TP

—

Le barbier

L3–UPJV

Octobre 2023

L'objectif de cet exercice est de simuler le fonctionnement d'un salon de barbier qui suit l'algorithme suivant :

1. quand le salon est vide le barbier dort ;
2. quand le barbier a fini de raser un client, et qu'il n'y a pas d'autre client en attente, le barbier se rendort, sinon il prend le client suivant dans la salle d'attente ;
3. quand un client arrive, que le barbier rase un client, et qu'il reste une chaise dans la salle d'attente, le client s'assoit et attend ;
4. quand un client arrive et que la salle d'attente est pleine, il s'en va et revient plus tard.

Ecrire un programme qui simule le fonctionnement du salon du barbier. Il y a un thread pour le barbier, et un thread pour chaque client. Les synchronisations s'effectueront avec des variables de conditions. Chaque client se fait raser dix fois. Au départ, il y a plus de clients que de place dans le salon. Quand tous les clients ont terminé, le programme s'arrête. Le code de base est donnée sur la figure 1.

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

#define N 4 //taille de la file d'attente.

int c[N]; //tableau pour la liste circulaire
int d=0, //debut de la liste circulaire
    f=0; //fin de la liste circulaire


int b; // 0 le barbier fait la sieste, 1 il travaille
int s=0; // nombre de clients dans la file d'attente
int a=-1; // numéro du client assis dans le fauteuil
int r=-1; // numéro du client appelé

void * barbier() {
       while (1) {
                if (s==0) printf("barbier : personne... sieste !\n");
                while (s==0) b=0; // la salle d'attente est vide, sieste !
                b=1;
                printf("Barbier réveillé !\n");

                r=c[d]; d=(d+1)%N; s--; // j'appel le client suivant dans la salle d'attente
                printf("barbier : j'appelle client %d\n",r);
                while (a!=r) {}; // j'attend que le client s'installe dans le fauteuil
                printf("barbier: je rase le client %d\n",r);
                sleep(rand()%2);
                printf("barbier: fin client %d\n",r);
                r=-1;
        }
}

void *client(void *arg) {
        int k=*((int *)arg);
        int i=10;
        while(i--){
          sleep(rand()%2);
          if (s==4) {
            printf("client(%d) : la salle est pleine, je vais faire un tour\n",k);
            i++;
            continue;
          }
          if (!b) printf("client(%d) : Heho barbier, reveil !\n",k);
          c[f]=k; f=(f+1)%N; s++; // je m'assois dans un siège de la salle d'attente

          while (r!=k) {}; // j'attend qu'on m'appel
          a=k;
          printf("client(%d) : je m'assois dans le fauteuil et je me fais raser\n",k);
          while (r==k) {}; // On me rase
          printf("client(%d) : Merci, c'est parfait ! Au revoir !\n",k);
        }
}


int main(int argc,char *argv[]) {
  pthread_t bar,clt[10];
  int i,num[10];
  time_t t;

  srand((unsigned) time(&t));

  setvbuf(stdout,NULL, _IONBF, 0);

  pthread_create(&bar,NULL,barbier,NULL);
  for (i=0;i<10;i++) {
    num[i]=i;
    pthread_create(clt+i,NULL,client,num+i);
  }


  for (i=0;i<10;i++) pthread_join(clt[i],NULL);
//  pthread_join(bar,NULL);  // Tous les clients ont fini, on arrete le programme.
  printf("Fin\n");
}
```

FIGURE 1 – Code avec attente active.