Processes

Qinglei Cao
CSCI 2510 – Principles of Computing Systems
Saint Louis University
St. Louis, MO 63103

Definition

A *process* is a program in execution

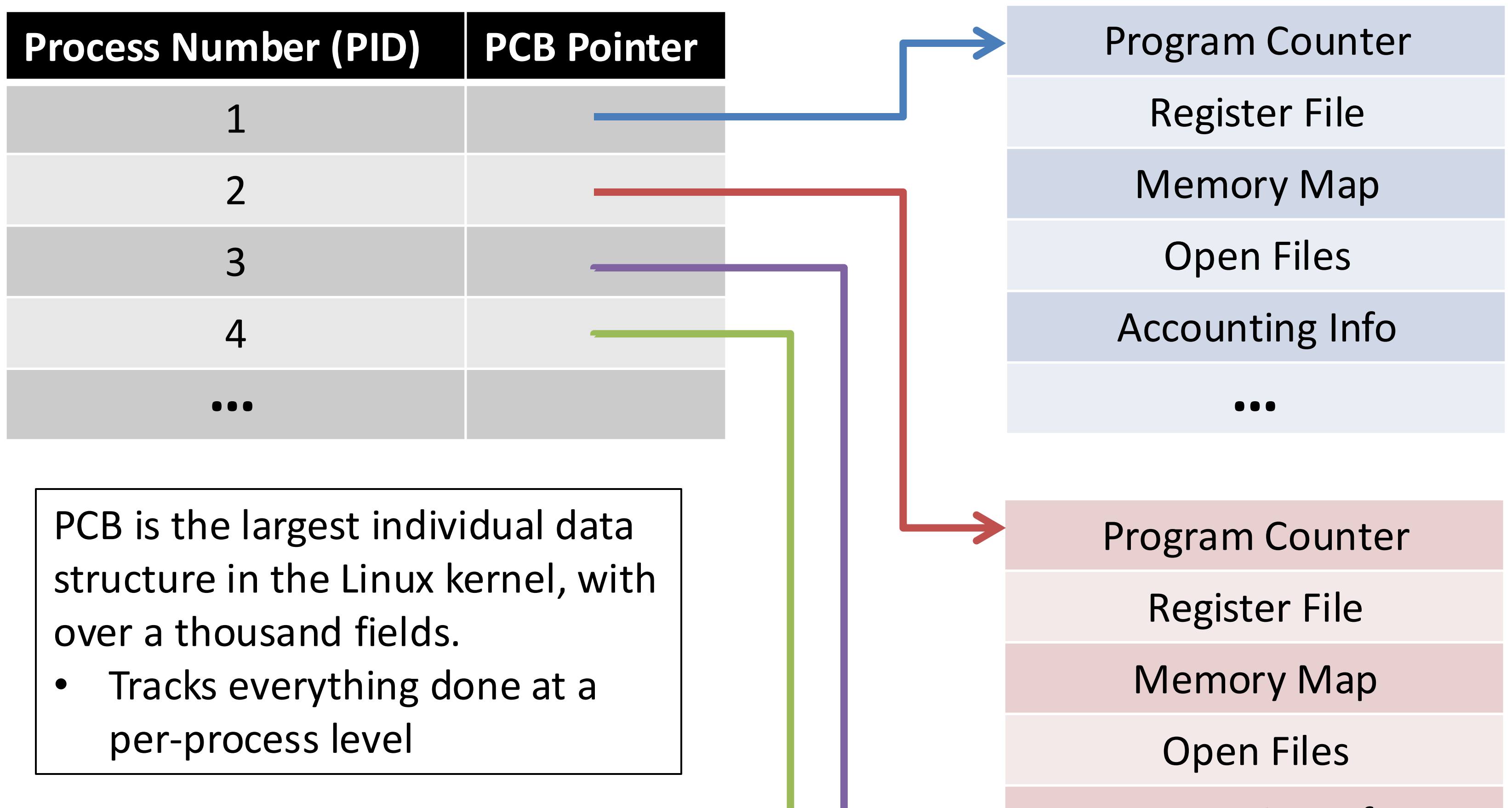
- A process has internal state that evolves, a program on the hard drive does not
- You can have multiple copies of the same program executing

Processes are the fundamental abstraction for managing a key computer resource: processor time

- Each process runs independently
- Allows for *pseudo-concurrency* by allowing multiple programs to execute "concurrently"

Kernel Process Data Structures

The OS must keep track of all running processes.



Process Table

Process Number (PID)	PCB Pointer
1	
2	
3	
4	
…	

Process Control Block (PCB)

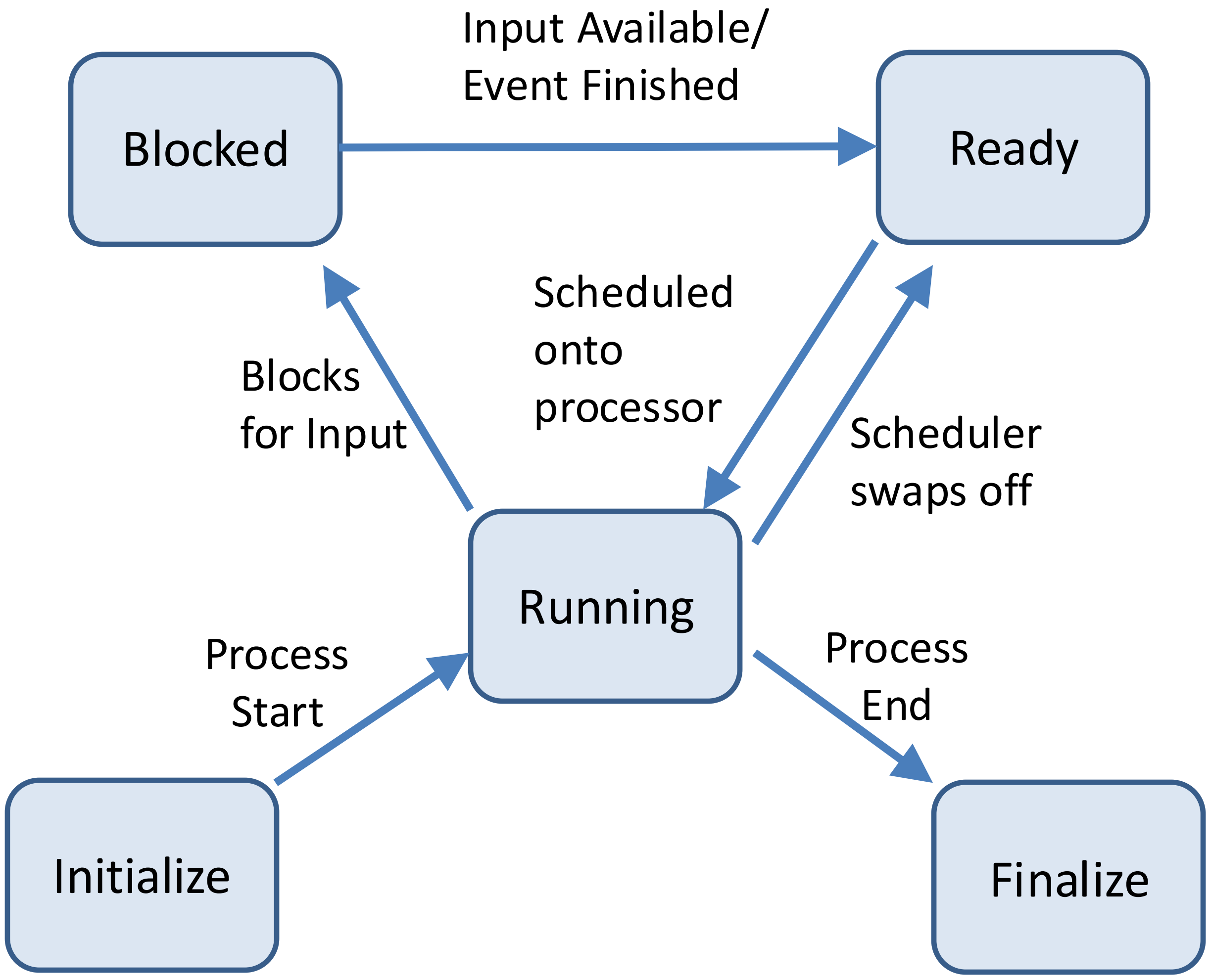
Program Counter
Register File
Memory Map
Open Files
Accounting Info
…

Program Counter
Register File
Memory Map
Open Files

PCB is the largest individual data structure in the Linux kernel, with over a thousand fields.

- Tracks everything done at a per-process level

Process State Diagram



Lifecycle of a process:

Spends most of its life between ready and running.

When a process can't continue it ***blocks***.

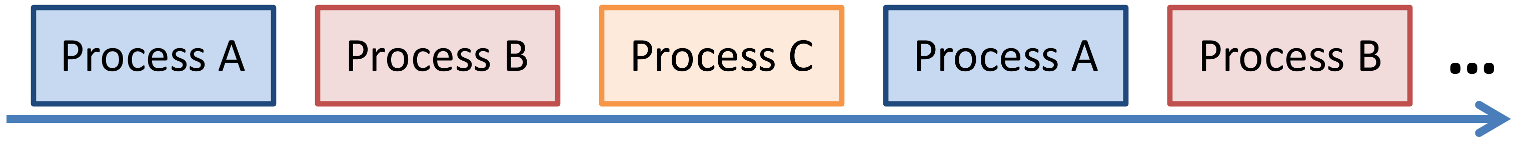
Blocking examples:

- "Press any key to continue"
- Wait for hard drive to become available
- Wait for lock/mutex to become available

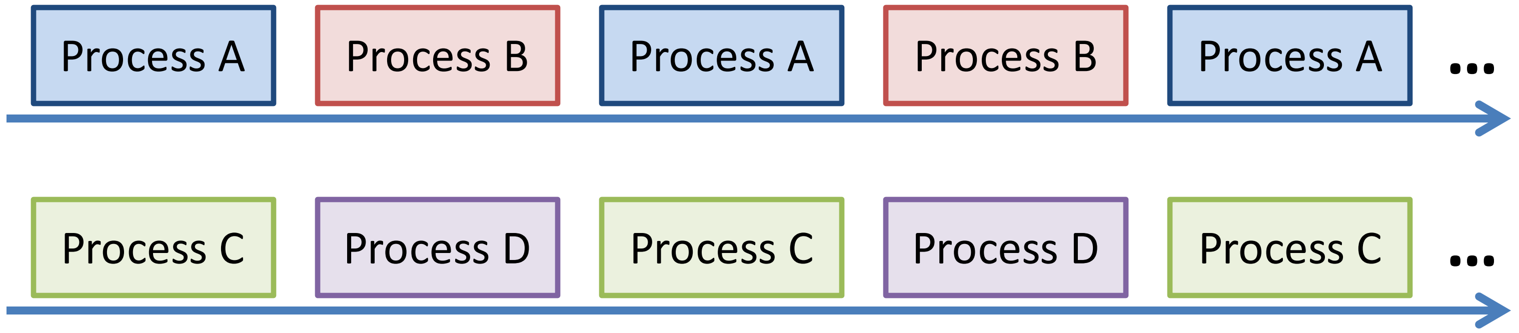
Multi-Programming

The *CPU scheduler* shares the processor among running processes

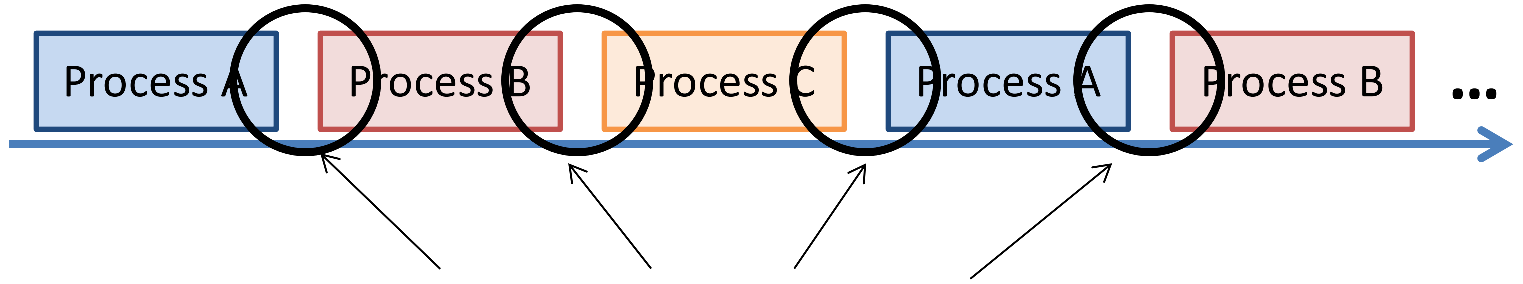
- Single processor



- Multi-processor



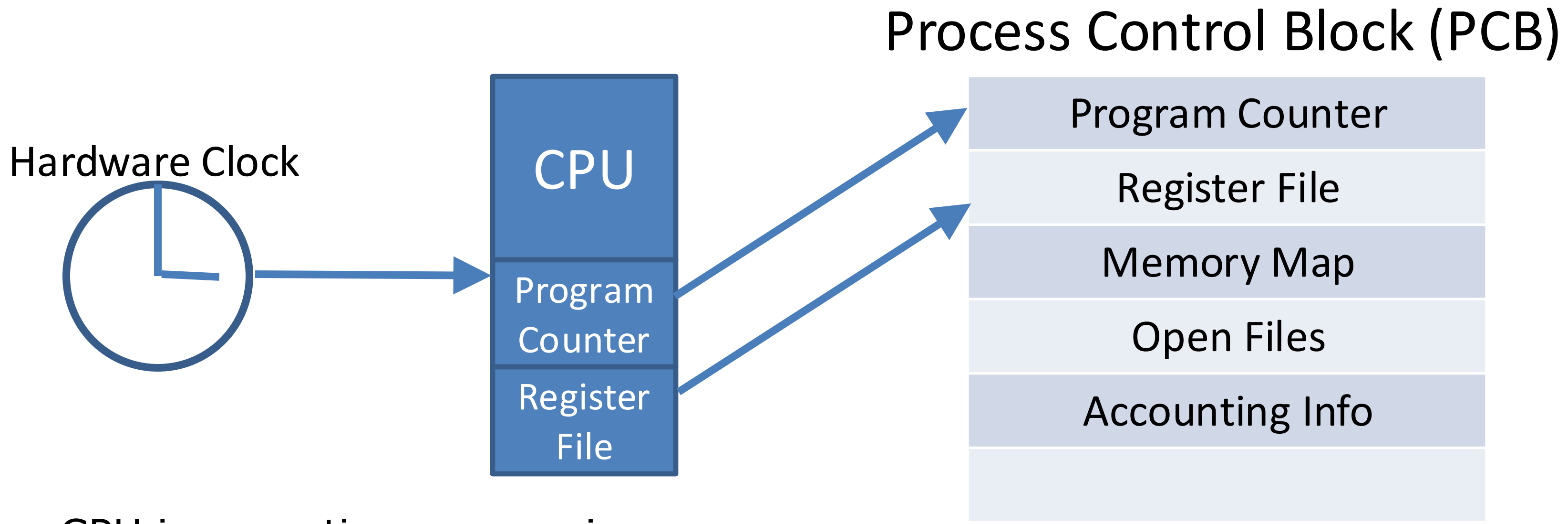
Context Switch



A *context switch* occurs when the process currently executing on a processor changes.

- So called because the processor context (program counter, register file) is swapped out.
- **Not apparent from the process point of view.**
- On desktop Linux, commonly happens 1000 times per second. Slower on mobile devices for better battery life.

Context Switch Mechanism



1. CPU is executing process in userspace
2. Hardware timer interrupt occurs
3. CPU saves PC and register file in PCB
4. Jumps to Interrupt Service Routine (ISR)
5. ISR saves other process state
6. ISR passes control to OS scheduler
7. OS Scheduler picks next process to run
8. Loads next process state from PCB to processor

Process never knows it was interrupted!