Threads
Readings

- Silberschatz et al: Chapter 4
Introduction

- Multiple applications run concurrently!
- This means that there are multiple processes running on a computer
A “Single Threaded” Program

main()
{
......
{
}
Applications often need to perform many tasks at once.

This requires multiple threads of execution within a single process.
Example: Word processor

- Tasks include:
  - Display graphics
  - Respond to keystrokes from the user
  - Perform spelling and grammar checking
Example

- Example: Web server
  - It is desirable to service requests concurrently
Earlier we discussed the use of forking to create a process.

For example we could:

- **Word processor example:** fork a process for each task
- **Web server example:** fork a process for each request

Not very efficient since a fork copies everything.
Threads

- A thread is a basic unit of CPU utilization.
- Threads of a process share memory but can execute independently.
- A traditional process can be viewed as a memory address space with a single thread.
A word processor program with three threads.
Thread Usage - Web Server

Web/FTP server

Client 1

Client 2

Client N

Process Request Client 2

Process Request Client N
Why Not Fork?

- You certainly can fork a new process
- In fact, the first implementation of Apache web servers (Apache 1.0) forked N processes when the web server was started
  - “N” was defined in a configuration file
  - Each child process handled one connection at a time
- **Problem:** Process creation is time consuming and resource intensive
- Creating threads is not as expensive
Why Not Fork?

- Let’s look at web servers
  - This allowed a child process to handle multiple connections at a time
  - Web servers have caches for read pages
  - Forking means that these caches cannot be shared
  - Using threads allows for these caches to be shared
Thread State

 Threads share
  - Process address space
    - Text
    - Data (global variables)
    - Heap (dynamically allocated data)
  - OS state
    - Open files, sockets, locks

 Threads have their own CPU context
  - Program counter (PC), Stack pointer (SP), register state, stack
### Single and Multithreaded Processes

<table>
<thead>
<tr>
<th>code</th>
<th>data</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>registers</td>
<td>stack</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>code</th>
<th>data</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>registers</td>
<td>registers</td>
<td>registers</td>
</tr>
<tr>
<td>stack</td>
<td>stack</td>
<td>stack</td>
</tr>
</tbody>
</table>

- **Single-threaded process**
- **Multithreaded process**
Benefits of Threads

- **Responsiveness**
  - Overlap computation and blocking due to I/O on a single CPU

- **Resource Sharing**
  - Example: Word processor
    - The document is shared in memory.
    - Forking would require replication

- Allocating memory and resources for process creation is costly

- Context-switching is faster
Questions

Can one thread access local variables of another thread?

If one thread allocates a data structure from the heap, can other threads access it?
Thread Libraries

- A thread library provides the programmer with an API for creating and managing threads.
- Three main libraries:
  - POSIX Pthreads
  - Win32
  - Java
Problem

- Sharing global variables is dangerous - two threads may attempt to modify the same variable at the same time.
- Use support for mutual exclusion primitives that can be used to protect against this problem.
- The general idea is to lock something before accessing global variables and to unlock as soon as you are done.
- More on this topic later in the course
Question

- Consider a multi-threaded web browser
  - What is the disadvantage of this design?
Processes vs Threads

- A beta version of Google’s browser, *Chrome*, was released on 2 September 2008.
- Before Chrome browsers allocated a thread for each tab
- *Chrome* allocates a process for each tab
Summary

- Introduction to the concept of threads
- There will be more discussion