10. Simulate sleeping barber problem.

**AIM:**

Implementation of sleeping barber.

**DESCRIPTION:**

The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair.

1. If there is no customer, then the barber sleeps in his own chair.

2. When a customer arrives, he has to wake up the barber.

3. If there are many customers and the barber is cutting a customer’s hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

**ALGORITHM:**

**Step 1:** start

**Step 2:** include stdio.h, unistd.h, stdlib.h, pthread.h, semaphore.h

**Step 3:** // The maximum number of customer threads.

Declare MAX\_CUSTOMERS 25 , REENTRANT (use pre processor directives)

**Step 4:**  // Function prototypes... void \*customer(void \*num);

void \*barber(void \*);

void randwait(int secs);

// Define the semaphores.

// waitingRoom Limits the # of customers allowed

// to enter the waiting room at one time.

sem\_t waitingRoom;

// barberChair ensures mutually exclusive access to

// the barber chair.

sem\_t barberChair;

// barberPillow is used to allow the barber to sleep

// until a customer arrives.

sem\_t barberPillow;

// seatBelt is used to make the customer to wait until

// the barber is done cutting his/her hair.

sem\_t seatBelt;

// Flag to stop the barber thread when all customers

// have been serviced.

int allDone = 0;

**Step 5:** int main(int argc, char \*argv[])

begin

pthread\_t btid; pthread\_t tid[MAX\_CUSTOMERS];

declare RandSeed as long

declare i, numCustomers, numChairs as integer

int Number[MAX\_CUSTOMERS];

// Check to make sure there are the right number of command line arguments.

if (argc != 4)

begin

write"Use: SleepBarber <Num Customers> <Num Chairs> <rand seed>\n"

exit(-1);

end if

// Get the command line arguments and convert them into integers.

numCustomers <- atoi(argv[1]);

numChairs <- atoi(argv[2]);

RandSeed <- atol(argv[3]);

// Make sure the number of threads is less than the number of customers we can support.

if (numCustomers > MAX\_CUSTOMERS)

begin

write"The maximum number of Customers is %d.\n", MAX\_CUSTOMERS

exit(-1);

end if

write"\nSleepBarber.c\n\n"

write"A solution to the sleeping barber problem using semaphores.\n"

// Initialize the random number generator with a new seed.

srand(RandSeed);

// Initialize the numbers array.

for (i=0; i<MAX\_CUSTOMERS; i++)

begin

Number[i] <- i;

End for

// Initialize the semaphores with initial values...

sem\_init(&waitingRoom, 0, numChairs);

sem\_init(&barberChair, 0, 1);

sem\_init(&barberPillow, 0, 0);

sem\_init(&seatBelt, 0, 0);

// Create the barber.

pthread\_create(&btid, NULL, barber, NULL);

// Create the customers.

for (i=0; i<numCustomers; i++)

begin

pthread\_create(&tid[i], NULL, customer, (void \*)&Number[i]);

end for

// Join each of the threads to wait for them to finish.

for (i=0; i<numCustomers; i++)

begin

pthread\_join(tid[i],NULL);

end for

// When all of the customers are finished, kill the barber thread.

allDone <- 1;

sem\_post(&barberPillow);

// Wake the barber so he will exit.

pthread\_join(btid,NULL);

end

**Step 6:** void \*customer(void \*number)

begin

int num = \*(int \*)number

//Leave for the shop and take some random amount of time to arrive.

write"Customer %d leaving for barber shop.\n", num

randwait(5);

write"Customer %d arrived at barber shop.\n", num

// Wait for space to open up in the waiting room...

sem\_wait(&waitingRoom);

write"Customer %d entering waiting room.\n", num

// Wait for the barber chair to become free.

sem\_wait(&barberChair);

// The chair is free so give up your spot in the waiting room.

sem\_post(&waitingRoom);

// Wake up the barber...

write"\n Customer %d waking the barber.\n", num

sem\_post(&barberPillow);

// Wait for the barber to finish cutting your hair.

sem\_wait(&seatBelt);

// Give up the chair.

sem\_post(&barberChair);

write"\n Customer %d leaving barber shop.\n", num

end

**Step 7:** void \*barber(void \*junk)

begin

// While there are still customers to be serviced... Our barber is omnicient and can tell if there are customers still on the way to his shop.

while (!allDone)

begin

// Sleep until someone arrives and wakes you..

write"The barber is sleeping\n"

sem\_wait(&barberPillow);

// Skip this stuff at the end...

if (!allDone)

begin

// Take a random amount of time to cut the customer's hair.

write"The barber is cutting hair\n"

randwait(3);

write"The barber has finished cutting hair.\n"

// Release the customer when done cutting...

sem\_post(&seatBelt);

end if

else

begin

write"The barber is going home for the day.\n"

end else

end if

end

**Step 8:** void randwait(int secs)

begin

int len;

// Generate a random number...

len = (int) ((rand() \* secs) + 1);

sleep(len);

end

**Step 9:** stop

**SOURCE CODE:**

**#include <stdio.h>**

**#include <unistd.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <pthread.h>**

**#include <semaphore.h>**

**// The maximum number of customer threads.**

**#define MAX\_CUSTOMERS 25**

**// Function prototypes…**

**void \*customer(void \*num);**

**void \*barber(void \*);**

**void randwait(int secs);**

**// Define the semaphores.**

**// waitingRoom Limits the # of customers allowed**

**// to enter the waiting room at one time.**

**sem\_t waitingRoom;**

**// barberChair ensures mutually exclusive access to**

**// the barber chair.**

**sem\_t barberChair;**

**// barberPillow is used to allow the barber to sleep**

**// until a customer arrives.**

**sem\_t barberPillow;**

**// seatBelt is used to make the customer to wait until**

**// the barber is done cutting his/her hair.**

**sem\_t seatBelt;**

**// Flag to stop the barber thread when all customers**

**// have been serviced.**

**int allDone = 0;**

**int main(int argc, char \*argv[]) {**

**pthread\_t btid;**

**pthread\_t tid[MAX\_CUSTOMERS];**

**long RandSeed;**

**int i, numCustomers, numChairs;**

**int Number[MAX\_CUSTOMERS];**

**printf("Enter the number of Custmors : ");**

**scanf("%d",&numCustomers) ;**

**printf("Enter the number of Charis : ");**

**scanf("%d",&numChairs);**

**// Make sure the number of threads is less than the number of**

**// customers we can support.**

**if (numCustomers > MAX\_CUSTOMERS) {**

**printf("The maximum number of Customers is %d.\n", MAX\_CUSTOMERS);**

**exit(-1);**

**}**

**// Initialize the numbers array.**

**for (i=0; i<MAX\_CUSTOMERS; i++) {**

**Number[i] = i;**

**}**

**// Initialize the semaphores with initial values…**

**sem\_init(&waitingRoom, 0, numChairs);**

**sem\_init(&barberChair, 0, 1);**

**sem\_init(&barberPillow, 0, 0);**

**sem\_init(&seatBelt, 0, 0);**

**// Create the barber.**

**pthread\_create(&btid, NULL, barber, NULL);**

**// Create the customers.**

**for (i=0; i<numCustomers; i++) {**

**pthread\_create(&tid[i], NULL, customer, (void \*)&Number[i]);**

**sleep(1);**

**}**

**// Join each of the threads to wait for them to finish.**

**for (i=0; i<numCustomers; i++) {**

**pthread\_join(tid[i],NULL);**

**sleep(1);**

**}**

**// When all of the customers are finished, kill the**

**// barber thread.**

**allDone = 1;**

**sem\_post(&barberPillow); // Wake the barber so he will exit.**

**pthread\_join(btid,NULL);**

**}**

**void \*customer(void \*number) {**

**int num = \*(int \*)number;**

**// Leave for the shop and take some random amount of**

**// time to arrive.**

**printf("Customer %d leaving for barber shop.\n", num);**

**randwait(2);**

**printf("Customer %d arrived at barber shop.\n", num);**

**// Wait for space to open up in the waiting room…**

**sem\_wait(&waitingRoom);**

**printf("Customer %d entering waiting room.\n", num);**

**// Wait for the barber chair to become free.**

**sem\_wait(&barberChair);**

**// The chair is free so give up your spot in the**

**// waiting room.**

**sem\_post(&waitingRoom);**

**// Wake up the barber…**

**printf("Customer %d waking the barber.\n", num);**

**sem\_post(&barberPillow);**

**// Wait for the barber to finish cutting your hair.**

**sem\_wait(&seatBelt);**

**// Give up the chair.**

**sem\_post(&barberChair);**

**printf("Customer %d leaving barber shop.\n", num);**

**}**

**void \*barber(void \*junk) {**

**// While there are still customers to be serviced…**

**// Our barber is omnicient and can tell if there are**

**// customers still on the way to his shop.**

**while (!allDone) {**

**// Sleep until someone arrives and wakes you..**

**printf("The barber is sleeping\n");**

**sem\_wait(&barberPillow);**

**// Skip this stuff at the end…**

**if (!allDone) {**

**// Take a random amount of time to cut the**

**// customer’s hair.**

**printf("The barber is cutting hair\n");**

**randwait(2);**

**printf("The barber has finished cutting hair.\n");**

**// Release the customer when done cutting…**

**sem\_post(&seatBelt);**

**}**

**else {**

**printf("The barber is going home for the day.\n");**

**}**

**}**

**}**

**void randwait(int secs) {**

**int len;**

**// Generate a random number…**

**len = (int) ((1 \* secs) + 1);**

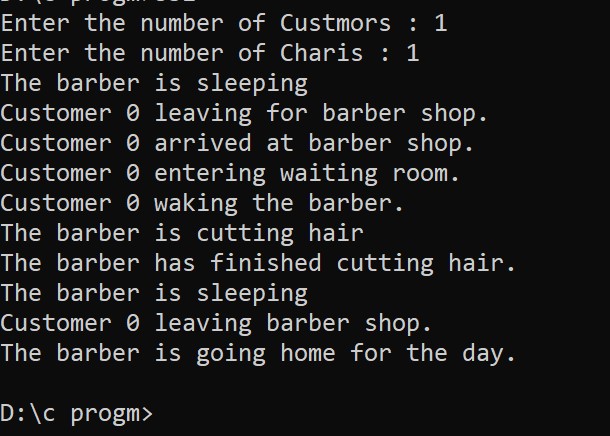
**sleep(len);**

**}**

**OUTPUT:**

**$gcc sb.c –o sb**

**$sb**

****