Threads and Real Time Processes

Modern Scheduling Issues

(Lecture 15)

Classical Processes

- One Address Space
- One Flow of Control

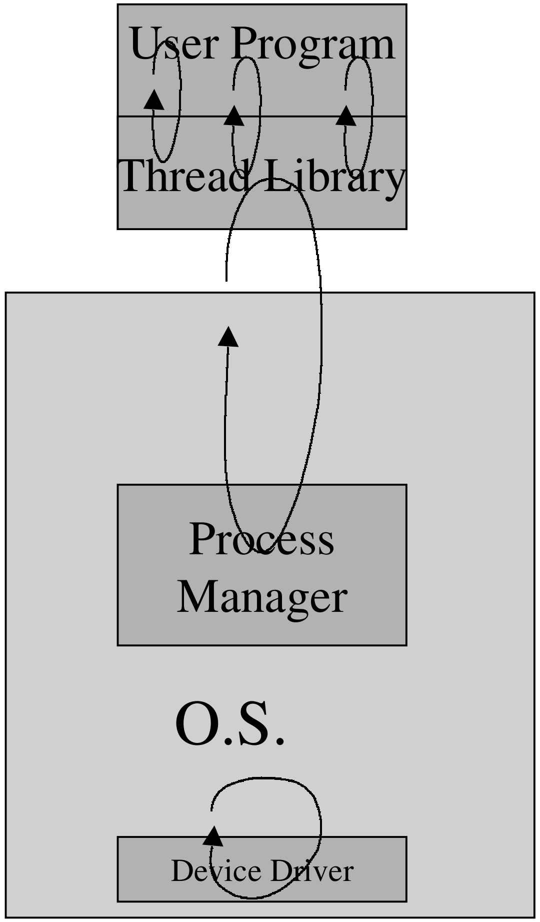
Is this ideal?

- OS has multiple flows of control
- May applications need them?
- Examples?

Threads

- Threads are multiple flows of control in one address space
- Each thread needs PC, register values, stack
- All threads share memory, heap, open files, network connections
- Threads need to interact with each other and with the Operating System

Implementing Threads



Threads implemented by user-level library

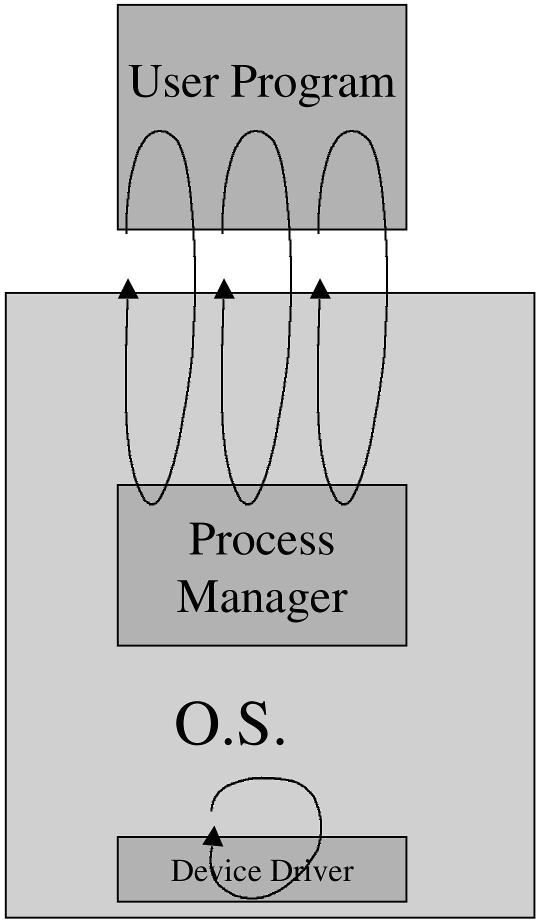
– Thread interaction is fast

But

– OS suspends or schedules whole program

– multiprocessor support?

Native Threads



OS supports Threads

- OS schedules individual threads
- multiprocessor support

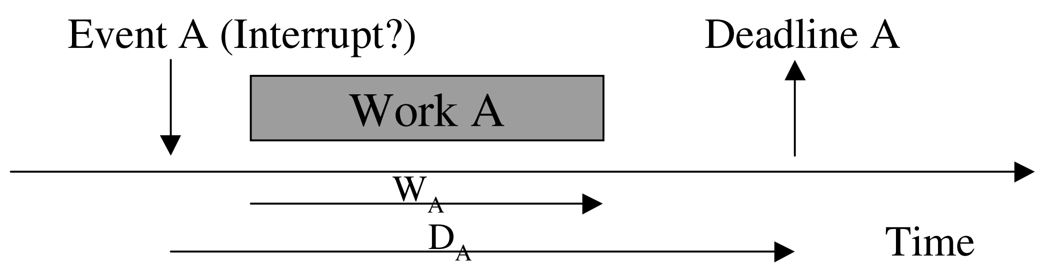
But

- Thread interaction is slow (need system call)

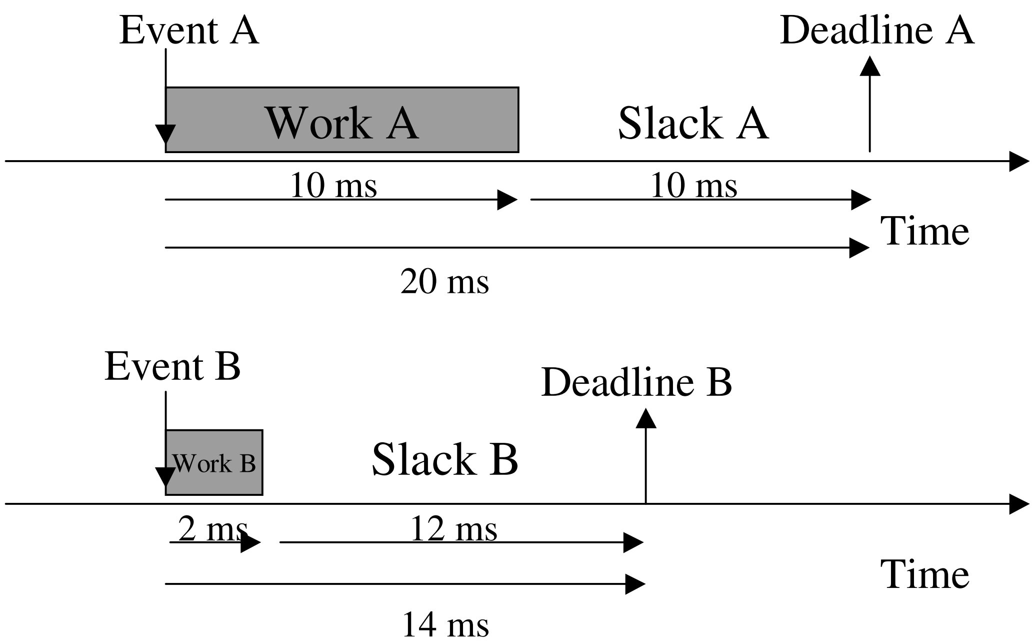
Real Time Scheduling

- Static Priorities?
  - consider multimedia Audio/Visual
  - is Video or Audio higher priority?

- General Requirement:



Example:


Event A
Deadline A
Work A
Slack A
10 ms
10 ms
Time
20 ms
Event B
Deadline B
Work B
Slack B
2 ms
12 ms
Time
14 ms

Earliest Deadline First, or Least Slack First?