Threads

Chapter 4

“A thread of execution” (aka locus of control)

- A thread is a concurrent flow of execution of a task in a program
- Program counter
- Registers
- Stack

- Sometimes called a “lightweight process”
  - Context switching a thread requires less effort than context switching a process (or heavyweight process)
  - Contrast this usage with pp. 142-143

Threads v Processes

A summary

<table>
<thead>
<tr>
<th>Process</th>
<th>Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Program</td>
</tr>
<tr>
<td>Data</td>
<td>Not directly shared</td>
</tr>
<tr>
<td>Owners</td>
<td>May be different</td>
</tr>
<tr>
<td>Relation</td>
<td>May be threaded</td>
</tr>
<tr>
<td>Created by</td>
<td>Operating system</td>
</tr>
<tr>
<td>Creation time</td>
<td>1</td>
</tr>
<tr>
<td>Switch time</td>
<td>1/5</td>
</tr>
</tbody>
</table>

Process Creation

Programs become processes

```
main()
{
    threadStart(fill)
    threadStart(empty)
}
fill()
{
    ...
}
empty()
{
    ...
}
```

Processes created from 1 program do not share data. Forking a process duplicates data; data still not shared.
Thread States

Thread Benefits

- Responsiveness
  - Application continues if a thread is blocked or busy
- Resource sharing
  - Reduces burden on OS
- Economy
  - Faster to create and switch between threads
- Multiprocessor utilization
  - Possible to run threads on different processors

Thread Types

- System level
  - Aka kernel threads implemented by the OS
  - Runs in the kernel address space
- Application level
  - Aka user threads
  - Implemented by a library and runs in user space
  - Still requires some OS support
    - Application does scheduling, context switch, and synchronization
    - OS provides timing interrupts

Many-to-One Thread Model

- Maps many user threads to one kernel thread
- Management in application
- Process blocks if one thread blocks
- Unable to utilize multiple processors
- Easiest to implement and validate

One-to-One Thread Model

- Maps one user thread to one kernel thread
- Process may continue when one thread blocks
- Threads may be distributed over many processors
- Management in kernel ➔ more burden on OS
- Number of threads may be limited

Many-to-Many Thread Model

- Maps many user threads to many kernel threads
- \( n(k) \leq n(u) \)
- 1 kernel thread may service multiple user threads
- Allows any number of user threads; supports true concurrency
- More difficult to implement and validate
Pthreads

From Figure 4.6

```c
pthread_t tid;
pthread_attr_t attr;

...;
pthread_attr_init(&attr);
pthread_create(&tid, &attr, runner, argv[1]);
pthread_join(tid, NULL);

...;
void* runner(void* param)
{
    ...;
pthread_exit(0);
}
```

Wind32 Threads

From Figure 4.7

```c
DWORD ThreadId;
HANDLE ThreadHandle;

...;
ThreadHandle = CreateThread(NULL, 0, Summation, &Param, 0, &Threadid);
WaitForSingleObject(ThreadHandle, INFINITE);

...;
DWORD WINAPI Summation(LPVOID Param)
{
    ...;
    return 0;
}
```

Java Threads

From Figure 4.8

```java
class Summation implements Runnable
{
    ...;
    public void run() { ... }
}

public class Driver
{
    Thread thrd = new Thread(new Summation(...));
    thrd.start();
    ...;
    thrd.join();
}
```

Clarifying Java Threads

See page 136

- Creating a Thread object
  - Creates a Thread
  - But it is not running concurrently as a thread
- The start() method
  - Puts the Thread on the scheduler’s queue; this makes the Thread eligible to execute (the run method does NOT make it eligible to execute)
  - It does NOT call run; the scheduler calls run
  - You can call run() directly but it will not execute concurrently

Threading Issues

1

- Does fork() duplicate threads
  - Not if followed by exec()
- Cancellation
  - Asynchronous
    - One thread kills another
    - Deadlock prone
  - Deferred
    - Thread is allowed to die when convenient

2

- Signals
  - Generally delivered to processes
  - Asynchronous are caused by external events
  - Synchronous are often caused by errors
  - Signals are sent to
    - only the thread to which the signal applies (synchronous)
    - every thread in the process (asynchronous)
    - only certain threads
    - a thread designated as signal handler
Thread Pools
- Faster (avoids creating a new thread)
- Limits the number of threads
- Number of threads tuned by heuristics and measurements