

Sistemas Operativos 5º semestre. Grado II

Segundo Parcial. Gestión de Procesos. 6 de noviembre de 2017.

Dispone de 50 minutos. Publicación el 24 de noviembre. Revisión el 28 de noviembre.

Ejercicio 1 (1.5 puntos)

En algunas ocasiones un proceso puede no querer recibir alguna señal, para lo cual puede ignorar la señal o bloquearla. ¿Cuál es la diferencia entre bloquear e ignorar una señal? ¿Son acciones incompatibles o excluyentes? ¿Qué servicios UNIX utilizaríamos para llevar a cabo una u otra acción?

Cuando se genera una señal, ésta debe ser entregada al proceso al que va dirigida. Si la señal está bloqueada, ésta queda pendiente de entrega, y será entregada cuando el proceso la desbloquee en su máscara de señales. Por otro lado, si la señal está ignorada, ésta es descartada, independientemente de la máscara de señales y no quedando pendiente para su posterior entrega. Ambas acciones son compatibles y pueden llevarse a cabo simultáneamente.

Los servicios que utilizaremos para bloquear una señal son los que nos permiten establecer/modificar una máscara de señales: `sigprocmask()` para manipular la máscara y `sigemptyset()`, `sigaddset()`, `sigdelset()`, etc. para establecer el conjunto de señales se aplicará el bloqueo/desbloqueo.

El servicio que usaremos para ignorar una señal será `sigaction()` con el valor `SIG_IGN` como handler.

Ejercicio 2 (1 punto)

¿Qué es el BCP y qué información contiene?

El BCP (Bloque de Control de Proceso) es una estructura de datos que maneja el sistema operativo y que almacena información relativa a un proceso. Cada proceso debe estar representado por un BCP que contiene, entre otra, la siguiente información:

- Información de identificación (PID, PPID, UID y GID reales y efectivos, etc.).
- Estado del procesador: copia de los registros del procesador cuando el proceso no está en ejecución.
- Información de control del proceso:
 - o Información de planificación y estado
 - o Regiones de memoria asociadas al proceso
 - o Recursos asignados: descriptores de ficheros abiertos, puertos, temporizadores, etc.
 - o Información sobre señales: armado y máscara de señales
 - o Información de contabilidad
 - o Etc.

Ejercicio 3 (1.5 puntos)

¿Para qué sirve el servicio `alarm()`? ¿Cómo funciona? ¿Qué efecto tiene sobre el proceso que lo invoca? ¿Qué uso típico podemos hacer de él?

El servicio `alarm()` establece un temporizador en el proceso que lo invoca. Dicho temporizador provoca que, una vez transcurrido el tiempo establecido en la invocación del servicio, se envíe al proceso la señal `SIGALRM`. Dicha señal tiene asociado como comportamiento por defecto la muerte del proceso que la recibe, por lo que es una forma muy sencilla de establecer un tiempo máximo de ejecución para un proceso hijo que ejecute una tarea potencialmente larga: después del `fork()`, en el código del proceso hijo, se puede establecer el temporizador por el tiempo máximo deseado y, a continuación, lanzar la tarea que queramos que pueda ser finalizada si excede el tiempo máximo establecido.

Ejercicio 4 (6 puntos)

Una conocida empresa de Internet quiere desarrollar una "App" para que sus usuarios puedan retocar sus fotos antes de subirlas a la Web. Durante el diseño del programa los ingenieros dudan de si es más conveniente una implementación basada en procesos pesados o en *threads*. El diseño actual contempla que el programa reciba como argumentos el nombre del fichero que contiene la imagen y los nombres de los filtros que se quieren aplicar. Un ejemplo de

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

```
typedef unsigned char image_t [ImageSize];
image_t * readImage(char * fname);
int writeImge(char * fname, image_t * outputImage);
image_t * applyFilter(image_t * inputImage, char * filterName);
```

La primera de las funciones se encarga de abrir y leer la imagen cuyo nombre se pasa como argumento y devuelve un puntero a un `image_t` que contiene la información de la imagen. La segunda de las funciones recibe como primer argumento el nombre de la imagen original y el resultado de aplicar el último filtro y lo escribe en un fichero del mismo nombre que el original, pero con el añadido anteriormente mencionado. Finalmente, la tercera recibe como argumentos un `image_t` que representa la imagen y el nombre del filtro que se debe aplicar y devuelve un nuevo `image_t` con el resultado de aplicar el filtro sobre la imagen de entrada.

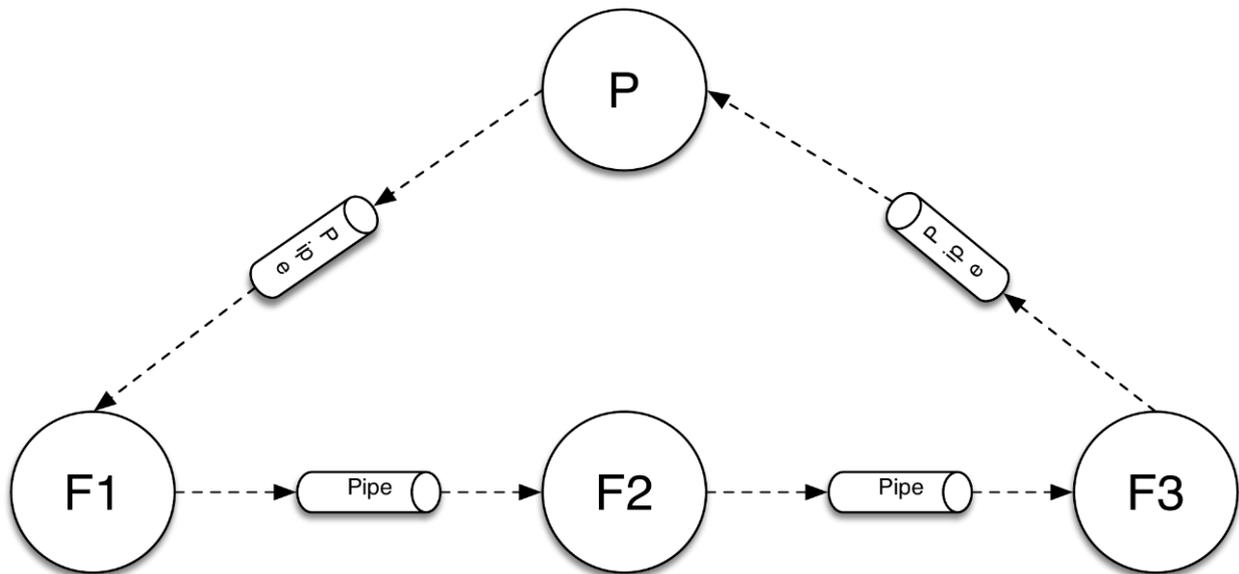
NOTAS:

- Para simplificar el problema, asumiremos que el tamaño de las imágenes es constante, tal y como se muestra en el que código que tenemos a continuación.
- Asimismo, no se tendrá en cuenta en esta implementación la necesidad de liberar los recursos reservados en las funciones auxiliares para almacenar la información de las imágenes.

Asumiendo que invocamos al programa de la siguiente forma:

```
./instamiligram pict20171031.jpg ludwig juno slumber
```

- a) Dibuje el diagrama de procesos en el momento en que todos los procesos están en ejecución. Dicho diagrama debe mostrar tanto la jerarquía de procesos como los mecanismos de comunicación entre los mismos.



El proceso padre crea las 4 tuberías necesarias para conectar los cuatro procesos que ejecutarán simultáneamente de acuerdo a la línea de ejecución de más arriba: el mismo proceso padre y los tres procesos hijos, uno por cada filtro solicitado. En esta jerarquía el proceso padre se encuentra al principio y al final de la secuencia de procesos encadenados, mientras que los procesos hijos ejecutan secuencialmente, uno tras otro.

- b) Indique en qué momento los procesos hijos terminarán su ejecución por completo y discuta en qué estado se encontrarían hasta ese momento.



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

d) Escriba una implementación equivalente utilizando hilos en vez de procesos.

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

#define ImageSize 1024*768
typedef unsigned char image_t [ImageSize];

image_t* readImage(char* fname);
int writeImage(char* fname, image_t* outputImage);
image_t* applyFilter(image_t* inputImage, char* filterName);

image_t *inputImage, *outputImage;

void* runTask(void *p) {
    inputImage=outputImage;

    outputImage=applyFilter(inputImage, (char*)p);
    return (void*)NULL;
}

int main(int argc, char*argv[]) {
    int nFiltros=argc-2;
    int i;

    pthread_t thid[nFiltros];

    pthread_attr_t attr;
    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);

    inputImage=readImage(argv[1]);
    printf("Imagen original: %s\n", *inputImage);

    for (i=0; i<nFiltros; i++) {
        int ret = pthread_create(&thid[i], &attr, &runTask, argv[i+2]);
        if (ret != 0) {perror("p_create()"); exit(1);}

        ret = pthread_join(thid[i], (void**)NULL);
        if (ret != 0) {perror("p_join()"); exit(1);}
    }

    pthread_attr_destroy(&attr);

    writeImage(argv[1], outputImage);

    return 0;
}
```

The logo for 'Cartagena99' features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the rest of the text. The logo is set against a light blue background with a white arrow pointing to the right, and a yellow and orange gradient bar at the bottom.

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

A continuación, se muestra la implementación basada en procesos pesados:

```
1: #include <stdlib.h>
2: #include <stdio.h>
3: #include <unistd.h>
4: #include <sys/wait.h>
5:
6: #define ImageSize 1024*768
7: typedef unsigned char image_t [ImageSize];
8: image_t * readImage(char * fname);
9: int writeImage(char * fname, image_t * outputImage);
10: image_t * applyFilter(image_t * inputImage, char * filterName);
11:
12: int main(int argc, char*argv[]) {
13:     int nFiltros=argc-2;
14:     int i;
15:
16:     int fd[(nFiltros+1)][2];
17:
18:     for (i=0; i<(nFiltros+1); i++)
19:         pipe(fd[i]);
20:
21:     for (i=0; i<nFiltros; i++) {
22:         if (fork()==0) {
23:             int j;
24:             for (j=0; j<(nFiltros+1); j++) {
25:                 if (j!=i)
26:                     close(fd[j][0]);
27:                 if(j!=(i+1))
28:                     close(fd[j][1]);
29:             }
30:
31:             image_t inputImage, *outputImage;
32:             read(fd[i][0], &inputImage, ImageSize);
33:             outputImage=applyFilter(&inputImage, argv[i+2]);
34:             write(fd[i+1][1], outputImage, ImageSize);
35:             exit(0);
36:         }
37:     }
38:
39:     for (i=0; i<(nFiltros+1); i++) {
40:         if (i!=nFiltros)
41:             close(fd[i][0]);
42:         if (i!=0)
43:             close(fd[i][1]);
44:     }
45:
46:     image_t * origImage=readImage(argv[1]);
47:     write(fd[0][1], origImage, ImageSize);
48:     image_t finalImage;
49:     read(fd[nFiltros][0], &finalImage, ImageSize);
50:
51:     writeImage(argv[1], &finalImage);
52:
53:     while (wait(NULL)>0) continue;
54: }
```

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the word 'Cartagena'. The text is set against a background of a light blue and white gradient with a subtle pattern.

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**