Lecture Notes 17

Concurrency Introduction

- Single Instruction, Multiple Data (SIMD) – Machines that can execute the same instruction on multiple pieces of data simultaneously
- Vector Processor – A type of SIMD machine
- Multiple Instruction, Multiple Data (MIMD) – Machines that can execute multiple instruction streams
- Multiprocessors – A type of MIMD machine
- Hidden Concurrency – The set of hardware features such as instruction pipelining that allowed for instructions to be executed concurrently
- Physical Concurrency – Multiple physical processing units are capable of executing concurrently
- Logical Concurrency – Interleaving of execution to provide illusion of multiple physical processing units
- Thread of Control – Sequence of program points reached as control flows through the program
- Multithreaded – Program designed for more than one thread of control

Subprogram-Level Concurrency

- Process – Execution of a single program
- Thread – Execution of thread of control, each process has one or more threads
- Heavyweight Thread – OS visible thread that can be scheduled by the OS
- Lightweight Thread – Thread that is only visible to the application, and must be scheduled within the process
- Synchronization – Mechanism that controls the order in which tasks are executed
- Cooperation Synchronization – Required when one task must wait until another completes it work
- Competition Synchronization – Required when multiple tasks require the same resource that cannot be used simultaneously
- Race Condition – Occurs when multiple tasks are accessing/manipulating the same data concurrently and result depends upon order of access
- Liveness – Characteristic of a process if it continues to execute eventually completing
- Deadlock – Situation where there is a loss of liveness

Synchronization Mechanisms

- Semaphore – Integer value that is access through atomic operations
  - P, V? (Proberen, Verhogen) or up/down or wait/signal or wait/resume
  - Counting Semaphore – Allows over unrestricted domain

    - Binary Semaphore (Mutex Lock) – Limited to 0 and 1
    - Semaphores considered harmful. – Separate lock and condition variables is more readable, and stateless condition variable is better that one with state
- Monitor – Abstract data type that provide mutual exclusion when accessing shared data within
    - Java `synchronized` – Keyword used in Java to specify that object will act as a monitor within the specified method(s)
- Message Passing – Messages are sent either synchronously or asynchronously between tasks
- Rendezvous – The transmission of a message between sender and receiver in synchronous message passing

Language Support for Concurrency

- Ada
    - `accept` clause – Used to create the receive side of a rendezvous
    - `select` statement – Allows for multiple different types of messages to be received
    - `when` clause – Allows for a guard to specify when the `accept` clause is open
- Java
    - `Thread` class – Only class available to create concurrent programs
        - `start` – Method to start the thread
        - `run` – Method that is the body of the thread
        - `yield` – Method to explicitly yield the processor
        - `sleep` – Method to put the thread to sleep for the specified number of milliseconds
        - `join` – Method to force delay of execution until the run method of another thread completes.
    - `synchronized` Methods or Statements – Locks the object's lock and guarantees mutual exclusion during execution
    - `wait` – Method that will yield control of the synchronized region until woken with `notify` or `nofiyAll`
    - `notify` – Wakes one waiting thread and allows it to execute once able
    - `notifyAll` – Wakes all waiting threads allowing each to execute once able
    - `java.util.concurrent.atomic` – Package that provides nonblocking synchronization of primitive types
- C#
    - `Thread` class – Can take `ThreadStart` delegate to execute any method as its own thread
        - `Start` – Method to start the thread
        - `Sleep` – Method to put the thread to sleep for the specified number of milliseconds

    - `Join` – Method to force delay of execution until the run method of another thread completes.
    - `Abort` – Raises `ThreadAbortExecution` exception for thread
    - `Invoke` – Invokes a method as own thread, will block until completed.
    - `BeginInvoke` – Begins asynchronous execution of method
    - `EndInvoke` – Returns the result of the method executed asynchronously
    - `IsCompleted` – Returns if the asynchronous method has completed
- `Interlocked` class – Used for incrementing/decrementing incrementing/decrementing integers
- `lock` – Used to create a critical section
- `Monitor` class – Provides a monitor object
    - `Enter` – Synchronizes on the object specified
    - `Wait` – Analogous to Java `wait`
    - `Pulse` – Analogous to Java `notify`
    - `PulseAll` – Analogous to Java `notifyAll`
    - `Exit` – Ends the critical section for the object