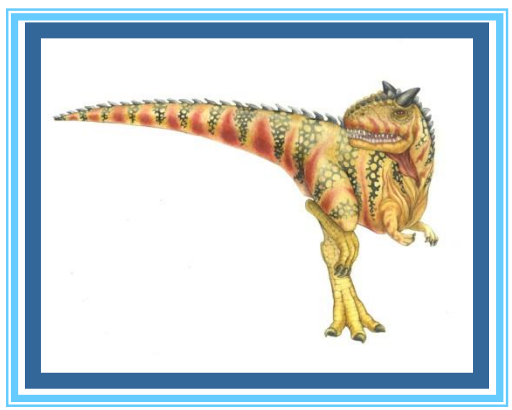
Chapter 3: Processes



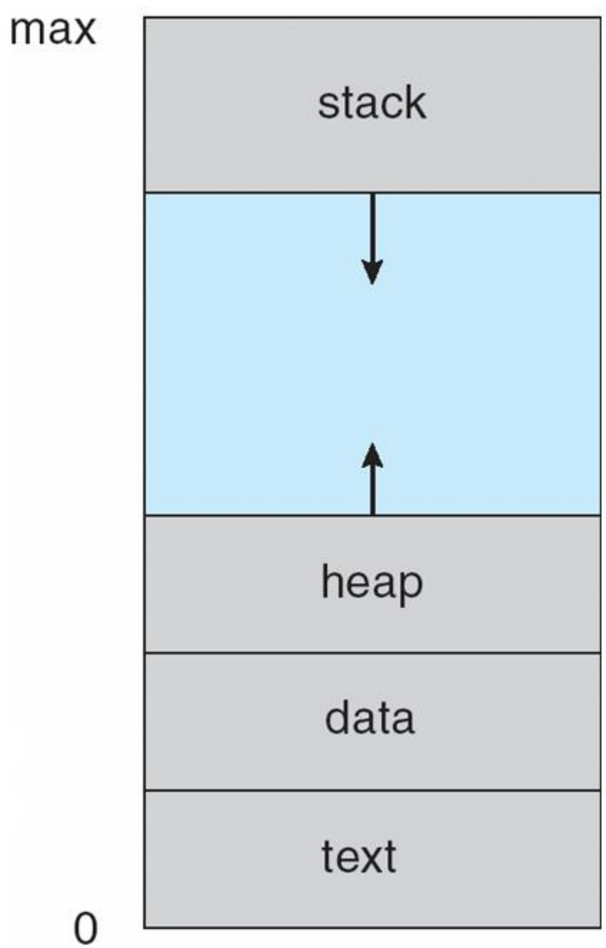
Chapter 3: Processes

- Process Concept
- Process Scheduling
- Operations on Processes
- Inter-process Communication

Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably.
- Process – A program in execution is process.
- In computing, a **process** is an instance of a computer program that is being executed. It contains the program code and its current activity.
- Process execution must progress in sequential fashion.
- A process includes:
  - program counter
  - stack
  - data section

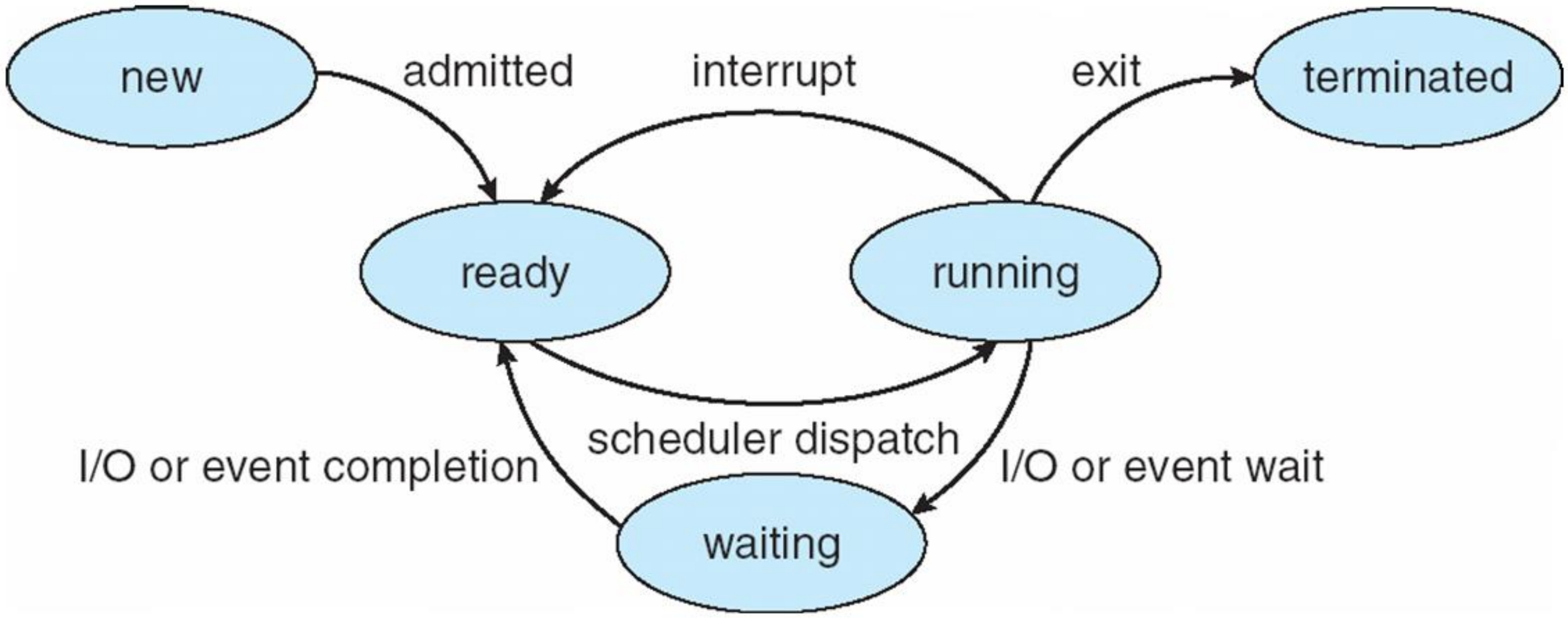
Process in Memory



Process State

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Diagram of Process State



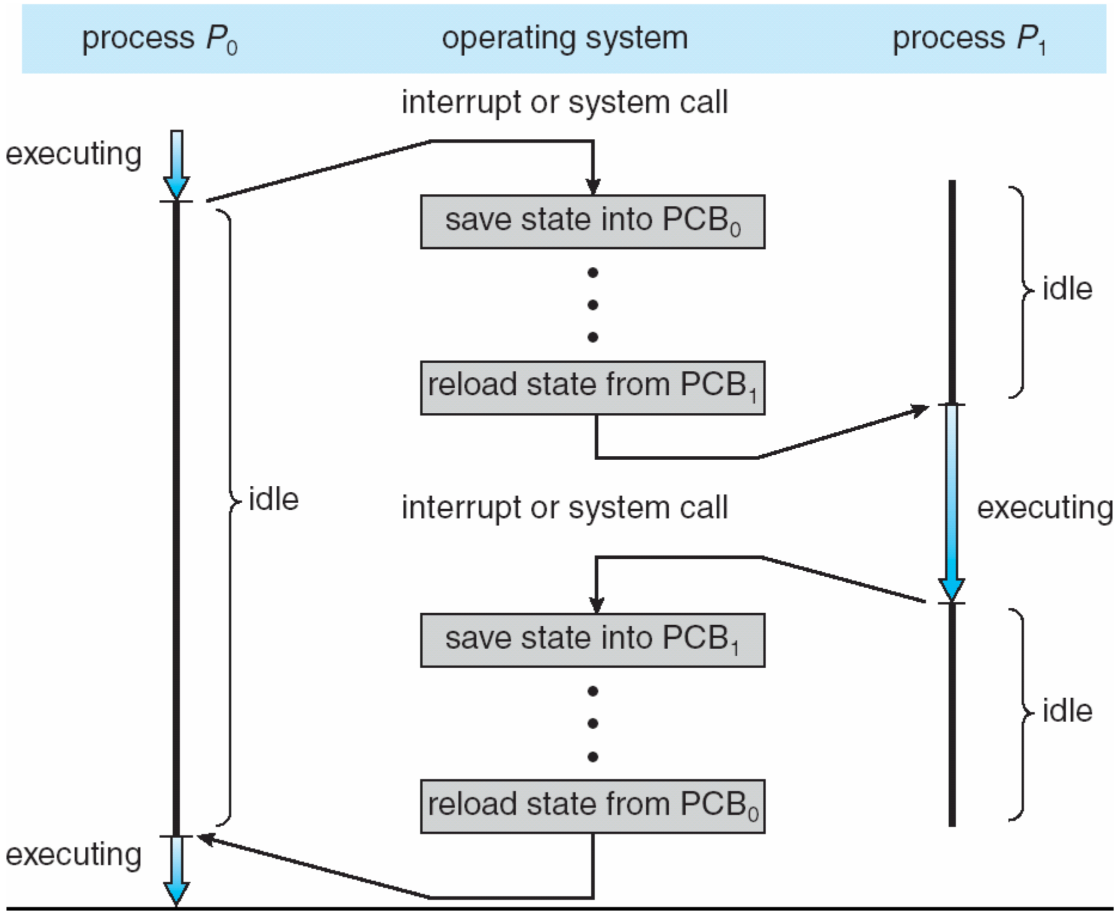
Process Control Block (PCB)

Information associated with each process

(also called **task control block**)

- **Process state**: Running, waiting, interrupted etc. information.
- **Program counter:** Indicates the address of the next instruction to be executed for this process.
- **CPU registers:** The registers vary in number and type, depending on the computer architecture. Keep contents of all process
- **CPU scheduling information:** This information includes a process priority, pointers to scheduling queues and any other scheduling parameters.
- **Memory-management information**: This information may include about memory allocated to the process.
- **Accounting information:** includes the amount of CPU and real time used, time limit, process number etc.
- **I/O status information:** I/O devices allocated to process, list of open files

CPU Switch From Process to Process

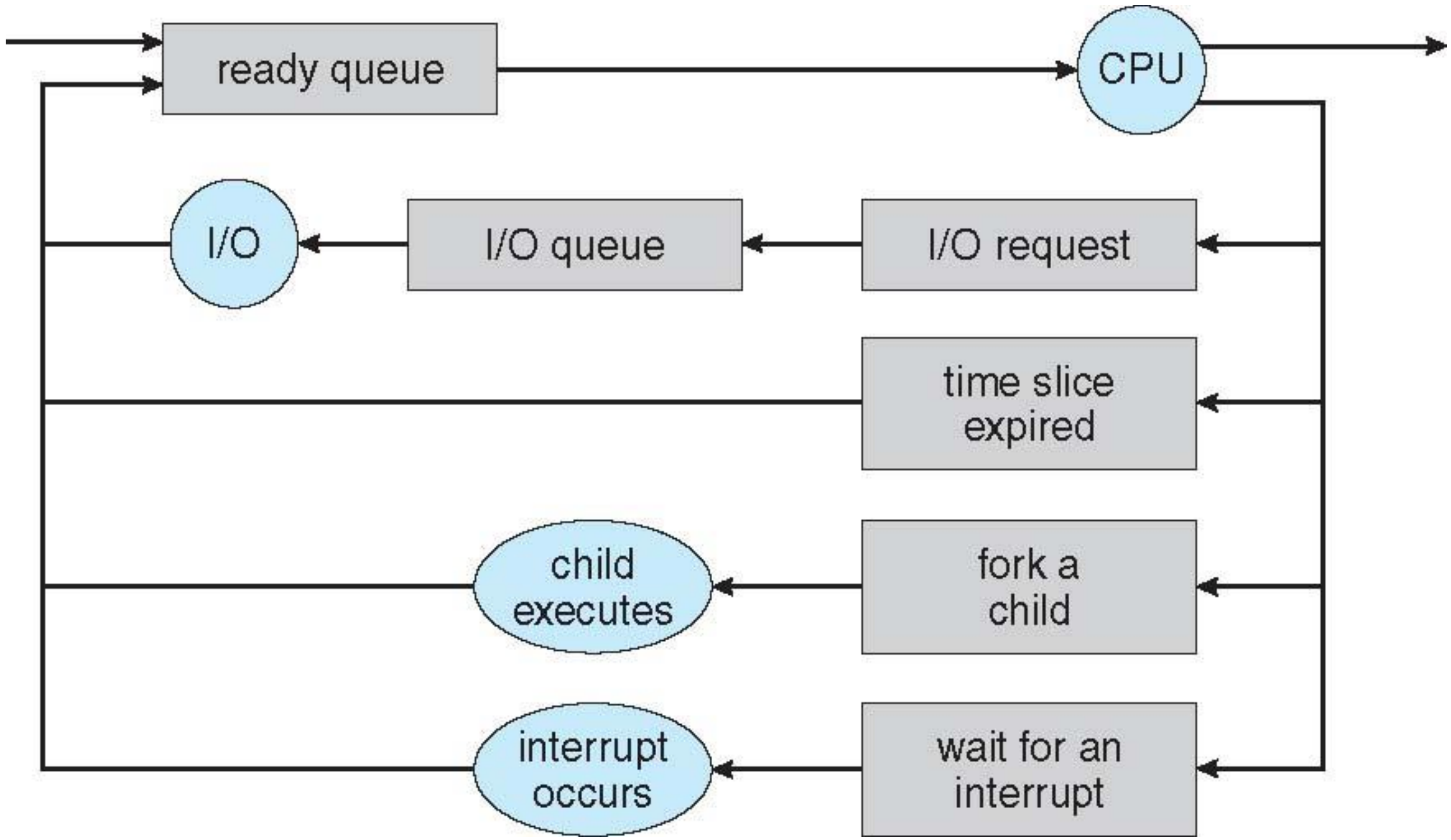


Process Scheduling

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues
- Once the process is assigned to the CPU and is executing, one of several events could occur:
  - The process could issue an I/O request and then be placed in an I/O queue
  - The process could create a new sub-process and wait for its termination
  - The process could be removed forcibly form the CPU, as a result of an interrupt and be put back in the ready queue.

Representation of Process Scheduling

- **Queueing diagram** represents queues, resources, flows



Schedulers

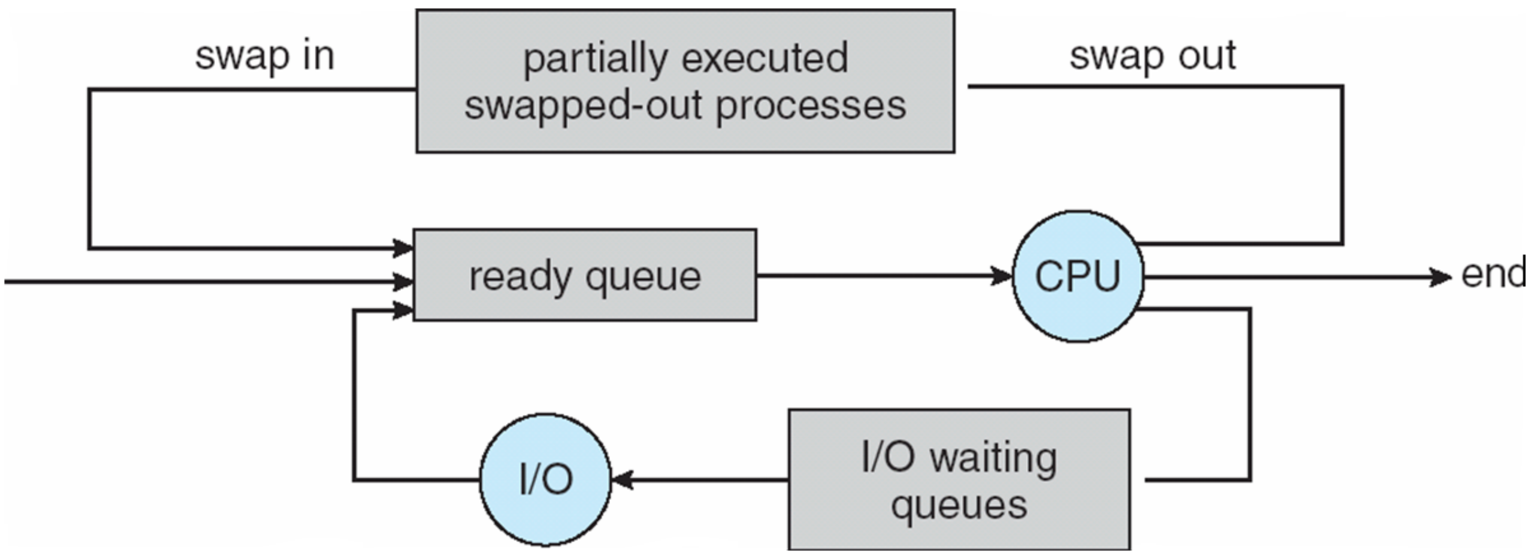
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
- Short-term scheduler is invoked very frequently (milliseconds) $\Rightarrow$ (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) $\Rightarrow$ (may be slow)
- The long-term scheduler controls the **degree of multiprogramming**

Schedulers…

- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts

Addition of Medium Term Scheduling

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**



Context Switch

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB -> longer the context switch
- Time dependent on hardware support

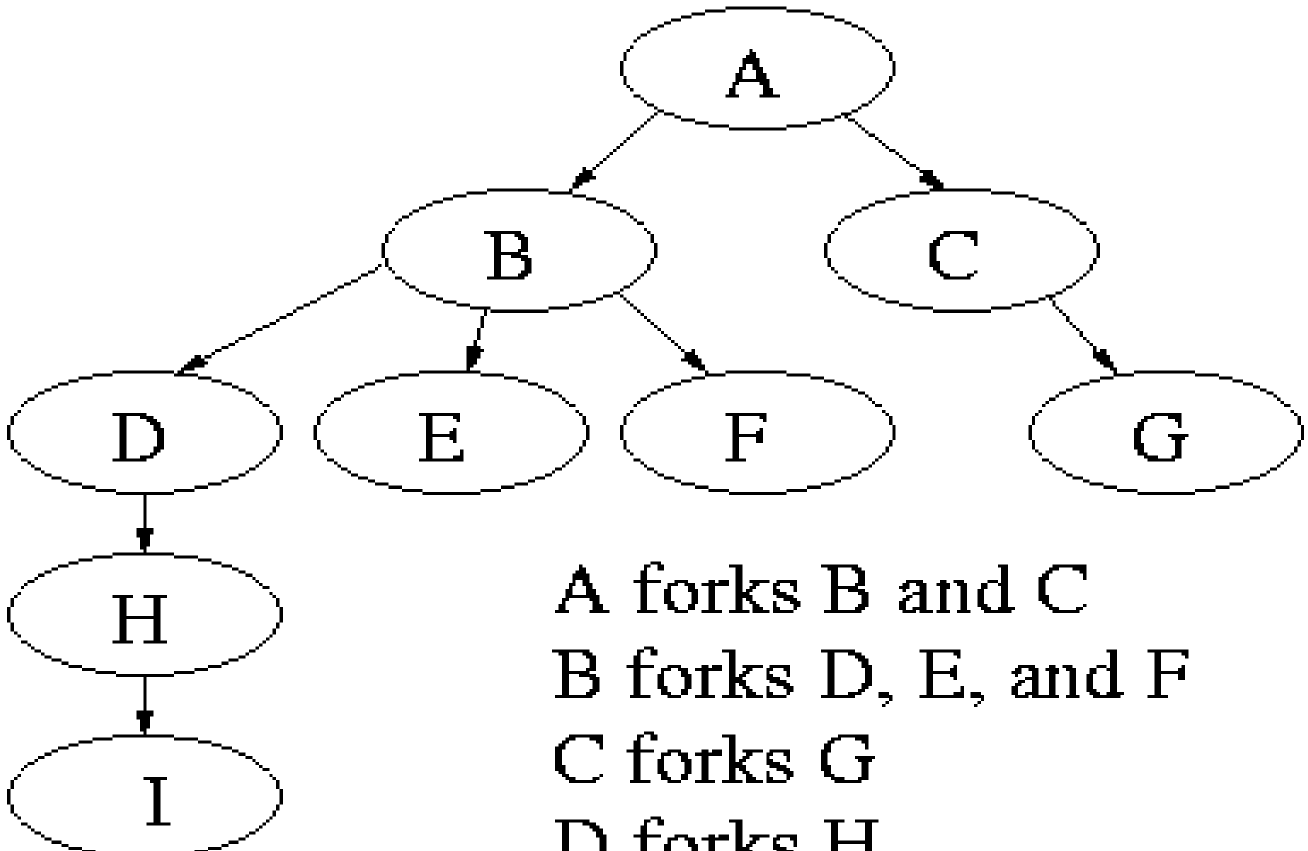
Operations on Processes

- Process creation
- Process termination

Process Creation

- **Parent** process create **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier** (**pid**)
- Resource sharing options
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution options
  - Parent and children execute concurrently
  - Parent waits until children terminate

A Tree of Processes in Linux



A forks B and C

B forks D, E, and F

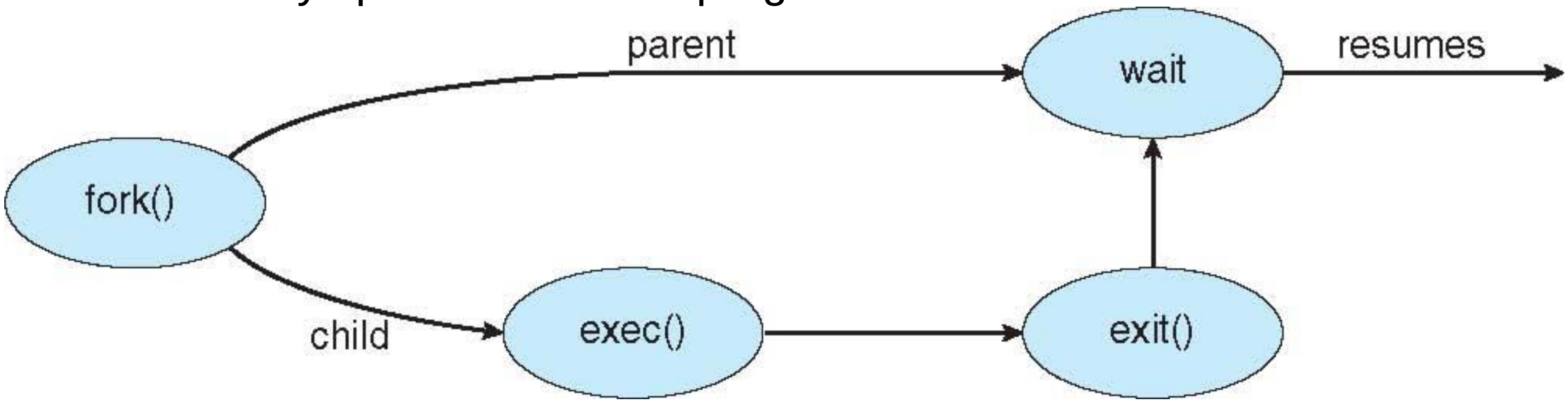
C forks G

D forks H

H forks I

Process Creation (Cont.)

- Address space
  - Child duplicate of parent
  - Child has a program loaded into it
- 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



Process Termination

- Process executes last statement and asks the operating system to delete it (**`exit()`**)
  - Output data from child to parent (via **`wait()`**)
  - Process' resources are de-allocated by operating system
- Parent may terminate execution of children processes (**`abort()`**)
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting
    - Some operating systems do not allow child to continue if its parent terminates
      - All children terminated - **cascading termination**

Multiprocess Architecture – Chrome Browser

- Many web browsers ran as single process (some still do)
  - If one web site causes trouble, entire browser can hang or crash
- Google Chrome Browser is multiprocess with 3 categories
  - **Browser** process manages user interface, disk and network I/O
  - **Renderer** process renders web pages, deals with HTML, Javascript, new one for each website opened.
  - **Plug-in** process for each type of plug-in.

Interprocess Communication

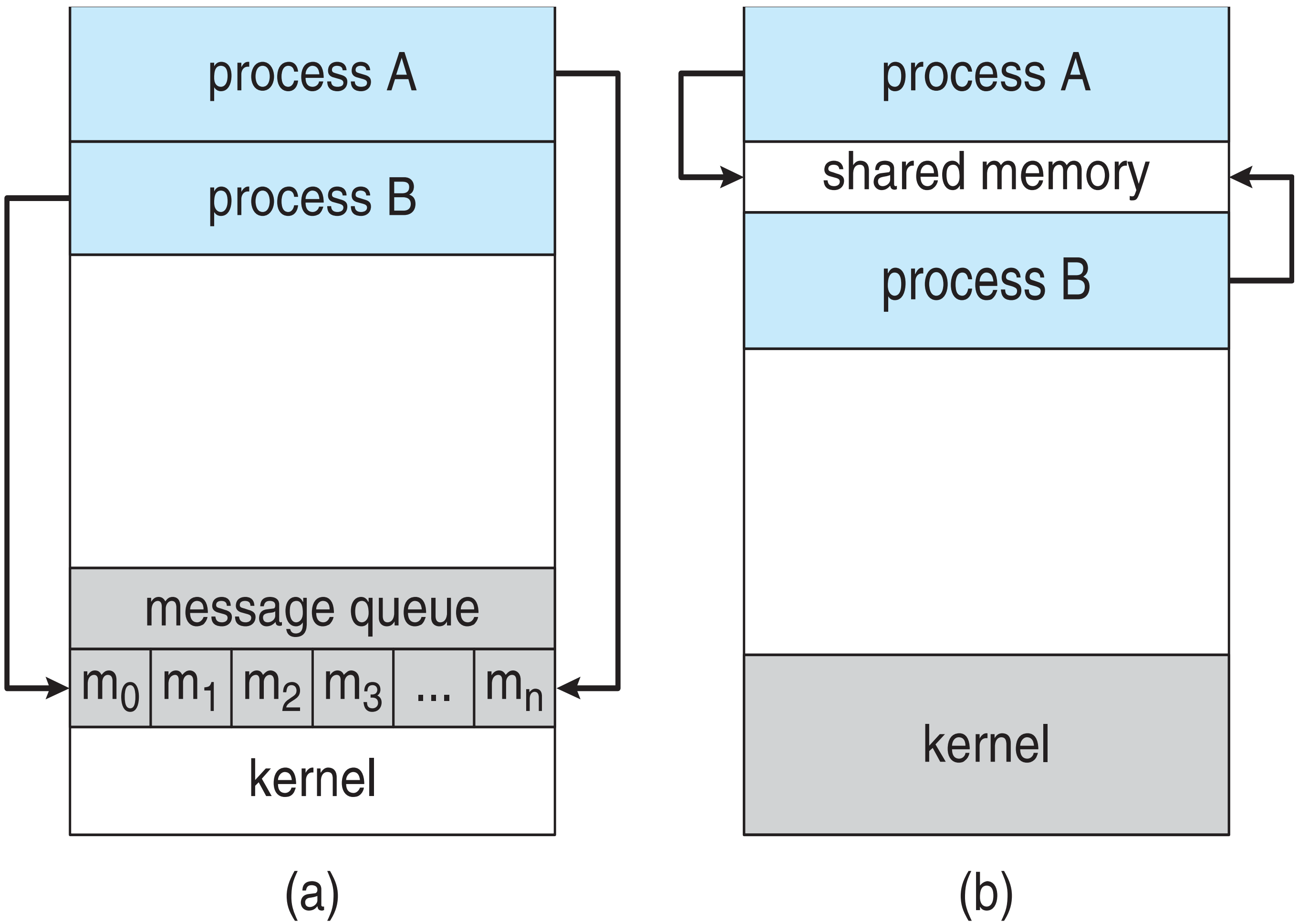
- Processes within a system may be ***independent*** or ***cooperating***
- Cooperating process can affect or be affected by other processes, including sharing data
- Advantages of process cooperation
  - Information sharing : Since several users may be interested in the same piece of information, we must provide and environment to allow concurrent access to these types of resources.
  - Computation speed-up: If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others.
  - Modularity: We may want to construct the system in a modular fashion, dividing the system functions into separate processes.
  - Convenience: Even an individual user may have many tasks on which to work at one time.

Inter-process Communication…

- Cooperating processes need **inter-process communication** (**IPC**)
- Two models of IPC
  - **Shared memory**
  - **Message passing**

- In shared memory model, a region of memory that is shared by cooperating processes is established. Processes can then exchange information by reading and writing data to the shared region.
- In the message passing model, communication takes place by means of message exchanged between the cooperating processes.

Communications Models



Assignment

Write a C program that illustrates the creation of child process using fork system call. One process finds sum of even series and other process finds sum of odd series