

Last lectures: Processes

- A process is the unit of execution
- Represented with PCBs in the OS: process state, scheduling, memory management information, ...
- A process goes through a number of states: New, Ready, Running, Waiting, ...
- The OS switches between processes – context switch
- Communication is through message-passing or shared memory.

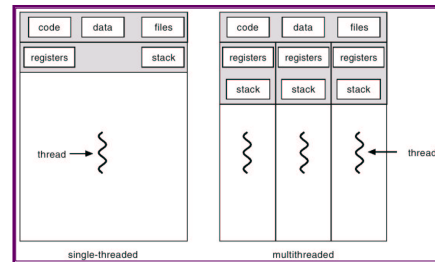
Chapter 5: Threads

- Overview – what are threads?
- Multithreading Models
- Threading Issues
- Pthreads
- Solaris 2 Threads
- Windows 2000 Threads
- Linux Threads
- Java Threads

Threads

- Single sequential execution stream within a process
- The address space of a process is potentially shared among multiple threads
- Each thread has its own threadID, PC, stack, registers and other things such as priority
- No system call is required to cooperate between threads!
- User and kernel level threads
- Forking a thread can be a system call or a simple user level library call.

Single and Multithreaded Processes



Benefits

- Responsiveness/concurrency
- Resource Sharing
 - each thread sees global memory for example
- Economy
 - it is faster than using processes
- Utilization of MP Architectures such as Simultaneous Multithreading Processors – parallelism across multiple threads

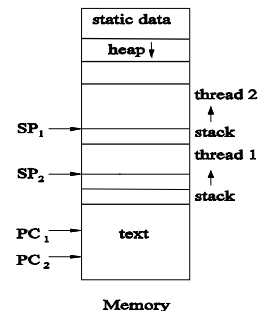
Memory Layout

```

main() {
    .....
    fork_thread(producer);
    fork_thread(consumer);
    .....
}

producer() {
    .....
}

consumer(){
    .....
}
    
```



User Threads

- Thread management done by user-level threads library
 - ◆ The OS knows only about the process
 - ◆ The OS schedules the process, not the threads within the process
 - ◆ User can define scheduling policy
 - ◆ The programmer uses a thread library to create, delete, synchronize, schedule threads.
- Examples
 - POSIX *Pthreads* (*IEEE standard*)
 - Mach *C-threads* (*Mach is a famous kernel*)
 - Solaris *threads*

Kernel Threads

- Supported by the Kernel
 - ◆ The OS knows about
 - ◆ The OS schedules the threads similar to processes (can use the same scheduling techniques as for processes)
 - ◆ Requires a small context switch, i.e., it is faster, no memory management information is changed
 - ◆ Kernel threads will/can get more time slices than user level threads but they are not flexible.
- Examples
 - Windows 95/98/NT/2000
 - Solaris
 - Tru64 UNIX
 - BeOS
 - Linux

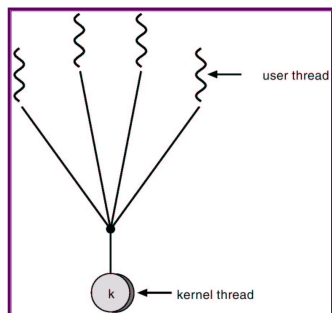
Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.

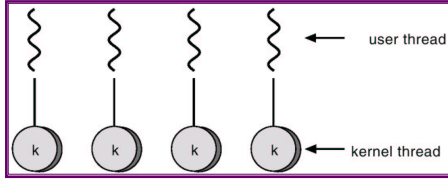
Many-to-One Model



One-to-One

- Each user-level thread maps to kernel thread.
- Examples
 - Windows 95/98/NT/2000
 - OS/2

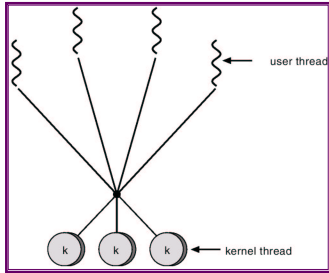
One-to-one Model



Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads.
- Allows the operating system to create a sufficient number of kernel threads.
- Solaris 2
- Windows NT/2000 with the *ThreadFiber* package

Many-to-Many Model



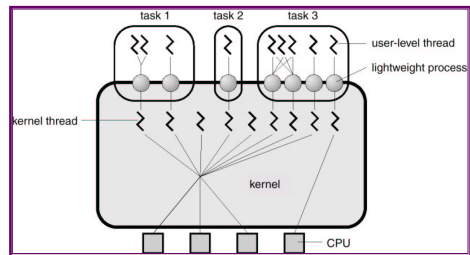
Threading Issues

- Semantics of `fork()` and `exec()` system calls.
- Thread cancellation.
- Signal handling
- Thread pools
- Thread specific data

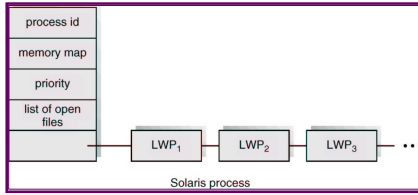
Pthreads

- 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 2 Threads



Solaris Process



Windows 2000 Threads

- Implements the one-to-one mapping.
- Each thread contains
 - a thread id
 - register set
 - separate user and kernel stacks
 - private data storage area

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)

Java Threads

- Java threads may be created by:
 - ◆ Extending Thread class
 - ◆ Implementing the Runnable interface
- Java threads are managed by the JVM.

Java Thread States

