# Chapter : Processes

***By: Navjot Kaur***

# Schedulers

- Schedulers are special system software which handle process scheduling in various ways.
- Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

# Long Term Scheduler / Job Scheduler

- It **determines which processes are admitted** to the system for processing.
- Selects processes from pool (disk) and bring them into the ready queue
- It **selects processes from the queue and loads them into memory** for execution.
- When a process changes the state from new to ready, then there is use of long-term scheduler.
- It controls **Degree of Multiprogramming (DoM)**
- **DoM:** The degree of multiprogramming describes the maximum number of processes that a single-processor system can accommodate efficiently.

# Long Term Scheduler / Job Scheduler

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor (CPU) bound.

- **I/O BOUND:** that spends its more time in doing I/O than computations.
- **CPU BOUND:** that spends its more time doing computations.
- LTS selects a good process.

- **Good Process: (I/O Bound + CPU Bound)**

# Short-Term Scheduler (CPU Scheduler)

- Short-term schedulers, also known as dispatchers.
- Selects which process should be executed next among the processes and allocates CPU to one of them.

**(From Ready Queue** Selects one process for execution and Allocate CPU **(Running Queue))**

- It must select process for CPU execution frequently.
- Short-term schedulers are faster than long-term schedulers.

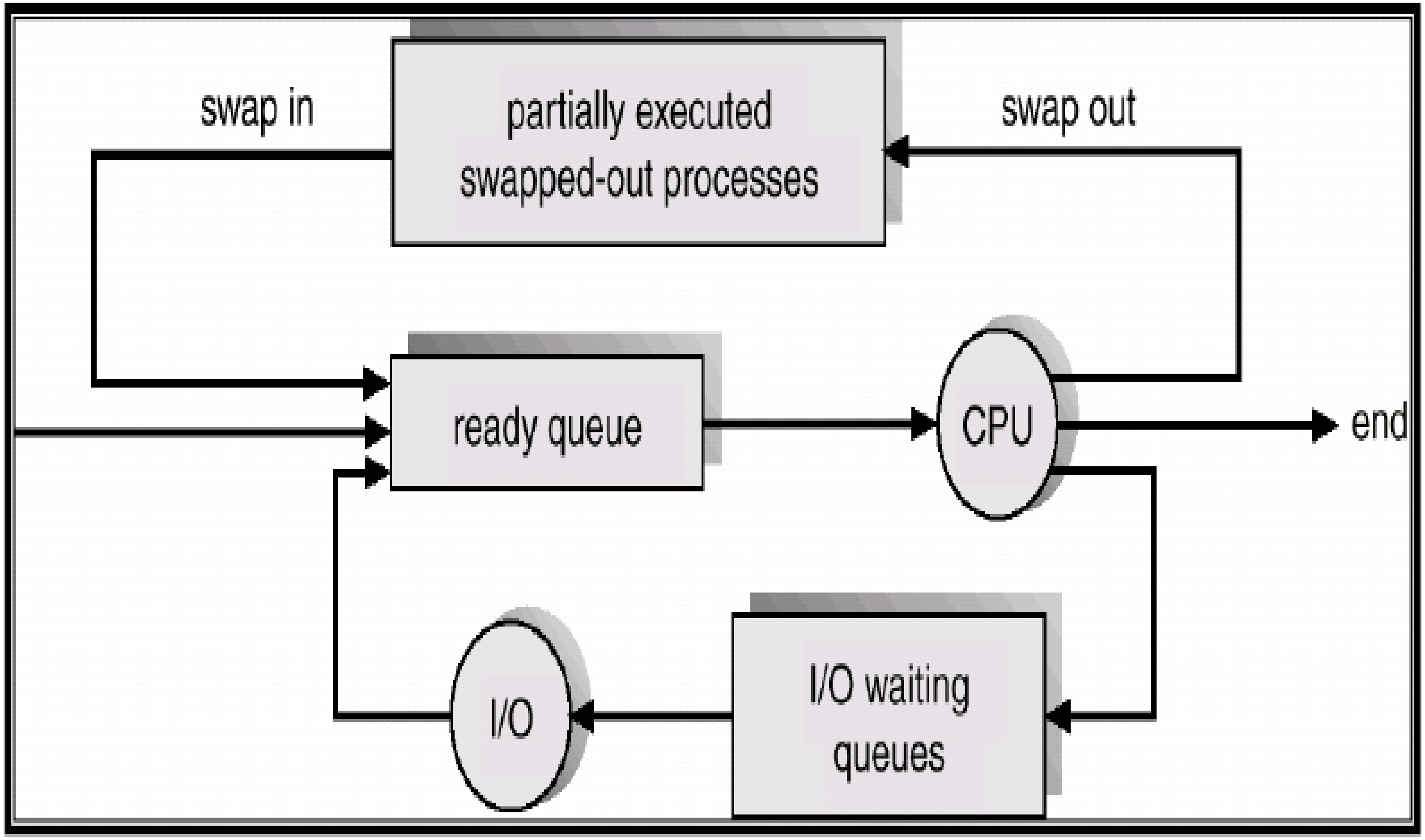
# Medium Term Scheduling

- Medium-term scheduling is a part of **swapping**.
- Reduces DOM(Degree of Multiprogramming)
- Processes are created and stored in Main Memory by Long Term Scheduler, But these processes has to wait for their turn to get execute.
- In Main Memory only those processes are kept that require execution.
- When ready queue is empty,

**MTS Swap-In and Swap-Out the processes from Main Memory and Secondary Memory**.

- Ready + Running + Waiting Queues are in Main Memory

# Medium Term Scheduling



# MCQ

1. CPU performance is measured through _________.

□ **a.** Throughput

□ **b.** MHz

□ **c.** Flaps

□ **d.** None of the above

# MCQ

1. a

# MCQ

2. Which of the following is a criterion to evaluate a scheduling algorithm?

a. CPU Utilization: Keep CPU utilization as high as possible

b. Throughput: number of processes completed per unit time

c. Waiting Time: Amount of time spent ready to run but not running

d. All of the above

# MCQ

2. d

# MCQ

3. __________ does the job of allocating a process to the processor.

**a.** Long term scheduler

**b.** Short term scheduler

**c.** Medium term scheduler

**d.** Dispatcher

# MCQ

3. d

# MCQ

4. In the multi-programming environment, the main memory consisting of __________ number of process.

a. Greater than 100

b. Only one

c. Greater than 50

d. More than one

# MCQ

4.d

# MCQ

5. CPU Scheduling is the basis of ___________ operating system.

a. Batch

b. Real time

c. Multiprogramming

d. Mono programming

# MCQ

5 c

# MCQ

6. _________ scheduler selects the jobs from the pool of jobs and loads into the ready queue.

a. Long term

b. Short term

c. Medium term

d. None of the above

# MCQ

6. a

# MCQ

7. Saving the state of the old process and loading the saved state of the new process is called ________.

a. Context Switch

b. State

c. Multi programming

d. None of the above

e. PCB

# MCQ

7. a

# Operations on Process

□ 1. Process Creation

□ 2. Process Termination

# Process Creation

- Parent process create children processes, which, in turn create other processes, forming a tree of processes
- The process which creates other process, is termed the **parent** of the other process, while the created sub-process is termed its **child**.
- Each process is given an integer identifier, termed as process identifier, or PID. The parent PID (PPID) is also stored for each process.

# Process Creation (Cont.)

- **Resource sharing**
  - Parent and children **share all resources**
  - **Children share subset** of parent’s resources
  - Parent and child **share no resources**
- **Execution Sequence**
  - Parent and children **execute concurrently**
  - Parent waits until children terminate

# Process Creation (Cont.)

- **Address Space**

**Address space** may refer to a range of either physical or virtual **addresses** accessible to a processor or reserved for a process.

  - Child process is duplicate of parent process (same program and data
  - Child process has new program loaded into it

- Each process is identified by its process identifier

- UNIX examples
  - **fork ()** system call creates new process
  - **exec()** system call used after a **fork** to replace the process' memory space with a new program

# fork()

- System call **fork()** is used to create processes.

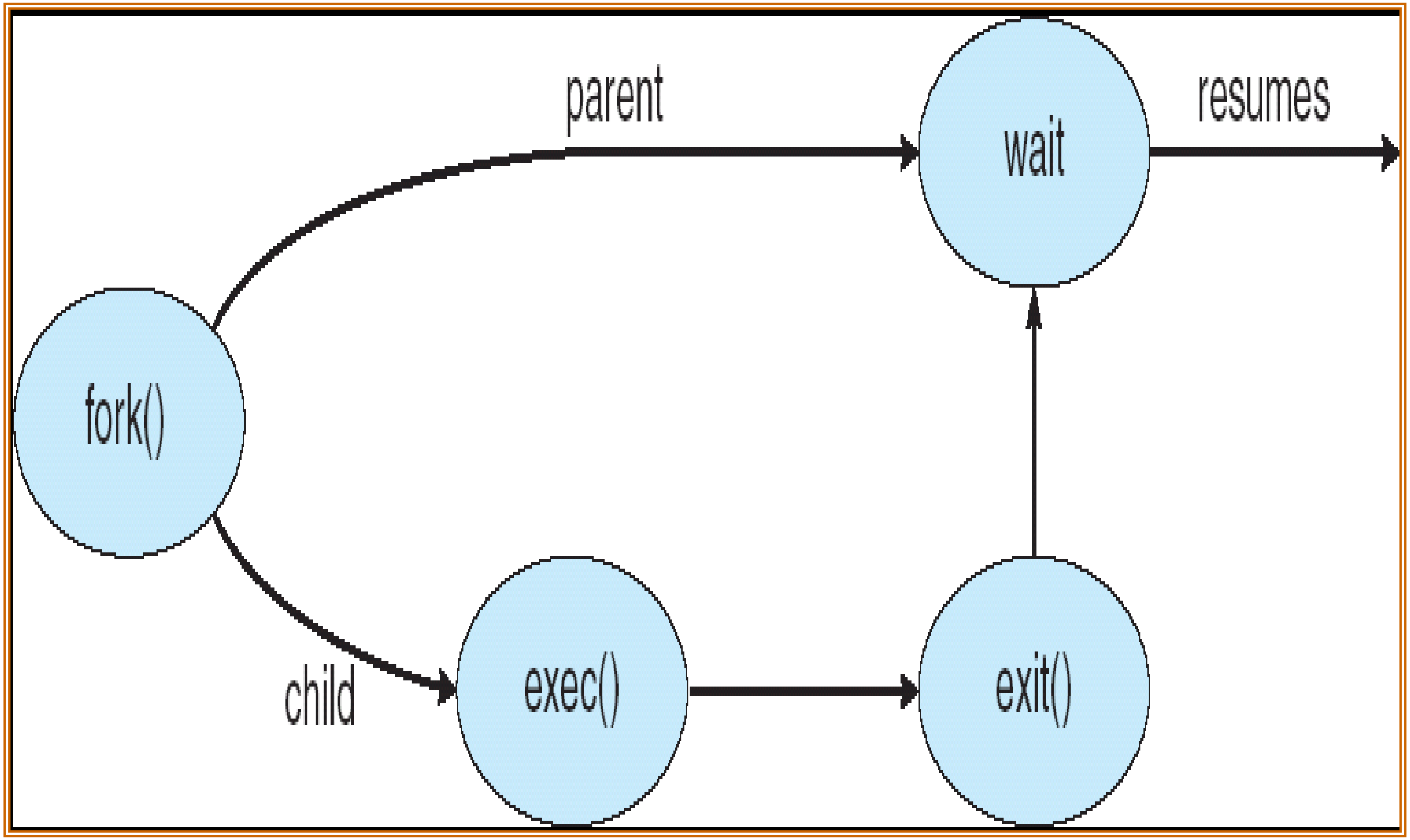
- It takes **no arguments** and returns a process ID.

- The purpose of **fork()** is to create a ***new*** process, which becomes the *child* process of the caller.

- After a new child process is created, ***both*** processes will execute the next instruction following the ***fork()*** system call.

This can be done by testing the returned value of **fork()**:

- If **fork()** returns a **negative value**, the creation of a child process was **unsuccessful**.
- **fork()** returns a zero to the newly created child process.
- **fork()** returns a positive value, the ***process ID*** of the child process, to the parent.


parent
wait
resumes
fork()
child
exec()
exit()

# Process Creation

- There are two options for the parent process after creating the child :

1. **Wait for the child process to terminate before proceeding.**
    - Parent process makes a wait() system call, for either a specific child process or for any particular child process, which causes the parent process to block until the wait() returns.
2. **Run concurrently with the child, continuing to process without waiting.**

There are also two possibilities **in terms of the address space** of the new process:

- The child process is a duplicate of the parent process.
- The child process has a program loaded into it.

# Process Termination

- Process executes last statement and asks the operating system to terminate it (by using **exit() system call**)
  - Output data from child to parent (via **wait**)
  - Process' resources are de-allocated by operating system
- **Parent may terminate execution of children processes** (**abort**)

**Why?**

  - Child has **exceeded allocated resources**
  - **Task** assigned to child is **no longer required**
  - If parent is exiting/Terminated
    - Some operating system do not allow child to continue if its parent terminates
      - All children terminated - *cascading termination*

# Process Termination

- **Processes may also be terminated** by the system for a variety of **reasons**,
  - The inability of the system to deliver the necessary system resources.
  - In response to a KILL command or other unhandled process interrupts.
  - A parent may kill its children if the task assigned to them is no longer needed.
  - If the parent does not exit, the system may or may not allow the child to continue without a parent (**orphaned processes** are generally inherited by init, which then proceeds to kill them.)

# Process Termination

-  **When a process ends, all of its system resources are freed up**. The process **termination status** and execution times are **returned to the parent** if the parent is waiting for the child to terminate,

or eventually returned to init if the process already became an orphan.

-  The processes which are trying to terminate but cannot do so because their parent is not waiting for them are termed **zombies**. These are eventually **inherited by init process as orphans** and killed off.

# MCQ

- Restricting the child process to a subset of the parent's resources prevents any process from :
- a) overloading the system by using a lot of secondary storage
- b) under-loading the system by very less CPU utilization
- c) overloading the system by creating a lot of sub-processes
- d) crashing the system by utilizing multiple resources

# MCQ

- c

# MCQ

- A parent process calling _____ system call will be suspended until children processes terminate.
- a) wait
- b) fork
- c) exit
- d) exec

# MCQ

- a

# MCQ

- Cascading termination refers to termination of all child processes before the parent terminates ______
- a) Normally
- b) Abnormally
- c) Normally or abnormally
- d) None of the mentioned

# MCQ

- a

# MCQ

- With ______________ only one process can execute at a time; meanwhile all other process are waiting for the processor. With _______________ more than one process can be running simultaneously each on a different processor.
- a) Multiprocessing, Multiprogramming
- b) Multiprogramming, Uniprocessing
- c) Multiprogramming, Multiprocessing
- d) Uniprogramming, Multiprocessing

# MCQ

- d

# MCQ

- In UNIX, each process is identified by its :
- a) Process Control Block
- b) Device Queue
- c) Process Identifier
- d) None of the the mentioned

# MCQ

- c

# MCQ

- In UNIX, the return value for the fork system call is _____ for the child process and _____ for the parent process.
- a) A Negative integer, Zero
- b) Zero, A Negative integer
- c) Zero, A nonzero integer
- d) A nonzero integer, Zero

# MCQ

- c

# MCQ

- The child process can :
- a) be a duplicate of the parent process
- b) never be a duplicate of the parent process
- c) cannot have another program loaded into it
- d) never have another program loaded into it

# MCQ

- a

# MCQ

-  aa