East West University

Course Title: Operating System
Course Code: CSE 325
Section: 01

Project: The Sandwich Problem
Group No :11

Submitted by:

Name: Samiu Esika Upoma
ID:2020-1-60-082

Name: Zarin Tasnim Nuzhat
ID: 2020-1-60-211

Name: Dewan Md. Anisur Rahman
ID: 2018-3-60-046

Submitted To:

Tanni Mitra(TM)
Senior Lecturer
Department of Computer Science and Engineering
East West University

Project Title: The Sandwich Problem

Project Description:
There are four students sitting in a room: three juniors and one senior. Each of the juniors will make a sandwich and eat it. To make a sandwich requires bread, cheese, and sausage. Each junior has one of the three items, that is, one has bread, another has cheese and the third has sausage. The senior has infinite supply of all three items. The senior places two of the three items on the table, and the junior that has the third item, makes the sandwich. However, the other two juniors cannot make a sandwich because they do not have the third required item. Thus, a junior can make a sandwich only when all the three items are available to him. Implement a synchronization method to solve the problem.

Project Analysis:

1. Three juniors and one senior
2. Total Items 3. Bread, Cheese, and Sausage
3. Senior who has infinite items always places any two items in the table.
4. Each of the juniors will make a sandwich who has third item.
5. The other two juniors cannot make a sandwich if they do not have the third required item.

Main Code:

1.There are some important header files in this code for semaphore, threads, piping and exec functions. Here I define some integers value and semaphore and threads. For 3 items I define a **MaxItems = 3**. As senior randomly places two items I define a buffer: **BufferSize= 2**
For thread synchronization I use semaphore where:
semEmpty is Number of empty slots in buffer.
semFull is Number of filled slots in buffer.
mutex use for locking the thread before entering the critical section.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <semaphore.h>
#define MaxItems 3
#define BufferSize 2

sem_t semEmpty;      //Number of empty slots in buffer
sem_t semFull;       //Number of filled slots in buffer
sem_t mutex;
```

```
int buffer[BufferSize];
int in = 0;
int out =0 ;

int item ;
int i ,temp =-1 ;
int count=0;
int  p1[2];    //pipe declare
char w1_number[30] ;
```

2.Here I use 2 functions for piping. Where Senior will write message and junior will read it.

```
void write_messege(){

    close (p1[0]);  //pipe() writing
    printf ("\nSenior send messeage : Take two items!\n");
    //fgets (w1_number,30,stdin);
    write (p1[1],w1_number,strlen(w1_number));
    close (p1[1]);

}
void read_messege(){
if (pipe(p1)!=-1){
        close (p1[1]);
        read (p1[0],w1_number,30);
        printf ("\nJuniors recived message:Take two items!\n");
        close (p1[0]);
   }
}
```

3.Here I use 2 functions: **senior give item ()** and **junior take item ().** In senior give item () function senior will randomly produce 2 items among cheese, bread, and sausage. I also use rand () function for random number. Also give some conditions for produce 2 items. I use **count** variable here for junior take item () function. Count variable will add the 2 numbers according to the random items. And check the conditions which one will be the third item.

```
Void senior_give_item ()
{
```

```
  printf (“\nSenior produced Two items :\n”);
    srand(time(NULL));
    for (i=0;i<2;i++)

{
       item = rand() % MaxItems;

       if (temp == item)
       {
           i--;
           continue;
       }

       if (item == 0 ){
       printf (“Bread\n”);

       }
       else if (item ==1 ){
       printf (“cheese\n”);

       }
       else if (item ==2){
       printf (“Sausage\n”);

       }

       count =count+item ;

       temp = item;
}
}
 void junior_take_item()
 {
  printf (“\nThird Item is :\n”);

if (count==1){
printf(“’Sausage’”);
}
else if (count==2){
printf(“’Cheese’”);
}

else if (count==3){
printf(“’Bread’”);

}
printf (“\n”);

 }
```

4. In this part, there are two thread functions**: producer () , consumer()** .

In **producer()** function :

- The semaphore **semEmpty** will decrease the buffer size from 2 to 1, 1 to 0. Because when senior produced items the empty slots will be filled up and decreased the empty space.
- The semaphore **mutex** will decrease the semaphore value from 1 to 0 and lock the threads and go to the critical section.
- In critical section we call those 2 functions: senior_give_item() and write_messege() because they were sharing global variable .
- The semaphore **semFull** will increased from 0 to 1 , 1 to 2 . Because when senior produced items the empty slots will be filled up and increased the filled space.
- Then again **mutex** will unlock the thread by increasing its value from 0 to 1.

In **Consumer()** function :

- The semaphore **semFull** will deccreased from 2 to 0. Because when junior consumed 2 items the filled slots will be empty and decreased the filled space.
- The semaphore **mutex** will decrease the semaphore value from 1 to 0 and lock the threads and go to the critical section.
- In critical section we call those 2 functions: read_messege() and junior_take_item() because they were sharing global variable .
- The semaphore **semEmpty** will increased from 0 to 1, 1 to 2 . Because when junior consumed items the filled slots will be empty and increased the empty space.
- Then again **mutex** will unlock the thread by increasing its value from 0 to 1.

```
Void* producer()  //senior produced 2 items
{


    sem_wait(&semEmpty);
    sem_wait(&mutex);   //semaphore value down (1=0)
    //<critical section>
    buffer[in] = item;

    senior_give_item ();
    write_messege();

    in = (in+1)%BufferSize;
    sem_post(&mutex);    //semaphore value up (0=1)
    sem_post(&semFull);


}

void* consumer()    //junior consumed 2 items
{
```

```
     sem_wait(&semFull);
     sem_wait(&mutex);
     //<Critical Section>
     int item = buffer[out];

     read_messege();
     junior_take_item();

     out = (out+1)%BufferSize;
     sem_post(&mutex);
     sem_post(&semEmpty);


}
```

5. In main function we just initialized, joined (Main will waiting for threads) and destroy/exit the semaphores ,threads and also call the execv functions .

```
int main(int argc ,char *argv[])
{
    pthread_t pro ,con ;
    sem_init(&semEmpty, 0, 2);
    sem_init(&semFull,  0, 0);
    sem_init(&mutex  ,  0, 1);


    pthread_create(&pro, NULL, &producer, NULL) ;
    pthread_create(&con, NULL, &consumer, NULL) ;


    pthread_join(pro, NULL);
    pthread_join(con, NULL);

    execv("./exec",argv);

    sem_destroy(&semEmpty);
    sem_destroy(&semFull);
    sem_destroy(&mutex);
    pthread_exit(NULL);         //terminates the calling thread
    return 0;
```

Exec file:

6. Here we use fork. In parent process we print “Sandwich make successfully!!!” which will print only when child process executes. Because of wait() function .

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
pid_t a=fork();
if (a<0){
printf("fork failed");
}

else if(a==0){

printf("\nJunior who has third item start making sandwich\n");
}
else{

wait(NULL);
printf("\nSandwich make successfully!!!\n\n");
}

return 0 ;
}
```

Output :

```
ubuntu@ubuntu-VirtualBox:~$ cd Desktop/Project
ubuntu@ubuntu-VirtualBox:~/Desktop/Project$ gcc project.c -lpthread
ubuntu@ubuntu-VirtualBox:~/Desktop/Project$ gcc exec.c -o exec
ubuntu@ubuntu-VirtualBox:~/Desktop/Project$ ./a.out

Senior produced Two items :
cheese
Sausage

Senior send messeage : Take two items!

Juniors recived message:Take two items!

Third Item is :
'Bread'

Junior who has third item start making sandwich

Sandwich make successfully!!!
```

```
Senior produced Two items :
Bread
cheese

Senior send messeage : Take two items!

Juniors recived message:Take two items!

Third Item is :
'Sausage'

Junior who has third item start making sandwich

Sandwich make successfully!!!
```

```
Senior produced Two items :
Bread
Sausage

Senior send messeage : Take two items!

Juniors recived message:Take two items!

Third Item is :
'Cheese'

Junior who has third item start making sandwich

Sandwich make successfully!!!
```