AYAZ HASAN
20K-1044
SE-5A

LAB 10

IPC.c

```
student@student-HA1: ~
File  Edit  View  Search  Terminal  Help
student@student-HA1:~$
student@student-HA1:~$ gedit ipc.c
student@student-HA1:~$ gcc -o obj -pthread ipc.c
ipc.c: In function ‘main’:
ipc.c:46:11: warning: implicit declaration of function ‘fork’ [-Wimplicit-functi
on-declaration]
     pid = fork();
           ^~~~
student@student-HA1:~$ ./obj
Parent : Wait for Child to Print
Child : I am done! Release Semaphore
Child : Done with sem_open
Parent : Child Printed!
Parent : Done with sem_open
student@student-HA1:~$ gedit ipc.c
```

```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include<sys/wait.h>
#include <stdlib.h>
#include <fcntl.h>

const char *semName = "asdfsd";

void parent(void){
    sem_t *sem_id = sem_open(semName, O_CREAT, 0600, 0);

    if (sem_id == SEM_FAILED){
        perror("Parent : [sem_open] Failed\n"); return;
    }

    printf("Parent : Wait for Child to Print\n");
    if (sem_wait(sem_id) < 0)
        printf("Parent : [sem_wait] Failed\n");
    printf("Parent : Child Printed! \n");

    if (sem_close(sem_id) != 0){
        perror("Parent : [sem_close] Failed\n"); return;
    }


    if (sem_unlink(semName) < 0){
        printf("Parent : [sem_unlink] Failed\n"); return;
    }
}
void child(void)
{
    sem_t *sem_id = sem_open(semName, O_CREAT, 0600, 0);

    if (sem_id == SEM_FAILED){
        perror("Child : [sem_open] Failed\n"); return;
    }

    printf("Child : I am done! Release Semaphore\n");
    if (sem_post(sem_id) < 0)
        printf("Child : [sem_post] Failed \n");
}
int main(int argc, char *argv[])
{
    pid_t pid;
    pid = fork();
    if (pid < 0){
        perror("fork"); exit(EXIT_FAILURE);
    }


    if (!pid){
        child();
        printf("Child : Done with sem_open \n");
    }
```

Share-var.c

```c
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
#include<unistd.h>
void *fun1();
 void *fun2();
 int shared=1; //shared variable
 sem_t s; //semaphore variable
 int main()
 {
 sem_init(&s,0,1); //initialize semaphore variable - 1st argument is address of variable, 2nd is number of processes sharing semaphore, 3rd argument is the initial value of semaphore variable
 pthread_t thread1, thread2;
 pthread_create(&thread1, NULL, fun1, NULL);
 pthread_create(&thread2, NULL, fun2, NULL);
 pthread_join(thread1, NULL);
 pthread_join(thread2,NULL);
 printf("Final value of shared is %d\n",shared); //prints the last updated value of shared variable
 }
void *fun1()
{
    int x;
    sem_wait(&s);
    x=shared;
    printf("Thread1 reads the value as %d\n",x);
    x++;  //thread1 increments its value
    printf("Local updation by Thread1: %d\n",x);
    sleep(1); //thread1 is preempted by thread 2
    shared=x; //thread one updates the value of shared variable
    printf("Value of shared variable updated by Thread1 is: %d\n",shared);
    sem_post(&s);
}
void *fun2()
{
    int y;
    sem_wait(&s);
    y=shared;//thread2 reads value of shared variable
    printf("Thread2 reads the value as %d\n",y);
    y--;  //thread2 increments its value
    printf("Local updation by Thread2: %d\n",y);
    sleep(1);
    shared=y;
    printf("Value of shared variable updated by Thread2 is: %d\n",shared);
    sem_post(&s);
}
```

student@student-HA1: ~

File Edit View Search Terminal Help

```
student@student-HA1:~$ gcc -o obj -pthread share_var.c
student@student-HA1:~$ ./obj
Thread2 reads the value as 1
Local updation by Thread2: 0
Value of shared variable updated by Thread2 is: 0
Thread1 reads the value as 0
Local updation by Thread1: 1
Value of shared variable updated by Thread1 is: 1
Final value of shared is 1
student@student-HA1:~$
```

FirstProgram.c

first.c

```c
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>
#include <semaphore.h>

void handler ( void *ptr );
sem_t mutex;
int counter;
int main() {
int i[2] = {0, 1};
pthread_t thread_a, thread_b;
counter = 0;
sem_init(&mutex, 0, 1);
pthread_create(&thread_a, 0, (void *) &handler, (void *) &i[0]);
pthread_create(&thread_b, 0, (void *) &handler, (void *) &i[1]);
pthread_join(thread_a,NULL);
pthread_join(thread_b,NULL);
sem_destroy(&mutex);
return 0;
}
void handler (void *ptr) {
int x = *((int*)ptr);
printf("sem [INFO] Thread %d Waiting to enter in critical region. \n", x);
sem_wait(&mutex);
printf("sem [INFO] Thread %d Enters in Critical Region. \n", x);
printf("sem [INFO] Thread %d Value of Counter is %d.\n",x,counter);
printf("sem [INFO] Thread %d Increamenting The Value of counter\n",x);
counter++;
printf("sem [INFO] Thread %d New value of counter is: %d\n",x, counter);
printf("sem [INFO] Thread %d Exiting Critical Region.\n", x);
sem_post(&mutex);
pthread_exit(0);
}
```

C Tab Width: 8 Ln 1, Col 1 INS

student@student-HA1: ~

File Edit View Search Terminal Help

```
student@student-HA1:~$ gcc -o obj -pthread first.c
student@student-HA1:~$ ./obj
sem [INFO] Thread 0 Waiting to enter in critical region.
sem [INFO] Thread 0 Enters in Critical Region.
sem [INFO] Thread 0 Value of Counter is 0.
sem [INFO] Thread 0 Increamenting The Value of counter
sem [INFO] Thread 0 New value of counter is: 1
sem [INFO] Thread 0 Exiting Critical Region.
sem [INFO] Thread 1 Waiting to enter in critical region.
sem [INFO] Thread 1 Enters in Critical Region.
sem [INFO] Thread 1 Value of Counter is 1.
sem [INFO] Thread 1 Increamenting The Value of counter
sem [INFO] Thread 1 New value of counter is: 2
sem [INFO] Thread 1 Exiting Critical Region.
student@student-HA1:~$ gedit first.c
```

Sleeping Barber.c

```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_CUSTOMERS 25
void *customer(void *num);
void *barber(void *);
sem_t waitingRoom;
sem_t barberChair;
sem_t barberPillow;
sem_t seatBelt;
int allDone = 0;
int main(int argc, char *argv[])
{
pthread_t btid;
pthread_t tid[MAX_CUSTOMERS];
int i, x, numCustomers, numChairs; int Number[MAX_CUSTOMERS];
printf("Maximum number of customers can only be 25. Enter number of customers and chairs.\n");
scanf("%d",&x);
numCustomers = x;
scanf("%d",&x);
numChairs = x;
if (numCustomers > MAX_CUSTOMERS) {
printf("The maximum number of Customers is %d.\n", MAX_CUSTOMERS);
return 0;
}
printf("A solution to the sleeping barber problem using semaphores.\n");
for (i = 0; i < MAX_CUSTOMERS; i++) {
Number[i] = i;
}

sem_init(&waitingRoom, 0, numChairs);
sem_init(&barberChair, 0, 1);
sem_init(&barberPillow, 0, 0);
sem_init(&seatBelt, 0, 0);
```

```
pthread_create(&btid, NULL, barber, NULL);

for (i = 0; i < numCustomers; i++) {
pthread_create(&tid[i], NULL, customer, (void *)&Number[i]);
}

for (i = 0; i < numCustomers; i++) {
pthread_join(tid[i],NULL);
}

allDone = 1;
sem_post(&barberPillow);
pthread_join(btid,NULL);
return 0;
}
void *customer(void *number) {
int num = *(int *)number;
printf("Customer %d leaving for barber shop.\n", num);
sleep(5);
printf("Customer %d arrived at barber shop.\n", num);
sem_wait(&waitingRoom);
printf("Customer %d entering waiting room.\n", num);
sem_wait(&barberChair);
sem_post(&waitingRoom);
printf("Customer %d waking the barber.\n", num);
sem_post(&barberPillow);
sem_wait(&seatBelt);
sem_post(&barberChair);
printf("Customer %d leaving barber shop.\n", num);
}
void *barber(void *junk)
{

while (!allDone) {
printf("The barber is sleeping\n");
sem_wait(&barberPillow);
if (!allDone)
{
printf("The barber is cutting hair\n");
sleep(3);
printf("The barber has finished cutting hair.\n");
sem_post(&seatBelt);
} else {
printf("The barber is going home for the day.\n");}}}
```

```
student@student-HA1:~$ ./obj
Maximum number of customers can only be 25. Enter number of customers and chairs.
3
6
A solution to the sleeping barber problem using semaphores.
The barber is sleeping
Customer 0 leaving for barber shop.
Customer 1 leaving for barber shop.
Customer 2 leaving for barber shop.
Customer 2 arrived at barber shop.
Customer 2 entering waiting room.
Customer 2 waking the barber.
Customer 1 arrived at barber shop.
Customer 1 entering waiting room.
The barber is cutting hair
Customer 0 arrived at barber shop.
Customer 0 entering waiting room.
The barber has finished cutting hair.
Customer 2 leaving barber shop.
Customer 1 waking the barber.
The barber is sleeping
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 1 leaving barber shop.
Customer 0 waking the barber.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 0 leaving barber shop.
The barber is going home for the day.
```

Second.c

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <semaphore.h>

#define THREAD_NUM 4

sem_t semaphore;

void* routine(void* args) {
    sem_wait(&semaphore);
    sleep(1);
    printf("Hello from thread %d\n", *(int*)args);
    sem_post(&semaphore);
    free(args);
}

int main(int argc, char *argv[]) {
    pthread_t th[THREAD_NUM];
    sem_init(&semaphore, 0, 2);
    int i;
    for (i = 0; i < THREAD_NUM; i++) {
        int* a = malloc(sizeof(int));
        *a = i;
        if (pthread_create(&th[i], NULL, &routine, a) != 0) {
            perror("Failed to create thread");
        }
    }

    for (i = 0; i < THREAD_NUM; i++) {
        if (pthread_join(th[i], NULL) != 0) {
            perror("Failed to join thread");
        }
    }
    sem_destroy(&semaphore);
    return 0;
}
```

```
student@student-HA1: ~
File  Edit  View  Search  Terminal  Help
student@student-HA1:~$ gedit second.c
student@student-HA1:~$ gcc -o obj -pthread second.c
student@student-HA1:~$ ./obj
Hello from thread 1
Hello from thread 0
Hello from thread 2
Hello from thread 3
student@student-HA1:~$ gedit second.c
```

Task1 (a)

```
ayaz@ayaz-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
ayaz@ayaz-VirtualBox:~$ gedit task1_a.c
ayaz@ayaz-VirtualBox:~$ gcc -o obj -pthread task1_a.c
ayaz@ayaz-VirtualBox:~$ ./obj

Person 1 is eating icecream

Person 3 is eating icecream

Person 2 is eating icecream
ayaz@ayaz-VirtualBox:~$
```

Open Save

```c
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<semaphore.h>

sem_t icecreamremaining;

void*thread_1(void*arg)
{

        sem_wait(&icecreamremaining);
        printf("\nPerson 1 is eating icecream\n");
        sleep(1);
        sem_post(&icecreamremaining);
}
void*thread_2(void*arg)
{

        sem_wait(&icecreamremaining);
        printf("\nPerson 2 is eating icecream\n");
        sleep(1);
        sem_post(&icecreamremaining);
}
void*thread_3(void*arg)
{
        sem_wait(&icecreamremaining);
        printf("\nPerson 3 is eating icecream\n");
        sleep(1);
        sem_post(&icecreamremaining);
```

C Tab Width: 8 Ln 27, Col 38 INS

```c
int main()
{
        sem_init(&icecreamremaining,0,1);
        pthread_t icecream;
        pthread_create(&icecream,NULL,thread_1,NULL);
        pthread_create(&icecream,NULL,thread_2,NULL);
        pthread_create(&icecream,NULL,thread_3,NULL);
        pthread_join(icecream,NULL);
}
```

Task 1_(b)

```
ayaz@ayaz-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
ayaz@ayaz-VirtualBox:~$ gcc -o obj1 -pthread task1_b.c
ayaz@ayaz-VirtualBox:~$ ./obj1

Person 1 is counting money for  icecream

Person 1 is eating icecream

Person 2 is counting money for icecream

Person 2 is eating icecream

Person 3 is counting money for icecream

Person 3 is eating icecream
ayaz@ayaz-VirtualBox:~$
```

Open Save

```c
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<semaphore.h>

sem_t icecreamremaining;

void*thread_1(void*arg)
{
        sem_wait(&icecreamremaining);
        printf("\nPerson 1 is counting money for  icecream\n");
        sleep(2);
        printf("\nPerson 1 is eating icecream\n");
        sem_post(&icecreamremaining);
}
void*thread_2(void*arg)
{
        sem_wait(&icecreamremaining);
        printf("\nPerson 2 is counting money for icecream\n");
        sleep(2);
        printf("\nPerson 2 is eating icecream\n");
        sem_post(&icecreamremaining);
}
void*thread_3(void*arg)
{
        sem_wait(&icecreamremaining);
        printf("\nPerson 3 is counting money for icecream\n");
        sleep(2);
        printf("\nPerson 3 is eating icecream\n");
```

C Tab Width: 8 Ln 38, Col 20 INS

```c
void*thread_3(void*arg)
{
        sem_wait(&icecreamremaining);
        printf("\nPerson 3 is counting money for icecream\n");
        sleep(2);
        printf("\nPerson 3 is eating icecream\n");
        sem_post(&icecreamremaining);
}

int main()
{
        sem_init(&icecreamremaining,0,1);
        pthread_t icecream;
        pthread_create(&icecream,NULL,thread_1,NULL);
        pthread_create(&icecream,NULL,thread_2,NULL);
        pthread_create(&icecream,NULL,thread_3,NULL);
        pthread_join(icecream,NULL);
}
```

TASK 2

```
ayaz@ayaz-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
Person 4  luggage is getting weighted

Person 4  luggage is getting checked for security

Person 4 is getting the boarding pass

Person 3  luggage is getting weighted

Person 3  luggage is getting checked for security

Person 3 is getting the boarding pass
====================================
Person 9  luggage is getting weighted

Person 9  luggage is getting checked for security

Person 9 is getting the boarding pass

Person 10  luggage is getting weighted

Person 10  luggage is getting checked for security

Person 10 is getting the boarding pass
ayaz@ayaz-VirtualBox:~$
```

Open Save

```c
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#include<semaphore.h>

sem_t s;

void*thread_1(void*arg)
{

sem_wait(&s);
printf("\nPerson 1  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 1  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 1 is getting the boarding pass\n");
sleep(3);
sem_post(&s);
}
void*thread_2(void*arg)
{

sem_wait(&s);
printf("\nPerson 2  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 2  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 2 is getting the boarding pass\n");
sem_post(&s);
```

C Tab Width: 8 Ln 147, Col 2 INS

Open Save

```c
void*thread_2(void*arg)
{

sem_wait(&s);
printf("\nPerson 2  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 2  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 2 is getting the boarding pass\n");
sem_post(&s);
}
void*thread_3(void*arg)
{

sem_wait(&s);
printf("\nPerson 3  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 3  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 3 is getting the boarding pass\n");
sleep(3);
printf("===================================");
sem_post(&s);
}
void*thread_4(void*arg)
{

sem_wait(&s);
printf("\nPerson 4  luggage is getting weighted\n");
sleep(4);
```

C Tab Width: 8 Ln 144, Col 2 INS

Open Save

```c
void*thread_5(void*arg)
{

sem_wait(&s);
printf("\nPerson 5  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 5  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 5 is getting the boarding pass\n");
sleep(3);
printf("====================================");
sem_post(&s);
}
void*thread_6(void*arg)
{

sem_wait(&s);
printf("\nPerson 6 luggage is getting weighted\n");
sleep(4);
printf("\nPerson 6  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 6 is getting the boarding pass\n");
sleep(3);
printf("====================================");
sem_post(&s);
}
void*thread_7(void*arg)
{

sem_wait(&s);
```

C Tab Width: 8 Ln 144, Col 2 INS

Open Save

```
sleep(3);
printf("==================================");
sem_post(&s);
}
void*thread_7(void*arg)
{

sem_wait(&s);
printf("\nPerson 7  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 7  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 7 is getting the boarding pass\n");
sleep(3);
sem_post(&s);
}
void*thread_8(void*arg)
{

sem_wait(&s);
printf("\nPerson 8  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 8  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 8 is getting the boarding pass\n");
sleep(3);
sem_post(&s);
}
void*thread_9(void*arg)
{
```

C Tab Width: 8 Ln 144, Col 2 INS

```
}
void*thread_10(void*arg)
{

sem_wait(&s);
printf("\nPerson 10  luggage is getting weighted\n");
sleep(4);
printf("\nPerson 10  luggage is getting checked for security \n");
sleep(7);
printf("\nPerson 10 is getting the boarding pass\n");
sleep(3);
sem_post(&s);
}

int main()
{
sem_init(&s,0,1);
pthread_t customer;
pthread_create(&customer,NULL,thread_1,NULL);
pthread_create(&customer,NULL,thread_2,NULL);
pthread_create(&customer,NULL,thread_3,NULL);
pthread_create(&customer,NULL,thread_4,NULL);
pthread_create(&customer,NULL,thread_5,NULL);
pthread_create(&customer,NULL,thread_6,NULL);
pthread_create(&customer,NULL,thread_7,NULL);
pthread_create(&customer,NULL,thread_8,NULL);
pthread_create(&customer,NULL,thread_9,NULL);
pthread_create(&customer,NULL,thread_10,NULL);
pthread_join(customer,NULL);
}
```