Part 2

Processes and Threads

2.1 Processes
2.2 Threads
2.3 Interprocess communication
2.4 Scheduling

2.1 Processes

Processes
The Process Model

- (a) Multiprogramming of four programs
- (b) Conceptual model of 4 independent, sequential processes
- (c) Only one program active at any instant

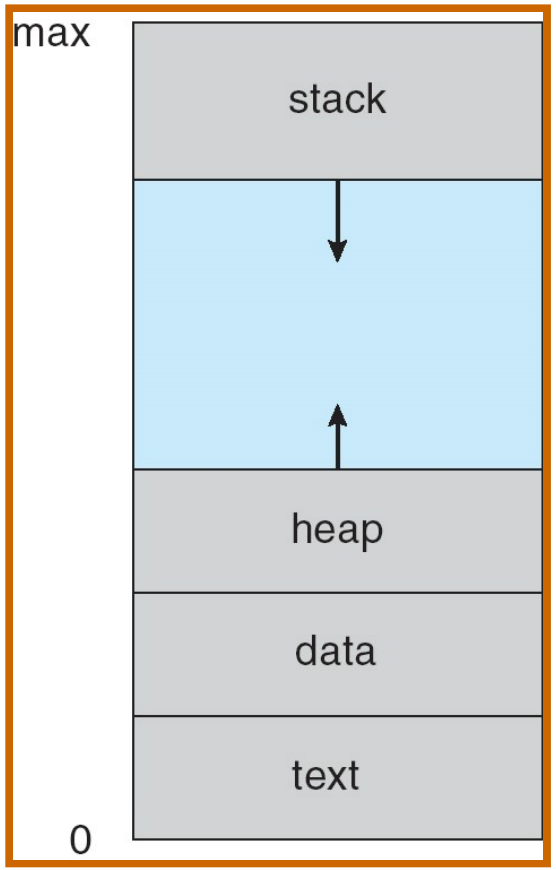
Processes

Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Process – a program in execution; process execution must progress in sequential fashion
- A process resources includes:
  - Address space (text segment, data segment)
  - CPU (virtual)
    - program counter
    - registers
    - stack
  - Other resource (open files, child processes…)

Processes

Process in Memory



Processes

Process Creation (1)

Principal events that cause process creation

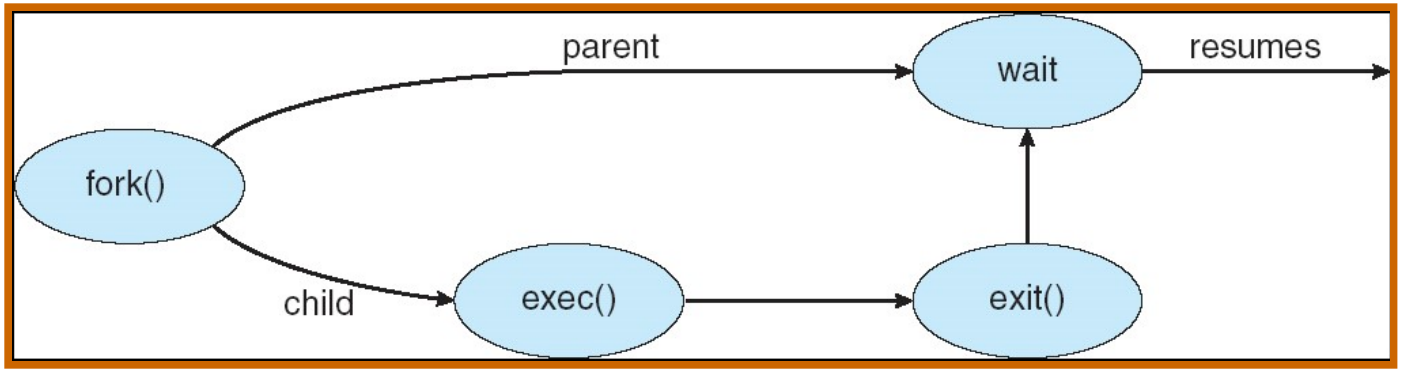
1. System initialization
2. Execution of a process creation system Call
3. User request to create a new process
4. Initiation of a batch job

Processes
Process Creation (2)

- 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

Processes

Process Creation (3) : Example



Processes
Process Termination

Conditions which terminate processes

1. Normal exit (voluntary)
2. Error exit (voluntary)
3. Fatal error (involuntary)
4. Killed by another process (involuntary)

Processes
Process Hierarchies

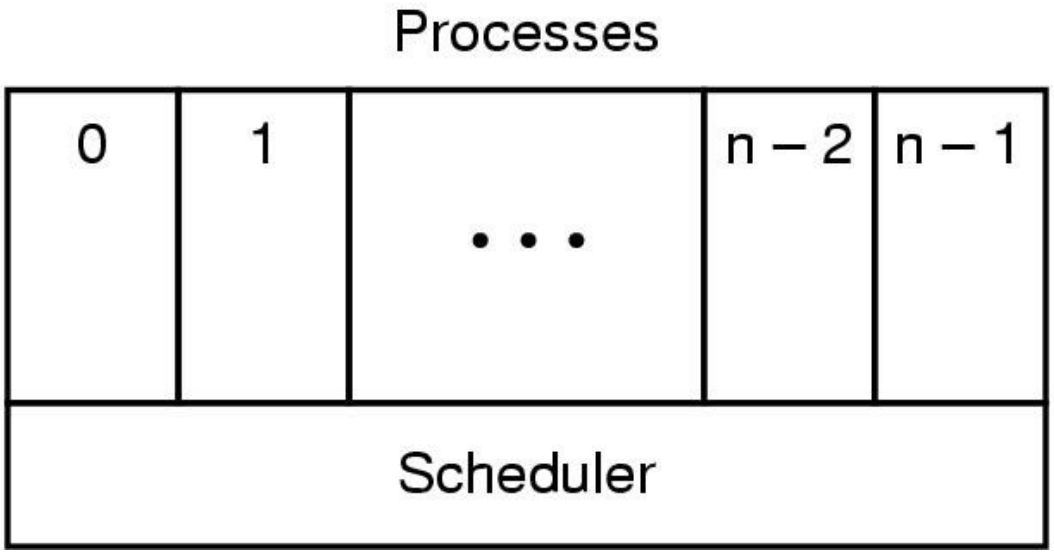
- Parent creates a child process, child processes can create its own process
- Forms a hierarchy
  - UNIX calls this a "process group"
- Windows has no concept of process hierarchy
  - all processes are created equal

Processes
Process States (1)

- Possible process states
  - running
  - blocked
  - ready
- Transitions between states shown

Processes

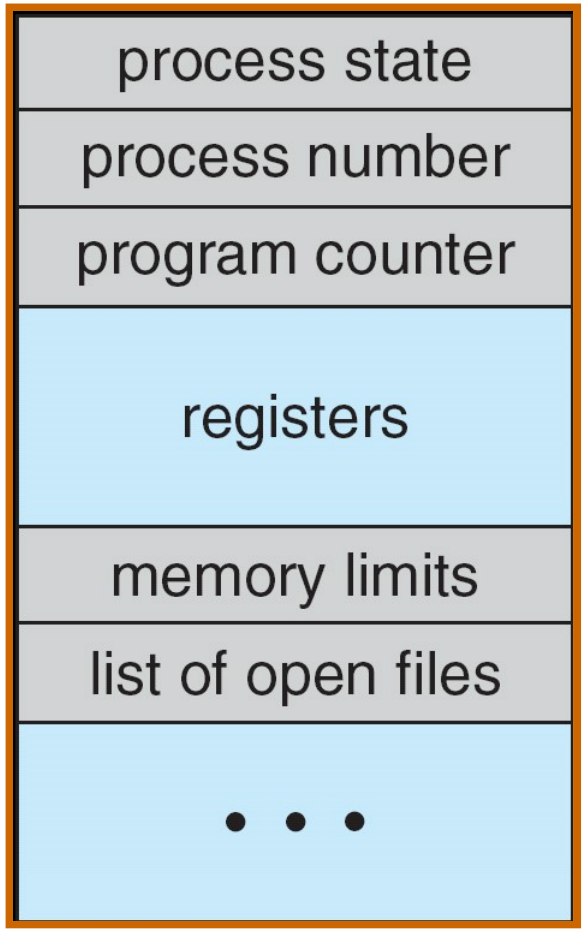
Process States (2)



- Lowest layer of process-structured OS
  - handles interrupts, scheduling
- Above that layer are sequential processes

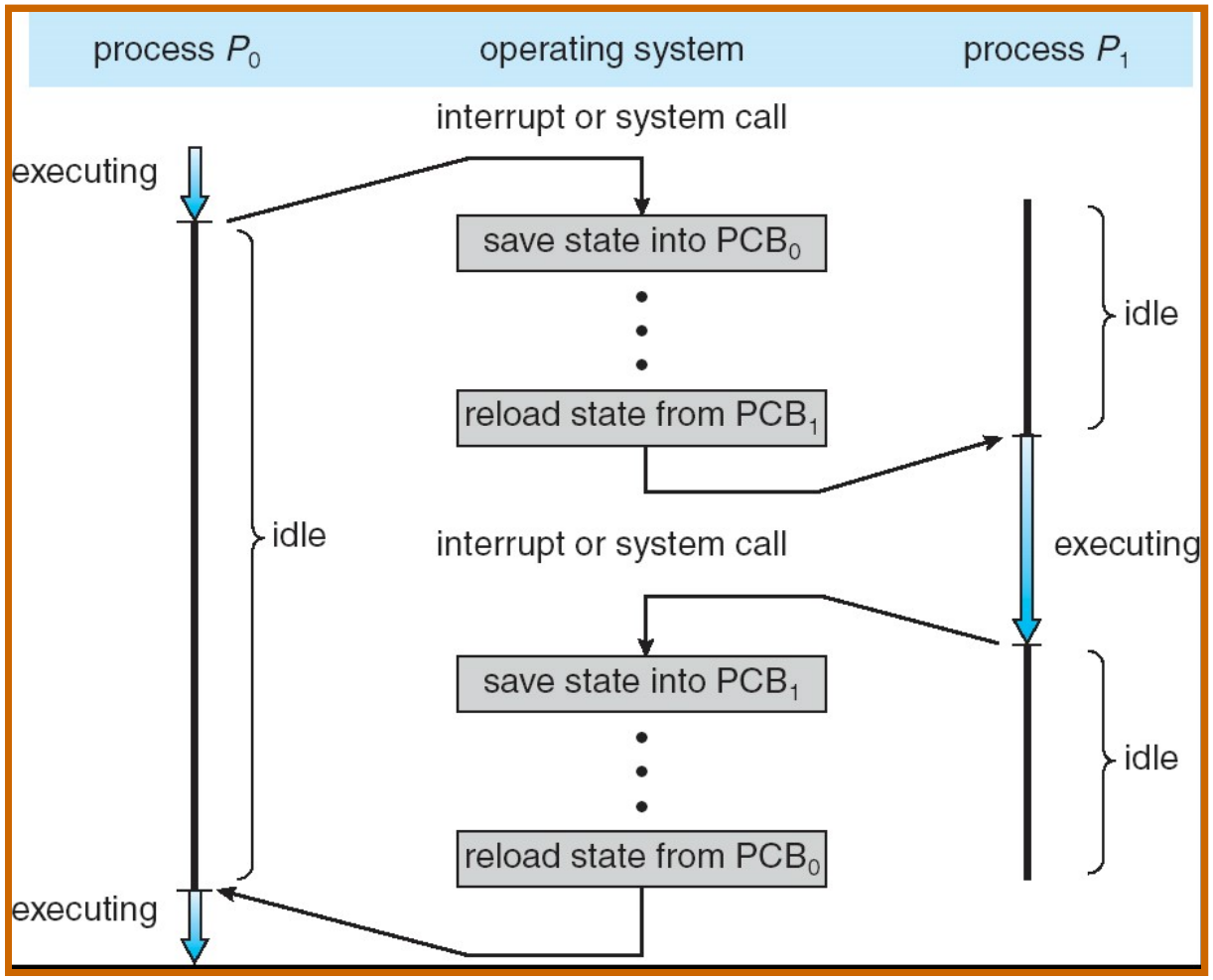
Processes

Process Control Block (PCB)



Processes

context switch



Processes

Implementation of Processes (1)

Fields of a process table entry

Modeling Multiprogramming

CPU utilization as a function of the number of processes in memory.

2.2 Threads

Threads

The Thread Model

(a) Three processes each with one thread

(b) One process with three threads

- A thread – Lightweight Process is a basic unit of CPU utilization

Threads
Process with single thread

- A process (heavyweight):
  - Address space (text section, data section)
  - Single thread of execution
    - program counter
    - registers
    - Stack
  - Other resource (open files, child processes…)

Threads

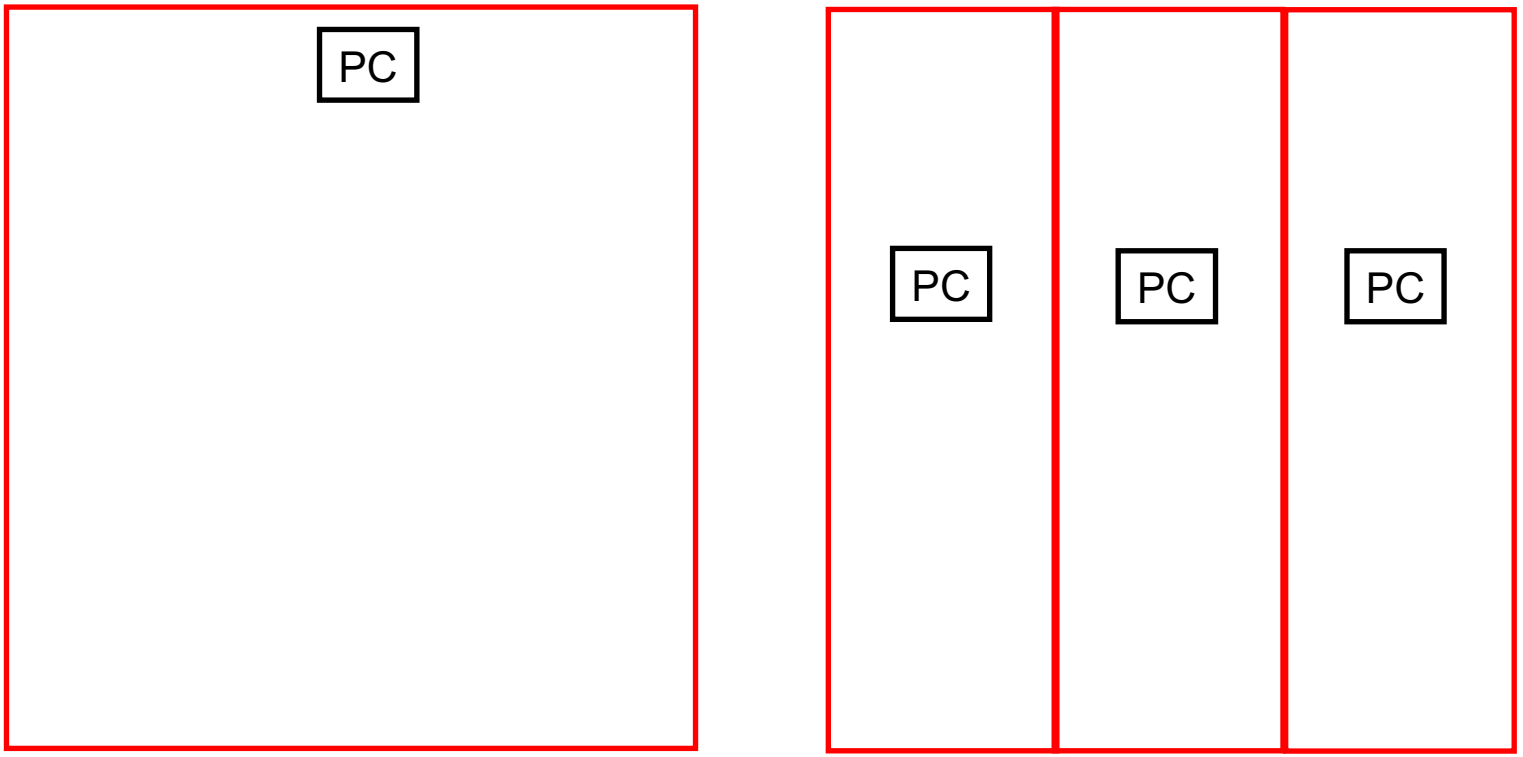
Process with multiple threads

Multiple threads of execution in the same environment of process

- Address space (text section, data section)
- Multiple threads of execution, each thread has private set:
  - program counter
  - registers
  - stack
- Other resource (open files, child processes…)

Threads

Single and Multithreaded Processes



Threads

Items shared and Items private

- Items shared by all threads in a process
- Items private to each thread

Threads
Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of Multiprocessor Architectures

Threads

Thread Usage (1)

A word processor with three threads

Threads

Thread Usage (2)

A multithreaded Web server

Threads
Thread Usage (3)

- Rough outline of code for previous slide

  (a) Dispatcher thread

  (b) Worker thread

Threads

Implementing Threads in User Space (1)

A user-level threads package

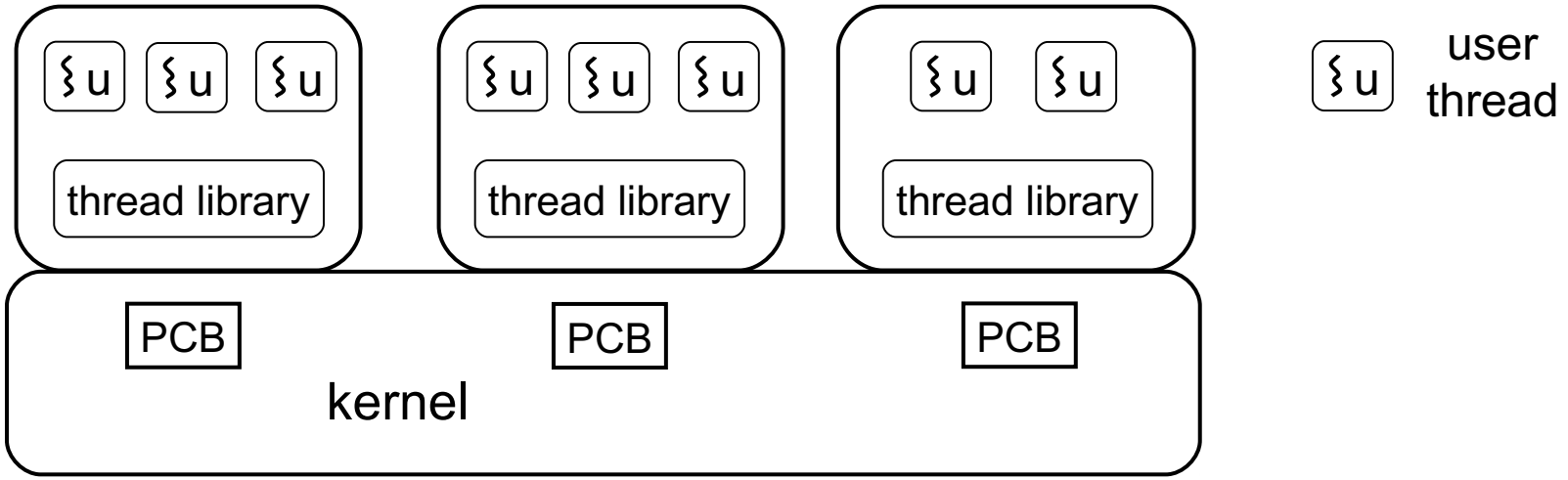
Threads

Implementing Threads in User Space (2)

- Thread library, (run-time system) in user space
  - thread_create
  - thread_exit
  - thread_wait
  - thread_yield (to voluntarily give up the CPU)
- Thread control block (TCB) (Thread Table Entry) stores states of user thread (program counter, registers, stack)
- Kernel does not know the present of user thread

Threads
Implementing Threads in User Space (3)

- Traditional OS provide only one "kernel thread" presented by PCB for each process.
  - *Blocking problem:* If one user thread is blocked ->the kernel thread is blocked, -> all other threads in process are blocked.



Threads

Implementing Threads in the Kernel (1)

A threads package managed by the kernel

Threads
Implementing Threads in the Kernel (2)

- Multithreading is directly supported by OS:
  - Kernel manages processes and threads
  - CPU scheduling for thread is performed in kernel
- Advantage of multithreading in kernel
  - Is good for multiprocessor architecture
  - If one thread is blocked does not cause the other thread to be blocked.
- Disadvantage of Multithreading in kernel
  - Creation and management of thread is slower

Threads
Hybrid Implementations

Multiplexing user-level threads onto kernel- level threads