

BUCKNELL UNIVERSITY  
Computer Science

CSCI 315 Operating Systems Design

## OpenMP: Multicore Programming

**Notice:** This set of slides is based on the notes by Professor Perrone of Bucknell and the textbook authors Silberschatz, Galvin, and Gagne, as well as the [tutorial](#) by Blaise Barney from Lawrence Livermore National Lab

# Multicore Programming

- We have seen synchronization issues when multiple threads and multiple processes are involved
- Many programming models can tackle this type of problems
- We briefly discuss one in the multi-core domain

# Multi-core Computers and Programming

- A modern processor may have multiple *cores*.
- A multi-core processor has multiple instruction execution units including registers and caches
  - Multiple instructions can be executed at the same time
- Multi-cores share memory
- Programming multi-core processor means writing programs that take advantage of the multiple cores.

# OpenMP


- OpenMP is a programming environment for C and Fortran that has programming structures to support multi-core programming
- OpenMP (Open Multi-Processing) works well for multi-threads on single core, too
- <http://en.wikipedia.org/wiki/OpenMP>

# A Simple Example

```
/* To compile, enter:
 *      gcc -fopenmp openmp.c
 * You should see the message "I am a parallel region" for each
 * processing core on your system.
 */
#include <omp.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    /* sequential code */
    #pragma omp parallel
    {
        printf("I am a parallel region\n");
    }
    /* sequential code */
    return 0;
}
```

# Execution Result

Running the program on a 4-core computer



```
[xmeng@polaris thread]$ ./openmp  
I am a parallel region  
I am a parallel region  
I am a parallel region  
I am a parallel region  
[xmeng@polaris thread]$
```

# A Example with Shared Data

```
/* To compile, enter: gcc -fopenmp openmp-m.c */
#include <omp.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    /* sequential code */
    int v = 0;  int tid;  int nthreads;
#pragma omp parallel shared(v, nthreads) private(tid)
    {
        tid = omp_get_thread_num();
        if (tid == 0) {
            nthreads = omp_get_num_threads();
            printf("Number of threads = %d\n", nthreads);
        }
#pragma omp critical (addv)
        {
            v ++;
        }
        printf("I am a parallel region (thread id == %d)\n", tid);
    }
    /* sequential code */
    printf("value of v = %d\n", v);
    return 0;
}
```

# Execution Result

Running the program on a 4-core computer

```
[xmeng@polaris thread]$ ./openmp-m  
I am a parallel region (thread id == 2)  
I am a parallel region (thread id == 1)  
Number of threads = 4  
I am a parallel region (thread id == 0)  
I am a parallel region (thread id == 3)  
value of v = 4  
[xmeng@polaris thread]$
```

Thread 0 printed this line

The last part of the sequential code printed this line