ;
  - *classification of OS*
  - *characteristics of types of OS*

Classification of types of OS

- Classified in terms of:
  - *Hardware they run*
  - *Number of programs that can be active*
  - *The type of interaction provided*

Microcomputer

- OS needs to:
  - *Initialise the system*
  - *Transfer data between memory and peripheral devices*
  - *Provide filing system*

Modern PC is evolved from microcomputer

- More powerful

Minicomputer

- Originally not much more powerful than microcomputer

- OS needs to:
  - *Support resource sharing*
  - *Error protection*
  - *Multi-user system*

Mainframe computer

- Late sixties

- OS needs to:
  - *Provide for many programs to be active*
  - *I/O performed by separate controller box*
  - *Terminals treated as block devices*
  - *Terminal controller echoes commands*

Single-programmed OS

- Single process operating
  - *MS-DOS running on stand-alone computer*

Multi-programmed OS

- More than one process in memory
  - *Switches execution between programs*
- Share system resources
  - *Protect user*
- Windows
- .....

Batch processing system

- User jobs submitted sequentially in batches
- No interaction between running processes and the user
- Input provided on a backing store device
  - *Single-programmed or multi-programmed OS*

Interactive system

- Users can interact with running program
- Can be:
  - *Single-user, single-programmed*
- Or:
  - *Allow time-sharing among many user-programs*
    - Each user appears to have sole use of the system,
    - Although CPU, memory and peripheral devices are, in fact, shared

Real time systems

- Time critical applications
  - *Response to a device must be handled within certain time span or data would be lost.*
    - Telecommunications
    - Air traffic control
    - Manufacturing control process
    - …

Further reading:

- Computer Science Illuminated
  - *Chapter 10*
    - P. 333-361

FUNCTIONS OF THE OS

By the end of this unit you will gain a basic understanding of:

- Principal functions of the OS
  - *File management*
  - *Process management*
  - *Memory management*
  - *Input/output functionality*
  - *General purpose functions – system information*

File management

- Files : collection of related data
- Filename
  - *Regardless of physical storage*
- Directory structure
  - *Containing information about file*
- More details next week

Process management

- Create and control processes
- Scheduling of processes
- Switching between processes
- Communication between processes
- Handling interruptions of processes
- Termination of processes

Memory management

- Allocate and de-allocate
- Protect between users
- Share use of devices
- Avoid conflict & corruption

Input/output functionality

- Normally invoked by OS itself
- Managing physical input / output of devices
  - *So that simple request from filing system can be converted to codes to:*
    - initiate I/O transfer
    - Perform transfer
    - Terminate transfer

General purpose functions to provide system information

- Process queues
- Disk quotas
- Time & Date

Additional functions

- OS are software like any other
- Developed to include more functionality
  - *Anti-virus*
  - *.....*

Further reading:

- Computer Science Illuminated
  - *Chapter 10*
    - Part 10.1 (p.334 – 340)

PROCESS MANAGEMENT

The Process Lifecycle

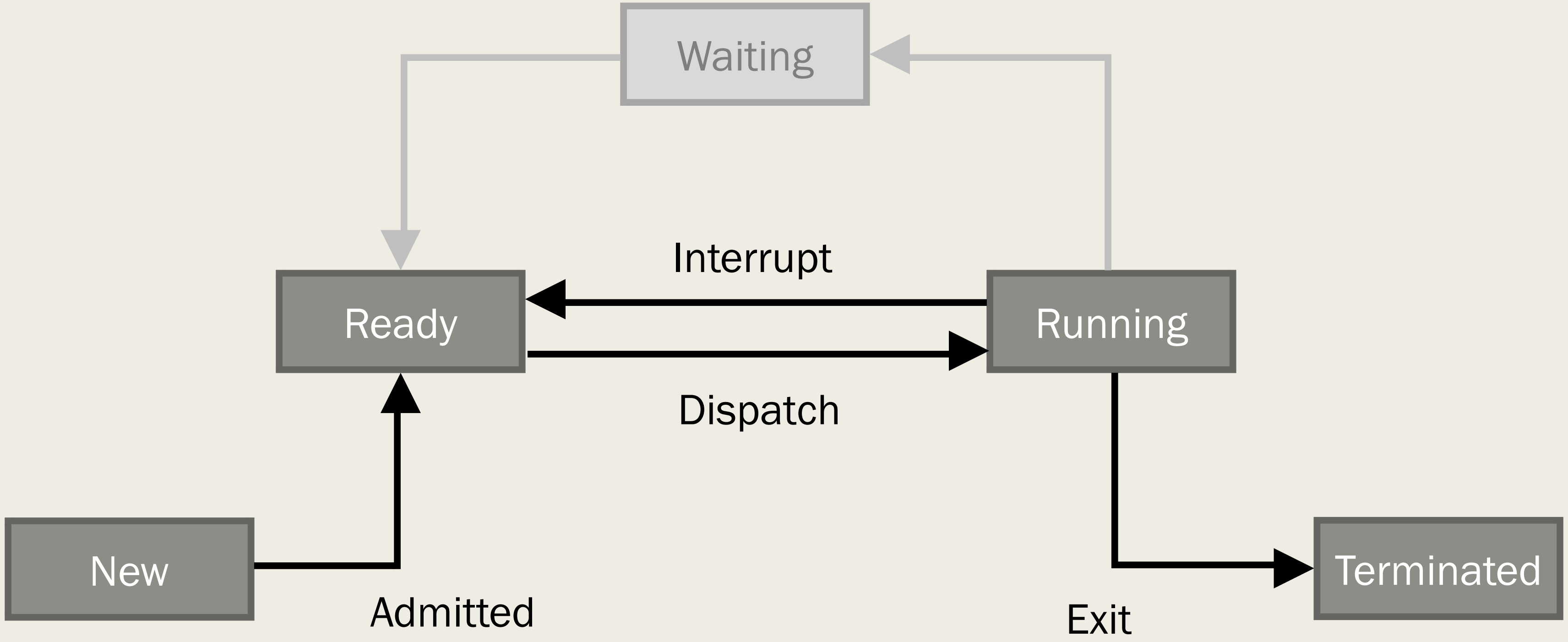
By the end of this unit, you will:

- Gain an appreciation of ;
  - *Process states*
  - *Process scheduling*
  - *CPU scheduling*

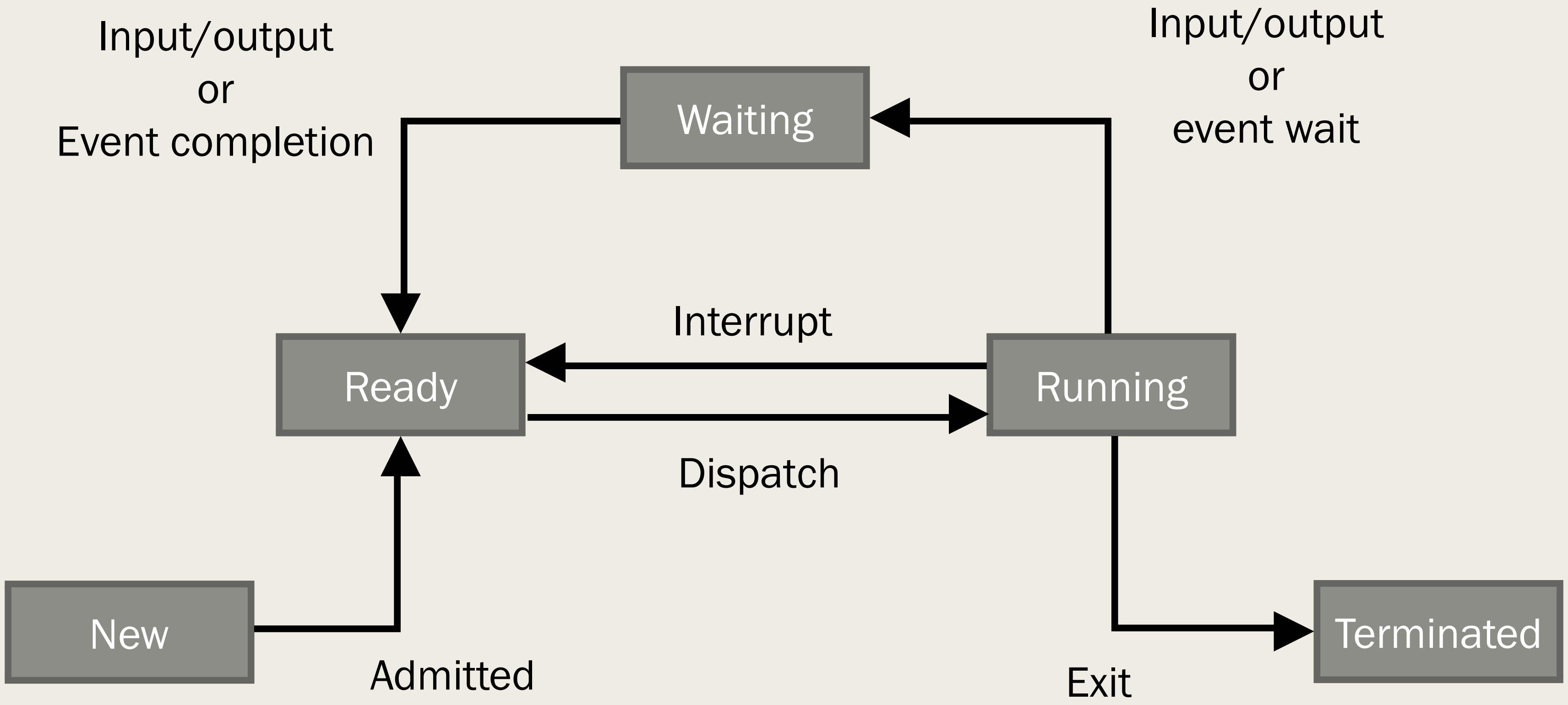
The Process:

- An instance of a computer program execution
- Machine code for process must reside in memory.
  - *May also require memory allocation for data*
- Requires CPU cycles – computer power
  - *OS manages this resource*

Process states



Process states



Process states

- New
  - *Being created*
  - *No resources yet allocated*
- Ready
  - *All resources are allocated*
  - *No more barriers to execution*
  - *No longer waiting for any events or data*
  - *Waiting for chance to use the CPU*
- Running
  - *Currently being executed*
  - *Instructions being processed in the fetch-execute cycle*
- Terminated
  - *Completed execution*
  - *No need to maintain data regarding process*

Process states : Waiting

- **Running**
  - *Currently being executed*
  - *Instructions being processed in the fetch-execute cycle*

- **Waiting**
  - *For resources (other than CPU)*
  - *For memory page*
  - *For process to send signal*
  - *For Input / Output*

- **Ready**
  - *Waiting for dispatch to CPU*

CPU scheduling

- Only processes in ready state can be moved to running state.
- Turnaround time
  - *Time between*
    - when process enters ready state,
    - and when it exits running state for the last time
- Scheduling approaches
  - *First Come, First Served*
  - *Shortest Job Next*
  - *Round Robin*

First Come, First Served (FCFS)

- Moved to CPU
  - *in the order in which the jobs arrive in the ready state*
- Non-preemptive

Process	Service time
p1	120
p2	80
p3	100
p4	30
p5	160

Shortest Job Next (SJN)

- Looks at all ready processes
  - *Selects shortest, runs it*
- Moves job to CPU
- Completes job

- Non-preemtive

FCFS	
Process	Service time
p1	120
p2	80
p3	100
p4	30
p5	160

SJN	
Process	Service time
p4	30
p2	80
p3	100
p1	120
p5	160

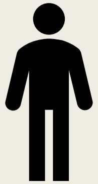
Round Robin

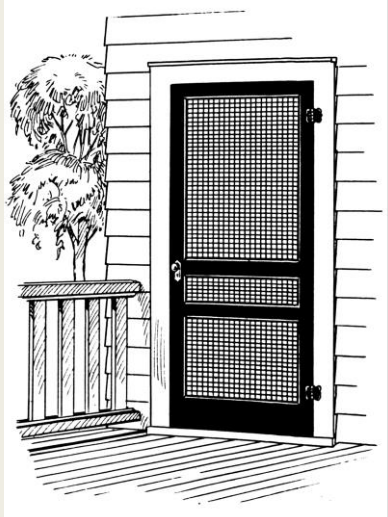
- Time Slice (Quantum)
  - *Suppose Time slice is 20*
- Preemptive
- Widely-used

Process	Service time
p1	120
p2	80
p3	100
p4	30
p5	160

0						120						230						350						
1	2	3	4	5	1	2	3	4	5	1	2	3	5	1	2	3	5	1	3	5	1	5	5	5

Imagine – single process:

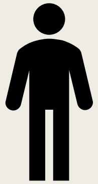




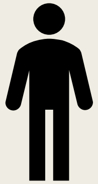
Office: 5.100

Imagine – Multiple processes:

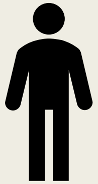
Imagine – First Come, First Served:



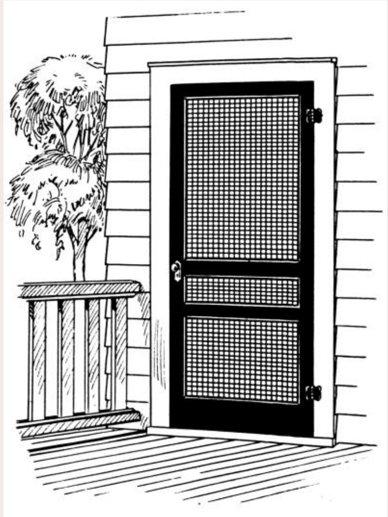
60



10



20

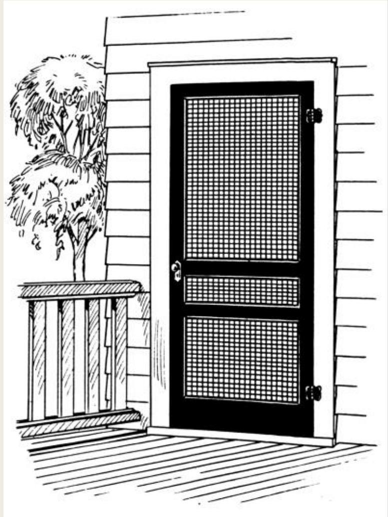


Office: 5.100

Imagine – First Come, First Served:

10

20



Office: 5.100

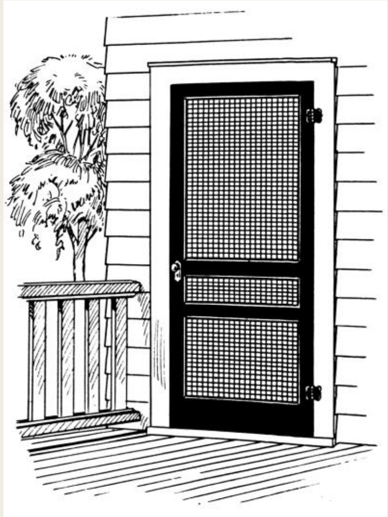
60

Imagine – Shortest Job Next

60

10

20



Office: 5.100

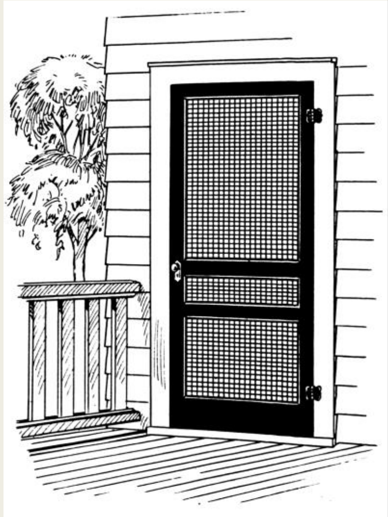
Imagine – Round Robin:

Quantum : 5

60

10

20



Office: 5.100

Further reading:

- Computer Science Illuminated
  - *Chapter 10*
    - 10.3 & 10.4 (p.347 – 361)
- Computer Systems
  - *Chapter 8*

MEMORY MANAGEMENT

By the end of this unit you will gain a basic understanding of:

- Memory management principles:
  - *Overview*
  - *Logical & Physical addressing*
  - *Partitioning of memory*
  - *Methods:*
    - Best-fit method
    - First-fit method
    - Worst-fit method
  - *Coalescing of holes*
  - *Paged memory*
  - *Swapping / Virtual memory*

Overview

- Computer memory:
  - *Stores data, information & instructions for all current processes*
  - *Work place for the CPU to use*
  - *Transient storage repository*
- More available memory enables:
  - *More processes to run (or be ready) simultaneously*
  - *Less dependence on swapping or paging*
  - *Less risk of running out of memory*

Principal tasks:

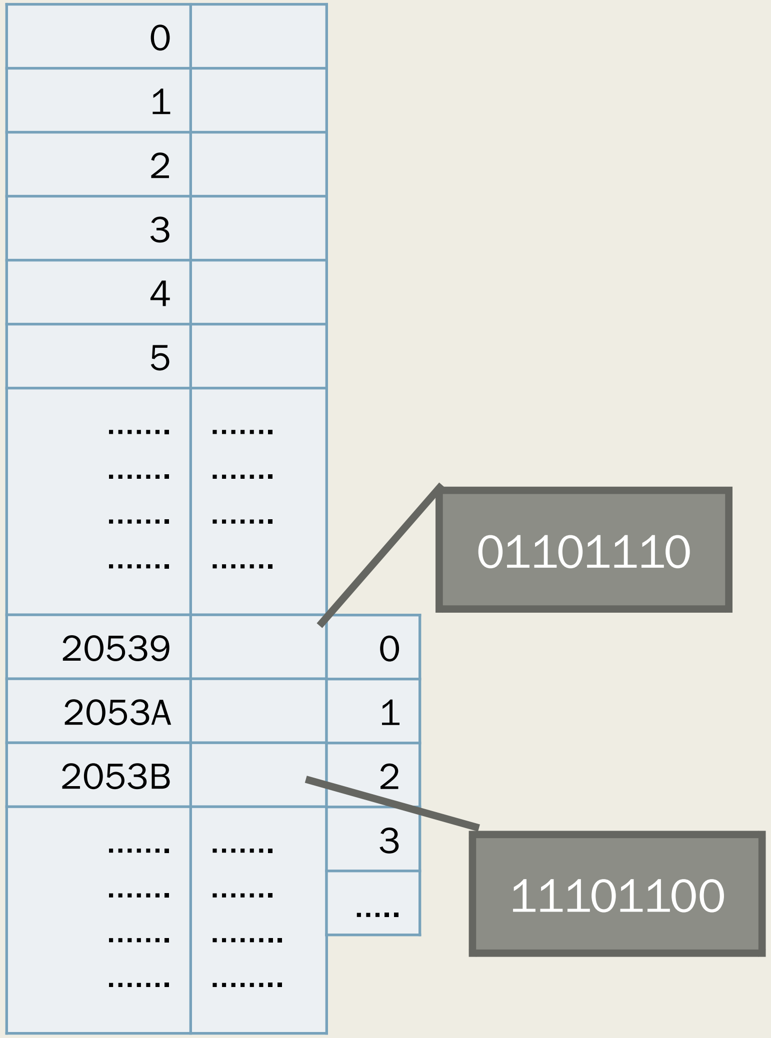
- Keep track of which areas of memory are in use
  - *And which NOT*
- Allocate memory to processes when required
- Deallocate when no longer required
- Enable sharing of memory between processes – when required
- Protect memory from other processes
- Manage swapping between memory and secondary devices
  - *Virtual memory*
- Provide satisfactory level of performance for computer
- Make addressing of memory space transparent to the programmer

What is Memory Management?

- Each process must reside in computer memory
  - *Assembly code occupies space*
  - *Process may require memory space to hold data (variables)*
- OS allocates resources
  - *Shared nicely*
  - *Keeps track of where program resides in memory*
  - *Releases resources after process terminates*
- Convert **Logical address** to **Physical address**

Logical address vs. Physical address

- Logical address:
  - *Location in memory relative to the program*
- Physical address:
  - *Actual address in the main memory*



Memory management – the process

- Partly in hardware called memory management unit
- Keeps track of which areas of memory are in use, and which are free
- Allocates memory to processes when the need it
  - *And deallocate when no longer required*
- Protect memory from other processes
- Manage Swapping & Paging

Fixed-size partition:

- Memory divided up into fixed sized spaces (eg. 300kB).
- Memory space allocated, just big enough to hold process.

- Problems:
  - *Wasted space*
  - *Unable to allocate resource to a process*

Variable-sized partition:

- Exact amount required is allocated.
- Memory can be allocated:
  - *until the whole memory space is filled or*
  - *until the remaining free space is too small to accommodate the new process*
- Memory from terminated processed is freed

Partition memory management

- At any point in time, memory is divided up into a set of partitions
  - *Some are empty*
  - *Some are allocated to programs*
- **Base register** is the register that holds the beginning address of the current partition
- **Bounds Register** is the register that holds the length of the current partition

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

A+L

A

Base register
A

Bounds register
length

Check:

L < length ?

Memory management methods

- Methods:
  - *Best-fit method*
  - *Worst-fit method*
  - *First-fit method*

Best-fit method

- New process is placed in a hole which is:
  - *Just big enough to accommodate it.*
- Unused space is minimized.

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

Best-fit method – New process E, 125kB

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	
	Process E (125 kB)
600 kB	Process A (150 kB)
450 kB	Free space (250 kB)
200 kB	Process B (100 kB)
100 kB	Operating System

Worst-fit method

- New process is placed in the hole which is :
  - *The largest hole in the memory*
- Maximises large holes of free space

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

Worst-fit method – New process E, 125kB

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

1450 kB	Process D (100 kB)
1250 kB	Free space (175 kB)
	Process E (125 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450 kB	Free space (250 kB)
200 kB	Process B (100 kB)
100 kB	Operating System

First-fit method

- New process is placed in:
  - *The first hole which is big enough to accommodate it.*
- Fast memory management
- Easy to implement
- But wasteful

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

First-fit method – New process E, 125kB

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (250 kB)
200kB	Process B (100 kB)
100 kB	Operating System

1450 kB	Process D (100 kB)
1250 kB	Free space (300 kB)
950 kB	Process C (150 kB)
800 kB	Free space (200 kB)
600 kB	Process A (150 kB)
450kB	Free space (75 kB)
	Process E (125 kB)
200kB	Process B (100 kB)
100 kB	Operating System

Coalescing of memory holes

1500 kB	Free space (400 kB)
1100 kB	Process B, 325 kB
675 kB	Free space (225 kB)
450 kB	Process A, 200 kB
250 kB	Free space (150kB)
100 kB	Operating System

1500 kB	Free space (400 kB)
1100 kB	*Process B finishes* Free space (325 kB)
675 kB	Free space (225 kB)
450 kB	Process A, 200 kB
250 kB	Free space (150kB)
100 kB	Operating System

1500 kB 1100 kB 675 kB	Free space (925 kB)
450 kB	Process A, 200 kB
250 kB	Free space (150kB)
100 kB	Operating System

Paged memory - general

- Processes can be divided up into **pages.**
- All these **pages** must be of a fixed size – determined by the architecture
- Stored in memory **frames** when loaded into memory.
  - *A **Frame** is a fixed-size portion of main memory that holds a process page.*
  - *A **Page** is a fixed-sized portion of a process that is stored into a memory frame.*
  - *A **Page-map Table** (**PMT)** is a table used by the OS to keep track of page/frame relationships.*

Paged Memory Management - addressing

P1 PMT	
Page	Frame
0	11
1	10
2	4
3	7

P2 PMT	
Page	Frame
0	6
1	5
2	1

Frame	Contents
0	
1	P2/Page2
2	
3	
4	P1/Page2
5	P2/Page1
6	P2/Page0
7	P1/Page3
8	
9	
10	P1/Page1
11	P1/Page0
12	
13	
……	…..

$$Physical\ address = (frame\ number\ \times frame\ size) + offset$$

Paged Memory Management – Demand, swapping and thrashing

- **Demand paging**
  - *Paging memory as demanded*
  - *Not all parts of program actually need to be in memory at the same time*
    - Pages are brought into memory on demand
- **Page swapping**
  - *Bringing in a page from secondary memory*
    - Writing back another page to secondary memory
- **Thrashing**
  - *Too much page swapping can seriously degrade performance*

Swapping / Virtual memory

- RAM (Primary memory) is very fast
- Hard disks are slower
- When more memory is required than exists in the system – processes cannot be allocated resources

- Swapping / virtual memory:
  - *Allocating part of the hard disk as an extension of the primary memory.*
  - *Disk area is called Swap Space*
  - *Swapped in , Swapped out*

Virtual memory:

- Addresses used within program refers to a ***virtual address*** in memory
  - *not the real address*
- Process has a virtual memory whose size and characteristics differ from those of real memory.
- Transparent
  - *Program does not need to know whether it is using real memory or VM*

What we have covered about OS functions:

- Process Management:
  - *What is a Process?*
  - *Process states*
  - *Process lifecycle*
  - *Process Scheduling*
  - *CPU Scheduling*
- Memory Management
  - *Physical & logical addressing*
  - *Memory Management Methods*
    - Best-fit
    - Worst-fit
    - First-fit
  - *Paging*
  - *Swapping*
  - *Virtual memory*

Further reading:

- File system Forensics
  - *Volumes & Partitions: p.70-71*
  - *PC-based partitions: p. 81 – 93*
- Module notes Volume II
  - *3.3 – 3.3.5*

Thank you