COMPUTER SYSTEMS FUNDAMENTALS (4COSC004W)

Lecture: Week 8. Part 3 of 3

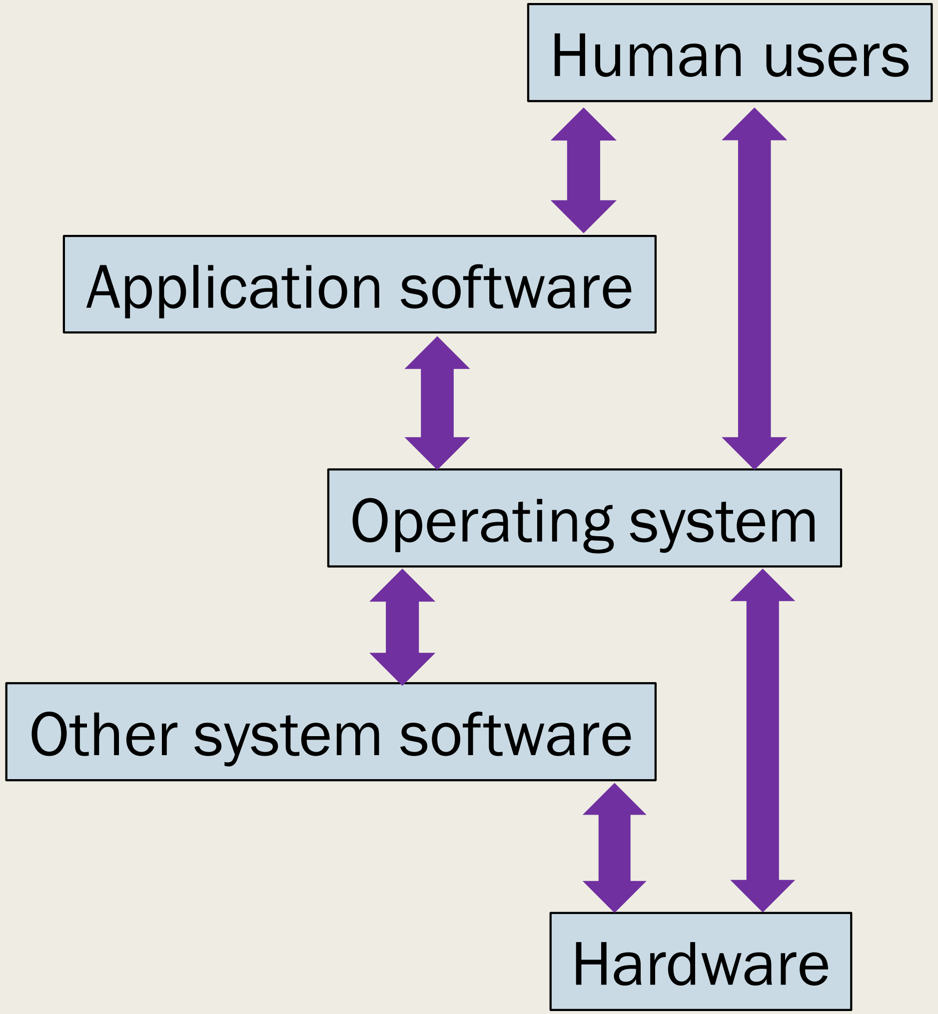
Contact details

- Module Leader:

Operating systems overview:

- Operation of Hardware is controlled by software.
  - *Operating system*
    - Responsibility: to manage resources
- Functions
  1. *File Management*
  2. *Process Management*
  3. *Memory Management*
  4. *Input/output functionality*
  5. *General purpose functions – system information*

Operating System interactions



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:

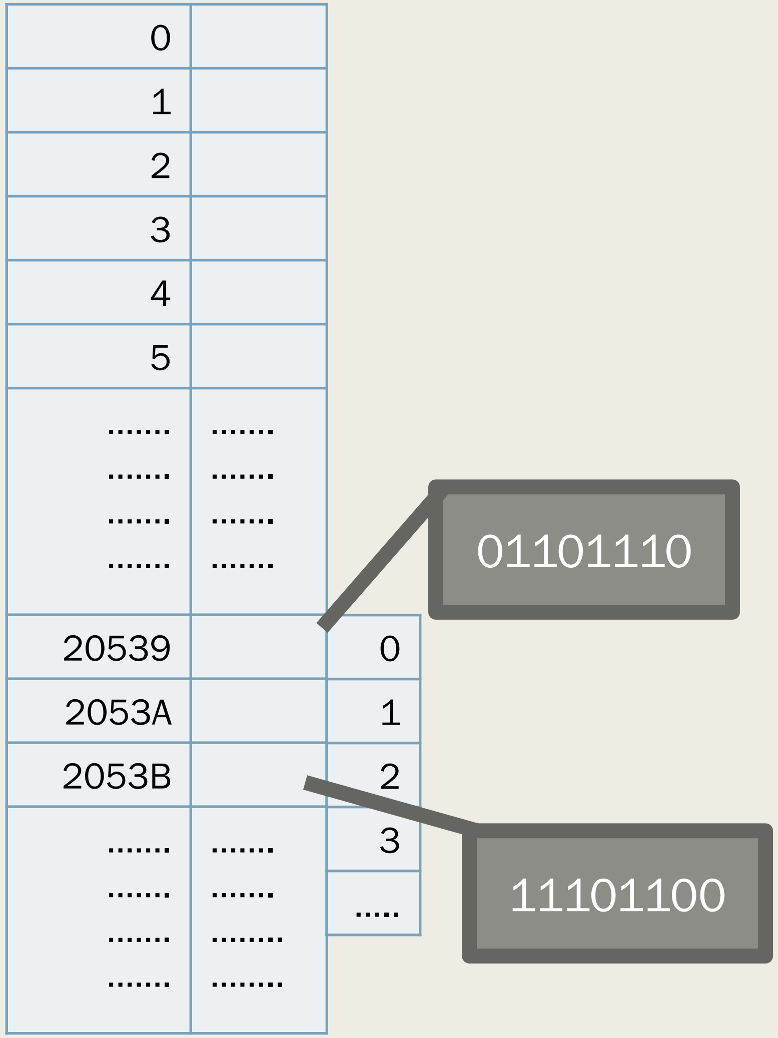
- Keep track of which areas of memory are in use
  - *And which NOT*
- Allocate memory to processes when required
- Deallocate when no longer required
- Enable sharing of memory between processes – when required
- Protect memory from other processes
- Manage swapping between memory and secondary devices
  - *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

Address	Contents
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

Address	Contents
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

Address	Contents
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
  - *Process states*
  - *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*