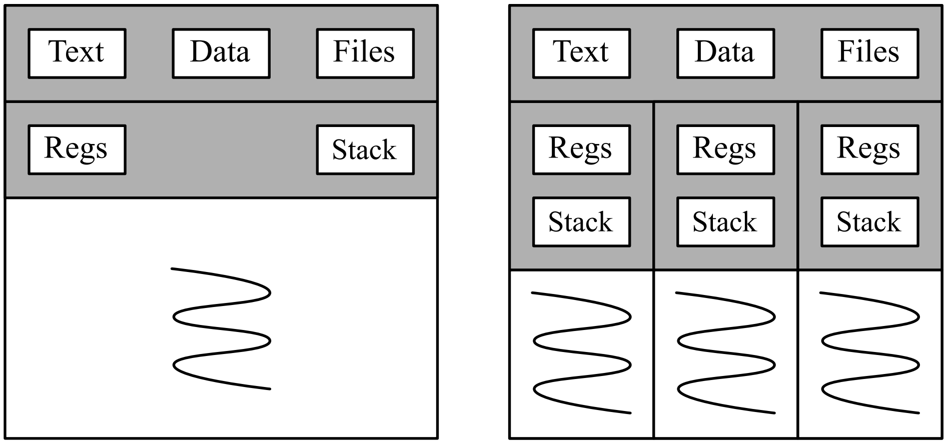
Lecture Notes 4

Threads

- Threading Models
  - Single Threaded – Single sequential execution context
  - Multithreaded – Multiple concurrent execution contexts



- Multithreaded Benefits
  - Responsiveness – Allows interactive applications to respond when other thread is blocked
  - Resource Sharing – Files and memory are automatically shared in multithreaded programs. Multi-process requires some IPC mechanism.
  - Economy – Lower resource cost, possibly lower overhead of context switch.
  - Scalability – Heavy weight threads may take advantage of multiple cores, single threaded process can't take advantage of multiple cores.
- Multicore Programming (or Multiprocessor Programming)
  - Parallelism – Multiple Tasks (or Threads) run simultaneously
  - Concurrency – Multiple Tasks (or Threads) continue to make progress
  - Amdahl's Law – Formula for potential speedup of multiple processing cores.
  - Programming Challenges
    - Identifying Tasks – Examine application to divide up concurrent tasks
    - Balance – Identifying and splitting up work equally
    - Data Splitting – Data accessed must be divided up between separate cores
    - Data Dependency – Must ensure dependencies are synchronized
    - Testing and Debugging – Increased difficulty due non-determinism
  - Types of Parallelism
    - Data Parallelism – Distributes subsets of data across multiple computing cores
    - Task Parallelism – Distributes data and tasks (threads) across multiple computing cores

- Multithreading Models
  - Heavy Weight Threads – Kernel space, visible to the OS, can concurrently executing on independent cores
  - Light Weight Threads – User space only, scheduled by application, all share a core
  - Fibers – Light Weight Threads that are cooperatively scheduled (non-preemptive threads)
  - Many-to-One Model – Many user space threads to one kernel space thread
  - One-to-One Model – One user space thread to one kernel space thread
  - Many-to-Many Model – Multiple user space threads to multiple (usually fewer) kernel space threads
    - Two-level Model – Many-to-Many that also allows user space to kernel space binding
- Thread Libraries – API for programmer to create and manage threads
  - Pthreads – POSIX standard library may be kernel or user space depending upon implementation
  - Windows Threads – Threading interface for Windows OS
  - Java Threads – Fundamental part of Java programming language
- Implicit Threading – Multiple threads are implied, not explicitly created
  - Thread Pools – Limited number of threads created and are taken from a pool
  - OpenMP – Open Multi-Processing library, compiler directives and API
    - Parallel Regions – A region of code that can execute in parallel
  - Grand Central Dispatch (GCD) – Apple technology similar to OpenMP
    - Blocks – A self contained unit of work
    - Dispatch Queue – The queue that holds the blocks
    - Main Queue – Per process serial queue
- Threading Issues
  - System Calls (fork() and exec())
    - fork() – Should all threads be duplicated?
    - exec() – If called after fork(), thread duplication not needed.
  - Signal Handling – Signals notify a process of an event is “handled” by a handler function
    - Default Signal Handler – Default handler defined by the kernel
    - User-Defined Signal Handler – User specified function to handle the signal
    - Which thread handles the signal?
    - What if all need to receive the signal?
  - Thread Cancellation – Termination of threads
    - Target Thread – Thread that is set to be cancelled
    - Asynchronous Cancelation – Thread immediately terminates Target Thread
    - Deferred Cancellation – Target Thread periodically checks for termination
    - Cancellation Point – Place where deferred cancellation occurs
    - Cleanup Handler – Function that is called to release of resources

- Thread-Local Storage – Thread own copy of data needed for processing
- Scheduler Activations – Communication point between user-thread library and kernel
  - Light Weight Process (LWP) – Each LWP is connected to a kernel thread
  - Upcall – Kernel notifying application about events
  - Upcall Handler – Function that handles the upcall event