Threads

- Objectives:
  - Concepts and issues associated with multithreaded computer systems.

- Thread – Lightweight process (LWP)
  - A basic unit of CPU utilization
    - A thread ID, program counter, a register set, and a stack space
  - Process – heavyweight process
    - A single thread of control
Threads

Motivation
- A web browser
  - Data retrieval
  - Text/image displaying
- A word processor
  - Displaying
  - Keystroke reading
  - Spelling and grammar checking
- A web server
  - Clients’ services
  - Request listening

Benefits
- Responsiveness
- Resource Sharing
- Economy
  - Creation and context switching
    - 30 times slower in process creation in Solaris 2
    - 5 times slower in process context switching in Solaris 2
- Utilization of Multiprocessor Architectures
User-Level Threads

User-level threads are implemented by a thread library at the user level.

Examples:
- POSIX Pthreads,
- Mach C-threads,
- Solaris 2 UI-threads

Advantages
- Context switching among them is extremely fast

Disadvantages
- Blocking of a thread in executing a system call can block the entire process.

Kernel-Level Threads

Kernel-level threads are provided a set of system calls similar to those of processes

Examples
- Windows 2000, Solaris 2, True64UNIX

Advantage
- Blocking of a thread will not block its entire task.

Disadvantage
- Context switching cost is a little bit higher because the kernel must do the switching.

* All rights reserved, Tei-Wei Kuo, National Taiwan University, 2005.
Multithreading Models

- Many-to-One Model
  - Many user-level threads to one kernel thread
  - Advantage:
    - Efficiency
  - Disadvantage:
    - One blocking system call blocks all.
    - No parallelism for multiple processors
  - Example: Green threads for Solaris 2

- One-to-One Model
  - One user-level thread to one kernel thread
  - Advantage: One system call blocks one thread.
  - Disadvantage: Overheads in creating a kernel thread.
  - Example: Windows NT, Windows 2000, OS/2
Multithreading Models

- Many-to-Many Model
  - Many user-level threads to many kernel threads
  - Advantage:
    - A combination of parallelism and efficiency
  - Example: Solaris 2, IRIX, HP-UX, Tru64 UNIX

Thread Libraries

- Goal: Provide an API for creating and managing threads!
- Two Approaches:
  - User Thread Library
  - Kernel-Level Thread Library
- Well-Known Examples
  - POSIX Pthreads – User or Kernel Level
  - Win32 threads – Kernel Level
  - Java threads – Level Depending on the Thread Library on the Host System
Pthreads

- Pthreads (IEEE 1003.1c)
  - API Specification for Thread Creation and Synchronization
  - UNIX-Based Systems, Such As Solaris 2.
- User-Level Library
- Header File: <pthread.h>
- pthread_attr_init(), pthread_create(), pthread_exit(), pthread_join(), etc.

```c
#include <pthread.h>
main(int argc, char *argv[]) {
  ...
  pthread_attr_init(&attr);
  pthread_create(&tid, &attr, runner, argv[1]);
  pthread_join(tid, NULL);
  ...
}

void *runner(void *param) {
  int i, upper = atoi(param), sum = 0;
  if (upper > 0)
    for(i=1;i<=upper,i++)
      sum+=i;
  pthread_exit(0);
}
```
Win32 Threads

- Kernel-Level Threads
  DWORD Sum; /* shared by threads */
  DWORD WINAPI Summation(LPVOID Param) {
    DWORD Upper = *(DWORD *) Param;
    for (DWORD I = 0; I <= Upper; i++)
      Sum += i;
    return 0;
  }
  Int main(int argc, char *argv[]) {
    ...
    Param = atoi(argv[1]);
    ...
    ThreadHandle = CreateThread(
      NULL, // default security attributes
      0, // default stack size
      Summation, // thread function
      &Param, // parameter
      0, // default creation flags
      &ThreadID);
    ...
    WaitForSingleObject(ThreadHandle, INFINITE);
    ...
  }

Java

- Thread Support at the Language Level
  - Mapping of Java Threads to Kernel Threads on the Underlying OS?
    - Windows 2000: 1:1 Model
- Thread Creation
  - Create a new class derived from the Thread class
  - Run its start method
    - Allocate memory and initialize a new thread in the JVM
    - start() calls the run method, making the thread eligible to be run by the JVM.
Java

class Summation extends Thread
{
public Summation(int n) {
    upper = n;
}
public void run() {
    int sum = 0;
    ...
}
 ...
}
public class ThreadTester
{
...
    Summation thrd = new Summation(Integer.parseInt(args[0]));
    thrd.start();
...


Threading Issues

- Fork and Exec System Calls
  - Fork: Duplicate all threads or create a duplicate with one thread?
  - Exec: Replace the entire process, including all threads and LWPs.
  - Fork \(\rightarrow\) exec?
Threading Issues

- Thread Cancellation
  - Target thread
  - Two scenarios:
    - Asynchronous cancellation
    - Deferred cancellation
      - Cancellation points in Pthread.
- Difficulty
  - Resources have been allocated to a cancelled thread.
  - A thread is cancelled while it is updating data.

Threading Issues

- Signal Handling
  - Signal
    - Synchronous – delivered to the same process that performed the operation causing the signal,
      - e.g., illegal memory access or division by zero
    - Asynchronous
      - e.g., ^C or timer expiration
  - Default or user-defined signal handler
  - Signal masking
Threading Issues

- Delivery of a Signal
  - To the thread to which the signal applies
    - e.g., division-by-zero
  - To every thread in the process
    - e.g., ^C
  - To certain threads in the process
  - Assign a specific thread to receive all threads for the process
    - Solaris 2

- Asynchronous Procedure Calls (APCs)
  - To a particular thread rather than a process

---

Threading Issues

- Thread Pools
  - Motivations
    - Dynamic creation of threads
    - Limit on the number of active threads
  - Awake and pass a request to a thread in the pool
  - Benefits
    - Faster for service delivery and limit on the # of threads
    - Dynamic or static thread pools
  - Thread-specific data – Win32 & Pthreads
Scheduler Activations

- Definition: A scheme for communication between the user-thread library and the kernel
  - The kernel provides an application with a set of virtual processors, i.e., light weight processes (LWP’s)
    - An upcall handler to stop or resume the execution of a thread
  - User threads on a LWP are blocked if any of the user threads is blocked!

Windows XP

- Win32 API
  - One-to-One Model
  - Fiber Library for the M:M Model
- A Thread Contains
  - A Thread ID
  - Context: A Register Set, A User Stack, A Kernel Stack, A Private Storage Space for Run-Time Libraries, and DLL’s
Windows XP

- Data Structures
  - ETHREAD (executive thread block)
    - A ptr to the process, a ptr to KTHREAD, the address of the starting routine
  - KTHREAD (kernel thread block)
    - Scheduling and synchronization information, a kernel stack, a ptr to TEB
  - TEB (thread environment block)
    - A user stack, an array for thread-specific data.

Linux

- Threads introduced in Version 2.2
  - clone() versus fork()
    - Term task for process & thread
    - Several per-process data structures, e.g., pointers to the same data structures for open files, signal handling, virtual memory, etc.
  - Flag setting in clone() invocation.
    - CLONE_FS, CLONE_VM, CLONE_SIGHAND, CLONE_FILES
    - Setting -> Threads or Processes