THE BARBER PROBLEM

PRESENTATION OF THE PROBLEM

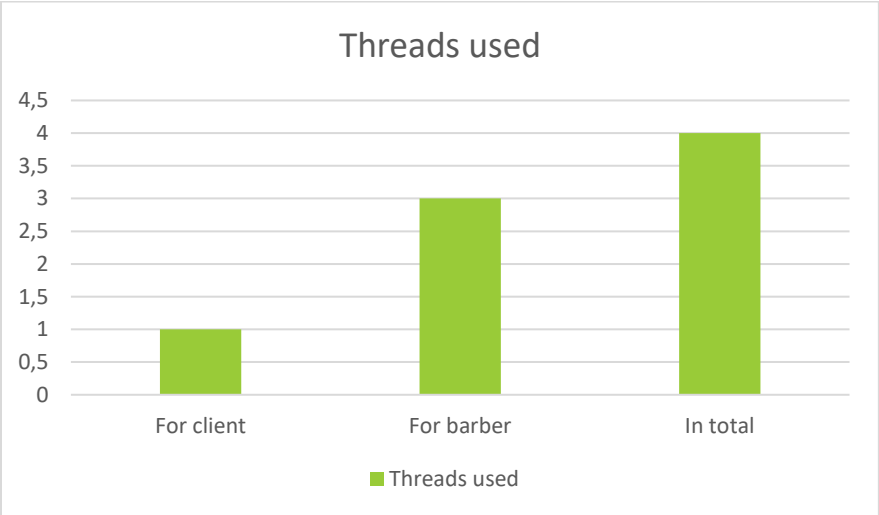
We had to create a program to emulate a barber shop. This shop has three different type of barber:

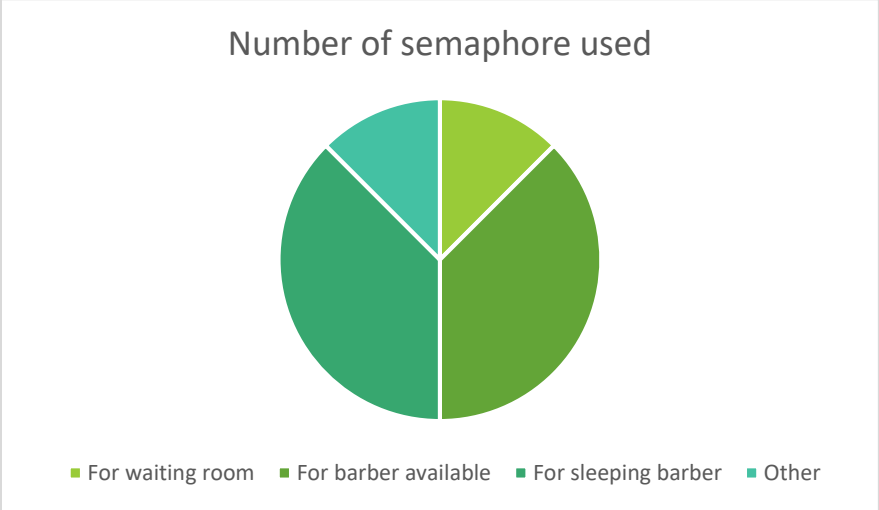
1. For man
2. For Woman
3. For both

Moreover, the shop has a waiting room (the number of seats is defined by the user) and if it is full, the client leaves the barber shop and go back home.
After a random time, the client arrives at the barber shop, enter the waiting room (if it is not full), check if a barber chair for him/her is free. If yes, the client wakes up the barber and get a haircut. After a certain amount of time, the barber finished his work, the client leaves the shop and the barber go back to sleep.
The end of the day come when all the clients where served and there is no more client in the waiting room.

RESOLUTION OF THE PROBLEM

To solve this problem, I used thread and semaphores to “do” actions in the same time.
I started with only on type of barber and one type of client. This first step let me work on the functionalities of the waiting room, the barber (sleeping, cutting hair, finishing his day etc…) and the client (arriving at the shop, waiting, waking up the barber, leaving etc…). When it was done, I work with a comrade to add a gender to the clients and to create the different kind of barber.





DESCRIPTION OF THE PROGRAM

DECLARATION OF VARIABLES AND FUCTIONS

```
11  #define NBRCLIENT 3     //Number of clients
12  #define NBRSEAT 1       //Number of seat in the waiting room
13  #define NBRBARBERM 1    //Number of barbers for man
14  #define NBRBARBERW 1    //Number of barbers for woman
15  #define NBRBARBERB 0    //Number of barbers for both
```

The first lines of the program declare how many clients, seats in the waiting room and barber of each type will be used.

```
17  //Declaration of functions
18  void *clientFunction(void *id);
19  void *barberFunctionM(void *id);
20  void *barberFunctionW(void *id);
21  void *barberFunctionB(void *id);
22
23  //Declaration global variable
24  sem_t seatWaitingRoom; //Define the umber of place in the waiting room  (NBRSEAT)
25
26  sem_t barberChairM; //Define the number of barber chair man
27  sem_t barberChairW; //Define the number of barber chair woman
28  sem_t barberChairB; //Define the number of barber chair both
29
30  sem_t barberAwakeM; //Define if the barber is awake (1==yes ; 0==no)
31  sem_t barberAwakeW; //Define if the barber is awake (1==yes ; 0==no)
32  sem_t barberAwakeB; //Define if the barber is awake (1==yes ; 0==no)
33
34  sem_t freeChair; //Use to make client wait until the barber finish is job
35
36  bool NMC = false; //NMC stand for No More Client, this variale is to end the program when all clients are served
```

The couple of lines above declares the program's functions and the different semaphores used to resolve this problem. As we can see, there is a function for the declaration of clients (the gender of the client is defines in it), one semaphore for the number of seats in the waiting room, three semaphores for each type of barber's chair and three for the state of the barber (awake or not). The last semaphore (freeChair) is used to "pause" the client thread when the barber is working. A boolean is declared and is used track when all clients are served, it means that the barbers have finished his day. Then we enter in the main().

MAIN() – PART 1

```
44  int main()
45  {
46      srand(time(NULL));
47
48      printf("Number of clients: %d\n", NBRCLIENT);
49      printf("Number of seats in the waiting room: %d\n\n", NBRSEAT);
50      printf("Number of barbers:\n\tMan: %d\n\tWomen: %d\n\tBoth: %d\n\n", NBRBARBERM, NBRBARBERW, NBRBARBERB);
51
52  //Declaration of variable
53      pthread_t barberThreadM[NBRBARBERM];
54      pthread_t barberThreadW[NBRBARBERW];
55      pthread_t barberThreadB[NBRBARBERB];
56
57      pthread_t clientThread[NBRCLIENT];
58
59      int idArrayClient[NBRCLIENT];
60
61      int idArrayBarberM[NBRBARBERM];
62      int idArrayBarberW[NBRBARBERW];
63      int idArrayBarberB[NBRBARBERB];
```

The fist line of the main (line n°46) initialize the random for next functionalities. Then we display how many clients, seat in the waiting room and different barber are used for the simulation. After that we declare all threads, more precisely array of threads needed. We declare also array of ID used to declare barbers and clients.

```
65  //Initialization of the arrays of ID
66      for (int i = 0 ; i < NBRCLIENT ; i++)
67      {
```

These line initialized arrays of ID.

```
86  //Initialization of the semaphores
87      sem_init(&seatWaitingRoom, 0, NBRSEAT);
88
89      sem_init(&barberChairM, 0, NBRBARBERM);
90      sem_init(&barberChairW, 0, NBRBARBERW);
91      sem_init(&barberChairB, 0, NBRBARBERB);
92
93      sem_init(&barberAwakeM, 0, 0);
94      sem_init(&barberAwakeW, 0, 0);
95      sem_init(&barberAwakeB, 0, 0);
96
97      sem_init(&freeChair, 0, 0);
98      }
99
81      for (int i = 0 ; i < NBRBARBERB ; i++)
82      {
83          idArrayBarberB[i] = i;
84      }
```

We declare initialized all semaphores. At the beginning of the program there are:

- NBRSEAT seats in the waiting room.
- NBRBARBERM who take care of man.
- NBRBARBERW who take care of man.
- NBRBARBERB who take care of man.

By default, all barbers are sleeping.

```
99  //Creation of one barbers
100     for (int i = 0 ; i < NBRBARBERM ; i++)
101     {
102         pthread_create(&barberThreadM[i], NULL, barberFunctionM, &idArrayBarberM[i]);
103     }
104
105     for (int i = 0 ; i < NBRBARBERW ; i++)
106     {
107         pthread_create(&barberThreadW[i], NULL, barberFunctionW, &idArrayBarberW[i]);
108     }
109
110     for (int i = 0 ; i < NBRBARBERB ; i++)
111     {
112         pthread_create(&barberThreadB[i], NULL, barberFunctionB, &idArrayBarberB[i]);
113     }
114
115     sleep(1);
116
117 //Creation of clients
118     for (int i = 0 ; i < NBRCLIENT ; i++)
119     {
120         pthread_create(&clientThread[i], NULL, clientFunction, &idArrayClient[i]);
121     }
```

After the initialisation of the different semaphore, we create all barbers and all clients. The program sleep (line 115) for one second to e sure that the display of information is done correctly.

The following picture show the output of this step:

```
Number of clients: 3
Number of seats in the waiting room: 1

Number of barbers:
        Man: 1
        Women: 1
        Both: 0

========== The barber [W0] is sleeping.==========
========== The barber [M0] is sleeping.==========
```

The next step is to join all the client threads to start them at the same time.

```
123   //Join all client threads
124       for (int i = 0 ; i < NBRCLIENT ; i++)
125       {
126           pthread_join(clientThread[i], NULL);
127       }
```

CLIENTFUNCTION() – PART 1

The beginning of the function declares the ID, the gender ("attribute" is a global variable which can take values 0 and 1) of the client. The variable "freeSeat" store the number of seat available in the waiting room. Variables "freeChair*" store the number of barber chairs available for each type of barber. And the boolean "BB" is true if the barber's type is both, false if not.

```
155   void *clientFunction(void *id)
156   {
157       int ID = *(int *) id;
158       int freeSeats;
159       char sex;
160
161       int freeChairM;
162       int freeChairW;
163       int freeChairB;
164
165       bool BB;
166
167   //Define sex of the client
168       if (attribute == 0)
169       {
170           attribute = 1;
171           sex = 'W';
172       }
173       else
174       {
175           attribute = 0;
176           sex = 'M';
177       }
```

Then the client leaves his house and arrive at the barber shop randomly between 0 and 3 seconds. To be realistic, we take 1 second equal to 5 minutes.

```
179   //Client leave home and go to the barber shop
180       printf("Client [%c%d] leaving home.\n", sex, ID);
181       sleep(rand()%4);
182       printf("Client [%c%d] enter the barber shop.\n", sex, ID);
```

```
184  //Can the client seat in the waiting room ?
185      sem_getvalue(&seatWaitingRoom, &freeSeats);
186
187      if (freeSeats != 0)
188      {
189      //Client is in the waiting room
190          sem_wait(&seatWaitingRoom);
191          printf("Client [%c%d] is in the waiting room.\n", sex, ID);
192
193          if (sex == 'M') //Client is a Man
194          {
195              do
196              {
197                  sem_getvalue(&barberChairM, &freeChairM);
198                  sem_getvalue(&barberChairB, &freeChairB);
199
200                  if(freeChairM != 0)
201                  {
202                      BB = false; //Remember that barber is not both
203
204                  //Client wait for an available barber chair
205                      sem_wait(&barberChairM);
206
207                  //Free a seat in the waiting room
208                      sem_post(&seatWaitingRoom);
209
210                  //Barber is sleeping so client waking him up
211                      printf("Client [%c%d] wake up the barber.\n", sex, ID);
212                      sem_post(&barberAwakeM);
213                  }
214                  else if (freeChairB != 0)
215                  {
216                      BB = true; //Remember that barber is both
217
218                  //Client wait for an available barber chair
219                      sem_wait(&barberChairB);
220
221                  //Free a seat in the waiting room
222                      sem_post(&seatWaitingRoom);
223
224                  //Barber is sleeping so client waking him up
225                      printf("Client [%c%d] wake up the barber.\n", sex, ID);
226                      sem_post(&barberAwakeB);
227                  }
228              }while(freeChairM == 0 && freeChairB == 0);
229          }
230          else //Client is a Woman
```

When the client arrives at the barber shop, he checks if he can seat in the waiting room (“freeSeat =! 0” mean that minimum one seat is available).
Then if the client is a man we check if there are free chair for male and both. We prioritize the barber type corresponding to the gender the client.
If a free chair for a barber chair male, we set the variable “BB” to false and we decrease the semaphore “baberChairM” by one. We increase the semaphore “seatWaitingRoom” by one because the client is no more in the waiting room. Then we wake up the barber.
You can see that it is the same thing if the barber’s type is both except that we set the “BB” variable to true. It is also the same code for woman, we just have to change all the functions for man by functions for woman.

When the client wakes up the barber, we execute the code of the barberFunction().

If there is no seat free in the waiting room the client goes back home.

```
289       else
290       {
291           printf("No more seat available in the waiting. The client [%c%d] go home.\n", sex, ID);
292       }
293   }
```

BARBER FUNCTIONS

```
299   void *barberFunctionM(void *id)
300   {
301       int ID = *(int *) id;
302       char typeClient = 'M';
303
304       while (!NMC)
305       {
306           printf("========== The barber [%c%d] is sleeping.==========\n", typeClient, ID);
307           sem_wait(&barberAwakeM);
308
309           if (!NMC)
310           {
311           //Barber take care of the client
312               printf("Barber [%c%d] is cutting hair...\n", typeClient, ID);
313               sleep(rand()%7);
314               printf("Barber [%c%d] has finished is job.\n", typeClient, ID);
315
316           //Release the client
317               sem_post(&freeChair);
318           }
319           else
320           {
321               printf("End of the day.\nThe barber [%c%d] wake up and go home.\n", typeClient, ID);
322               sem_post(&barberAwakeM);
323           }
324       }
325   }
```

The first two lines set the ID and the type of the barber (here the barber is for men). The line 306 is display when the barber threads are created and after we wait until the barer is waking up by a client (“barberAwakeM” has value 1). When the value of “barberAwakeM” is equal to one, the line 307 re-set the value to zero and the program continue after this line. If all clients are not served, the barber start to cut the hair of the client and finish after a random time between 0 and 6 seconds (in real life: 0 and 30 minutes).When the barber has finished his job, the client free the chair (line 317) and the program return in the client function.
If there are no more client (“NMC == true”) the barber goes home, and it is the end of the simulation.

Of course, the functions barberFunctionW() and barberFunctionW() are identical, we have just to adapt the functions according to the type of barber.

CLIENTFUNCTION() – PART 2

```
268       //Wait the end of the barber's work
269           sem_wait(&freeChair);
270
271       //Free the barber chair
272           if (BB == true)
273           {
274               sem_post(&barberChairB); //Free a chair for both
275           }
276           else
277           {
278               if (sex = 'M')
279               {
280                   sem_post(&barberChairM); //Free a chair for Man
281               }
282               if (sex = 'W')
283               {
284                   sem_post(&barberChairW); //Free a chair for Woman
285               }
286           }
287           printf("Customer [%c%d] leaving barber shop.\n", sex, ID);
288       }
```

When the barber has done his work, the line 269 set the semaphore “freeChair” to 0 and re-start the function after this line.
In this part we have to free a barber chair corresponding to the type of barber who have cut the air (we use here the variable “BB”).
Then the client goes home.

MAIN() – PART 2

When there is no more client to serve (“NMC = 1”), we wake up all barbers and they finish their day.

```
129   //When all the client are served
130       printf("========================================\n");
131       NMC = 1;
132       for (int i = 0 ; i < NBRBARBERM ; i++)
133       {
134           sem_post(&barberAwakeM);
135           pthread_join(barberThreadM[i], NULL);
136       }
137
138       for (int i = 0 ; i < NBRBARBERW ; i++)
139       {
140           sem_post(&barberAwakeW);
141           pthread_join(barberThreadW[i], NULL);
142       }
143
144       for (int i = 0 ; i < NBRBARBERB ; i++)
145       {
146           sem_post(&barberAwakeB);
147           pthread_join(barberThreadB[i], NULL);
148       }
149   }
```

EXAMPLE ONE SIMULATION

```
Number of clients: 3
Number of seats in the waiting room: 1

Number of barbers:
        Man: 1
        Women: 1
        Both: 0

========== The barber [W0] is sleeping.==========
========== The barber [M0] is sleeping.==========
Client [W2] leaving home.
Client [M1] leaving home.
Client [W0] leaving home.
Client [W2] enter the barber shop.
Client [W2] is in the waiting room.
Client [W2] wake up the barber.
Barber [W0] is cutting hair...
Client [M1] enter the barber shop.
Client [M1] is in the waiting room.
Client [M1] wake up the barber.
Barber [M0] is cutting hair...
Client [W0] enter the barber shop.
Client [W0] is in the waiting room.
Barber [M0] has finished is job.
========== The barber [M0] is sleeping.==========
Customer [W2] leaving barber shop.
Client [W0] wake up the barber.
Barber [W0] has finished is job.
========== The barber [W0] is sleeping.==========
Barber [W0] is cutting hair...
Customer [W1] leaving barber shop.
Barber [W0] has finished is job.
========== The barber [W0] is sleeping.==========
Customer [W0] leaving barber shop.
========================================
End of the day.
The barber [M0] wake up and go home.
End of the day.
The barber [W0] wake up and go home.
```

TEST WITH DIFFERENT PARAMETERS, TIME TO CUT, TIME ARRIVE AT SHOP, ETC …

Number of clients	Number of seats in the waiting room	Number of barbers			Arrival time < haircut time ?	Result
		Man	Woman	Both		
3	1	1	1	0	Yes	Everything is OK
3	1	1	1	0	No	Everything is OK
5	1	1	1	0	No	2 clients left due to the lack of place in the waiting room
5	1	1	1	1	No	1 client left due to the lack of place in the waiting room
5	2	1	1	1	Yes	Everything is OK
5	2	0	0	2	Yes	1 client left due to the lack of place in the waiting room