

PROGRAMACIÓN CONCURRENTE Y TIEMPO REAL

PRÁCTICA Nº 2

SEMÁFOROS Y MEMORIA COMPARTIDA
(CURSO 2011/2012)

ESCUELA SUPERIOR DE INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA

Cristian Carretón Ruiz 2º A

Indice:

- a) Descripción del problema a solucionar
- b) Explicación de mi solución propuesta para el problema
- c) Cómo compilar y ejecutar
- d) Código:
 - d1) manager.c
 - d2) barbero.c
 - d3) cliente.c

a) Descripción del problema a solucionar

Se pide simular la solución a la siguiente variante del problema del barbero dormilón. En esta ocasión, la barbería tiene una sala de espera con 'n' sillas y una capacidad máxima de 'm' clientes que pueden estar en el interior de la barbería (sentados en las sillas o esperando de pie en la sala). Si la barbería está llena ('m' clientes en total) y llega un cliente nuevo, este cliente no se esperará y abandonará la barbería. La barbería cuenta además, con una sala donde se atiende a los clientes. En esta sala hay tres sillones y tres barberos (el barbero lento, el medio y el rápido) que cortarían el pelo a los clientes. Los barberos son muy vagos y si no tienen clientes, se duermen. El cliente que ocupe el sillón del barbero se encargará de despertarlo. Únicamente los clientes que están sentados en las sillas podrán pasar a ocupar un sillón para que algún barbero les corte el pelo. De esta forma, cuando un cliente se levante de su silla para ocupar un sillón, algún cliente de los que está de pie, ocupará su sitio en la silla libre. Cuando acaben de cortar el pelo, cada cliente pagará al barbero que le corte el pelo una cantidad de dinero fija D, más una propina (aleatoria) entre 0 y P Euros. La barbería cuenta con una caja compartida para todos los barberos y clientes. El cliente depositará el dinero total en una caja, de la que el barbero extraerá únicamente la propina, dejando la cantidad fija D como beneficio de la barbería.

El alumno debe plantear y realizar una solución correcta, con el mayor grado posible de paralelismo, que coordine los barberos y sus clientes de forma adecuada utilizando semáforos y memoria compartida. Se debe definir el código de programa que debe ejecutar cada proceso cliente y los procesos barbero, de forma que puedan evolucionar, sin ningún tipo de interbloqueo.

Implementar una simulación del problema planteado mediante el uso de las primitivas que ofrece el IPC de POSIX y con los siguientes requerimientos específicos e imprescindibles:

- 1) La simulación debe coordinar a los clientes y a los barberos correctamente.
- 2) Cada cliente será un proceso Unix (generado a partir de un único programa).
- 3) Cada barbero será otro proceso Unix.
- 4) Se hará la siguiente distinción en el tiempo que emplea cada barbero en cortar el pelo (se simulará con un simple sleep). El primer barbero tardará S segundos, el segundo 2S segundos y el tercero 3S segundos.
- 5) Debe haber un proceso que realice las inicializaciones necesarias y lance los procesos que intervienen en la resolución del problema. Éste se encargará de lanzar inicialmente los barberos y posteriormente, en tiempos aleatorios, a los clientes.
- 6) El proceso que se encarga de las inicializaciones deberá pasar todas las constantes necesarias para la simulación por línea de órdenes a los procesos barbero y cliente, de forma que no será necesario su recompilación si se cambiara el nombre de algún recurso compartido o constante.
- 7) Cuando finalice la ejecución (porque han finalizado todos los clientes o se ha forzado la finalización mediante Control+C) el proceso principal deberá liberar todos aquellos recursos inicialmente creados.
- 8) Tras lanzar los procesos necesarios, se ha de mostrar información sobre qué está haciendo cada uno de ellos y sobre su estado. Se podrá visualizar cualquier otra información que se considere interesante con la adecuada justificación.
- 9) Al finalizar la simulación se deberá mostrar el dinero total recaudado por la barbería

b) Explicación de mi solución propuesta para el problema

Para solucionar el problema propuesto en esta práctica he empleado tres archivos .c; el manager, el proceso barbero y el proceso cliente. Los barberos pueden ser de 3 tipos distintos: rápidos, medios y lentos. Para sincronizar los procesos cliente y barbero he utilizado varios semáforos y variables en memoria compartida, que explicaré a continuación:

SEMÁFOROS

- MSILLAS: para acceder a la variable compartida SENTADOS que lleva la cuenta de los clientes que se encuentran sentados en las sillas. Para consultar/modificar su valor hay que hacerle un wait a MSILLAS, y un signal al acabar.
- MPIE: para acceder a la variable compartida ENPIE que lleva la cuenta de los clientes que se encuentran dentro de la barbería pero de pie. Para consultar/modificar su valor hay que hacerle un wait a MPIE, y un signal al acabar.
- MCAJA: para acceder a la variable CAJA en la cual se encuentran las ganancias de la barbería por cada pelado, a la que pueden acceder todos los barberos, pero solo uno simultáneamente.

Los siguientes semáforos existen 3 tipos de cada uno (R – barbero rápido, M – barbero medio y L – barbero lento)

- DESPERTAR_ : se utiliza para que cuando un cliente que espera para pelarse le llega su turno, avisar al barbero correspondiente que permanecerá dormido. Se inicializa a 0, porque inicialmente el barbero duerme.
- PREPARADO_ : se utiliza para que el barbero avise al cliente que ya esta listo para pelarlo, justos después de que el cliente lo despierte. Se inicializa a 0.
- SENTADO_ : se utiliza para que una vez el barbero este preparado, el cliente este ya sentado en el sillón para pelarse, avisar al barbero que puede comenzar con el pelado. Se inicializa a 0.
- PAGADO_ : una vez el barbero ha cortado el pelo al cliente, espera a que este le pague y cuando el cliente pague al barbero también le avisa por medio de este semáforo. Inicialmente se encuentra con un valor 0.
- AGRADECER_ : este semáforo se utiliza para avisar al cliente que ya se puede marchar, porque ya ha pagado y en caso de entregar propina al barbero este se lo ha agradecido. Inicialmente se encuentra a 0.
- SILLAS: semáforo contador que lleva la cuenta del número de sillas disponibles para sentarse los clientes que van llegando. Inicialmente cuenta con el valor del número total de sillas en la peluquería.

VARIABLES COMPARTIDAS

- ENPIE: variable en la que se lleva la cuenta de los clientes que actualmente se encuentran en pie en la barbería. Como he indicado anteriormente se accede a ella después de hacer un wait(MPIE) y una vez utilizado se hace un signal(MPIE)
- SENTADOS: variable en la que se lleva la cuenta de los clientes que actualmente se encuentran sentados en la barbería. Como he indicado anteriormente se accede a ella después de hacer un wait(MSILLAS) y una vez utilizado se hace un signal(MSILLAS)

- CAJA: variable en la que se introduce el importe de cada corte de pelo y que comparten los 3 tipos de barberos. Como he indicado anteriormente se accede a ella después de hacer un wait(CAJA) y una vez utilizado se hace un signal(CAJA)
- PROPINA: esta variable es independiente para cada barbero, por lo que no es necesario un semáforo para poder acceder a ella. En ella se guarda la propina que cada cliente entrega a su barbero (en caso de que lo haga)

PSEUDOCÓDIGO DE LA SOLUCIÓN

Cliente

```

1. wait(msillas)
2. //Consultar gente sentada
3. wait(mpie)
4. //Consultar gente levantada
5. Si(sentados+levantados<CAPACIDAD_BARBERIA)
6.   signal(mpie)
7.   Si(sentados>=N_SILLAS)
8.     signal(msillas)
9.     wait(mpie)
10.    //Incrementar en uno la gente levantada
11.    signal(mpie)
12.    wait(sillas)
13.    //Incrementar en uno la gente sentada
14.    signal(msillas)
15.    wait(mpie)
16.    //Decrementar gente levantada en uno
17.    signal(mpie)
18.  Sino
19.    //Incrementar gente sentada en uno
20.    wait(mpie)
21.    signal(msillas)
22.    signal(mpie)
23.    wait(sillas)
24.  Fin_Sino
25.  signal(despertar)
26.  wait(preparado)
27.  wait(msillas)
28.  //Decrementar en uno la gente sentada
29.  signal(msillas)
30.  signal(sillas)
31.  signal(sentado)
32.  wait(mcaja)
33.  //Añadir importe a la caja
34.  signal(mcaja)
35.  //Generar propina aleatoriamente
36.  signal(pagado)
37.  wait(agradecer)
38.  Sino
39.    signal(mpie)
40.    signal(msillas)
41. Fin_Sino

```

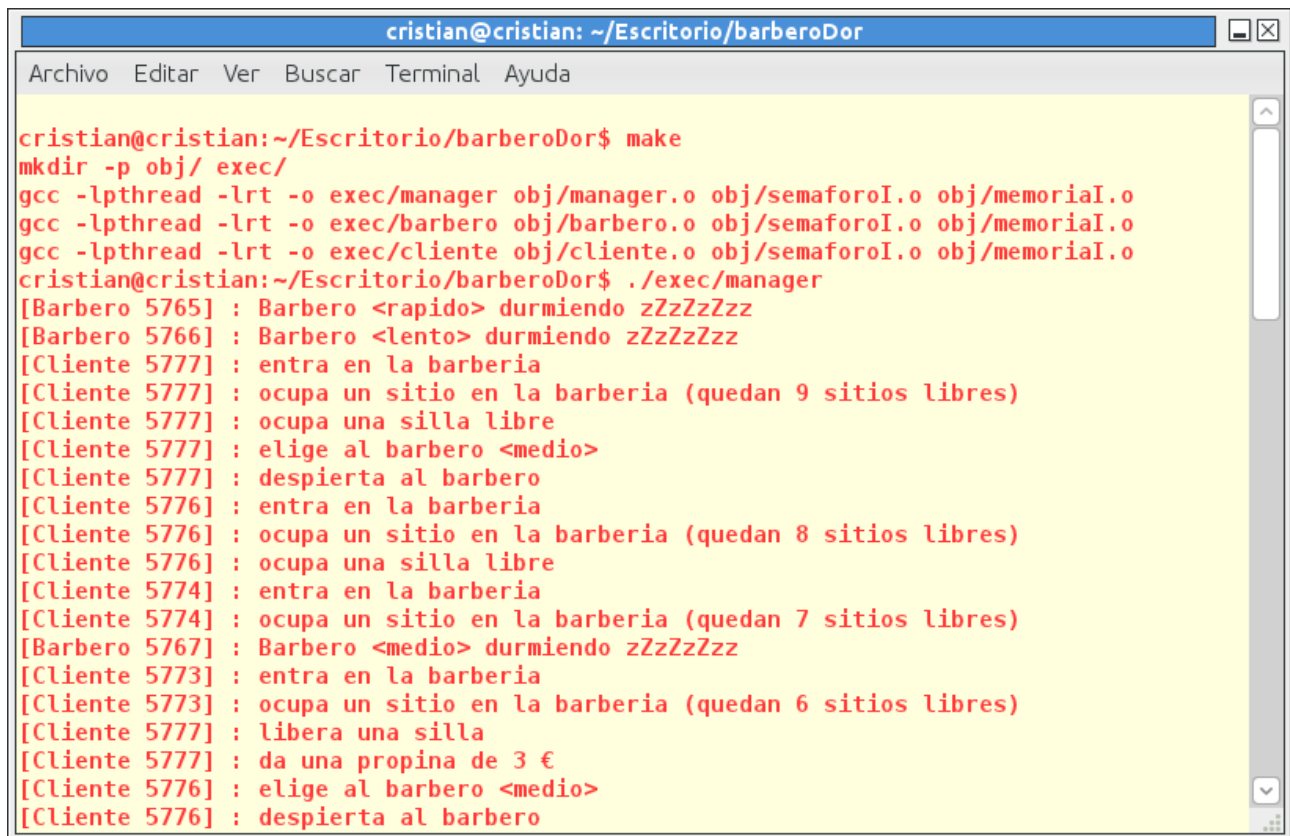
Barbero

1. while(1)
2. wait(despertar)
3. signal(preparado)
4. wait(sentado)
5. //Cortando el pelo
6. wait(pagado)
7. //Obtener propina
8. Si(propina!=0) mostrar("barbero agradece propina")
9. signal(agradecer)
10. fin_while

c) Cómo compilar y ejecutar

Para compilar la práctica, se descomprime el archivo 70587447.tar.gz mediante el comando "tar xzvf 70587447.tar.gz", una vez descomprimido se quedará en una carpeta que por defecto tiene el mismo nombre que el archivo comprimido pero sin la extensión 'tar.gz', es decir: 70587447. Se sitúa en dicha carpeta con los comandos pertinentes en la terminal (cd: change directory seguido del nombre de la carpeta a la que acceder), una vez en el interior de carpeta '70587447' se ejecuta la orden: 'make' que se encarga de compilar el código de todos los archivos .c , actualizar los ya existentes y crear un archivo ejecutable.

Una vez realizado el make, ya tendremos nuestro archivo ejecutable en la ubicación (exec/manager), que ejecutaremos con el comando: ./exec/manager; y se mostrará por pantalla la acción que realiza cada clientes y cada barbero. Una vez se agoten todos los clientes, el programa para los procesos barbero y muestra lo recaudado por los barberos en dicha ejecución.



```
cristian@cristian: ~/Escritorio/barberoDor
Archivo Editar Ver Buscar Terminal Ayuda

cristian@cristian:~/Escritorio/barberoDor$ make
mkdir -p obj/ exec/
gcc -lpthread -lrt -o exec/manager obj/manager.o obj/semaforoI.o obj/memoriaI.o
gcc -lpthread -lrt -o exec/barbero obj/barbero.o obj/semaforoI.o obj/memoriaI.o
gcc -lpthread -lrt -o exec/cliente obj/cliente.o obj/semaforoI.o obj/memoriaI.o
cristian@cristian:~/Escritorio/barberoDor$ ./exec/manager
[Barbero 5765] : Barbero <rapido> durmiendo zZzZzZzz
[Barbero 5766] : Barbero <lento> durmiendo zZzZzZzz
[Cliente 5777] : entra en la barberia
[Cliente 5777] : ocupa un sitio en la barberia (quedan 9 sitios libres)
[Cliente 5777] : ocupa una silla libre
[Cliente 5777] : elige al barbero <medio>
[Cliente 5777] : despierta al barbero
[Cliente 5776] : entra en la barberia
[Cliente 5776] : ocupa un sitio en la barberia (quedan 8 sitios libres)
[Cliente 5776] : ocupa una silla libre
[Cliente 5774] : entra en la barberia
[Cliente 5774] : ocupa un sitio en la barberia (quedan 7 sitios libres)
[Barbero 5767] : Barbero <medio> durmiendo zZzZzZzz
[Cliente 5773] : entra en la barberia
[Cliente 5773] : ocupa un sitio en la barberia (quedan 6 sitios libres)
[Cliente 5777] : libera una silla
[Cliente 5777] : da una propina de 3 €
[Cliente 5776] : elige al barbero <medio>
[Cliente 5776] : despierta al barbero
```

d)Código:

d1)manager.c:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <signal.h>
#include <memoriaI.h>
#include <semaforoI.h>

#define N_CLIENTES 10
#define N_SILLAS 1
#define BARBEROS 3
//Mutex sobre variables compartidas
#define MSILLAS "mtx_sillas"
#define MPIE "mtx_pie"
#define MCAJA "mtx_caja"
#define S 3 //Tiempo base que tarda cada barbero en cortar el pelo
//Semaforos
//*****RAPIDO*****//
#define DESPERTAR_R "despertarRapido"
#define PREPARADO_R "preparadoRapido"
#define SENTADO_R "sentadoRapido"
#define PAGADO_R "pagadoRapido"
#define AGRADECER_R "agradecerRapido"
//*****MEDIO*****//
#define DESPERTAR_M "despertarMedio"
#define PREPARADO_M "preparadoMedio"
#define SENTADO_M "sentadoMedio"
#define PAGADO_M "pagadoMedio"
#define AGRADECER_M "agradecerMedio"
//*****LENTO*****//
#define DESPERTAR_L "despertarLento"
#define PREPARADO_L "preparadoLento"
#define SENTADO_L "sentadoLento"
#define PAGADO_L "pagadoLento"
#define AGRADECER_L "agradecerLento"
//*****//
#define SILLAS "sillasEspera"
#define ENPIE "EsperanEnPie"
#define SENTADOS "EsperanSentados"
#define CAJA "CajaBarberia"
#define PROPINA "PropinaBarbero"

void controlador (int senhal);
```



```

void finalizarprocesos(int);
void liberarecursos();
pid_t pids[BARBEROS+N_CLIENTES];

int main (int argc, char *argv[]) {
    int i;
    int j=0;
    memset (pids, 0, sizeof(pid_t)*(BARBEROS+N_CLIENTES));
    srand((int) getpid());

    // Creación de semáforos y segmentos de memoria compartida.

    // Manejo de variables compartidas (mutex)
    crear_sem(MSILLAS,1);
    crear_sem(MPIE,1);
    crear_sem(MCAJA,1);
    // Semaforos lentos
    crear_sem(PREPARADO_L,0);
    crear_sem(DESPERTAR_L,0);
    crear_sem(SENTADO_L,0);
    crear_sem(PAGADO_L,0);
    crear_sem(AGRADECER_L,0);
    // Semaforos medios
    crear_sem(PREPARADO_M,0);
    crear_sem(DESPERTAR_M,0);
    crear_sem(SENTADO_M,0);
    crear_sem(PAGADO_M,0);
    crear_sem(AGRADECER_M,0);
    // Semaforos rapidos
    crear_sem(PREPARADO_R,0);
    crear_sem(DESPERTAR_R,0);
    crear_sem(SENTADO_R,0);
    crear_sem(PAGADO_R,0);
    crear_sem(AGRADECER_R,0);

    crear_sem(SILLAS, N_SILLAS);

    //Variables compartidas
    crear_var(ENPIE,0);
    crear_var(SENTADOS,0);
    crear_var(CAJA,0);
    crear_var(PROPINA,0);

    // Manejo de Ctrol+C.
    if (signal(SIGINT, controlador) == SIG_ERR) {
        fprintf(stderr, "Abrupt termination.\n");    exit(1);
    }

    // Lanzar los 3 barberos
    int time=0;
    char *tiempo;
    tiempo=(char*)malloc(sizeof(int));

```

```

for (i = 0; i < BARBEROS; i++) {
    if((pids[j++] = fork()) == 0) {
        switch(i){
            // Mismo proceso; distintos datos.
            case 0: //Barbero <rapido>
                time=S;
                sprintf(tiempo,"%d",time);
                execl("./exec/barbero", "barbero","rapido",tiempo,
                    DESPERTAR_R, PREPARADO_R, SENTADO_R, PAGADO_R,
                    AGRADECER_R, PROPINA,NULL);

                break;
            case 1://Barbero <lento>
                time=S*3;
                sprintf(tiempo,"%d",time);
                execl("./exec/barbero", "barbero","lento", tiempo,
                    DESPERTAR_L, PREPARADO_L, SENTADO_L, PAGADO_L,
                    AGRADECER_L, PROPINA, NULL);

                break;
            case 2://Barbero <medio>
                time=S*2;
                sprintf(tiempo,"%d",time);
                execl("./exec/barbero", "barbero","medio",tiempo,
                    DESPERTAR_M, PREPARADO_M, SENTADO_M, PAGADO_M,
                    AGRADECER_M, PROPINA,NULL);

                break;
        }//Fin switch
    }// Fin if
}

for (i = 1; i <= N_CLIENTES; i++) {
    if ((pids[j++] = fork()) == 0) {
        int n=(rand()%BARBEROS)+1;
        if (n==1)
            execl("./exec/cliente", "cliente","rapido", MSILLAS,
                MPIE, MCAJA,DESPERTAR_R, PREPARADO_R, SENTADO_R,
                PAGADO_R, AGRADECER_R, ENPIE, SENTADOS, CAJA,
                PROPINA,SILLAS, NULL);
        else if(n==2)
            execl("./exec/cliente", "cliente","lento", MSILLAS,
                MPIE, MCAJA,DESPERTAR_L, PREPARADO_L,SENTADO_L,
                PAGADO_L, AGRADECER_L, ENPIE, SENTADOS, CAJA,
                PROPINA,SILLAS, NULL);
        else
            execl("./exec/cliente", "cliente","medio", MSILLAS,
                MPIE, MCAJA,DESPERTAR_M, PREPARADO_M,SENTADO_M,
                PAGADO_M, AGRADECER_M ,ENPIE, SENTADOS, CAJA,
                PROPINA, SILLAS, NULL);
    }// Fin if
}

for (; j>=BARBEROS; j--) waitpid(pids[j], 0, 0);
finalizarprocesos(0);
int negocio, ganancias;
ganancias=obtener_var(CAJA);

```

```

    consultar_var(ganancias,&negocio);
    liberarecursos();
    printf("\n----- Ganancias Finales ----- \n");
    printf("La barbería ha ganado %d\n",negocio);
    return 0;
}

void controlador (int senhal) {
    printf("\nCtrl+c capturada.\n");
    printf("Finalizando...\n\n");
    finalizarprocesos(1);
    liberarecursos();
    printf("OK!\n");
    // Salida del programa.
    exit(0);
}

void liberarecursos (void) {
    printf ("\n----- Liberando recursos ----- \n");
    printf("Recursos generales...\n");
    destruir_sem(MSILLAS);
    destruir_sem(MPIE);
    destruir_sem(MCAJA);
    printf("Recursos relativos a los 3 barberos ...\n");
    // Semaforos lentos
    destruir_sem(PREPARADO_L);
    destruir_sem(DESPERTAR_L);
    destruir_sem(SENTADO_L);
    destruir_sem(PAGADO_L);
    destruir_sem(AGRADECER_L);
    // Semaforos medios
    destruir_sem(PREPARADO_M);
    destruir_sem(DESPERTAR_M);
    destruir_sem(SENTADO_M);
    destruir_sem(PAGADO_M);
    destruir_sem(AGRADECER_M);
    // Semaforos rapidos
    destruir_sem(PREPARADO_R);
    destruir_sem(DESPERTAR_R);
    destruir_sem(SENTADO_R);
    destruir_sem(PAGADO_R);
    destruir_sem(AGRADECER_R);

    destruir_sem(SILLAS);

    //Variables compartidas
    destruir_var(ENPIE);
    destruir_var(SENTADOS);
    destruir_var(CAJA);
    destruir_var(PROPINA);
}

void finalizarprocesos(int todos) {

```

```

/* Si todos == 0, se manda señal sólo a procesos B */
/* Si todos == 1, se manda señal a todos los procesos */
int i, nproc;

if (todos) nproc = BARBEROS+N_CLIENTES; else nproc = BARBEROS;
printf ("\n----- Finalización de procesos -----
                                             \n");

for (i=0; i<nproc; i++) {
    if (pids[i]) {
        printf ("Finalizando proceso [%d]...", pids[i]);
        kill(pids[i], SIGINT); printf ("<Ok>\n");
    }
}
}

d2)barbero.c:
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <semaforoI.h>
#include <memoriaI.h>

void barbero ();
void controlador (int senhal) {
    printf ("[Barbero %ld] Finalizado (SIGINT)\n", (long)getpid());
    exit(1);
}

int main (int argc, char *argv[]) {
    if (signal(SIGINT, controlador) == SIG_ERR) {
        fprintf(stderr, "Abrupt termination.\n");    exit(1);
    }
    barbero(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],
                                                    argv[8]);

    return 0;
}

void barbero (char *tipo, char *tiempo, char *id_despertar, char
    *id_preparado, char *id_sentado, char *id_pagado, char
    *id_agradecer, char *propina) {
    sem_t *despertar, *preparado, *sentado, *pagado, *agradecer;
    int propina_handle, P;
    int pelando=atoi(tiempo);
    despertar = get_sem(id_despertar);
    preparado = get_sem(id_preparado);
    sentado = get_sem(id_sentado);
    pagado = get_sem(id_pagado);
    agradecer = get_sem(id_agradecer);
    propina_handle=obtener_var(propina);

```

```

while(1){
    printf("[Barbero %ld] : Barbero <%s> durmiendo
           zZzZzZzz\n", (long) getpid(), tipo);
    wait_sem(despertar);
    signal_sem(preparado);
    wait_sem(sentado);
    //CORTANDO PELO
    printf("[Barbero %ld] : Barbero <%s> cortando el
    pelo\n", (long) getpid(), tipo);
    sleep(pelando);
    printf("[Barbero %ld] : Barbero <%s> fin de corte\n",
    (long) getpid(), tipo);
    wait_sem(pagado);
    consultar_var(propina_handle, &P);
    if(P!=0) printf("[Barbero %ld] : Barbero <%s> agradece
    propina %d\n", (long) getpid(), tipo, P);
    signal_sem(agradecer);
} //fin WHILE
} //fin BARBERO

```

d3)cliente.c:

```

#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <semaforoI.h>
#include <memoriaI.h>

#define CAPACIDAD_TOTAL 10
#define N_SILLAS 2
#define K 10 //importe por cada corte de pelo
#define MAX_PROPINA 5 //propina maxima
void cliente ();

void controlador (int senhal) {
    printf ("[Cliente %ld] Finalizado (SIGINT)\n", (long) getpid());
    exit(1);
}

int main (int argc, char *argv[]) {
    if (signal(SIGINT, controlador) == SIG_ERR) {
        fprintf(stderr, "Abrupt termination.\n");        exit(1);
    }

    cliente(argv[1], argv[2], argv[3], argv[4], argv[5], argv[6], argv[7],
            argv[8], argv[9], argv[10], argv[11], argv[12], argv[13],
            argv[14]);

    return 0;
}

```

```

void cliente (char *tipo, char *idm_sillas, char *idm_pie, char
              *idm_caja, char *id_despertar, char *id_preparado,
              char *id_sentado, char *id_pagado, char *id_agradecer,
              char *enpie, char *cl_sentados, char *caja, char *propi,
              char *id_sillas){
    sem_t *despertar, *preparado, *sentado, *pagado, *agradecer,
    *sillas, *msillas, *mpie, *mcaja;
    //Semaforos que controlan acciones
    despertar=get_sem(id_despertar);
    preparado=get_sem(id_preparado);
    sentado = get_sem(id_sentado);
    pagado = get_sem(id_pagado);
    agradecer = get_sem(id_agradecer);
    sillas = get_sem(id_sillas);
    //Cerrojos sobre variables compartidas
    msillas = get_sem(idm_sillas);
    mpie = get_sem(idm_pie);
    mcaja = get_sem(idm_caja);
    //Obtener variables compartidas
    int cajaBarb_handler, sillas_handler, propi_handler,
    depie_handler;
    cajaBarb_handler = obtener_var(caja);
    sillas_handler = obtener_var(cl_sentados);
    propi_handler = obtener_var(propi);
    depie_handler = obtener_var(enpie);
    int sentados, levantados, cajaBarberia, P;
    wait_sem(msillas);
    consultar_var(sillas_handler, &sentados);
    wait_sem(mpie);
    consultar_var(depie_handler, &levantados);
    if(sentados+levantados<CAPACIDAD_TOTAL){
        printf("[Cliente %ld] : entra en la barberia\n",
              (long)getpid());

        signal_sem(mpie);
        if(sentados>=N_SILLAS){
            signal_sem(msillas);
            wait_sem(mpie);
            consultar_var(depie_handler, &levantados);
            modificar_var(depie_handler, ++levantados);
            printf("[Cliente %ld] : ocupa un sitio en la
            barberia (quedan %d sitios libres)\n"
              (long)getpid(),
              CAPACIDAD_TOTAL-(sentados+levantados));
            signal_sem(mpie);
            wait_sem(sillas);
            wait_sem(msillas);
            printf("[Cliente %ld] : ocupa una silla libre\n",
              (long)getpid());
            consultar_var(sillas_handler, &sentados);
            modificar_var(sillas_handler, ++sentados);
        }
    }
}

```

```

        signal_sem(msillas);
        wait_sem(mpie);
        consultar_var(depie_handler,&levantados);
        modificar_var(depie_handler,--levantados);
        signal_sem(mpie);
    }else{
        consultar_var(sillas_handler,&sentados);
        modificar_var(sillas_handler,++sentados);
        wait_sem(mpie);
        consultar_var(depie_handler,&levantados);
        printf("[Cliente %ld] : ocupa un sitio en la
barberia (quedan %d sitios libres)\n"
               (long)getpid(),CAPACIDAD_TOTAL-
               (sentados+levantados));

        printf("[Cliente %ld] : ocupa una silla libre\n",
               (long)getpid());

        signal_sem(msillas);
        signal_sem(mpie);
        wait_sem(sillas);
    }
    printf("[Cliente %ld] : elige al barbero <%s>\n",
           (long)getpid(),tipo);
    printf("[Cliente %ld] : despierta al barbero\n",
           (long)getpid());

    signal_sem(despertar);
    wait_sem(preparado);
    printf("[Cliente %ld] : libera una silla\n",
           (long)getpid());

    wait_sem(msillas);
    consultar_var(sillas_handler,&sentados);
    modificar_var(sillas_handler,--sentados);
    signal_sem(msillas);
    signal_sem(sillas);
    signal_sem(sentado);
    wait_sem(mcaja);
    consultar_var(cajaBarb_handler,&cajaBarberia);
    modificar_var(cajaBarb_handler,cajaBarberia+K);
    signal_sem(mcaja);
    srand((int)getpid());
    int clipro=rand()%MAX_PROPINA;
    consultar_var(prop_i_handler,&P);
    modificar_var(prop_i_handler,clipro);
    printf("[Cliente %ld] : da una propina de %d €\n",
           (long)getpid(),clipro);

    signal_sem(pagado);
    wait_sem(agradecer);
    printf("[Cliente %ld] : se va de la barberia\n",
           (long)getpid());
}
}else{
    printf("[Cliente %ld] : No coge en la barberia, pase mas

```

```
tarde\n", (long) getpid());  
signal_sem(msillas);  
signal_sem(mpie);  
exit(1);  
}  
  
}
```