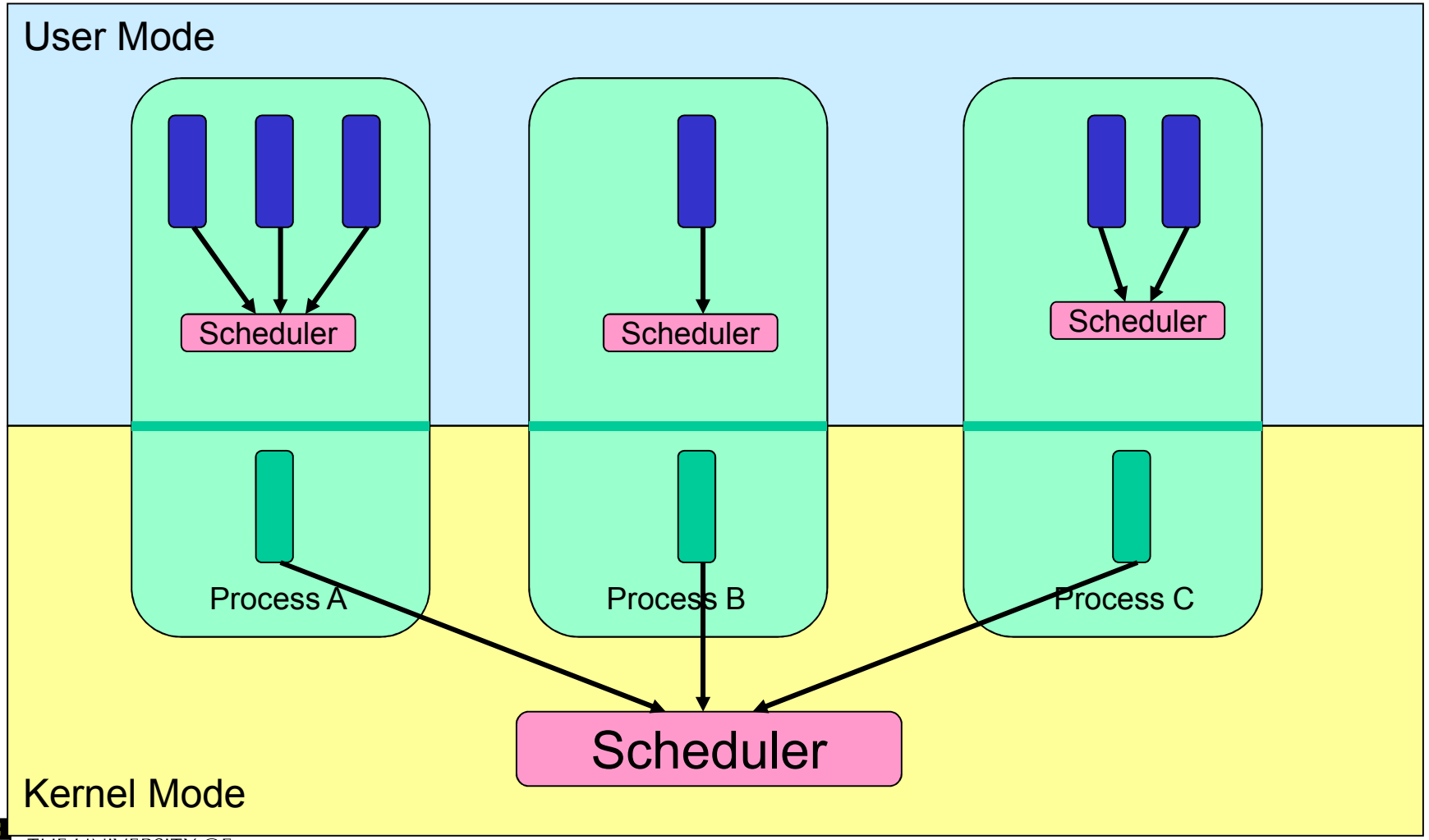
Scheduler Activations

With some slides modified from
Raymond Namyst, U. Bordeaux

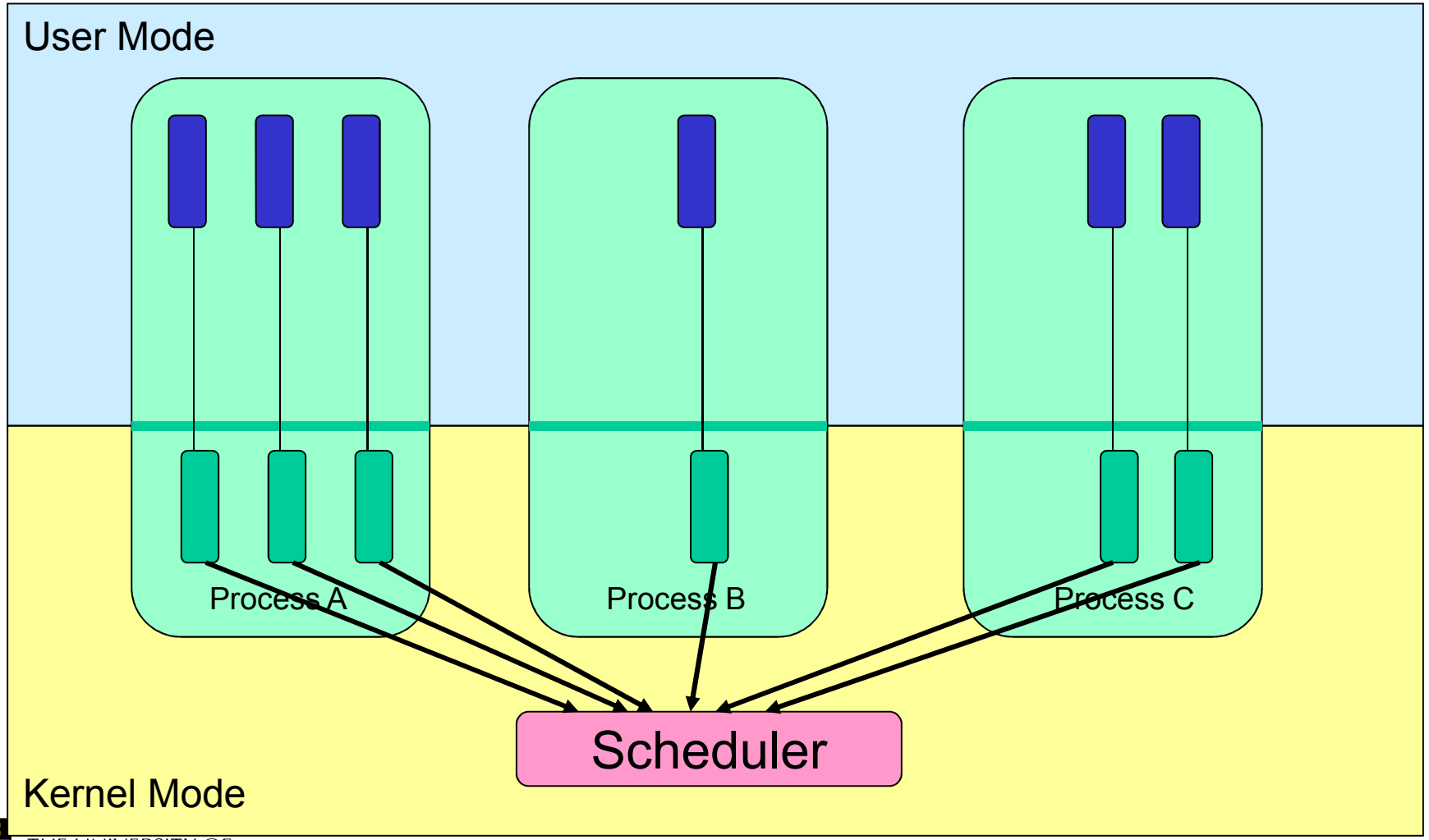
User-level Threads



User-level Threads

- ✓ Fast thread management (creation, deletion, switching, synchronisation…)
- ✘ Blocking blocks all threads in a process
  - Syscalls
  - Page faults
- ✘ No thread-level parallelism on multiprocessor

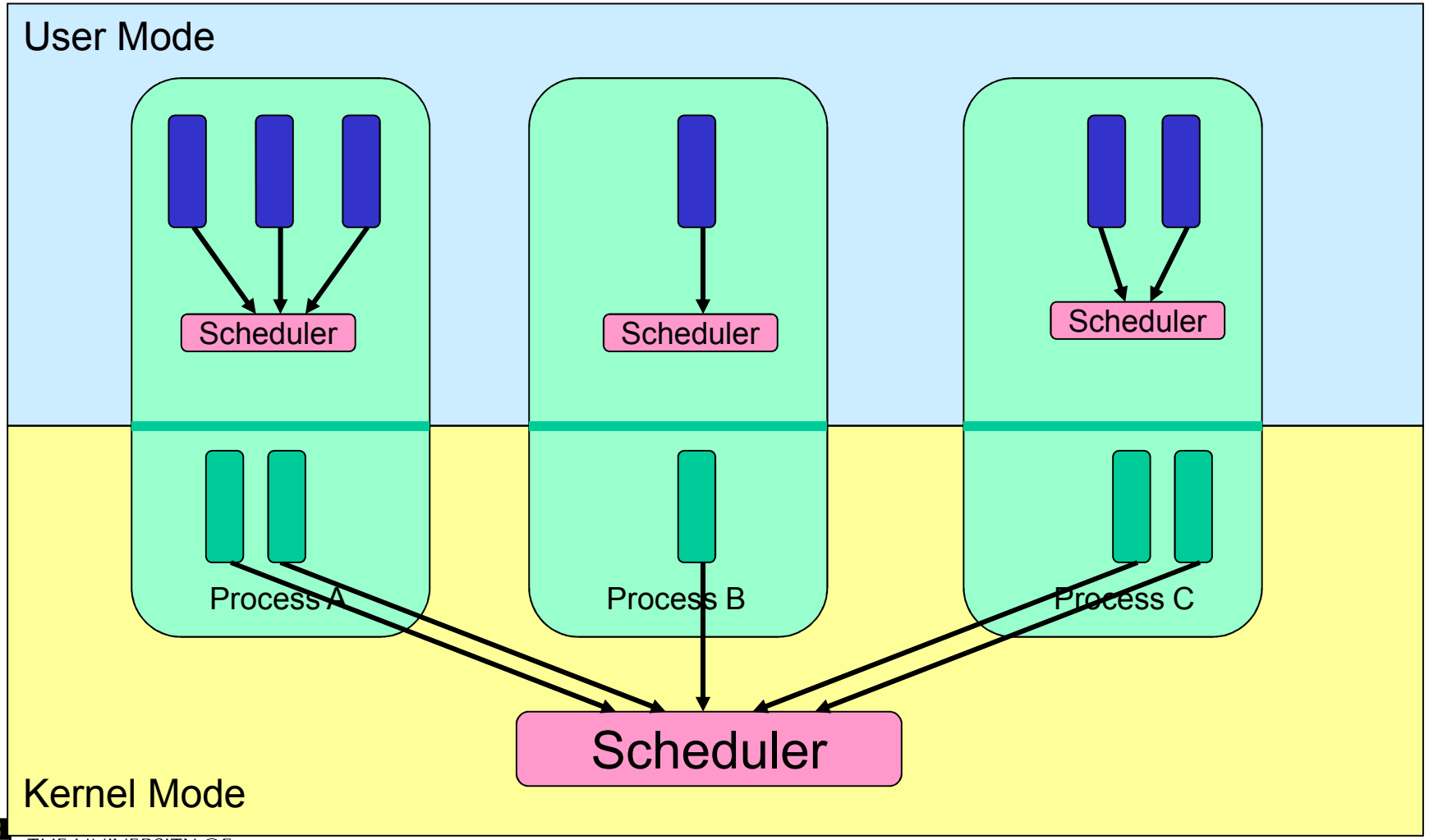
Kernel-Level Threads



Kernel-level Threads

- ✘ Slow thread management (creation, deletion, switching, synchronisation…)
  - System calls
- ✓ Blocking blocks only the appropriate thread in a process
- ✓ Thread-level parallelism on multiprocessor

Hybrid Multithreading



Hybrid Multithreading

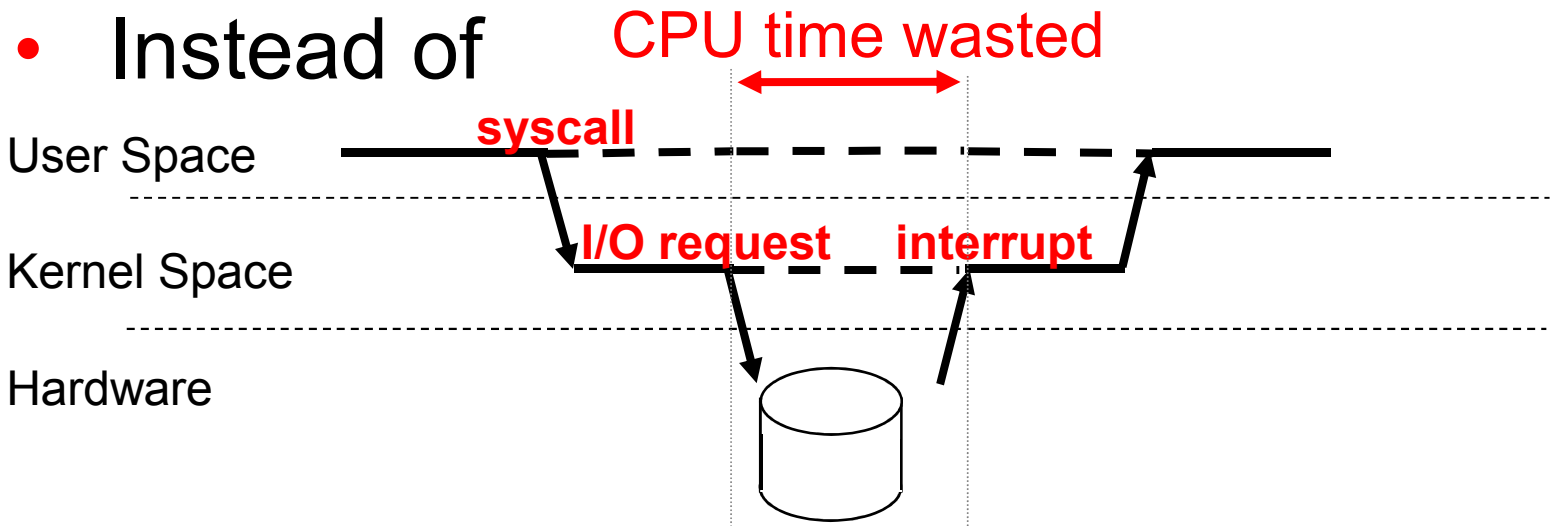
- ✓ Can get real thread parallelism on multiprocessor
- ✘ Blocking still a problem!!!

Scheduler Activations

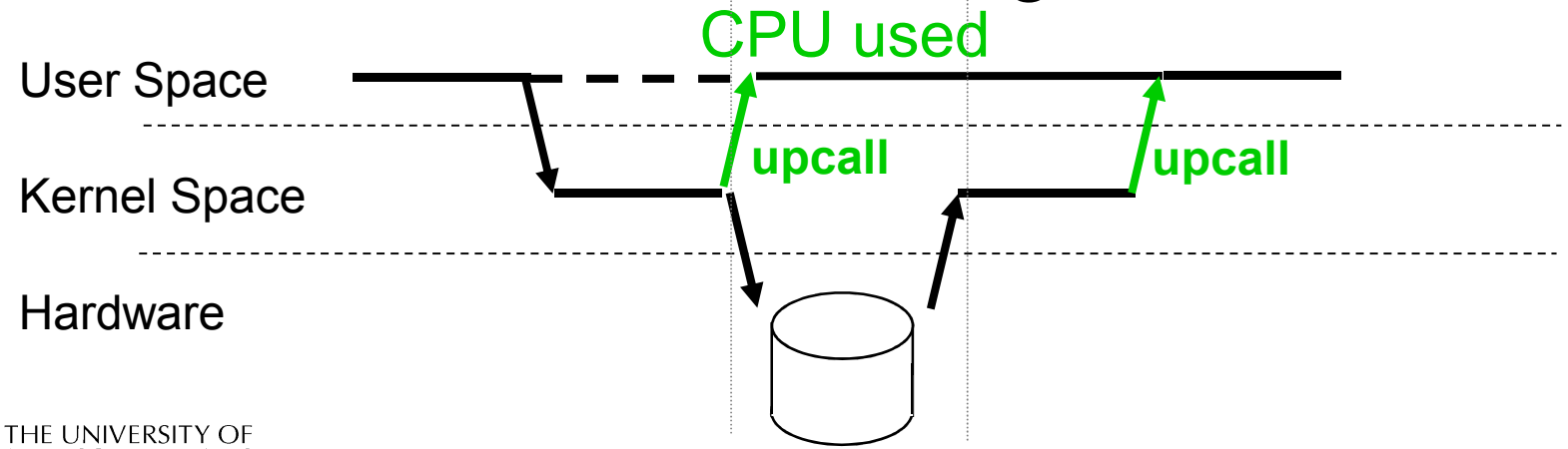
- First proposed by [Anderson et al. 91]
- Idea: Both schedulers co-operate
  - User scheduler uses system calls
  - Kernel scheduler uses upcalls!
- Two important concepts
  - Upcalls
    - Notify the user-level of kernel scheduling events
  - Activations
    - A new structure to support upcalls and execution
      - approximately a kernel thread
    - As many running activations as (allocated) processors
    - Kernel controls activation creation and destruction

Scheduler Activations

- Instead of



- …rather use the following scheme:

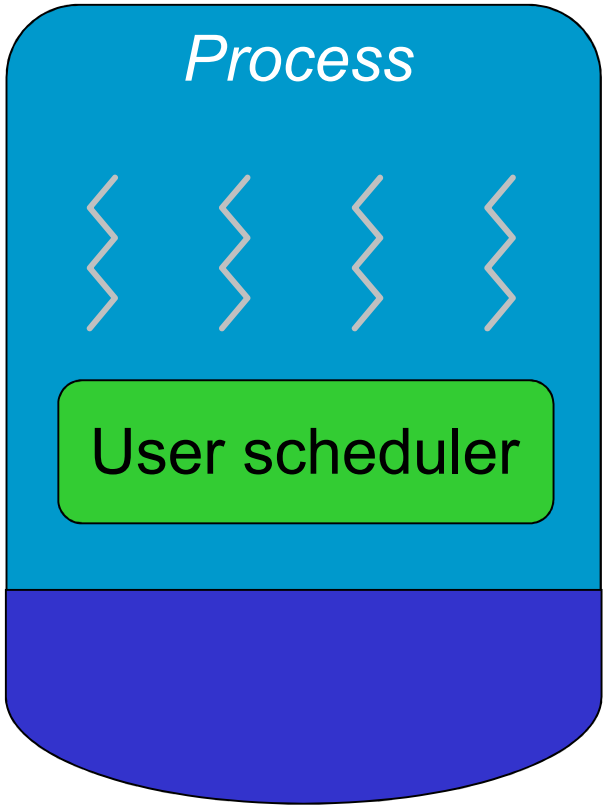


Upcalls to User-level scheduler

- New
  - Allocated a new virtual CPU
  - Can schedule a user-level thread
- Preempted
  - Deallocated a virtual CPU
  - Can schedule one less thread
- Blocked
  - Notifies thread has blocked
  - Can schedule another user-level thread
- Unblocked
  - Notifies a thread has become runnable
  - Must decided to continue current or unblocked thread

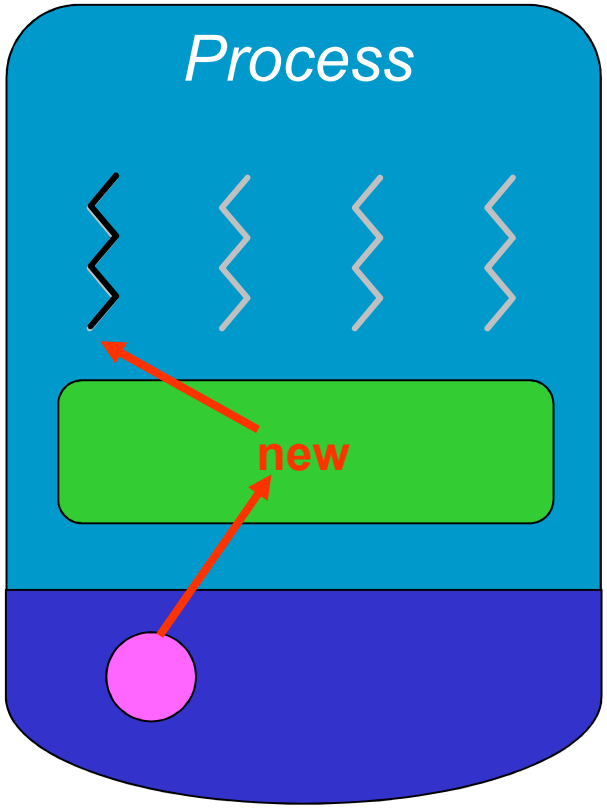
Working principle

- Blocking syscall scenario on 2 processors



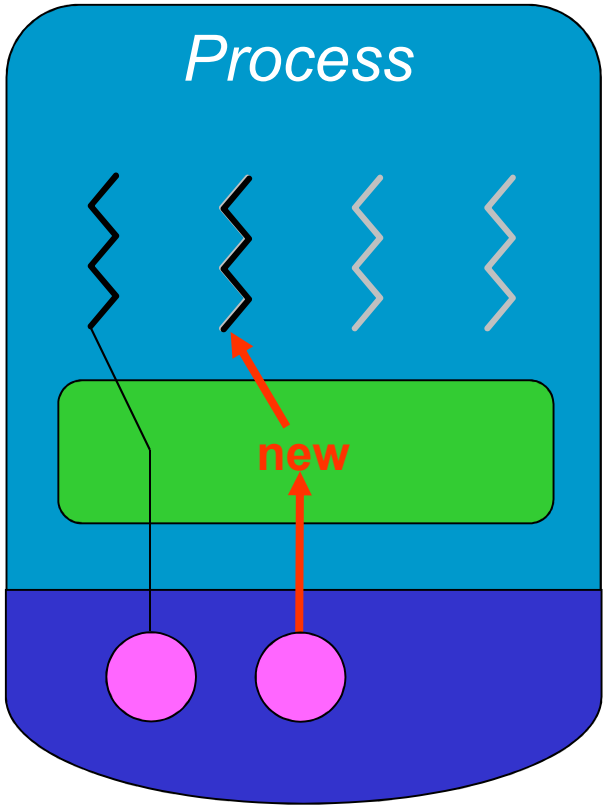
Working principle

- Blocking syscall scenario on 2 processors



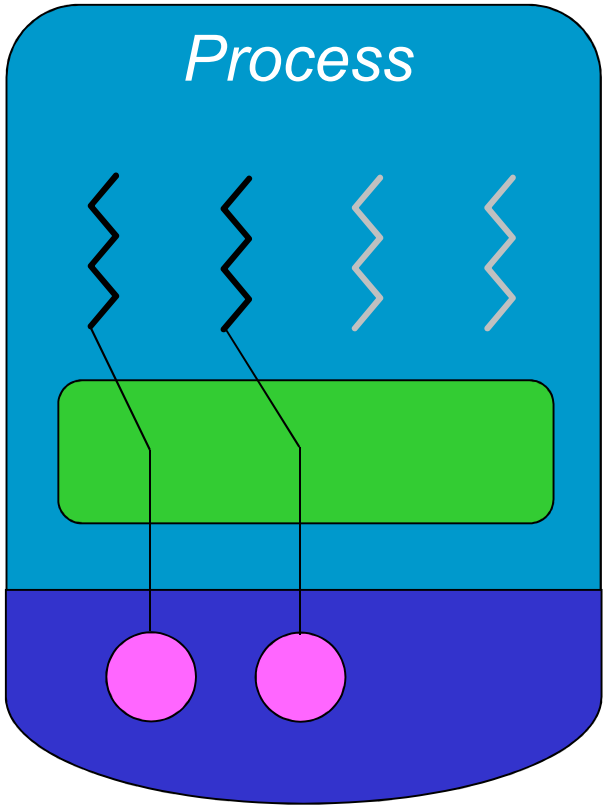
Working principle

- Blocking syscall scenario on 2 processors



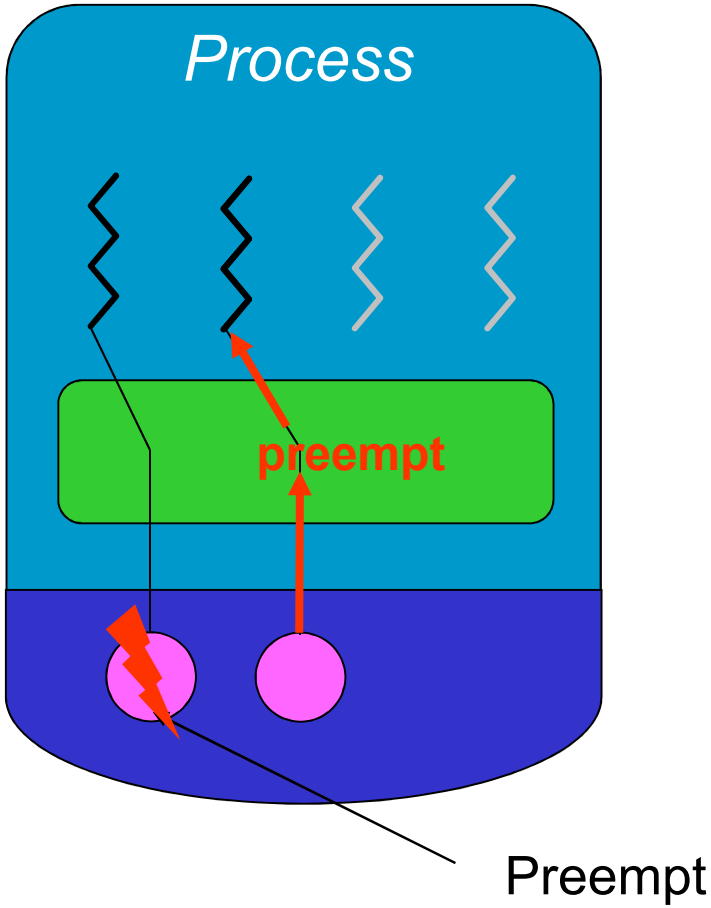
Working principle

- Blocking syscall scenario on 2 processors



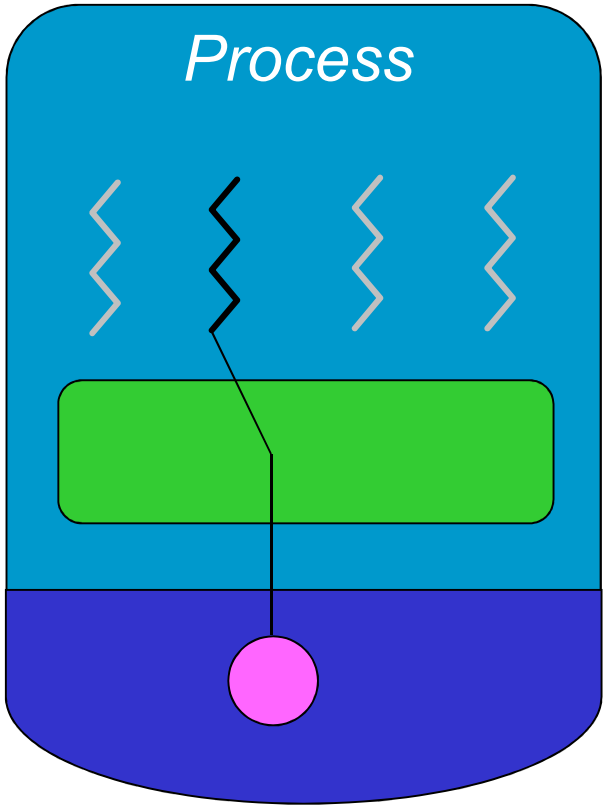
Working principle

- Blocking syscall scenario on 2 processors



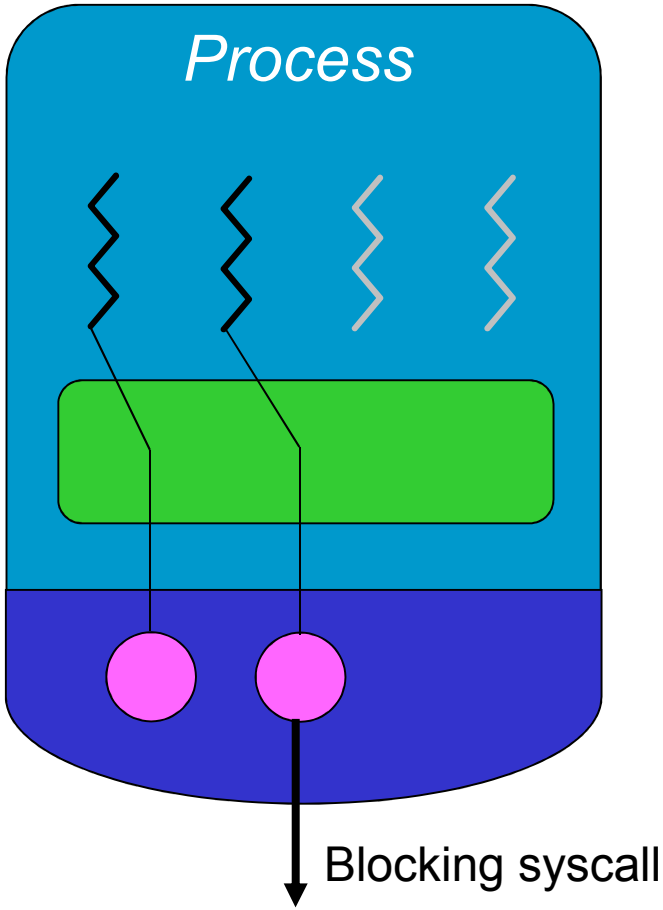
Working principle

- Blocking syscall scenario on 2 processors



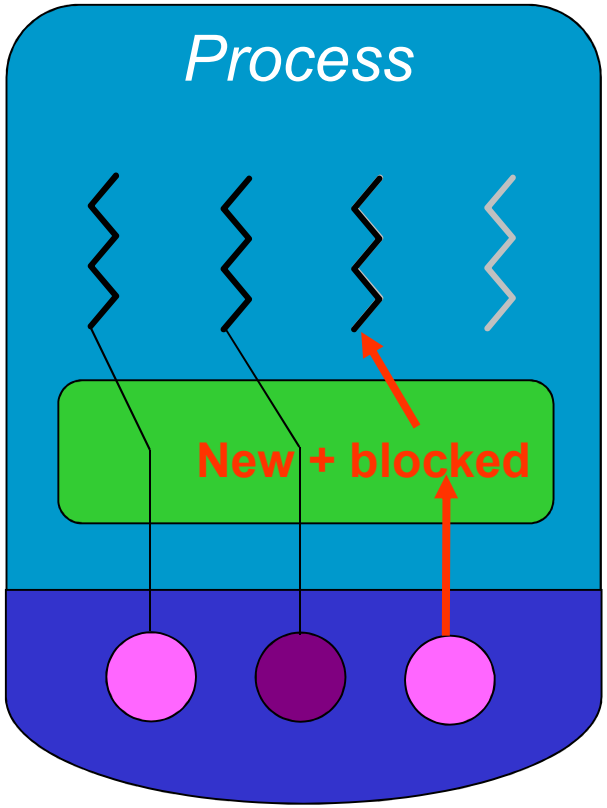
Working principle

- Blocking syscall scenario on 2 processors



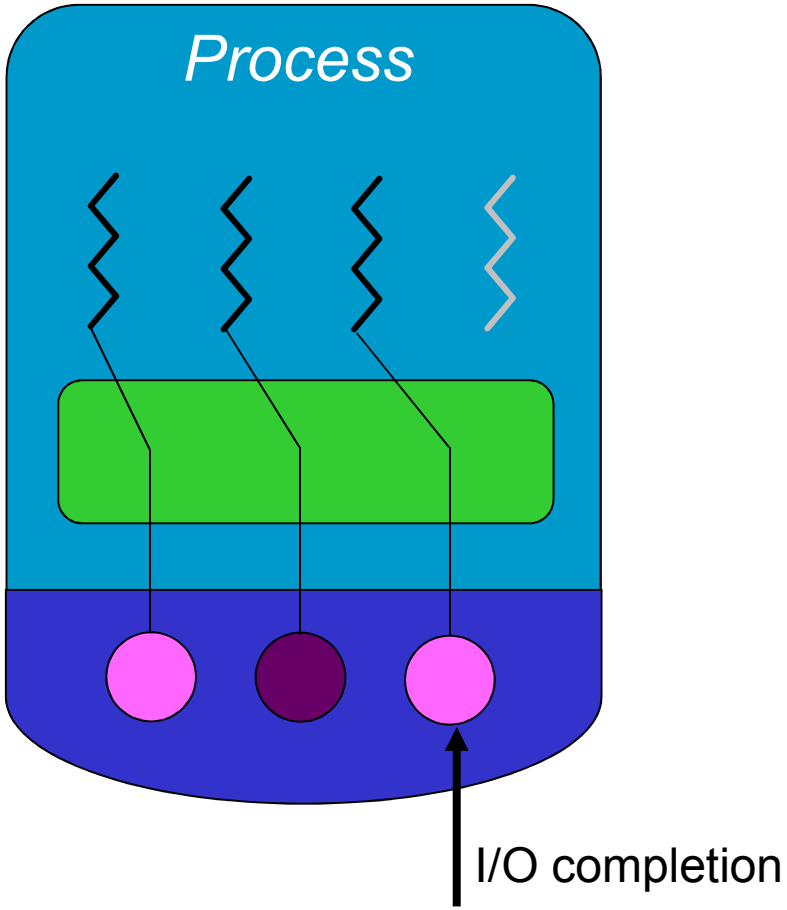
Working principle

- Blocking syscall scenario on 2 processors



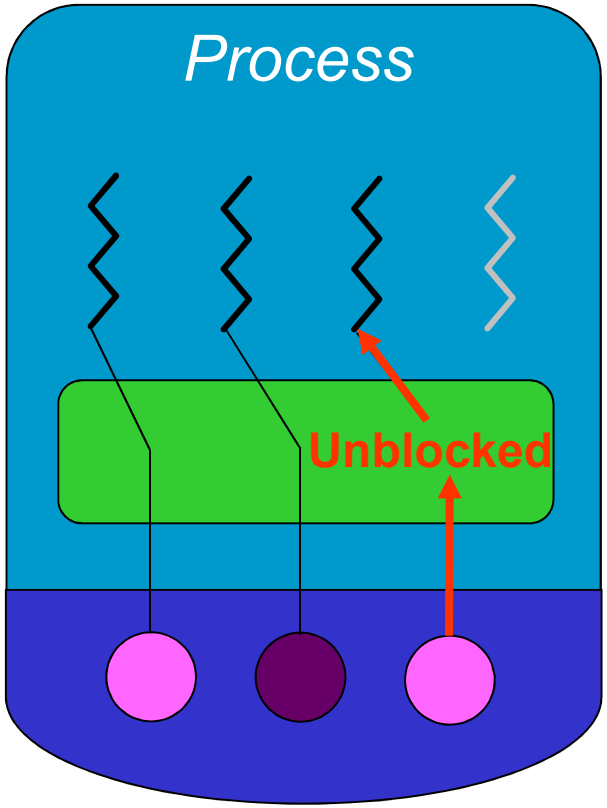
Working principle

- Blocking syscall scenario on 2 processors



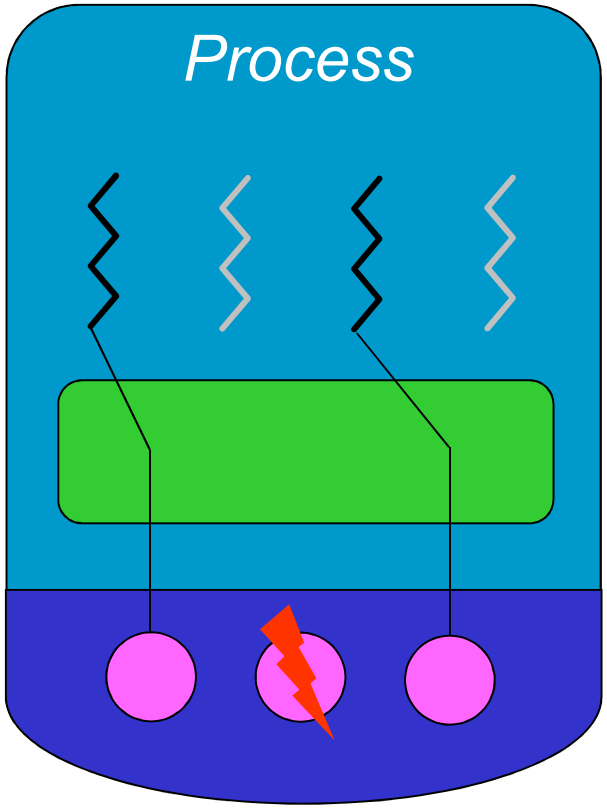
Working principle

- Blocking syscall scenario on 2 processors



Working principle

- Blocking syscall scenario on 2 processors



Scheduler Activations

- Thread management at user-level
  - Fast
- Real thread parallelism via activations
  - Number of activations (virtual CPU) can equal CPUs
- Blocking (syscall or page fault) creates new activation
  - User-level scheduler can pick new runnable thread.
- Fewer stacks in kernel
  - Blocked activations + number of virtual CPUs

Adoption

- Adopters
  - BSD “Kernel Scheduled Entities”
  - K42
  - Digital UNIX
  - Solaris
  - Mach
- Linux -> kernel threads