

**Examen de Sistemas Operativos (Parte práctica). Convocatoria Diciembre.  
Curso 2007/08. Tercero Ingeniería Técnica Informática**

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| Nombre:    | <input type="checkbox"/> Gestión <input type="checkbox"/> Sistemas |
| Apellidos: | 12 de Diciembre de 2008                                            |

**[Cuestión 1.]**      Resuelva con una línea de comandos UNIX las siguientes tareas:

**Apartado a.-** Imprimir el listado de las señales disponibles en el sistema al final del fichero “senales.txt” situado en su directorio HOME.

```
kill -l >> $HOME/senales.txt
```

**Apartado b.-** Un usuario ha generado un fichero ejecutable, *ejercicio*, en el subdirectorio bin de su directorio HOME. El ejecutable funciona correctamente y tiene los privilegios necesarios. Al intentar ejecutarlo introduciendo el nombre del ejecutable ocurre lo siguiente:

```
BASH@/home/practica/bin>ejercicio  
bash: ejercicio: command not found
```

Realice la modificación necesaria para que al volver a insertar la orden de igual forma el programa se ejecute correctamente.

```
PATH=$PATH:$HOME/bin
```

**Apartado c.-** Obtenga un listado de todos los ficheros del sistema terminados en .c sin permiso de ejecución para el grupo

```
ls -lR / | grep "\.c$" | grep "^-....-"
```

**Apartado d.-** Añada permiso de ejecución para el grupo y otros a los ficheros *ejercicio1* y *ejercicio2* situados en su directorio HOME.

```
chmod go+x $HOME/ejercicio1 $HOME/ejercicio2  
chmod go+x $HOME/ejercicio[12]
```

**Puntuación: 2 ptos.**

**Tiempo estimado: 10 min.**



**Examen de Sistemas Operativos (Parte práctica). Convocatoria Diciembre.  
Curso 2007/08. Tercero Ingeniería Técnica Informática**

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| Nombre:    | <input type="checkbox"/> Gestión <input type="checkbox"/> Sistemas |
| Apellidos: | 12 de Diciembre de 2008                                            |

**[Cuestión 2.]** Realice un programa en C cuya ejecución resulte en un proceso padre que lanzará un hijo cada 10 segundos, lanzando un total de 10 procesos hijos. Cada proceso hijo contará el número de ficheros regulares que tengan un tamaño mayor a 1024 bytes que hay en un cierto directorio, y enviará el resultado al padre mediante un mecanismo de comunicación entre procesos visto en clase. El padre a su vez mostrará el resultado por pantalla. El directorio del cual se contarán el número de ficheros regulares se recibirán como primer argumento al invocar al programa.

**NOTAS:**

- No deje procesos zombis/huérfanos
- Recuerde que dos procesos cualesquiera nunca comparten memoria: No emplee variable compartidas para comunicar padre e hijo.
- No use sleep para esperar 10 segundos, use señales en su lugar.

**Puntuación: 4 ptos.**

**Tiempo estimado: 30 min.**

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <sys/stat.h>
#include <fcntl.h>

char *fichero;

void alarma(int foo)
{
    int contador = 0;
    char path[PATH_MAX], entrada[PATH_MAX];
    int fd, ret, fds[2];
    DIR *d;
    struct dirent *dent;
    struct stat statbuf;

    if (pipe(fds) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    ret = fork();
    if (ret == 0) {
        close(fds[0]);

        fd = open(fichero, O_RDONLY);
        if (fd == -1) {
            perror("open");
            exit(EXIT_FAILURE);
        }

        ret = read(fd, path, sizeof(path));
        if (ret == -1) {
            perror("read");
            exit(EXIT_FAILURE);
        }
    }
```

```

        path[ret-1]='\0';

        close(fd);

        d = opendir(path);
        if (d == NULL) {
            perror("opendir");
            exit(EXIT_FAILURE);
        }

        while ((dent = readdir(d)) != NULL) {
            snprintf(entrada, PATH_MAX, "%s/%s", path, dent->d_name);

            if (stat(entrada, &statbuf) == -1)
                continue;

            if (statbuf.st_mode & S_IFREG && statbuf.st_size > 1024)
                contador++;
        }

        closedir(d);

        if (write(fds[1], &contador, sizeof(int)) == -1) {
            perror("write");
            exit(EXIT_FAILURE);
        }

        close(fds[1]);
        exit(EXIT_SUCCESS);
    } else {
        close(fds[1]);
        if (read(fds[0], &contador, sizeof(int)) == -1) {
            perror("read");
            exit(EXIT_FAILURE);
        }
        printf("ficheros regulares: %d\n", contador);

        close(fds[0]);

        wait(NULL);
    }
}

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

    if (argc != 2) {
        printf("Uso: %s [directorio]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    fichero = argv[1];

    if (signal(SIGALRM, alarma) == SIG_ERR) {
        perror("signal");
        exit(EXIT_FAILURE);
    }

    for (i=0; i<10; i++) {
        alarm(10);
        pause();
    }
}

```

# Examen de Sistemas Operativos (Parte práctica). Convocatoria Diciembre. Curso 2007/08. Tercero Ingeniería Técnica Informática

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| Nombre:    | <input type="checkbox"/> Gestión <input type="checkbox"/> Sistemas |
| Apellidos: | 12 de Diciembre de 2008                                            |

**[Cuestión 3.]** Se desea resolver, haciendo uso de semáforos, el problema del barbero dormilón. Tenemos un barbero y varios clientes (no existe ninguna relación filial entre dichos procesos). Dado que el número de clientes va cambiando, para almacenar dicho valor se utilizará un fichero denominado *nClientesEsperando.txt* (`#define ESPERANDO "/nClientesEsperando.txt"`), accesible tanto para el barbero como para cada uno de los clientes. Hay una silla de peluquero y N sillas de espera para que se sienten los clientes que están esperando. Se tienen que tener en cuenta los siguientes aspectos de sincronización:

- Si no hay clientes, el barbero se sienta en su silla de peluquero y se va a dormir.
- El barbero se despertará cuando llegue un cliente.
- Si llegan más clientes mientras el barbero corta el pelo a un cliente, se sientan en el caso de que haya sillas de espera libres, o en caso contrario se va.
- Debe garantizarse la exclusión mutua en el acceso al fichero.

Para la lectura y escritura en ficheros a bajo nivel se pueden emplear las siguientes funciones ya implementadas:

**int escribir (char \*ruta, char \*cadena);** en caso de éxito devuelve 0 y en caso contrario -1.

**int leer (char \*ruta, char \*cadena\_leida);** en caso de éxito devuelve 0 y en caso contrario -1. Lee todo el contenido del fichero indicado; considere que como máximo tendrá 255 caracteres.

La implementación se ha modularizado de la siguiente manera:

- Un proceso **barbero** que será el que se encargue de ir atendiendo a cada uno de los clientes. El número de sillas de espera, N, tomará el valor 5 (`#define N 5`). Considere que el tiempo de cada corte de pelo es de 5 segundos. El barbero en cada ejecución realizará 15 iteraciones.
- Una serie de procesos **clientes**, que se encargarán de ir leyendo el contenido del fichero para determinar cuantos clientes hay esperando y por tanto el número de sillas de espera ocupadas, y en caso de ser posible irán tomando asiento. Para mayor facilidad de ejecución cada proceso cliente llevará a cabo 3 iteraciones.
- Un proceso **iniciador** que será el encargado de la inicialización de los semáforos y de la creación del fichero ESPERANDO.

En primer lugar se lanzará el proceso iniciador. Y a continuación el barbero o los clientes (el orden no influye en la sincronización). Por tanto, debe implementar **tres programas** (iniciador, barbero y cliente).

## NOTAS:

- Es necesario hacer uso de la **librería de semáforos** empleada en el laboratorio.
- No debe preocuparse de codificar la destrucción de los semáforos, dado que existe un manejador de señal, llamado *Terminador*. Lo único que tendrá que hacer será instalarlo adecuadamente en el programa *barbero*.
- No se permite el uso de variables compartidas.

**Puntuación: 4 ptos.**

**Tiempo estimado: 40 min.**

```
// iniciador.h

#include "../Semaph.h"

#define ESPERANDO "/nClientesEsperando.txt"
#define SILLAS 5

Semaph excmut, clientes, barbero;

void Iniciador(void);
void Terminador(int);

// iniciador.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```

#include <fcntl.h>

#include "../Semaph.h"
#include "iniciador.h"

int main ()
{
    Iniciador();
    escribir(ESPERANDO,"0");
    return 0;
}

void Iniciador(void)
{
    excmut = Semaph_Create("excmut",1);
    barbero = Semaph_Create("barbero",0);
    clientes = Semaph_Create("clientes",0);
}

// barbero.c

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

#include "../Semaph.h"
#include "iniciador.h"

int main(void)
{
    extern Semaph excmut, clientes, barbero;
    int nClientesEsperando;
    char cadena[255];
    int i;

    if (signal(SIGINT, Terminador) != SIG_ERR)
    {
        Iniciador();
        for(i=0;i<20;i++)
        {
            printf("\n[BARBERO, PID:%d]: \t \t ESPERANDO A QUE LLEGUEN CLIENTES A LA BARBERIA\n", getpid());
            Semaph_Down(clientes);
            Semaph_Down(excmut); // Acceso exclusivo al contador de clientes esperando (almacenado en el Fichero)
            if (leer(ESPERANDO, cadena) != -1)
            {
                nClientesEsperando=atoi(cadena);
                nClientesEsperando=nClientesEsperando-1;
                printf("[BARBERO, PID:%d]: \t \t nClientesEsperando=%d\n", getpid(), nClientesEsperando);
                sprintf(cadena, "%d", nClientesEsperando);
                escribir(ESPERANDO, cadena);
                Semaph_Up(barbero);
                Semaph_Up(excmut);

                printf("[BARBERO, PID:%d]: \t Estoy cortandole el pelo a un cliente. nClientesEsperando=%d\n", getpid(), nClientesEsperando);
                sleep (5);
            }
            else
            {
                printf ("Error al leer el número de clientes");
                Semaph_Up(excmut);
                exit(-1);
            }
        }
    }
}

```

**Examen de Sistemas Operativos (Parte práctica). Convocatoria Diciembre.  
Curso 2007/08. Tercero Ingeniería Técnica Informática**

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| Nombre:    | <input type="checkbox"/> Gestión <input type="checkbox"/> Sistemas |
| Apellidos: | 12 de Diciembre de 2008                                            |

```
    }
}
return 0;
}

// cliente.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include "../Semaph.h"
#include "iniciador.h"

int main(void)
{
    extern Semaph excmut, clientes, barbero;
    int nClientesEsperando;
    char cadena[255];
    int i;

    Iniciador();

    for(i=0;i<3;i++)
    {
        Semaph_Down(excmut); // Acceso exclusivo al FICHERO
        if (leer(ESPERANDO, cadena)!=-1)
        {
            nClientesEsperando=atoi(cadena);
            if (nClientesEsperando < SILLAS)
            {
                nClientesEsperando=nClientesEsperando+1;
                printf("\n[CLIENTE PID: %d]: \n \t nClientesEsperando=%d\n",
getpid(),nClientesEsperando);
                sprintf (cadena, "%d", nClientesEsperando);
                escribir(ESPERANDO,cadena);
                Semaph_Up(clientes);
                Semaph_Up(excmut);

                printf("[CLIENTE PID: %d]: Estoy esperando recibir el corte de pelo.
nClientesEsperando=%d\n", getpid(),nClientesEsperando);
                Semaph_Down(barbero);

                printf("[CLIENTE PID: %d]: Ya he recibido el corte de pelo. ABANDONO la
BARBERIA. nClientesEsperando=%d\n", getpid(), nClientesEsperando);
            }
            else
            {
                printf("[CLIENTE PID: %d]: NO HAY SILLAS LIBRES. Vuelvo otro dia.
nClientesEsperando=%d\n", getpid(), nClientesEsperando);
                Semaph_Up(excmut);
            }
        }
        else
        {
            printf ("Error al leer el número de clientes");
            Semaph_Up(excmut);
        }
    }
}
```

```
}  
}  
return 0;  
}
```