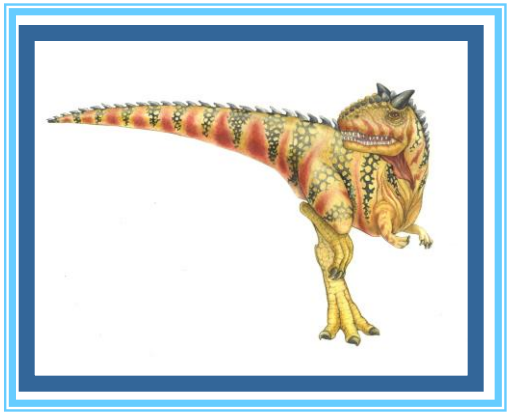
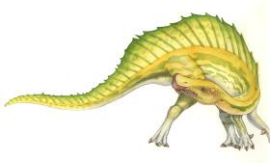
Chapter 4: Threads



Thread Libraries

- Thread library provides programmer with API for creating and managing threads

- Two primary ways of implementing
  - Library entirely in user space
  - Kernel-level library supported by the OS



Pthreads

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- API specifies behavior of the thread library, implementation is up to development of the library
- Common in UNIX operating systems (Solaris, Linux, Mac OS X)
- Examine Pthreads code sample

Forks and Threads

fork ()

1. Expensive
2. Creates “heavyweight process” exact copy of parent including parent’s descriptors except for a few differences such as:
    1. child has unique process id
    2. child has different parent id
    3. child has its own copy of parent’s descriptors although descriptors reference same underlying objects
3. Duplicates all threads
    1. can corrupt external resources (e.g. writing duplicate records to a file) if not careful

exec()

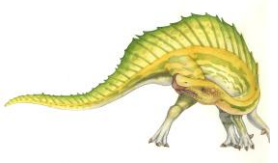
1. Works the same … if a thread calls exec (), entire process including all threads are replaced

Signal Handling

- Signals are used in UNIX systems to notify a process that a particular event has occurred
- A signal handler is used to process signals
  1. Signal is generated by particular event
  2. Signal is delivered to a process
  3. Signal is handled
- Options:
  - Deliver the signal to the thread to which the signal applies
  - Deliver the signal to every thread in the process
  - Deliver the signal to certain threads in the process
  - Assign a specific thread to receive all signals for the process
- Signals vs Interrupts (http://stackoverflow.com/questions/13341870/signals-and-interrupts-a-comparison)
  - interrupts - communication between CPU & kernel
  - signal - communication between kernel & processes

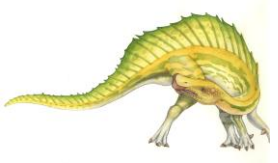
Thread Pools

- Create a number of threads in a pool where they await work
  - Advantages:
    - Usually slightly faster to service a request with an existing thread than create a new thread
    - Allows the number of threads in the application(s) to be bound to the size of the pool
- Threads in pool can be set heuristically based on # of CPUs, amount of memory, …
- Sophisticated thread-pool architectures can adjust thread pools dynamically according to usage patterns
  - Advantages
    - low load .. smaller pool .. less memory consumption



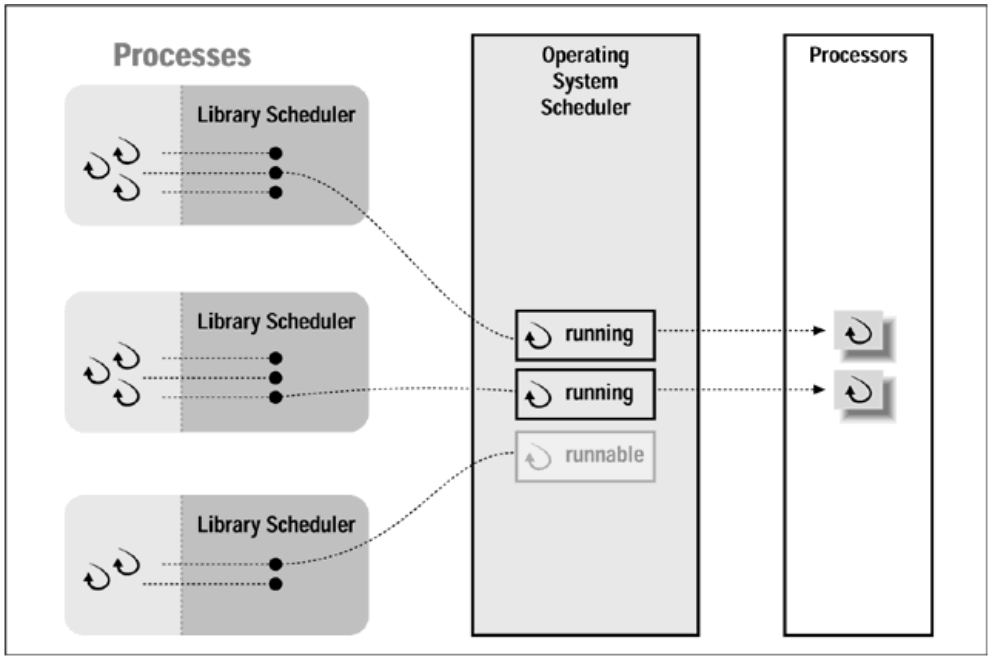
Linux Threads

- Linux refers to them as *tasks* rather than *threads*
- Thread creation is done through **clone()** system call
- **clone()** allows a child task to share the address space of the parent task (process)

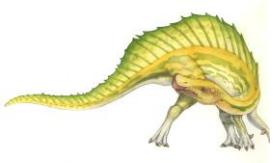


User Threads

- Advantages?
- Disadvantages?

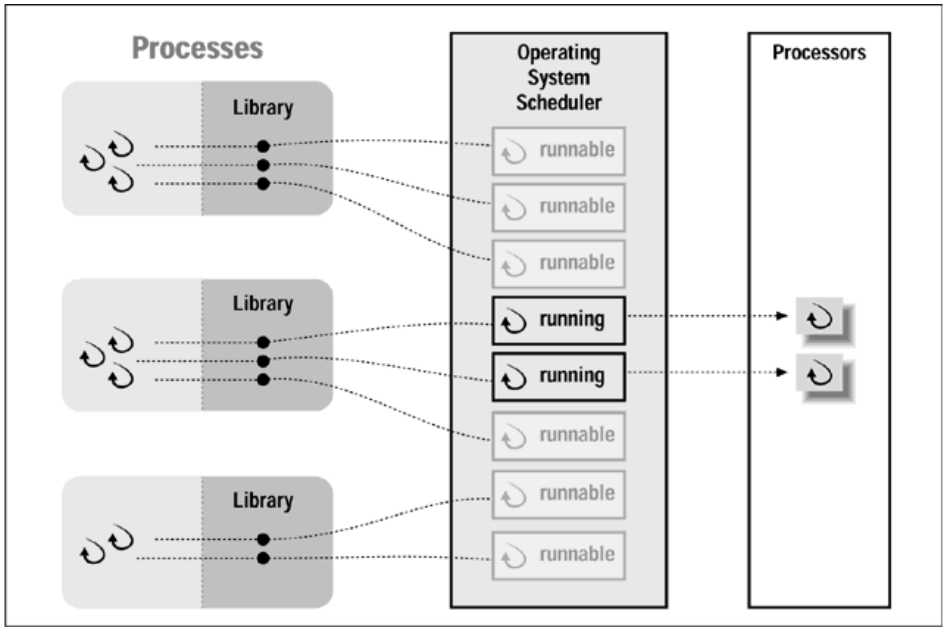


Pthreads Programming O'reilly

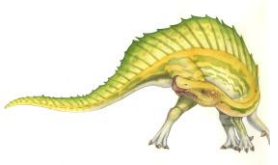


Kernel Threads

- Advantages
- Disadvantages

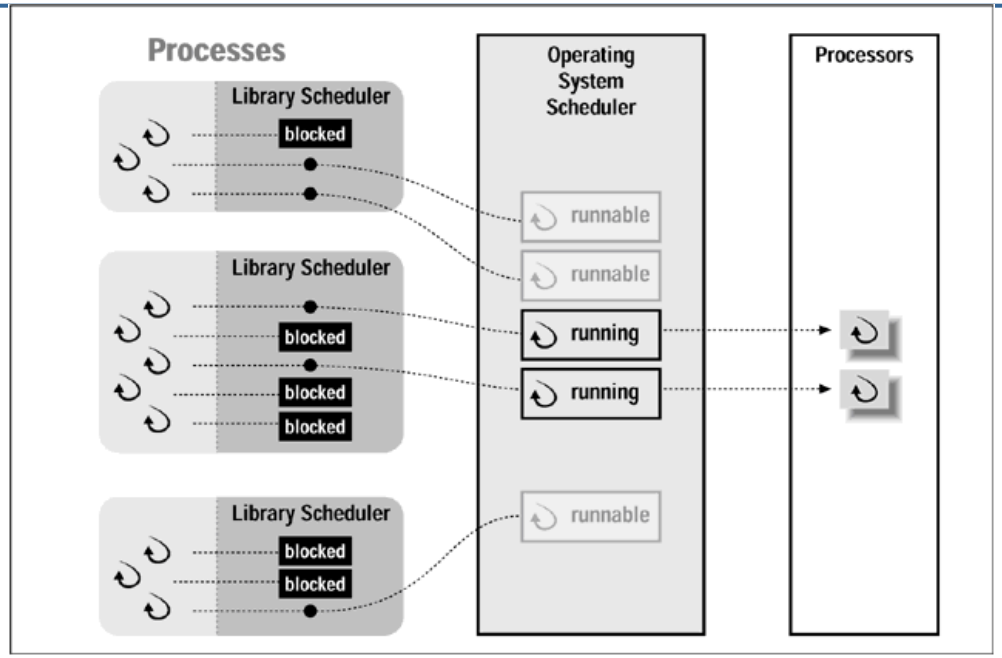


Pthreads Programming O'reilly



Hybrid Threads

- Best of both worlds



- NULL fork - time to create, schedule, execute, and complete process/thread that invokes NULL procedure
- Signal Wait - time to signal a waiting process/thread then wait on a condition (i.e. synchronization)

Thread and Process Operationg Latencies (micro-seconds)			
Operation	User Threads	Kernel Threads	Processes
NULL fork	34	948	11,300
Signal Wait	37	441	1,840

Operating Systems 5th Stallings