Project 5

**Name**: Gao Ning

**ID**: 288197

**Supervisor**: Figat Maksym

1. Task description:

The analogy is based upon a hypothetical barber shop with many barbers serving women and man.

N1: barbers serving only for women

N2: barbers serving only for men

N3: barbers serving for both women and men

M: the number of chairs in waiting room

1. Implementation

Pseudocode:

Semaphore Customers = 0;

Semaphore Barber = 0;

Mutex = 1;

int Clients = N;

int Chairs;

Barber {

      while(true) {

/\* for one case barbers serving for male\*/

if Clients > 0

         Clients--;

else {

/\*barbers go to sleep\*/

Unlock(mutex,0);

Lock (barber Semaphore, barber ID);

}

/\* Client in the queue display \*/

Unlock (mutex, 0);

Sleep (cutting time);

      }

}

Clients {

      while(true) {

          lock (mutex, 0);

// generate new client

Clients++;

/\* here just one case we ca generate clients for male or female \*/

If Clients number more than Chairs

/\* new client leave \*/

If Clients more than 0 and barbers more than 1

/\* wake up barber \*/

Unlock (barber semaphore, index);

Unlock (mutex, 0);

Sleep (wait time);

      }

}

A screenshot of text

Description automatically generated

**void** barberWorkDistrbution (**int** i, **int** barbers);

/\*

The function goes to distrbute the barbers by index number, like we have 9 clients, so I decided 0, 2, 4 … even number for female,

1, 3, 5 … odd number for male and each condition will call the barber function, barber function will get the barber type, the number of all barbers.

\*/

**void** barber (**int** barberType, **int** allBarbers, **int** barberID);

/\*

According the barber type there will be three case for male barbers, female barbers, and barbers serving for both male and female. The algorithm as before pseudocode.

\*/

**void** client (**int** allBarbers, **int** barberIndex);

/\*

Client process will do some job to wake the barbers and generate new clients (male clients or female clients), it will generate randomly.

\*/

**void** terminateProcess (**int** CreatedProcess,pid\_t\*processList);

**void** lock (**int** semid, **int** idx);

A screen shot of a social media post

Description automatically generated

**void** unlock (**int** semid, **int** idx);

A screenshot of a cell phone

Description automatically generated

The semaphore set functions are:

[semctl()](https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/apis/ipcsemct.htm) (Perform Semaphore Control Operations) provides semaphore control operations as specified by cmd on the semaphore specified by semnum in the semaphore set specified by semid.

[semget ()](https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/apis/ipcsemgt.htm)(Get Semaphore Set with Key) returns the semaphore ID associated with the specified semaphore key.

[semop()](https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/apis/ipcsemop.htm) (Perform Semaphore Operations on Semaphore Set) performs operations on semaphores in a semaphore set. These operations are supplied in a user-defined array of operations.

Something about shared memory:

A close up of a sign

Description automatically generated

#include <sys/ipc.h>

#include <sys/shm.h>

int shmget(key\_t key, size\_t size, int shmflg)

#include <sys/types.h>

#include <sys/shm.h>

void \* shmat(int shmid, const void \*shmaddr, int shmflg)

1. Test

A close up of a logo

Description automatically generated

Output:

A close up of text on a black background

Description automatically generated

A close up of a logo

Description automatically generated

Output：

A close up of text on a black background

Description automatically generated

A picture containing holding

Description automatically generated

Output：

A close up of text on a black background

Description automatically generated