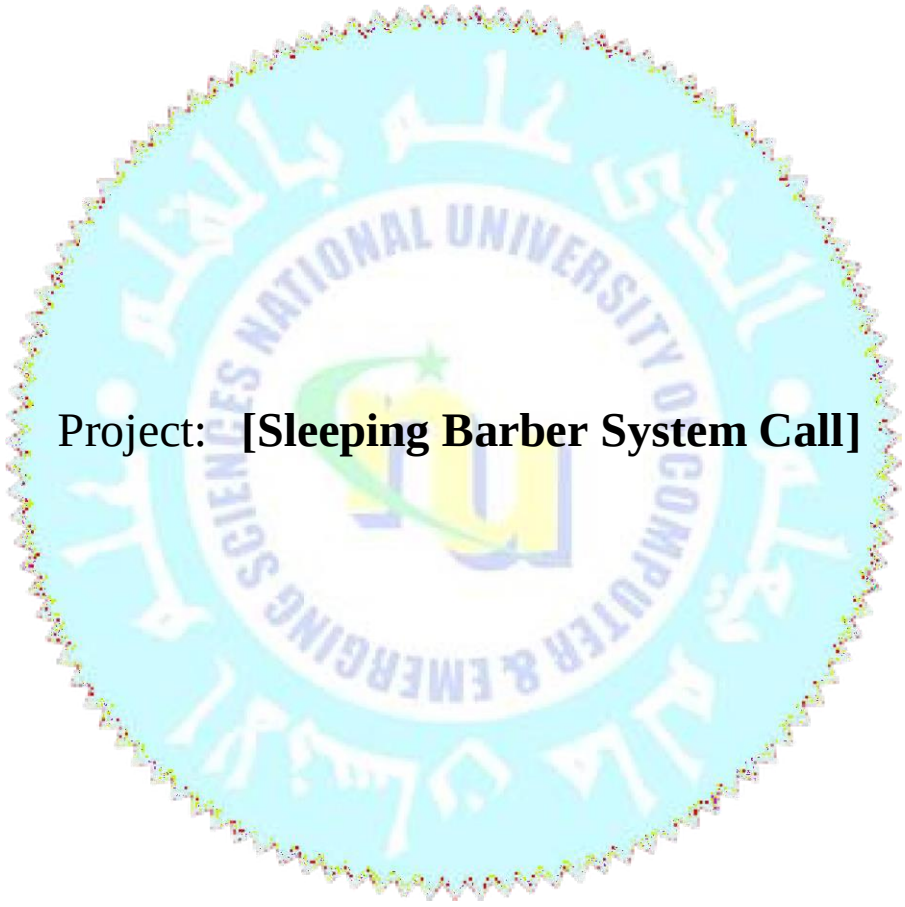
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(KARACHI CAMPUS)
Department of Computer Science
Spring 2022



Project: **[Sleeping Barber System Call]**

Group Members:

[Syed Aun Ali Zaidi (Group Leader)] - [20K-0286]

[Muhammad Anas]-[20K-0179]

[Ammar Amin]-[20K-0285]

Introduction:

Sleeping Barber problem is a classical process synchronization problem. The context of the problem is as follows: The barber shop has one barber, one barber chair, and n (3 in our case) chairs for waiting customers to sit on. If there are no customers present, the barber sits down in the barber chair and falls asleep. When a customer arrives, he has to wake up the sleeping barber. If additional customers arrive while the barber is cutting a customer's hair, they either sit down (if there are empty chairs) or leave the shop (if all chairs are full). The problem is to program the barber and the customers without getting into race conditions.

Approach:

Our solution uses semaphores, sleepbarber, which tells whether the barber is asleep or not. Waitroom semaphore, which allows the customers to take a seat in the waiting area (if there are more customers than the waiting chairs available, then the customer leaves the shop). Chairbarber semaphore, used for mutual exclusion and finally cust_wait semaphore which prevents the customer from leaving until his haircut is finished. When the barber opens up his shop, he executes the barber procedure and if there are no customers to service, he goes to sleep. When a customer arrives, he executes customer, starting by acquiring mutex to enter a critical region. If another customer enters shortly thereafter, the second one will not be able to do anything until the first one has released mutex. The customer then checks to see if the number of waiting customers is less than the number of chairs. If not, he releases mutex and leaves without a haircut. If there is a chair available, the customer acquires it and wakes up the barber. The customer leaves the shop once his haircut is finished. Barber then checks for waiting customer, if there is not any, the barber goes to sleep.

.C Implementaion:

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define max_cust 8
//////////////////////////////////semaphores/////////////////////////////////////
sem_t chairbarber;
sem_t sleepbarber;
sem_t waitroom;
sem_t cust_wait;
int job_done = 0;
//////////////////////////////////functions/////////////////////////////////////
void *customer(void *number){
int num = *(int *)number;
printf("Customer %d left to get a haircut.\n", num);
sleep(5);
printf("Customer %d reached the barber shop.\n", num);
sem_wait(&waitroom);
printf("Customer %d entered the waiting room.\n", num);
sem_wait(&chairbarber);
sem_post(&waitroom);
printf("Customer %d asked the sleeping barber for a haircut.\n", num);
sem_post(&sleepbarber);
sem_wait(&cust_wait);
sem_post(&chairbarber);
printf("Customer %d left the barber shop.\n", num);
}
void *barber(void *junk){
while (!job_done){
printf("The barber is sleeping\n");
sem_wait(&sleepbarber);
if (!job_done){
printf("The barber is busy servicing the customer\n");
sleep(5);
printf("The barber has finished servicing the customer.\n");
sem_post(&cust_wait);
}
else{
printf("The Barber's Job Is All Done For The Day!'.\n");}}}
//////////////////////////////////main() & Input/////////////////////////////////////////////////////
int main(){
pthread_t btid;
pthread_t tid[max_cust];
int Number[max_cust];
int count, x, num_cust, num_chairs;
printf("Enter Number Of Chairs And Customers (Make Sure The Customers Are Lesser Than 8): \n");
scanf("%d",&x);
num_chairs = x;
scanf("%d",&x);
num_cust = x;
//////////////////////////////////initializing/////////////////////////////////////////////////////
sem_init(&chairbarber, 0, 1);
sem_init(&sleepbarber, 0, 1);
sem_init(&waitroom, 0, num_chairs);
sem_init(&cust_wait, 0, 0);
//////////////////////////////////implementation/////////////////////////////////////////////////////
if (num_cust > max_cust){
printf("The maximum number of customers allowed in the store is %d.\n", max_cust);
return 0;
}
printf("Sleeping Barber Problem Implementation Using Semaphores:\n");
for (count = 0; count < max_cust; count++) {
Number[count] = count;
}
//////////////////////////////create customers and barber/////////////////////////////////////
for (count = 0; count < num_cust; count++) {
pthread_create(&tid[count], NULL, customer, (void *)&Number[count]);
}
pthread_create(&btid, NULL, barber, NULL);
//////////////////////////////joining & finishing/////////////////////////////////////////////////////
for (count = 0; count < num_cust; count++) {
pthread_join(tid[count],NULL);
}
job_done = 1;
sem_post(&sleepbarber);
pthread_join(btid,NULL);
return 0;
}
```

Kernel Implementation:

```c
#include<linux/delay.h>
#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/kthread.h>
#include<linux/sched.h>
#include<linux/time.h>
#include<linux/timer.h>
#include<linux/semaphore.h>
#define max_cust 10
//////////////////////////////////semaphores/////////////////////////////////////

static struct semaphore chairbarber;
static struct semaphore sleepbarber;
static struct semaphore waitroom;
static struct semaphore cust_wait;

static struct task_struct *btid;
static struct task_struct *tid[7];

int job_done = 0;
//////////////////////////////////functions/////////////////////////////////////
int customer(void *number){
int num = *(int *)number;
printk("Customer %d is leaving for the barber shop\n",num);
msleep(5);
printk("Customer %d has reached the barber shop\n",num);
down(&waitroom);
printk("Customer %d has reached the barber shop\n",num);
down(&chairbarber);
up(&waitroom);
printk("Customer %d is waking the barber up\n",num);
up(&sleepbarber);
down(&cust_wait);
up(&chairbarber);
printk("Customer %d is leaving the barber shop\n",num);
return 0;
}

int barber(void *junk){
while (!job_done){
printk("The Barber is sleeping\n");
down(&sleepbarber);
if (!job_done){
printk("The Barber is cutting hair\n");
msleep(5);
printk("The Barber has finished cutting hair\n");
up(&cust_wait);
}
else{
printk("The Barber has completed his work for the day\n");
return 0;}}}
//////////////////////////////////main() & Input/////////////////////////////////////////////////////////
asmlinkage long sys_hello(void){
int Number[max_cust];
int count, x, num_cust, num_chairs;
num_chairs = 3;
num_cust = 7;
printk("Number Of Chairs are 3\nNumber of Customers are 7\n");
//////////////////////////////////initializing/////////////////////////////////////////////////
sema_init(&chairbarber, 1);
sema_init(&sleepbarber, 1);
sema_init(&waitroom, num_chairs);
sema_init(&cust_wait, 0);
//////////////////////////////////implementation/////////////////////////////////////////////////
if (num_cust > max_cust){
printk("Max allowed Customers are %d\n",max_cust)
return 0;
}
printk("--------------------SOLVING THE SLEEPING BARBER PROBLEM-------------------------\n");
for (count = 0; count < max_cust; count++) {
Number[count] = count;
}
//////////////////////////create customers and barber/////////////////////////////////
for (count = 0; count < num_cust; count++) {
tid[count] = kthread_create(customer, (void *)&Number[count], "customerthread");
if(tid[count]){
wake_up_process(tid[count]);}
}
btid = kthread_create(barber,NULL,"barberthread");
if(btid){
wake_up_process(btid);}
//////////////////////////////joining & finishing/////////////////////////////////////////
for (count = 0; count < num_cust; count++) {
kthread_stop(tid[count]);
}
job_done = 1;
up(&sleepbarber);
kthread_stop(btid);
return 0;
}
```

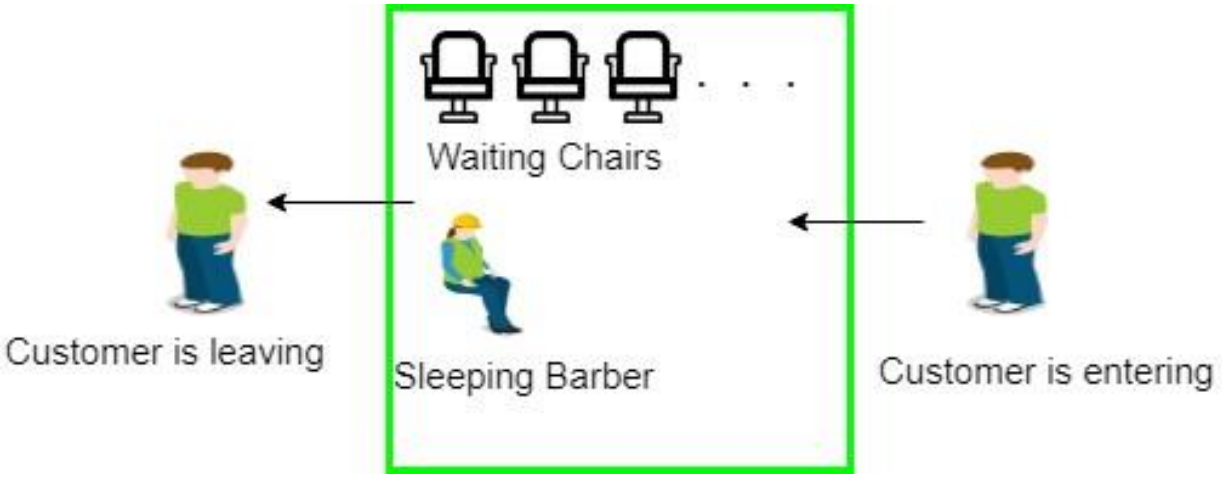
Userpace.c:

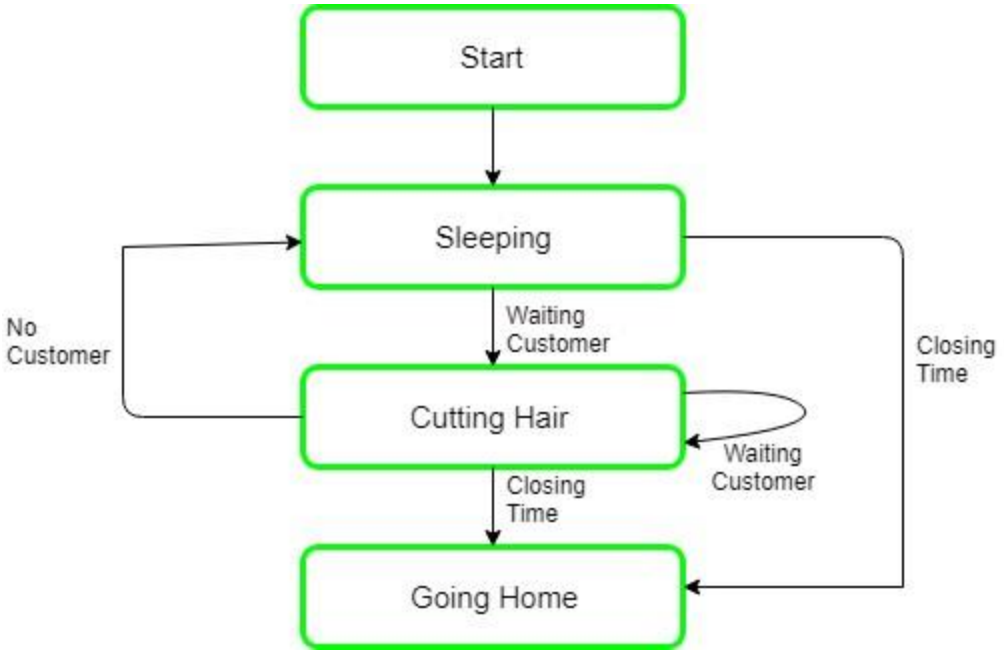
*userspace.c (~/) - gedit

Open

```c
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
long int i = syscall(335);
printf("System call slpeeping barber returned %ld\n", i);
return 0;
}
```

Sleeping Barber Problem Visualized:





Output Of .C:

```
Sleeping Barber Problem Implementation Using Semaphores:
Customer 0 left to get a haircut.
Customer 1 left to get a haircut.
Customer 2 left to get a haircut.
Customer 3 left to get a haircut.
Customer 4 left to get a haircut.
Customer 5 left to get a haircut.
Customer 6 left to get a haircut.
The barber is sleeping
The barber is busy servicing the customer
Customer 1 reached the barber shop.
Customer 1 entered the waiting room.
Customer 1 asked the sleeping barber for a haircut.
Customer 0 reached the barber shop.
Customer 0 entered the waiting room.
Customer 2 reached the barber shop.
Customer 2 entered the waiting room.
Customer 6 reached the barber shop.
Customer 6 entered the waiting room.
Customer 5 reached the barber shop.
The barber has finished servicing the customer.
Customer 1 left the barber shop.
Customer 0 asked the sleeping barber for a haircut.
Customer 4 reached the barber shop.
Customer 4 entered the waiting room.
The barber is sleeping
The barber is busy servicing the customer
Customer 3 reached the barber shop.
The barber has finished servicing the customer.
Customer 0 left the barber shop.
Customer 2 asked the sleeping barber for a haircut.
Customer 5 entered the waiting room.
The barber is sleeping
The barber is busy servicing the customer
The barber has finished servicing the customer.
Customer 2 left the barber shop.
Customer 6 asked the sleeping barber for a haircut.
Customer 3 entered the waiting room.
The barber is sleeping
The barber is busy servicing the customer
The barber has finished servicing the customer.
The barber is sleeping
The barber is busy servicing the customer
Customer 6 left the barber shop.
Customer 4 asked the sleeping barber for a haircut.
The barber has finished servicing the customer.
The barber is sleeping
The barber is busy servicing the customer
Customer 4 left the barber shop.
Customer 5 asked the sleeping barber for a haircut.
The barber has finished servicing the customer.
The barber is sleeping
The barber is busy servicing the customer
Customer 5 left the barber shop.
Customer 3 asked the sleeping barber for a haircut.
The barber has finished servicing the customer.
The barber is sleeping
The barber is busy servicing the customer
Customer 3 left the barber shop.
The barber has finished servicing the customer.
aunyx@ubuntu:~/Desktop$
```

Output Of Kernel (return & dmesg):

```
ammar@ammar-VirtualBox:~$ gedit userspace.c
ammar@ammar-VirtualBox:~$ gcc userspace.c
ammar@ammar-VirtualBox:~$ ./a.out
System call slpeeping barber returned 0
ammar@ammar-VirtualBox:~$
```

```
[  252.990710] Customer 1 is in the waiting room
[  253.002027] The Barber has finished cutting hair
[  253.002032] The Barber is sleeping
[  253.002037] Customer 3 is leaving the barber shop
[  253.002087] Customer 6 is waking the barber up
[  253.002091] The Barber is cutting hair
[  253.002347] Customer 7 is in the waiting room
[  253.014250] The Barber has finished cutting hair
[  253.014256] The Barber is sleeping
[  253.014274] Customer 6 is leaving the barber shop
[  253.014402] Customer 5 is waking the barber up
[  253.014496] The Barber is cutting hair
[  253.014537] Customer 4 is in the waiting room
[  253.027274] The Barber has finished cutting hair
[  253.027291] The Barber is sleeping
[  253.027399] Customer 5 is leaving the barber shop
[  253.027458] Customer 1 is waking the barber up
[  253.027466] The Barber is cutting hair
[  253.038249] The Barber has finished cutting hair
[  253.038258] The Barber is sleeping
[  253.038286] Customer 1 is leaving the barber shop
[  253.038490] Customer 7 is waking the barber up
[  253.038585] The Barber is cutting hair
[  253.050344] The Barber has finished cutting hair
[  253.050364] The Barber is sleeping
[  253.050536] Customer 7 is leaving the barber shop
[  253.050681] Customer 4 is waking the barber up
[  253.050807] The Barber is cutting hair
[  253.062143] The Barber has finished cutting hair
[  253.062160] The Barber is sleeping
[  253.062387] Customer 4 is leaving the barber shop
[  253.062915] The Barber has completed his work for the day
```

Conclusion:

Our project is giving the required output via the required manner. The sleeping barber problem was implemented both as a system call and as a .c file. Thank You!