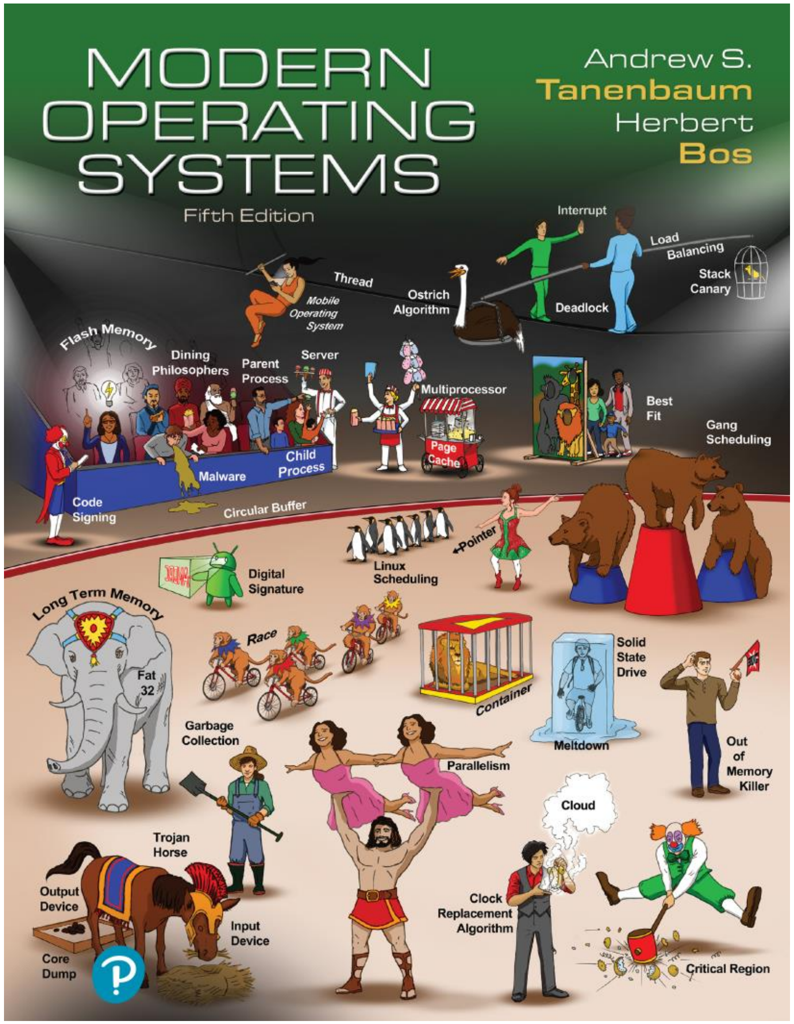
Modern Operating Systems

Fifth Edition



Chapter 2

Processes

Pearson

The Process Model

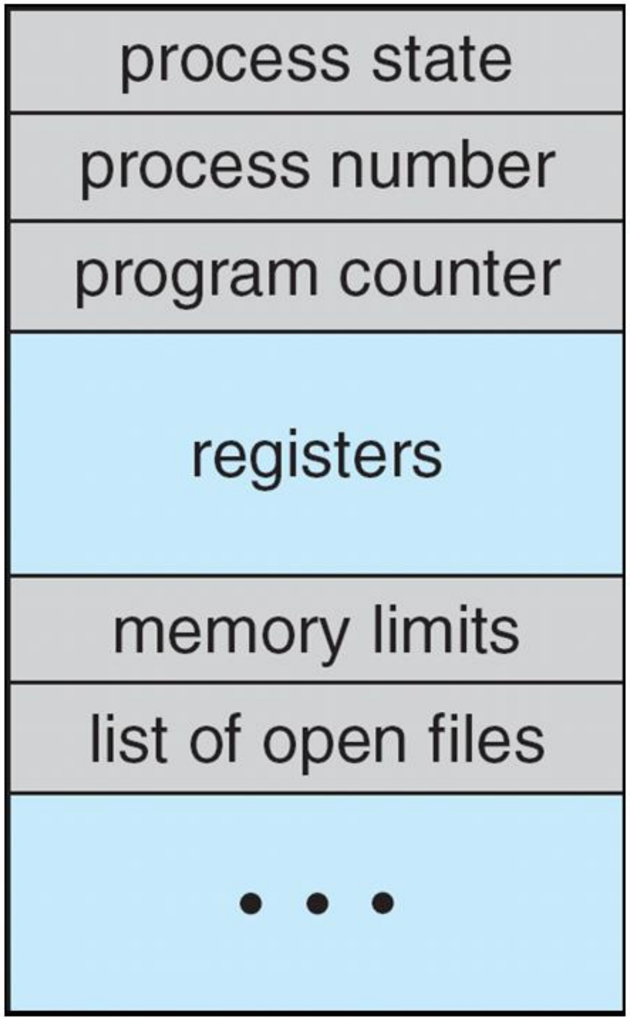
Process = Program in execution

- How many processes for each program?
- A fundamental operating system **abstraction**
- Allows the OS to simplify:
  - Resource **allocation**
  - Resource **accounting**
  - Resource **limiting**
- OS maintains information on the resources and the internal state of every single process in the system

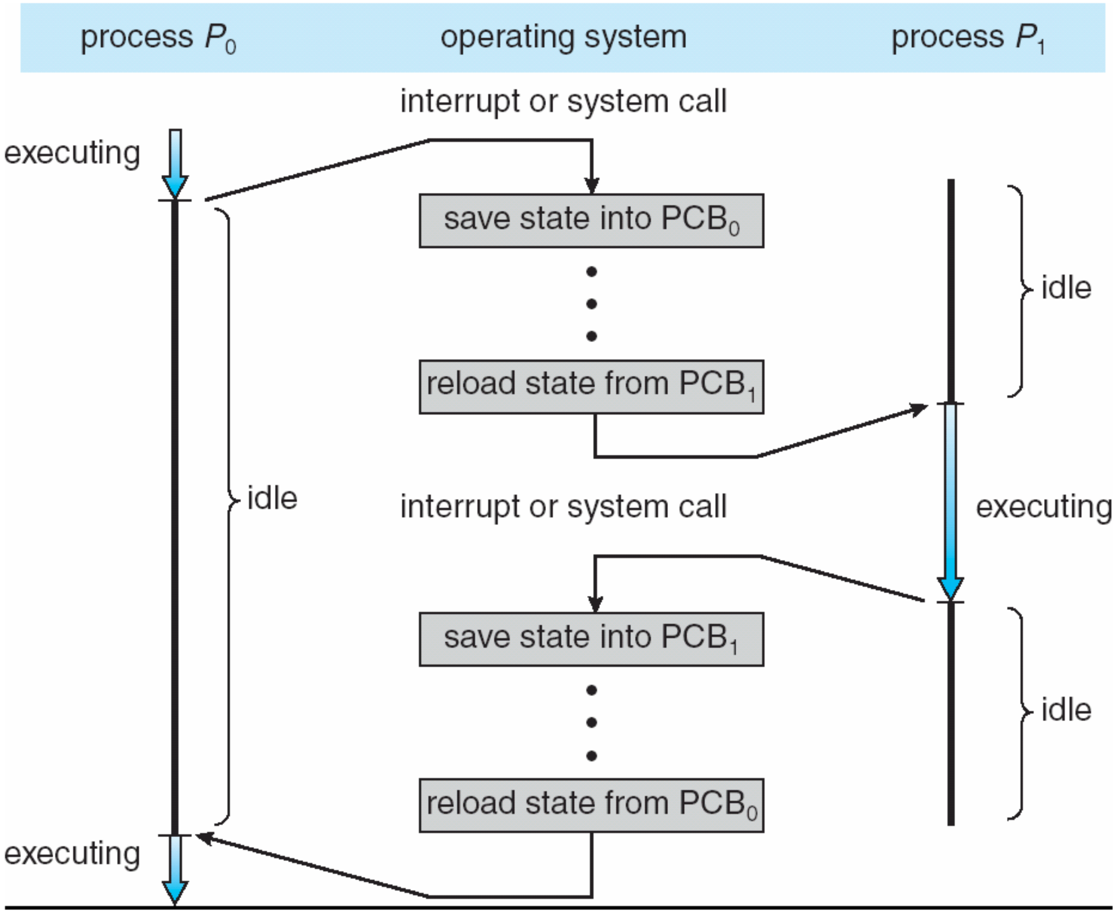
The Process Model

Information associated with each process (**process control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files



CPU Switch From Process to Process

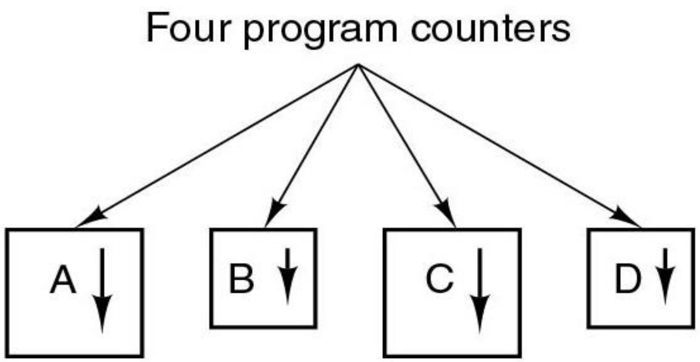


Context Switch

- 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 ➔ the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU ➔ multiple contexts loaded at once

The Process Model

- Each process has own flow of control
(own logical program counter)
- Each time we switch processes, we save the program counter of first process and restore the program counter of the second

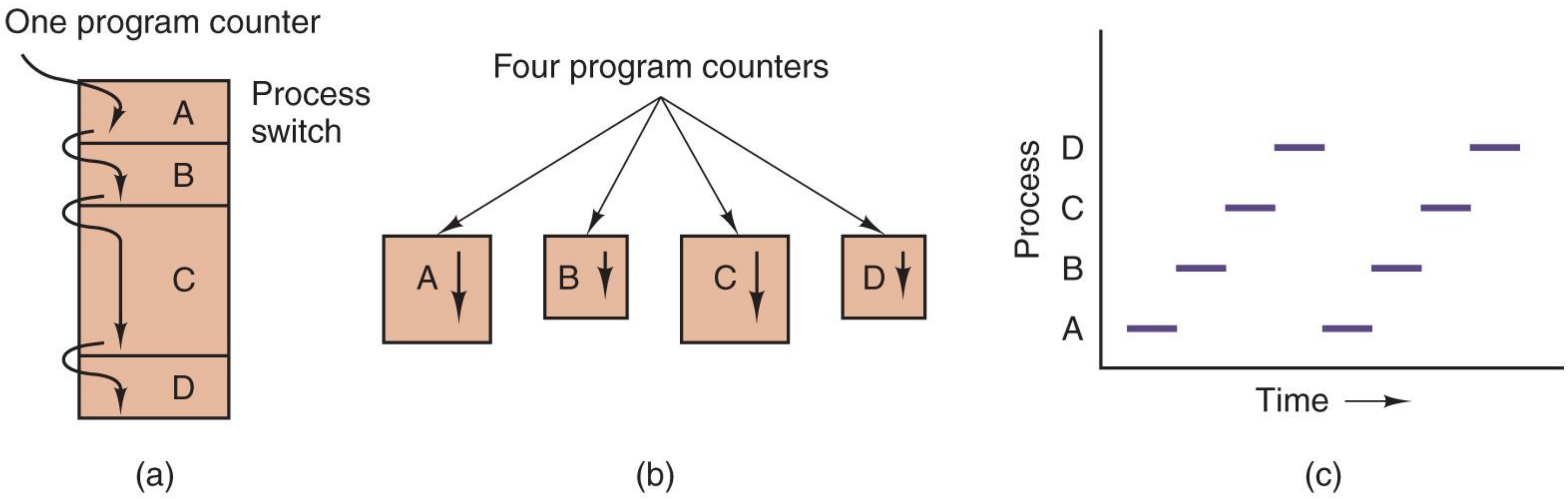


(b)

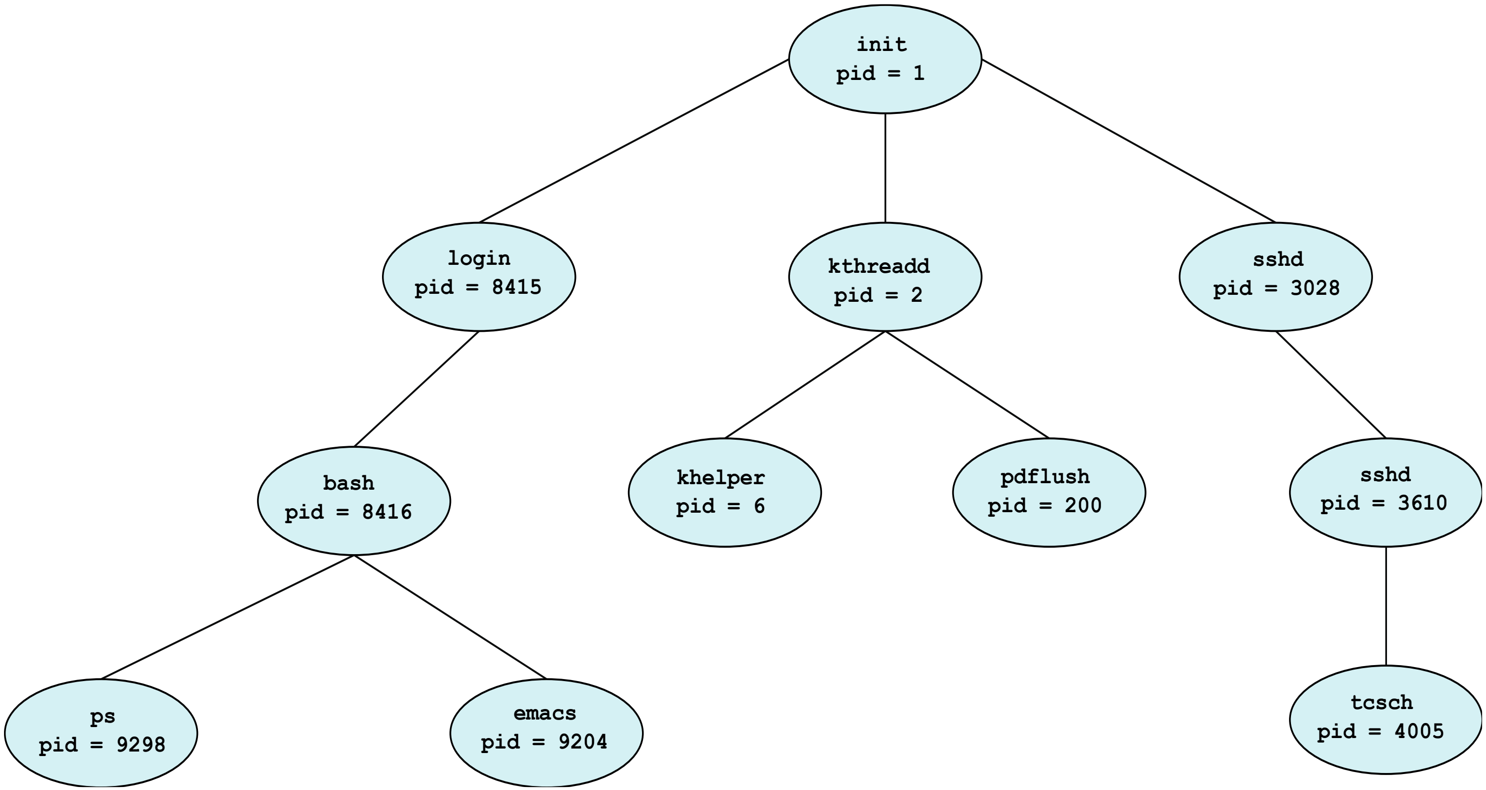
(b) Conceptual model of four independent, sequential processes.

Concurrent Processes

- So far, process has a single thread of execution
  - Consider having multiple program counters per process
  - Multiple locations can execute at once
  - Multiple threads of control -> threads



Process Hierarchies



OS typically creates only 1 `init` process

Subprocesses created independently:

- A parent process can create a child process
- This results in a tree-like structure and process groups

Process Creation

Four principal events that cause processes to be created:

1. System initialization
2. Execution of a process creation system call by a running process
3. A user request to create a new process
4. Initiation of a batch job

Pearson

Process Termination

Typical conditions which terminate a process:

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

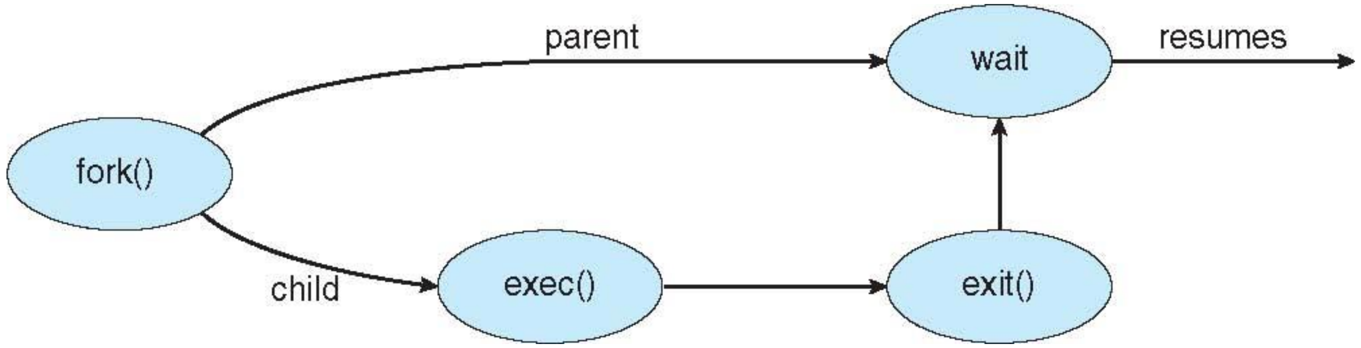
- The parent process may wait for termination of a child process by using the **`wait()`** system call. The call returns status information and the pid of the terminated process

```
pid = wait(&status);
```

- If no parent waiting (did not invoke **`wait()`**) process is a **zombie**
- If parent terminated without invoking **`wait`**, process is an **orphan**

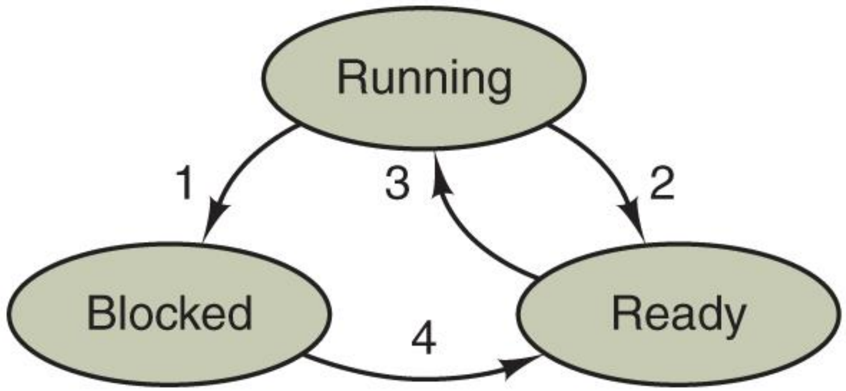
Process Management

- `fork:` create a new process
  - Child is a “private” **clone** of the parent
  - Shares **some** resources with the parent
- `exec:` execute a new process image
  - Used in combination with `fork`
  - `exec` on Windows?
- `exit:` cause voluntary process termination
  - `Exit` status returned to the parent
  - Involuntary process termination?
- `kill:` send a signal to a process (or group)
  - Can cause involuntary process termination



Process States

A process can be in running, blocked, or ready state. Transitions between these states are as shown.

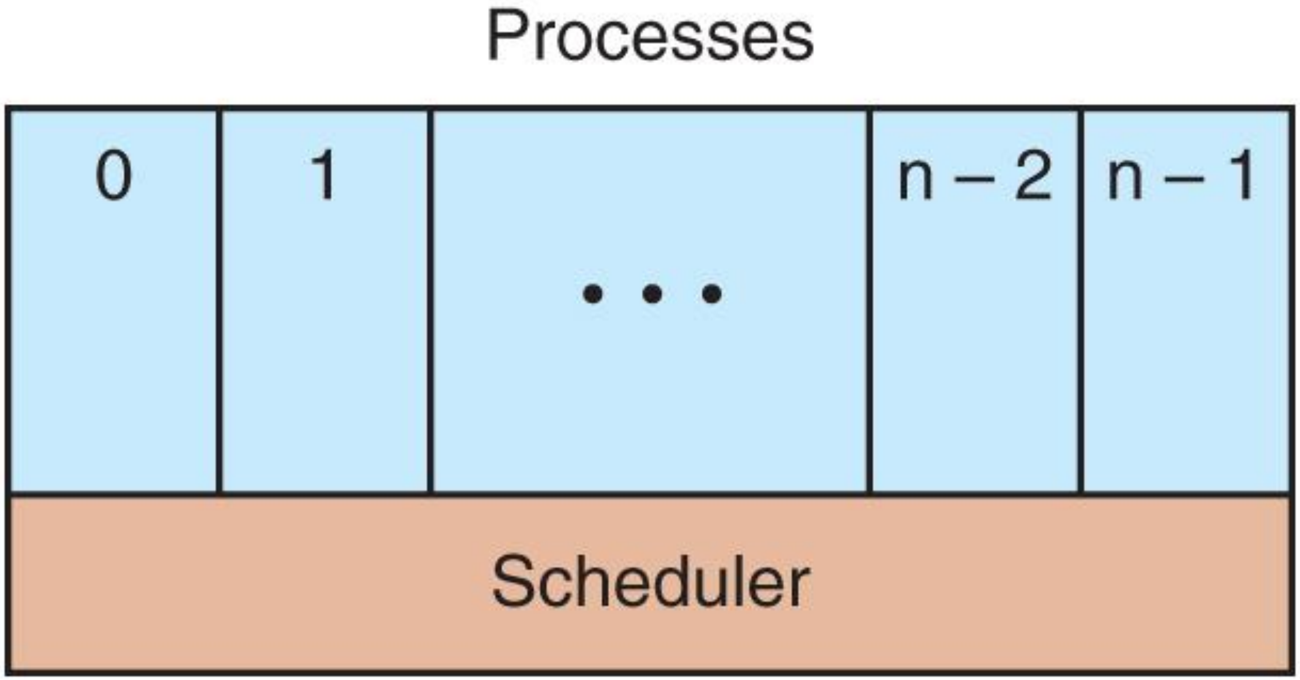


1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

A process can be in running, blocked, or ready state.
Transitions between these states are as shown.

Process States

- Scheduler periodically switches processes
- Sequential processes lay on the layer above
- This leads to a simple process organization



The lowest layer of a process-structured operating system handles interrupts and scheduling. Above that layer are sequential processes.

Interrupts

- Idea: to deallocate the CPU in favor of the scheduler, we rely on hardware-provided interrupt handling support
- Allows the scheduler to periodically get control, i.e., whenever the hardware generates an interrupt
- Interrupt vector:
  - Associated with each I/O device and interrupt line
  - Part of the interrupt descriptor table (IDT)
  - Contains the start address of an OS-provided internal procedure (interrupt handler)
- The interrupt handler continues the execution
- Interrupt types: sw, hw device (async), exceptions

Implementation of Processes

Overview of what the lowest level of the operating system does when an interrupt occurs.

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Every time an interrupt occurs, the scheduler gets control → acts as a mediator

A process cannot give the CPU to another process (context switch) without going through the scheduler

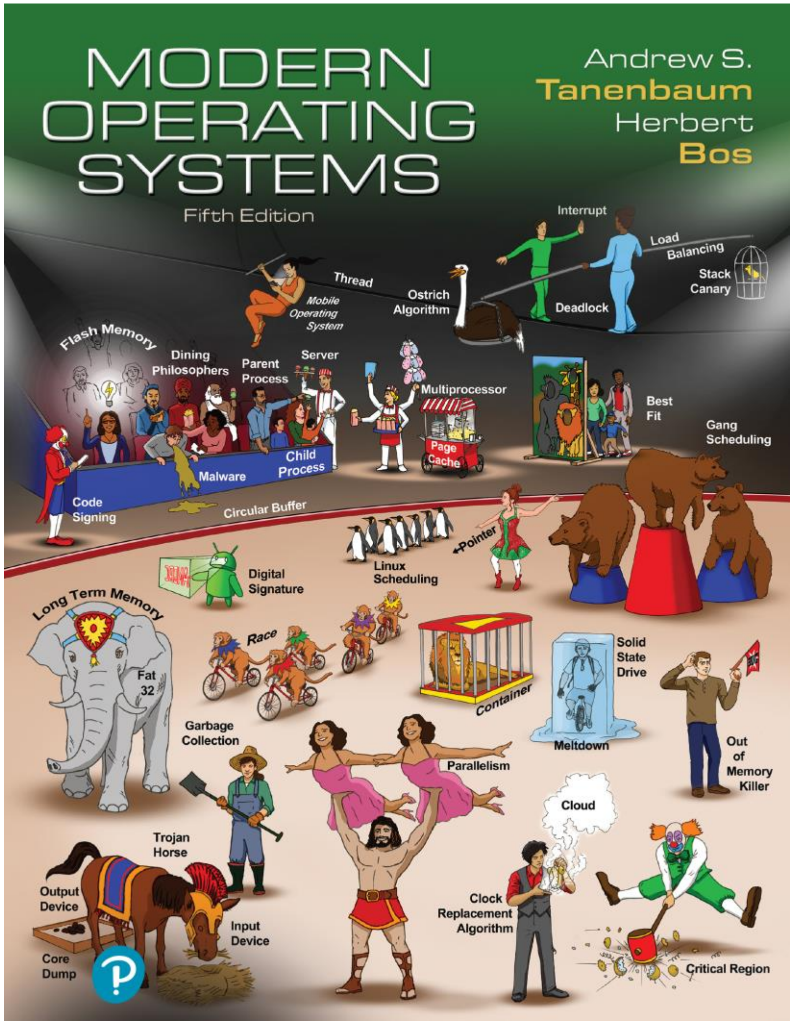
C Program Forking Separate Process

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
      fprintf(stderr, "Fork Failed");
      return 1;
    }
    else if (pid == 0) { /* child process */
      execlp("/bin/ls","ls",NULL);
    }
    else { /* parent process */
      /* parent will wait for the child to complete */
      wait(NULL);
      printf("Child Complete");
    }

    return 0;
}
```

Pearson

Modern Operating Systems

Fifth Edition



End of Processes

Pearson