

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Deadlock

Lab 07



اللهم علمنا ما ينفعنا،، وانفعنا بما علمتنا،، وزدنا علماً



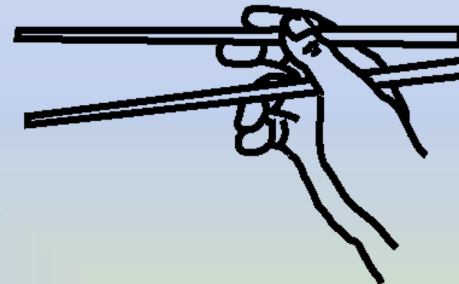
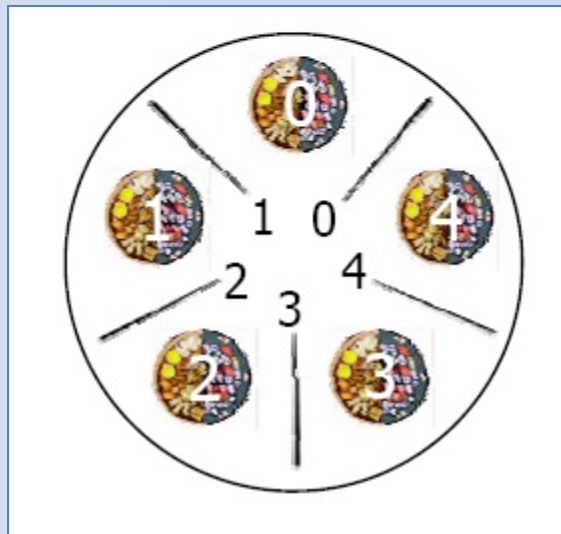
Lab Objective

- To practice resolving deadlock using Mutexes.



A Classic Example of Deadlock

- Dinning lawyers (philosophers)
- Each needs two chopsticks to eat



- If each first grabs the chopstick on their left before the one on their right, and all grab at the same time, we have a deadlock



Practice

- The following code implements the “*dinning lawyers*” deadlock example.
- Each lawyer needs two chopsticks to eat.
- The code creates 5 threads, one per lawyer, and it has five mutexes to represent the chopsticks.
- The `sleep(1)` in the code allows all threads (lawyers) to grab (lock) the chopstick on their left before the one on their right.
- The `sleep(2)` in the code represent the time taken for eating!
- Compile and run the program as shown then explain the output (**Do we have deadlock?**).



Practice (Cont.)

- Rewrite the code to resolve the deadlock problem.
- One way to solve the deadlock is to make every lawyer grab both chopsticks at the same time.
- To implement that, you need to define the following mutex:

```
pthread_mutex_t Chopstick_mutex =  
PTHREAD_MUTEX_INITIALIZER;
```

- In the thread function you will need to lock the new mutex before grabbing the first chopstick and unlock it after grabbing the second chopstick.



To lock use: `pthread_mutex_lock(&Chopstick_mutex);`



To unlock use: `pthread_mutex_unlock(&Chopstick_mutex);`

- Re-compile and run the program then record the new results.



```
#include <iostream>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <cstdint>
#include "pthread.h"
using namespace std;
```

**Define and initialize
five Mutexes to
represent the
chopsticks**

```
pthread_mutex_t Chopstick_mutex [5]=
{PTHREAD_MUTEX_INITIALIZER, PTHREAD_MUTEX_INITIALIZER,
PTHREAD_MUTEX_INITIALIZER, PTHREAD_MUTEX_INITIALIZER,
PTHREAD_MUTEX_INITIALIZER};
```

```
void *doit(void *);
```

**doit function
declaration**



```
int main()
{
    pthread_t Lawyer[5];
    int i;
    cout<<dec; // to display numbers in decimal format

    //Creating the Lawyers thread
    for(i=0;i<5;i++)
        pthread_create(&Lawyer[i], NULL, doit, (void *) (intptr_t)i);

    // wait for threads to terminate
    for(i=0;i<5;i++)
        pthread_join(Lawyer[i], NULL);

    //Leaving a mutex without destroying it can affect
    performance
    for(i=0;i<5;i++)
        pthread_mutex_destroy(&Chopstick_mutex[i]);

    exit(0);
}
```

**Creating 5
threads,
one per
lawyer**

**Waiting for
threads
termination**

**Destroying all
Mutexes**




```
void * doit(void *vptr)
```

```
{
```

```
//Get the Left Chopstick
```

```
pthread_mutex_lock(&Chopstick_mutex[(intptr_t)vptr]);
```

```
cout<<"Lawyer " << (intptr_t)vptr << " got chopstick number ";
```

```
cout <<(intptr_t) vptr << endl;
```

```
sleep(1);
```

```
//Get the Right Chopstick
```

```
pthread_mutex_lock(&Chopstick_mutex[((intptr_t) vptr+1)%5]);
```

```
cout<<"Lawyer " << (intptr_t)vptr << " got chopstick number ";
```

```
cout << ((intptr_t)vptr+1)%5 << endl;
```

```
//Eating
```

```
cout<<"Lawyer " << (intptr_t) vptr << " is eating with  
chopsticks ";
```

```
cout << (intptr_t) vptr << " & " << (((intptr_t)vptr+1)%5) <<  
endl;
```

```
sleep(2);
```

```
//Unlock both left and right Chopsticks
```

```
pthread_mutex_unlock(&Chopstick_mutex[(intptr_t)vptr]);
```

```
pthread_mutex_unlock(&Chopstick_mutex[((intptr_t)vptr+1)%5]);
```

```
return(NULL);
```

```
} //end doit function
```



Check Off

- 1) Compile and run the above program as shown then record the output showing how deadlock happens.
- 2) Resolve the deadlock by making every lawyer grab both chopsticks at the same time.
Re-compile and run the program then explain how the deadlock is resolved.





??? ANY QUESTIONS ???

