CISC 3320
C12c: Implicit Threads

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook

Outline

- Implicit Threading
    - Thread Pools
    - Fork-Join
    - OpenMP
    - Grand Central Dispatch
    - Intel Threading Building Blocks
- Threading Issues
- Operating System Examples

Implicit Threading

- Difficult to write correct program with explicit threads, in particular, with lots of threads
- Creation and management of threads done by compilers and run-time libraries rather than programmers

Methods of Implicit Threading

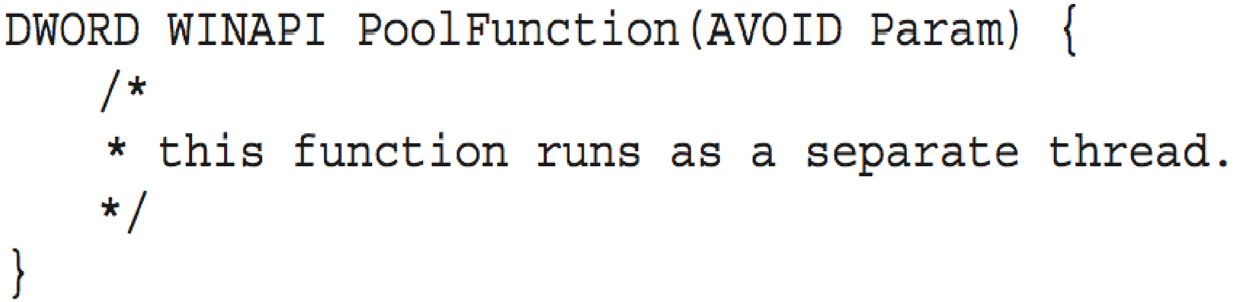
- Explore these following 5 methods
  - Thread Pools
  - Fork-Join
  - OpenMP
  - Grand Central Dispatch
  - Intel Threading Building Blocks

Thread Pools

- Create a number of threads in a pool where they await work
- Advantages:
  - Usually slightly faster to service a request with an existing thread than create a new thread
  - Allows the number of threads in the application(s) to be bound to the size of the pool
  - Separating task to be performed from mechanics of creating task allows different strategies for running task
    - i.e., Tasks could be scheduled to run periodically

Windows Thread Pools

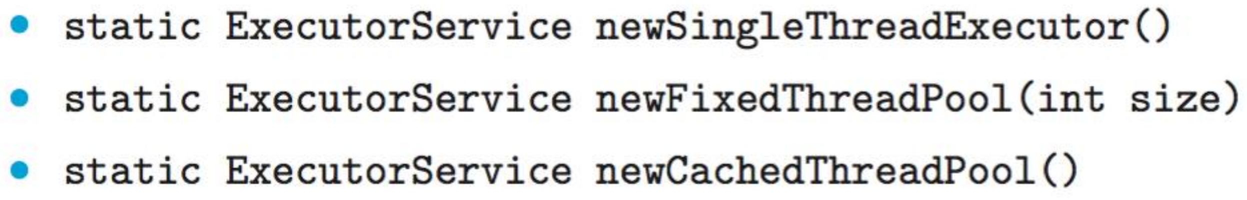
- Windows API supports thread pools:



```
DWORD WINAPI PoolFunction(AVOID Param) {
    /*
     * this function runs as a separate thread.
     */
}
```

Java Thread Pools

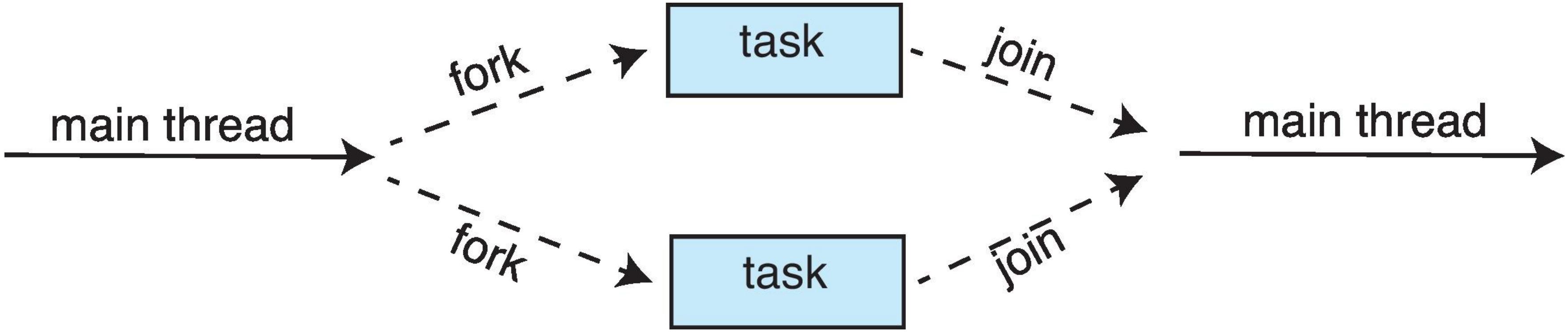
- Three factory methods for creating thread pools in Executors class:



- `static ExecutorService newSingleThreadExecutor()`
- `static ExecutorService newFixedThreadPool(int size)`
- `static ExecutorService newCachedThreadPool()`

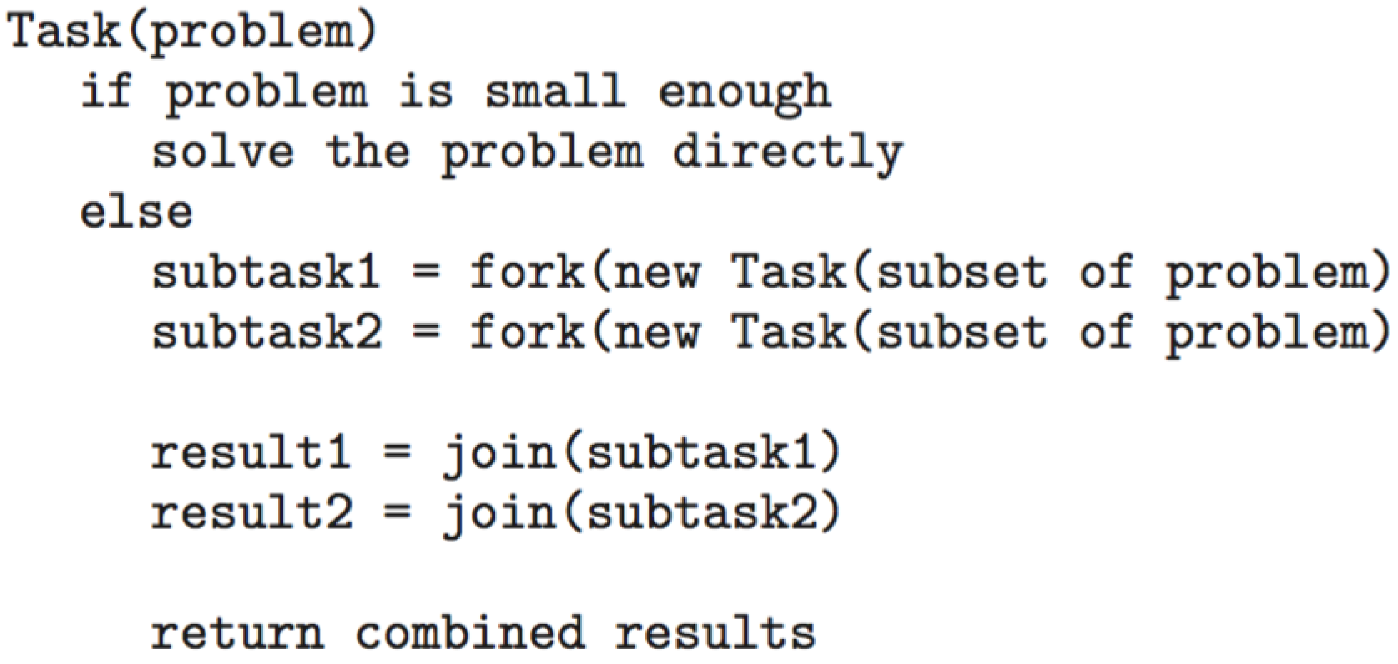
Fork-Join Parallelism

- Multiple threads (tasks) are **forked**, and then **joined**.



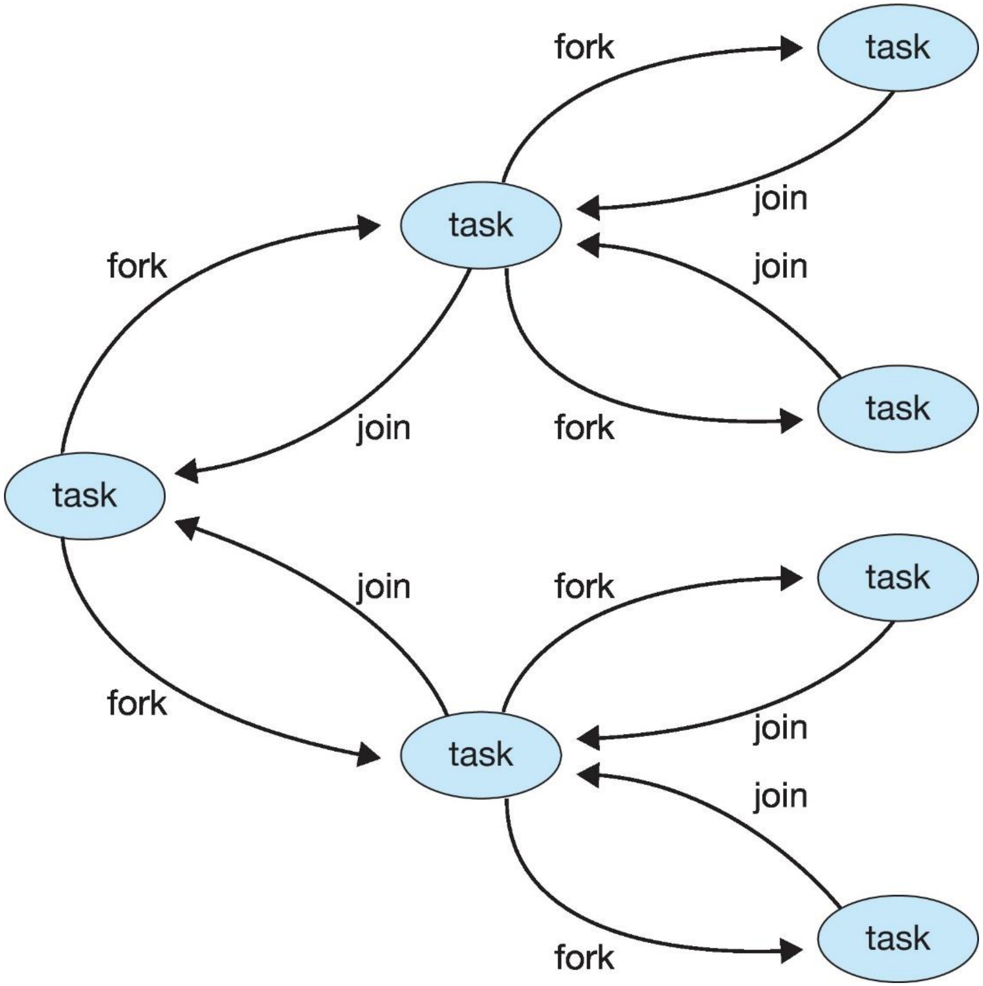
Fork-Join Strategy

- General algorithm for fork-join strategy:



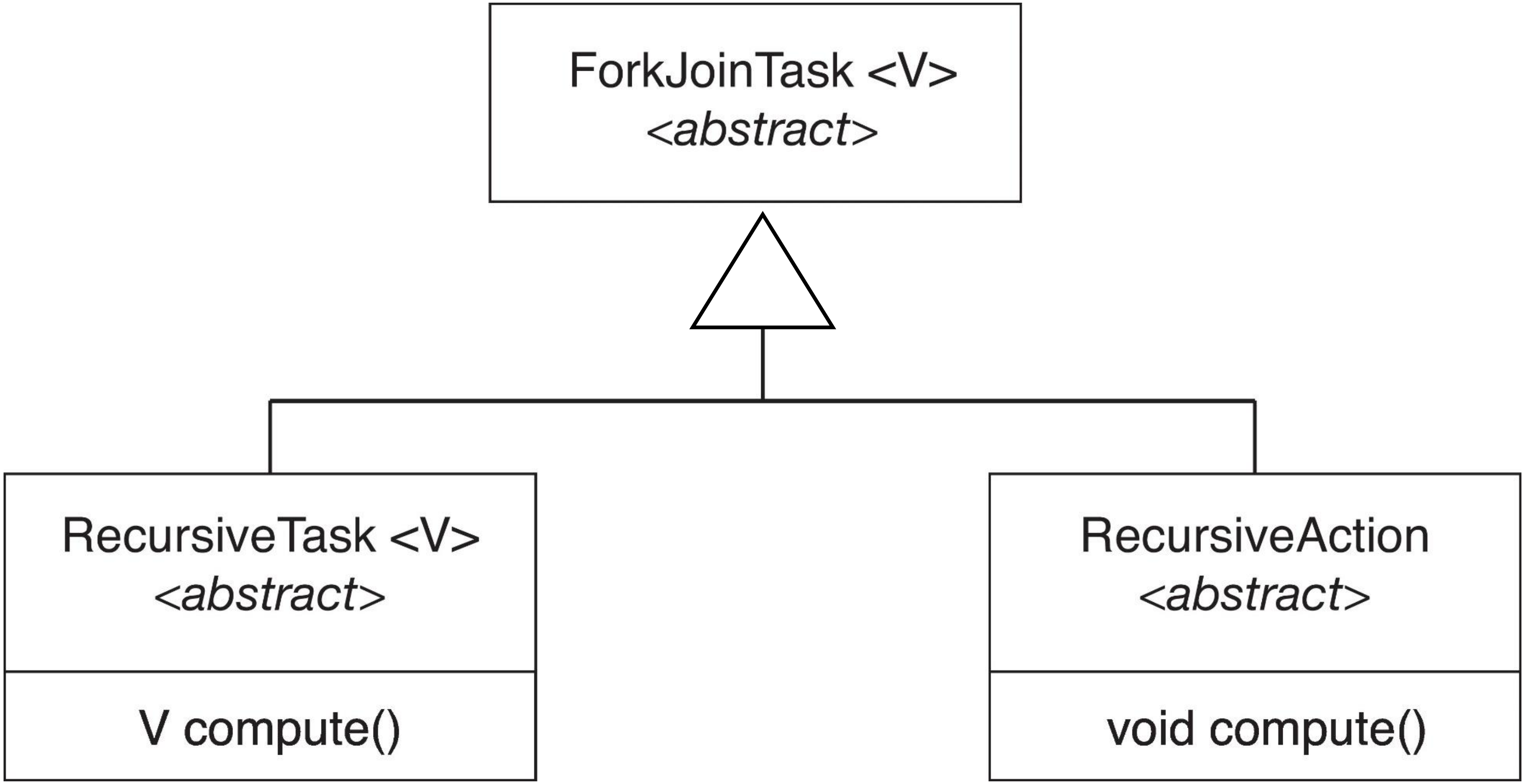
```
Task(problem)
   if problem is small enough
      solve the problem directly
   else
      subtask1 = fork(new Task(subset of problem)
      subtask2 = fork(new Task(subset of problem)

      result1 = join(subtask1)
      result2 = join(subtask2)

      return combined results
```

Fork-Join Parallelism



Fork-Join Parallelism in Java

- The ForkJoinTask is an abstract base class
- RecursiveTask and RecursiveAction classes extend ForkJoinTask
- RecursiveTask returns a result (via the return value from the compute() method)
- RecursiveAction does not return a result



ForkJoinTask <V>
<abstract>
RecursiveTask <V>
<abstract>
V compute()
RecursiveAction
<abstract>
void compute()

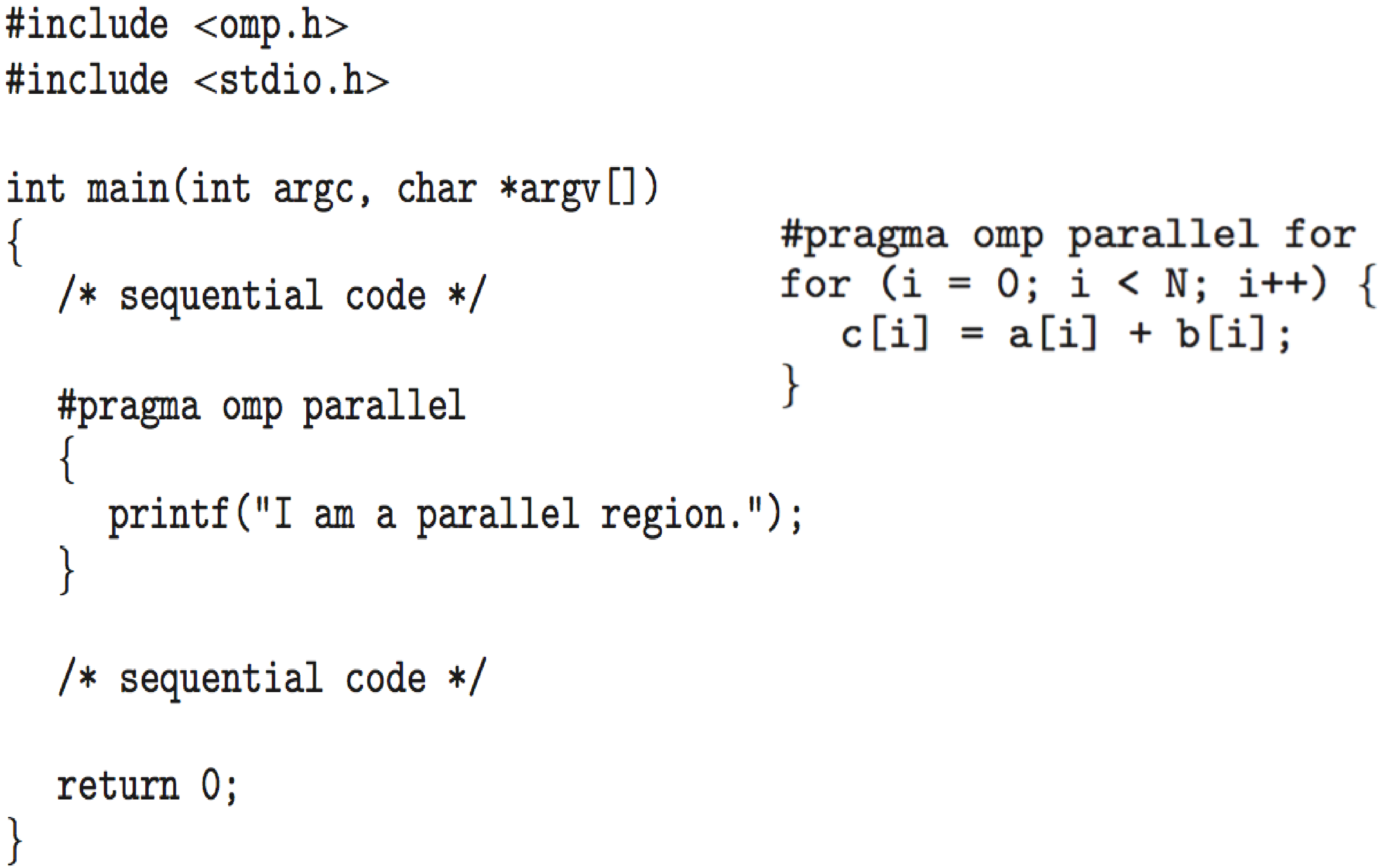
OpenMP

- Set of compiler directives and an API for C, C++, FORTRAN
- Provides support for parallel programming in shared-memory environments
- Identifies **parallel regions** – blocks of code that can run in parallel

```
#pragma omp parallel
```

- Create as many threads as there are CPU cores

OpenMP: Example



```
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
  /* sequential code */

  #pragma omp parallel
  {
    printf("I am a parallel region.");
  }

  /* sequential code */

  return 0;
}
```

```
#pragma omp parallel for
for (i = 0; i < N; i++) {
   c[i] = a[i] + b[i];
}
```

Grand Central Dispatch

- Apple technology for MacOS and iOS operating systems
- Extensions to C, C++ and Objective-C languages, API, and run-time library
- Allows identification of parallel sections
- Manages most of the details of threading
- Block is in “^{ }” :

```
^{ printf("I am a block"); }
```

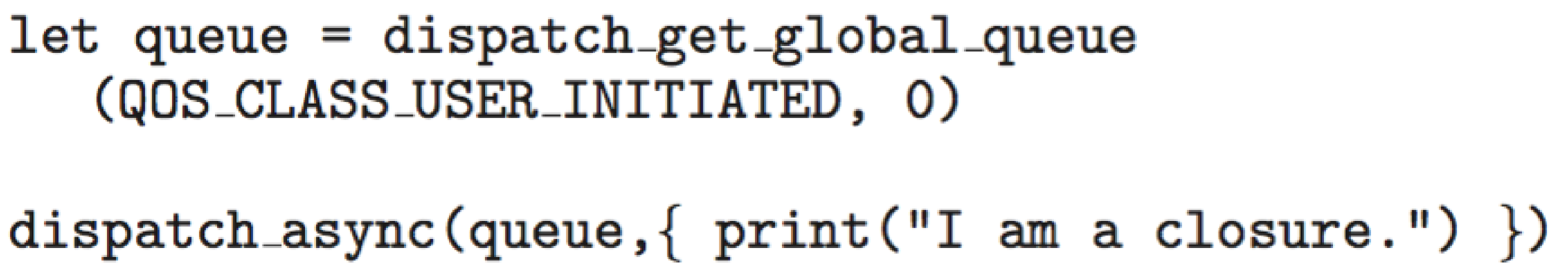
- Blocks placed in dispatch queue
  - Assigned to available thread in thread pool when removed from queue

Grand Central Dispatch: Dispatch Queues

- Two types of dispatch queues:
  - **serial** – blocks removed in FIFO order, queue is per process, called **main queue**
    - Programmers can create additional serial queues within program
  - **concurrent** – removed in FIFO order but several may be removed at a time
    - Four system wide queues divided by quality of service:
    - `QOS_CLASS_USER_INTERACTIVE`
    - `QOS_CLASS_USER_INITIATED`
    - `QOS_CLASS_USER_UTILITY`
    - `QOS_CLASS_USER_BACKGROUND`

Grand Central Dispatch: Swift

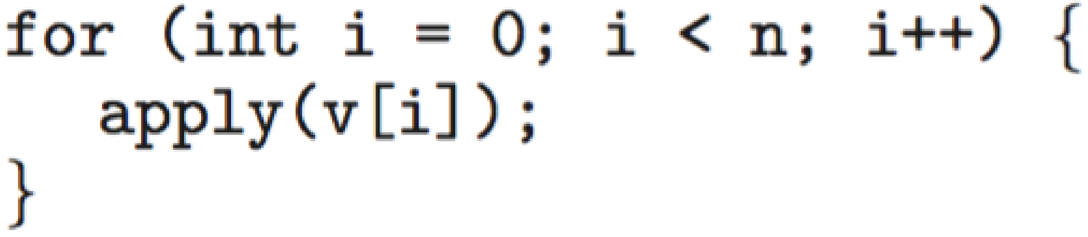
- For the Swift language a task is defined as a closure – similar to a block, minus the caret
- Closures are submitted to the queue using the `dispatch_async()` function:



```
let queue = dispatch_get_global_queue
   (QOS_CLASS_USER_INITIATED, 0)

dispatch_async(queue,{ print("I am a closure.") })
```

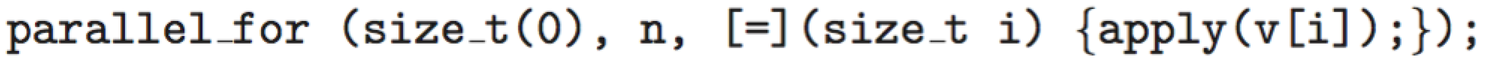
Intel Threading Building Blocks (TBB)

- Template library for designing parallel C++ programs
- A serial version of a simple for loop



```
for (int i = 0; i < n; i++) {
   apply(v[i]);
}
```

- The same for loop written using TBB with **`parallel_for`** statement:



```
parallel_for (size_t(0), n, [=](size_t i) {apply(v[i]);});
```

Questions?

- Implicit Threading
  - Thread Pools
  - Fork-Join
  - OpenMP
  - Grand Central Dispatch
  - Intel Threading Building Blocks