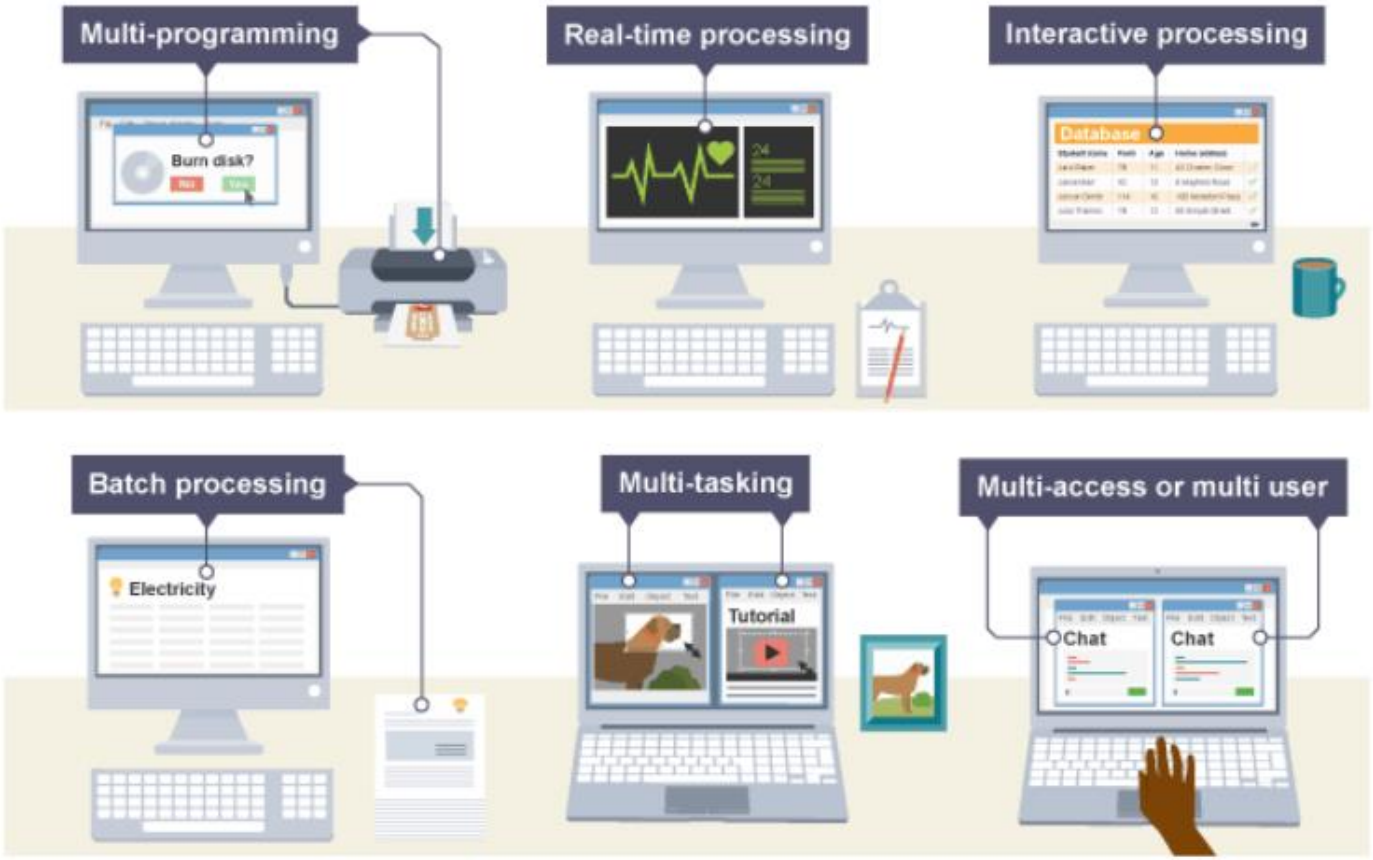
Week 12 – Operating Systems

Caroline Cahill

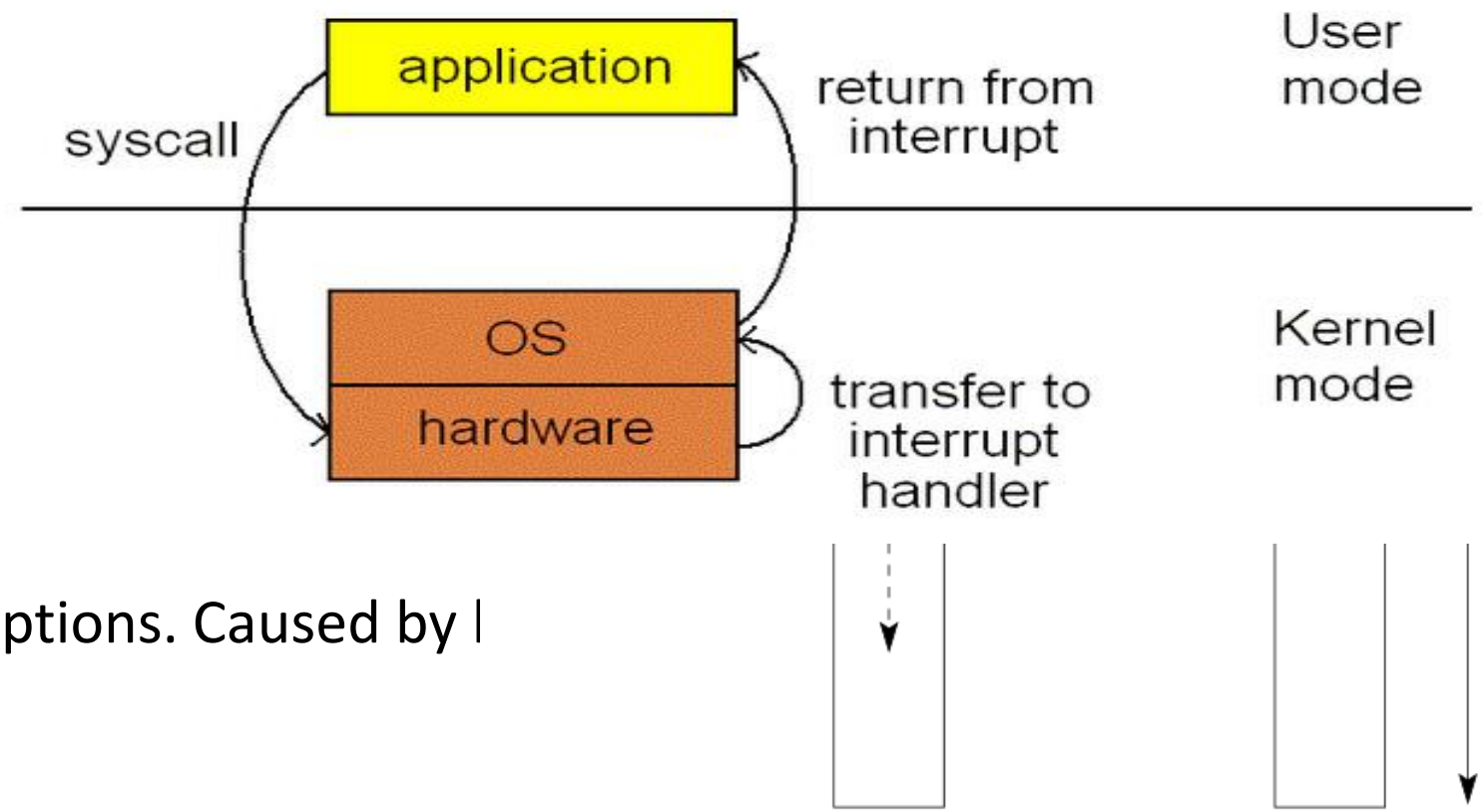
Waterford Institute of Technology

Week 8: Operating Systems

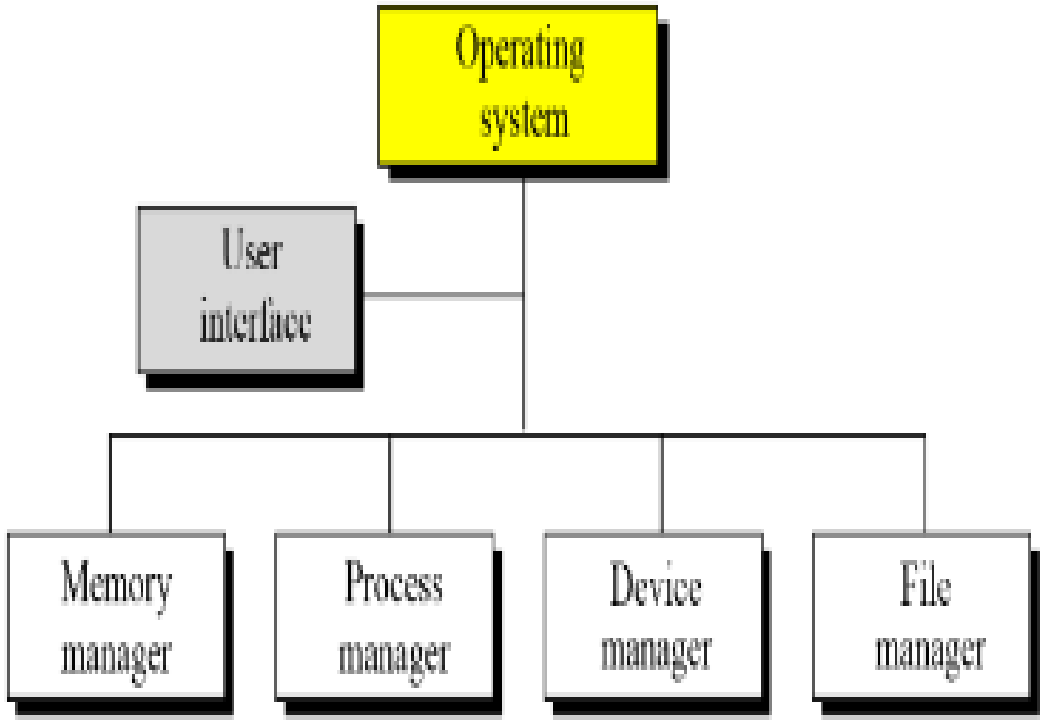


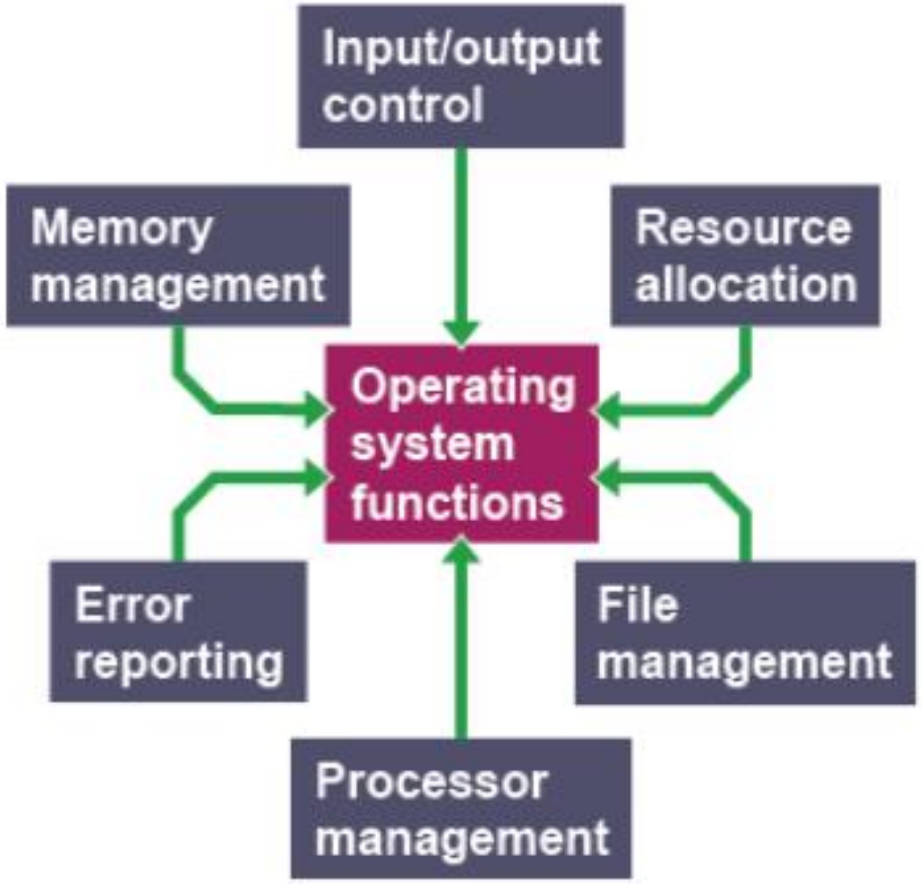
The OS needs to be able to respond to events

- System Calls
- Exceptions
  - detected by t
- Interrupts
  - similar to exceptions. Caused by I program



OS Functions/Responsibilities/Duties





Process management by an OS

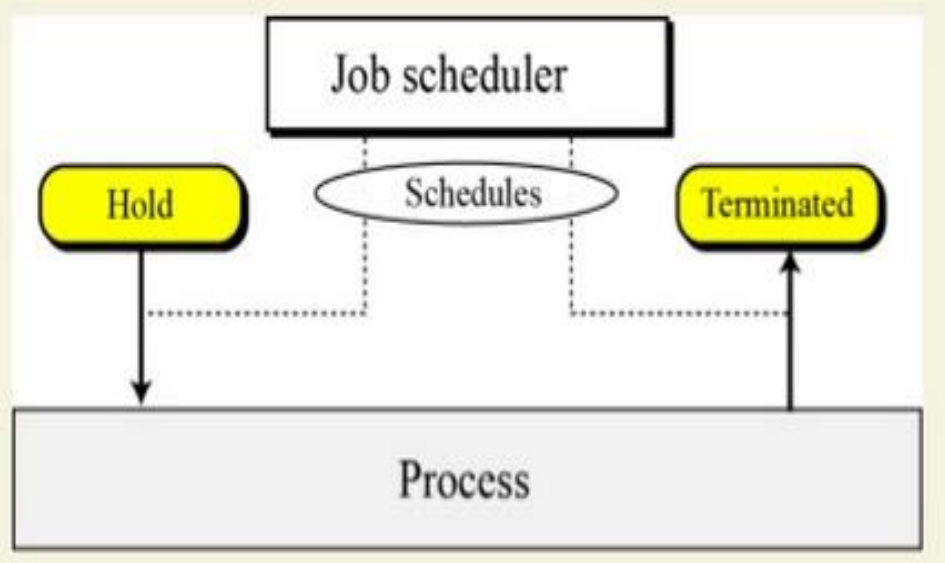
i. A ***program*** is a non-active, passive collection of instructions stored on disk

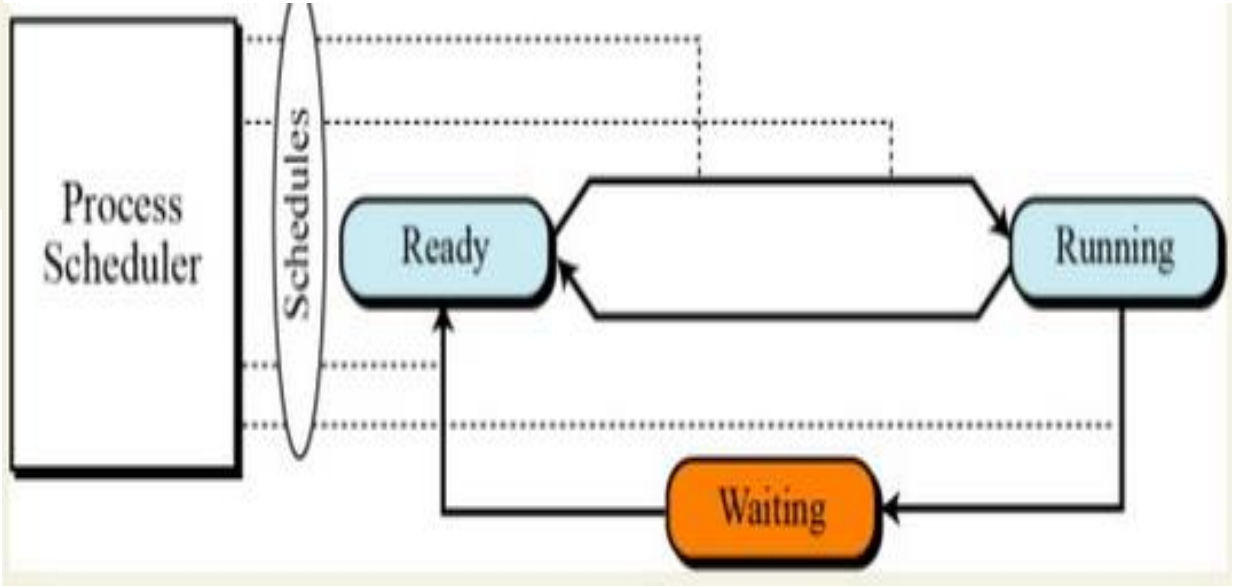
ii. A program becomes a ***job*** from the moment it is selected for execution until it has finished running and becomes a program again.

iii. A ***process*** is a program in execution. It is a program that has started but has not finished. A process has one or more threads, along with their execution state. A process is the actual execution of program instructions

iv. ***Thread***: Executes a series of instructions in order (only one thing happens at a time). A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history

Job and Process Schedulers

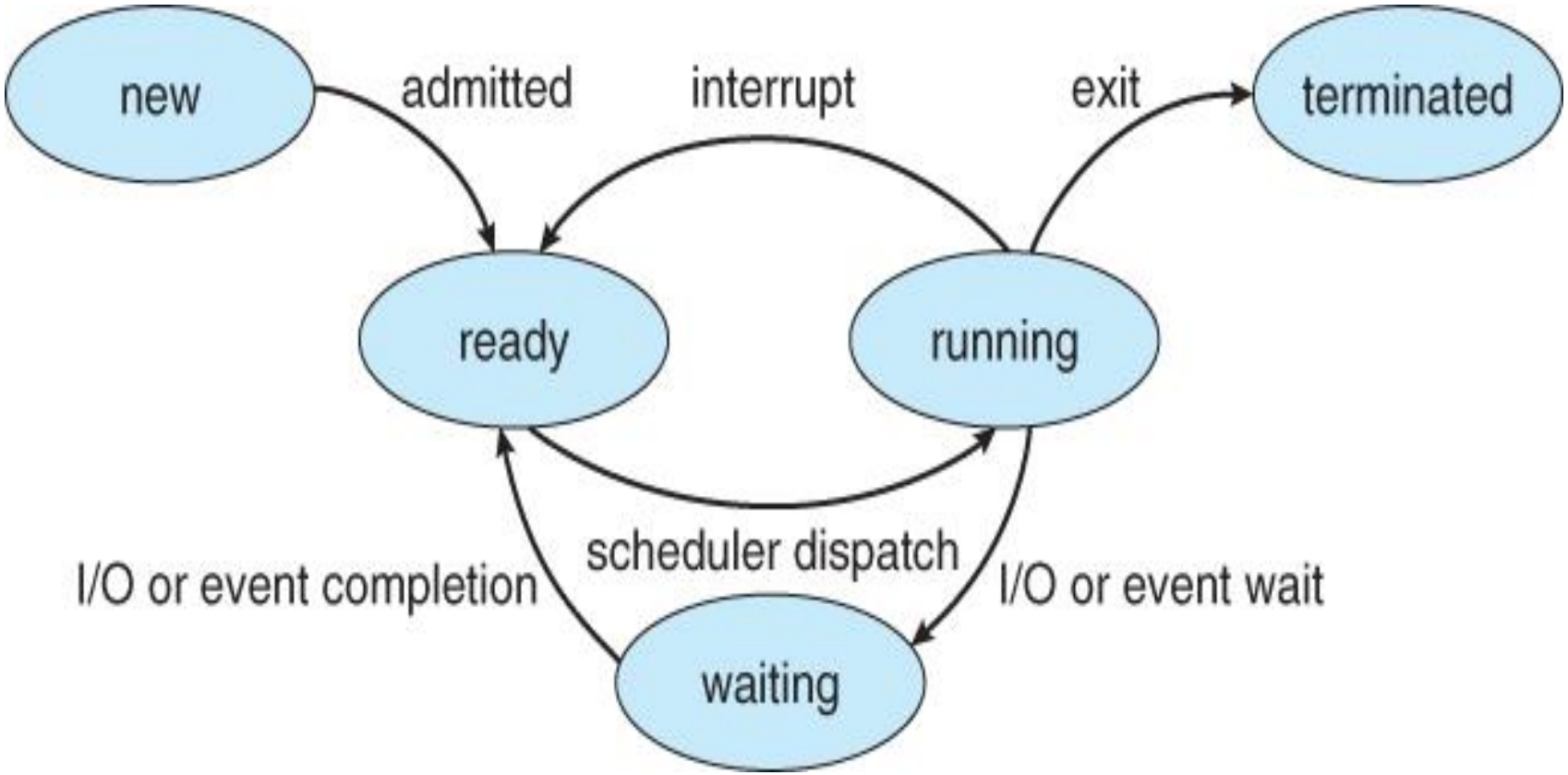




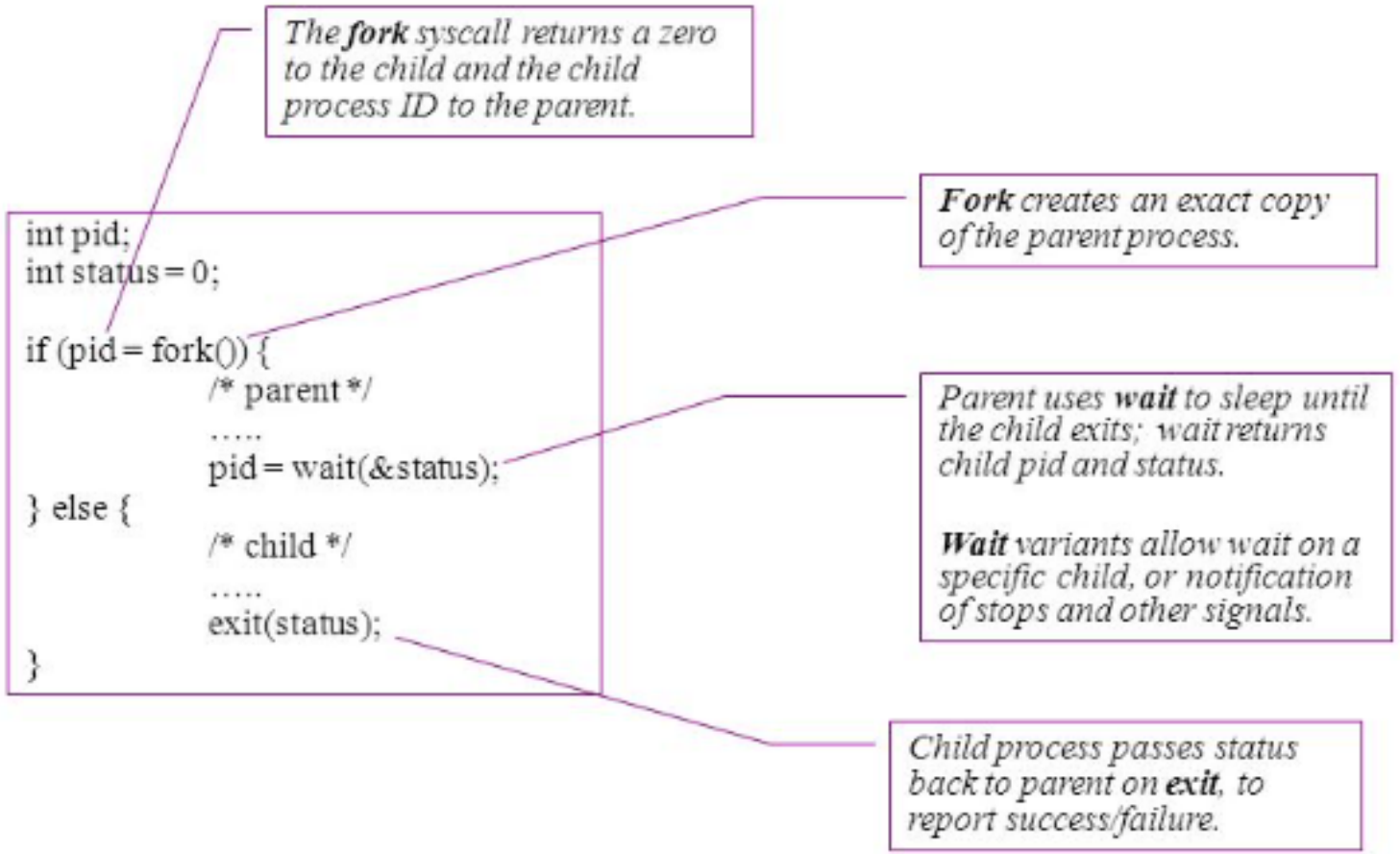
PCB


Process ID
State
Pointer
Priority
Program counter
CPU registers
I/O information
Accounting information
etc....

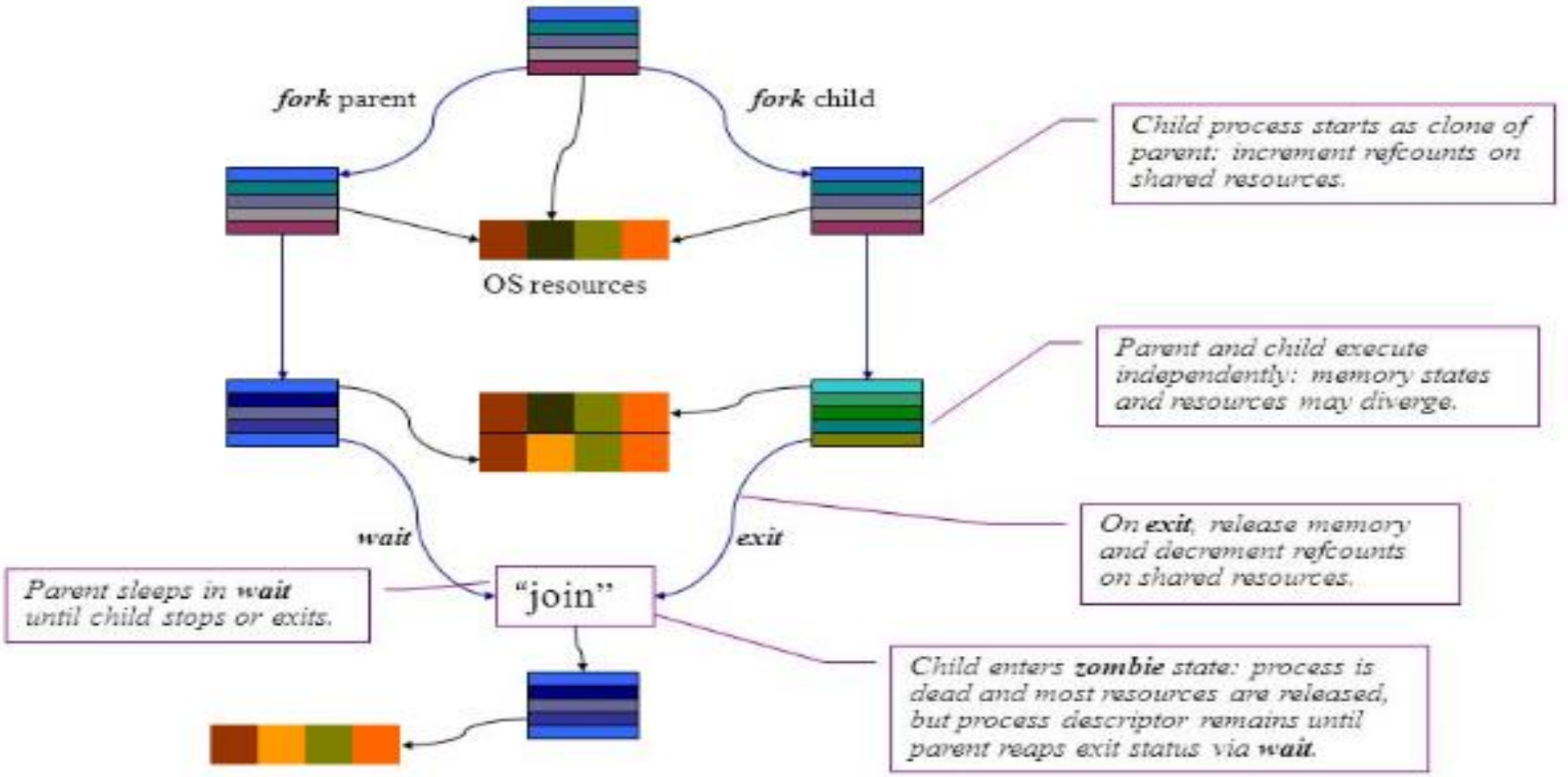
States of a Process



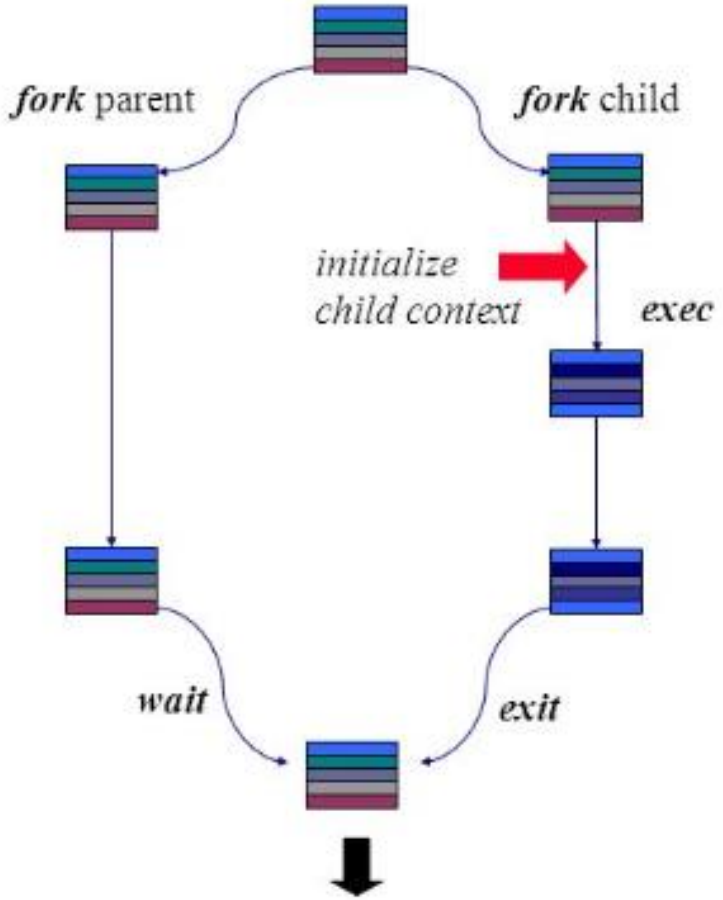
Unix Process Control



Fork/Exit/Wait



Fork/Exec/Exit/Wait



int pid = fork();
Create a new process that is a clone of its parent.

exec*("program" *[, argvp, envp]*);
Overlay the calling process virtual memory with a new program, and transfer control to it.

exit(status);
Exit with status, destroying the process.

int pid = wait*(&status);
Wait for exit (or other status change) of a child.

CPU Scheduling Techniques

- **Preemptive Scheduling Techniques**
  - the low level scheduler can remove a process from the RUNNING state in order to allow another process to run
  - SRT
  - Round Robin (RR)
- **Non-preemptive Scheduling Techniques**
  - "run to completion“ technique
  - FCFS Scheduling.
  - SJN Scheduling.
- Priority Scheduling

Memory Management

- transferring programs into and out of memory,
- allocating free space between programs,
- keeps track of each and every memory location,
- Check how much memory is to be allocated to processes.
- decides which process will get memory at what time
- tracks whenever some memory gets freed or unallocated
- updates the status.

Memory Management

- Swapping
- Paging, page tables and TLB
- Partitions
- Segmentation
- Virtual memory

Input/Output Device Management

- Efficiency is paramount, as is
- Generality

- The need for a *human* to input information and receive output from a computer.
- The need for a device to input information and receive output from a computer.
- The need for computers to communicate (receive/send information) over networks.

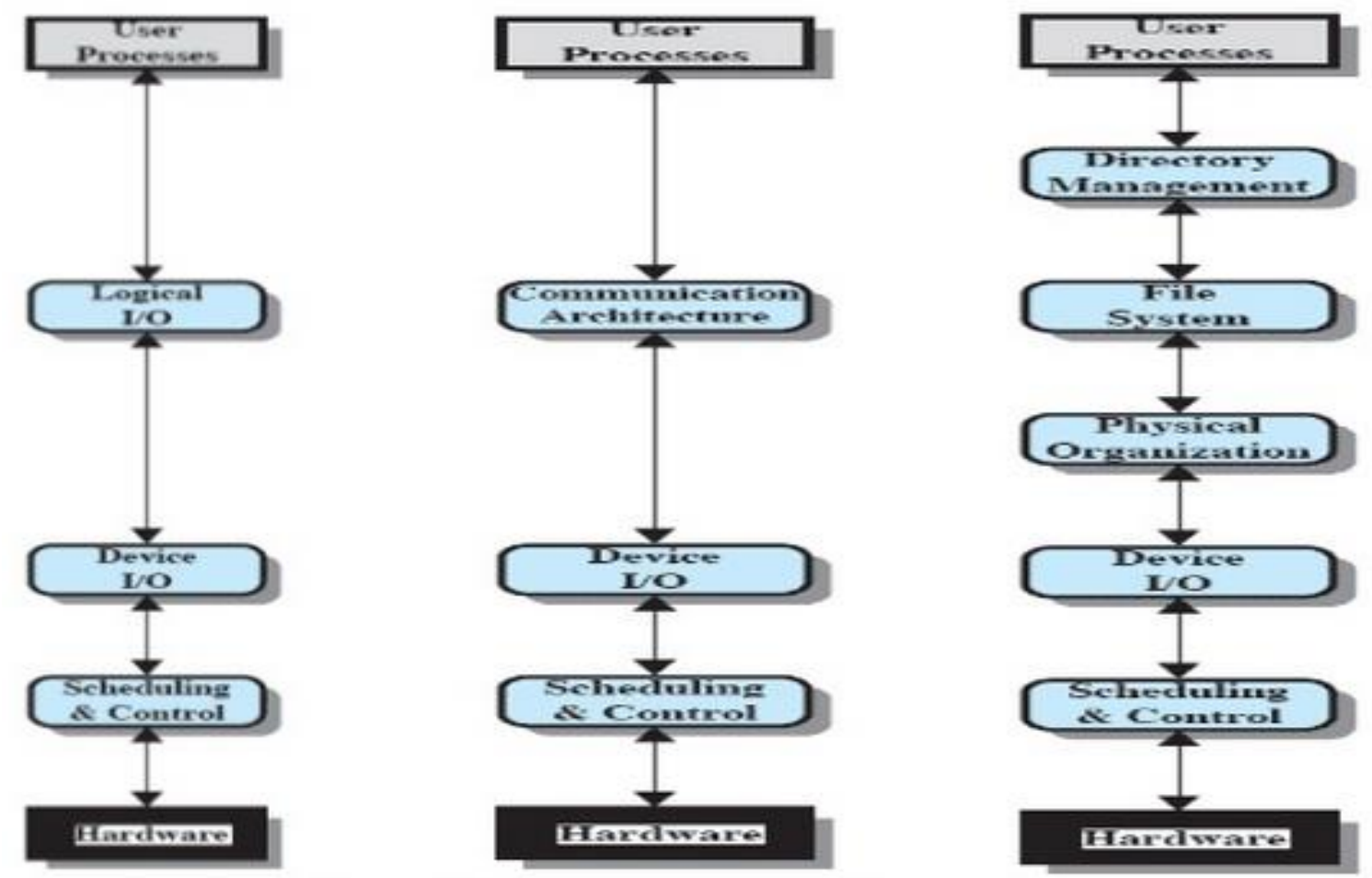
Modes of Data Transfer

- Consider device characteristics, synchronisation and reliability

3 main techniques for performing I/O:

1. Programmed I/O
2. Interrupt Driven I/O and
3. Direct Memory Access I/O
4. Polling I/O

I/O Organisation



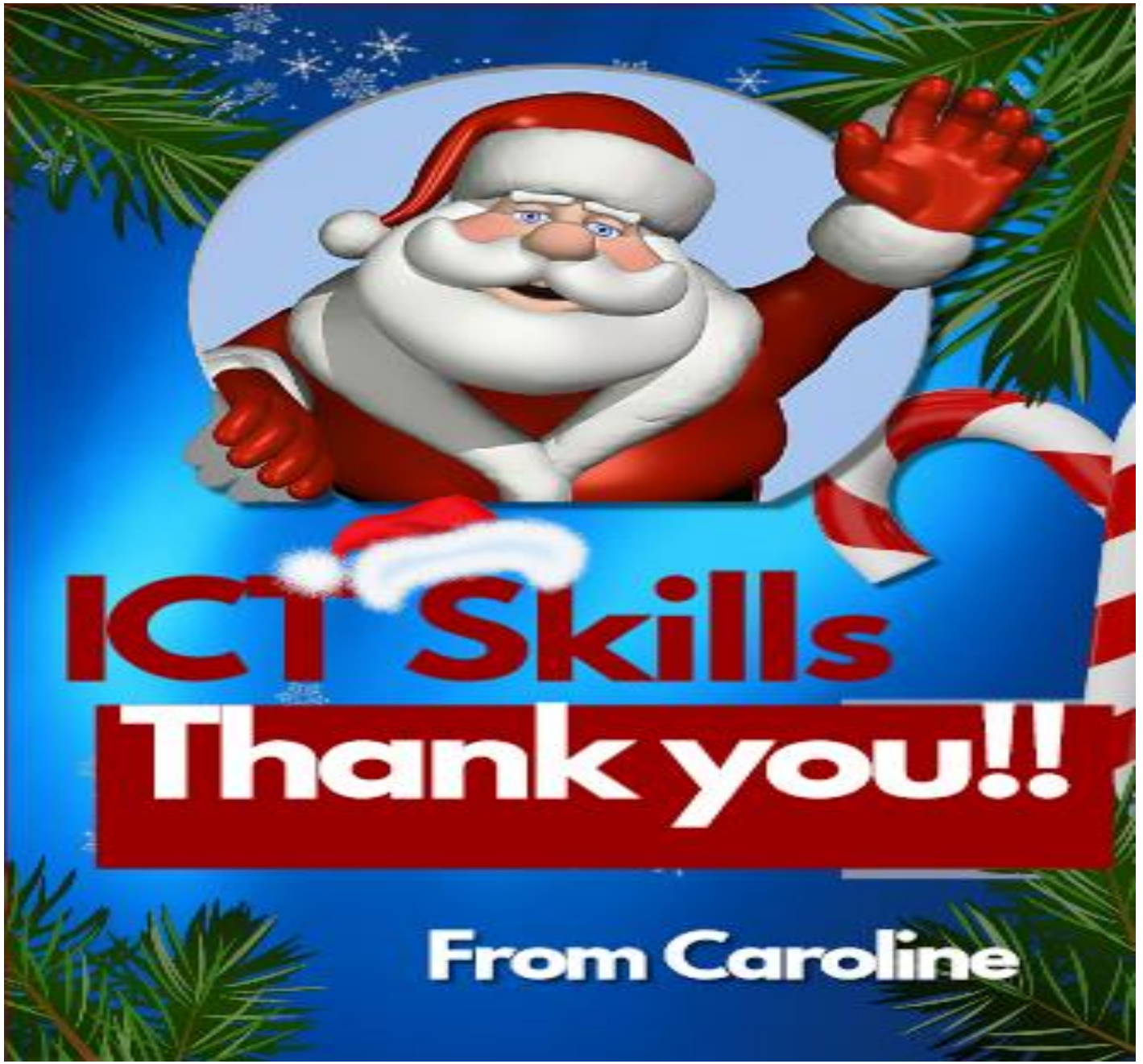
Improving Disk I/O Performance:

Disk Scheduling Techniques

- FIFO Disk Scheduling
- Priority
- LIFO
- SSTF (Shortest Service Time First)
- SCAN
  - C-SCAN
  - N-step-SCAN
  - FSCAN

Disk Cache

- a buffer in main memory for disk sectors
- contains a copy of some of the sectors on the disk
- Cache Replacement Policies
  - LRU
  - LFU
  - Freq-Based
  - LRU disk performance


ICT Skills
Thank you!!
From Caroline