**Parallel Programming Skills**

**Fall 2019**

**Johanna Beatty**

**Michael Rousell**

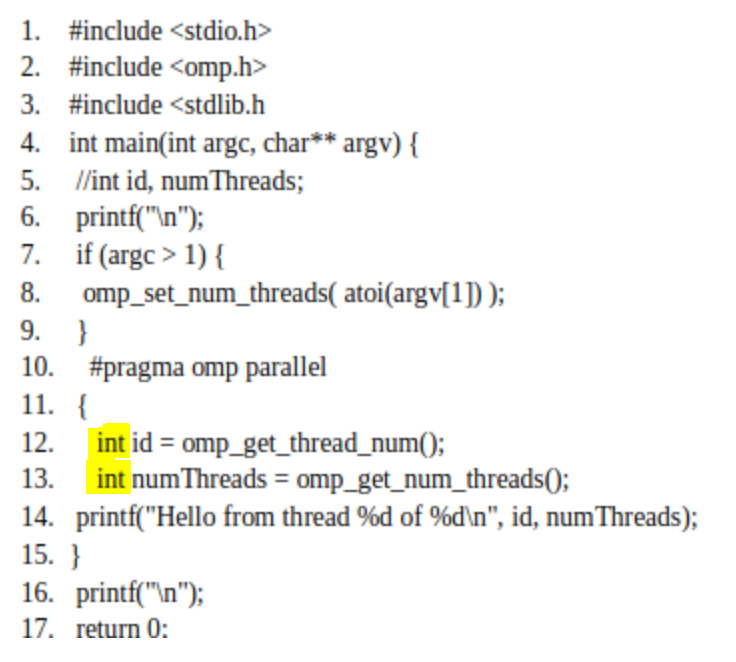
**Eric Drummond**

**Richard Lin**

**Wha Jong**

Race Condition:

Race condition is when a system’s output depends on the sequence or timing of the independent variables in the program. Race condition is hard to reproduce and debug because the result is unpredictable and depends on the timing of other threads. Because of this, it’s better to fix the problem before the debugging step. One can do this by declaring variables specifically. For example, in a previous assignment, we were given a program (spmd2.c) that’s output also depended on the sequence of other uncontrollable events as well as shared memory.



Without specifying the type of variable in the variable declaration, race condition will occur; however, if the programmer changes this to a full variable declaration by including the type “int”, each thread will have its own private copy of the variables, and race condition will not occur.

Parallel Programming Patterns:

Patterns are important in parallel programming because they define valuable techniques that can be used repeatedly by any programmer to write and organize parallel programs.

Patterns in parallel programming can be divided into two categories: Strategies and Concurrent Execution Mechanisms. Strategies consist of two other categories: algorithmic strategies and implementation strategies.

Algorithmic strategies concern how decisions are made about which tasks can be done at the same time on multiple processing units working at the same time. On the other hand, implementation strategies concern how the structure of the program is affected by the patterns and the data that’s being computed by the multiprocessor.

Concurrent Execution Mechanisms include Process/Thread control patterns, patterns that show how processors of parallel execution on hardware are controlled at runtime and Coordination patterns, patterns that arrange how tasks running at the same time coordinate to successfully output the parallel computation. Coordination patterns involve the use of message passing between concurrent processors on single multi-processors (MPI library) or distributed computer clusters and mutual exclusion between threads executing at the same time on a single shared memory system (OpenMP library). Furthermore, hybrid computation is a combination of both Process and Coordination patterns.

Categorizing Patterns:

1. Collective Synchronization (barrier) and Collective Communication (reduction) are both coordination patterns under concurrent execution mechanisms in parallel application. A barrier blocks all processes until other processes reach the barrier while reduction simply simplifies elements in an array into one result using commutative and associative properties.
2. Master-Worker and Fork-Join are similar program structures categorized under implementation strategies. Both patterns use a divide and conquer model. Fork join is a pattern in which execution is divided in parallels and then merged back together at certain points. It can be more efficient when using multiple threads. Master-Worker is a pattern where a master divides work among workers and combines the finished results.

Dependency:

1. Parallelism can be found in parallel execution. Parallel execution can be restricted by the process of operations necessary to yield the right outcome and revolves around control, data, and system dependences.
2. Dependency is when one operation depends on an earlier operation to complete and produce a result before it can be performed or when an operation depends on a particular resource. Specifically, where the data is located will affect the operations.
3. A statement is independent when the order of execution doesn’t affect the result. Statements are independent in an instance such that:

Statement 1: a = 1;

Statement 2: b = 1;

In contrast, a statement is dependent when the order of their execution affects the computation result. Statements are dependent in an instant such that:

Statement 1: a = 1;

Statement 2: b = a;

1. Two statements can execute in parallel if and only if statement 1 and 2 are completely independent of each other. That means true dependences, anti-dependences, and output dependences cannot exist.
2. Dependency can be removed by rearranging or eliminating statements.
3. Dependency can be computed by comparing the variables used in a statement with the variables modified in a statement. We can compute dependency for the loops below by unraveling the loop into separate statements and analyzing each statement from there.

