pthreads and the Dangers of Threading

Jonathan Misurda
jmisurda@cs.pitt.edu

pthreads

- pthreads (POSIX threads) is a library for doing threading
- Can transparently be used under User or Kernel threads

POSIX

- Portable Operating System Interface
- Standard to unify the programs and system calls that many different OSes provide.

pthread_create()

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

void *do_stuff(void *p) {
   printf("Hello from thread %d\n", *(int *)p);
}

int main() {
   pthread_t thread;
   int id, arg1, arg2;

   arg1 = 1;
   id = pthread_create(&thread, NULL, do_stuff, (void *)&arg1);
   arg2 = 2;
   do_stuff((void *)&arg2);
   return 0;
}
```

Output

```
Hello from thread 2
```

Yield!

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

void *do_stuff(void *p)
{
        printf("Hello from thread %d\n", *(int *)p);
}

int main()
{
        pthread_t thread;
        int id, arg1, arg2;

        arg1 = 1;
        id = pthread_create(&thread, NULL, do_stuff, (void *)&arg1);
        pthread_yield();
        arg2 = 2;
        do_stuff((void *)&arg2);

        return 0;
}
```

Output

Hello from thread 1

Hello from thread 2

pthread_join

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

void *do_stuff(void *p)
{
        printf("Hello from thread %d\n", *(int *)p);
}

int main()
{
        pthread_t thread;
        int id, arg1, arg2;

        arg1 = 1;
        id = pthread_create(&thread, NULL, do_stuff, (void *)&arg1);
        arg2 = 2;
        do_stuff((void *)&arg2);
        pthread_join(thread, NULL);
        return 0;
}
```

Output

Hello from thread 2

Hello from thread 1

Compile

- Need the –pthread option to gcc
- Links in the library

```
gcc –o threadtest threadtest.c -pthread
```

pthread_create()

```
int pthread_create(
   pthread_t *restrict thread,
   const pthread_attr_t *restrict attr,
   void *(*start_routine)(void*),
   void *restrict arg
);
```

- A unique identifier for the thread
- Thread attributes or NULL for the default
- A C Function Pointer
- The argument to pass to the function

Start Routine Prototype

```
void *(*start_routine)(void*)
```

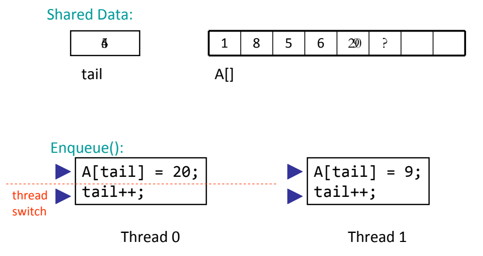
Java Threads

```
class TestThread implements Runnable {
        private int x;
        public static void main(String[] args) {
                Thread t1 = new Thread(new TestThread(1));
                Thread t2 = new Thread(new TestThread(2));

                t1.start();
                t2.start();
        }

        public void run() {
                System.out.println("Hello from thread " + x);
        }

        public TestThread(int y) { x = y; }
}
```

Output

```
Hello from thread 1
Hello from thread 2
```

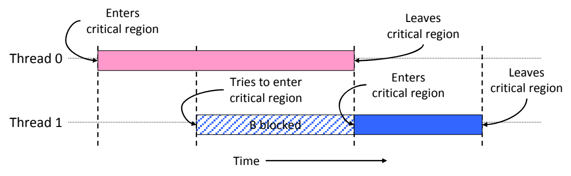
Race Condition



```
A[tail] = 20;
tail++;
```

```
A[tail] = 9;
tail++;
```

Critical Regions



Synchronization

- Scheduling can be random and preemption can happen at any time
- Need some way to make critical regions "atomic"