Examen de Sistemas Operativos.
20 de junio de 2007.

Nombre y apellidos, Grupo y NIA:

NOTAS:

* La fecha de publicación de las notas, así como de revisión se notificarán por Aula Global
* Para la realización del presente examen se dispondrá de **3 horas**
* **No** se pueden utilizar libros **ni** apuntes, ni usar móvil (o similar)
* Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

---

Ejercicio 1 (3 puntos)

Responda a las siguientes preguntas:

1) ¿Cuál es la diferencia entre enlace físico y enlace simbólico?

En el caso del enlace físico dos nombres apuntan al mismo i-nodo. El archivo sólo se elimina cuando se han borrado todos los enlaces. Este tipo de enlace sólo permite enlazar archivos.
En los enlaces simbólicos por su parte existen dos nombres y dos i-nodos que referencian al mismo fichero. El archivo se elimina cuando se borra el enlace físico.

2) Indique que mecanismo recomendaría para sincronizar dos procesos independientes que se ejecutan en una misma máquina.

Dos ejemplos válidos son la instrucción Test-and-set o la instrucción Swap

3) ¿Qué condiciones debe cumplir cualquier solución que se utilice para resolver el problema de la sección crítica?

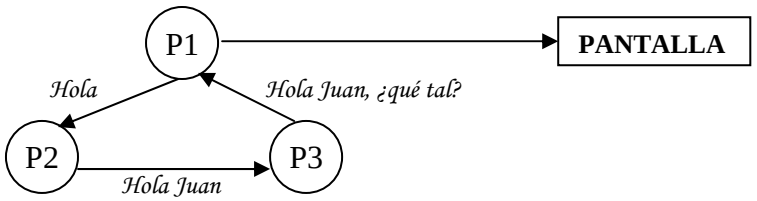
Exclusión muta,
progreso y
espera acotada.

Ejercicio 2 (3,5 puntos)

El siguiente diagrama de procesos muestra el flujo de datos entre los procesos P1, P2 y P3. El comportamiento de los procesos es el siguiente:

- El proceso P1 escribe una palabra *(p.ej. "Hola")* para el proceso P2.
- El proceso P2 lee dicha palabra y la concatena con otra palabra *(p.ej. "Juan").* El resultado de la concatenación lo escribe para el proceso P3.
- El proceso P3 lee el resultado *("Hola Juan")* y lo concatena con otra palabra. El resultado de la concatenación lo escribe para el proceso P1.
- El proceso P1 toma dicho resultado y lo imprime en pantalla.

Suponga que los datos enviados entre los procesos son los que aparecen etiquetando las flechas en el diagrama.



Se pide:

a) Enumere y justifique razonadamente qué recursos vistos en clase relacionados con procesos *pesados* podría utilizar para comunicar los procesos P1, P2 y P3.

b) Implementar en C el código que permita que el proceso P1 muestre en la pantalla la frase: *"Hola Juan, ¿qué tal?"* utilizando *pipes.*

NOTA: Tenga en cuenta el siguiente código de lectura y escritura en la tubería:

```
/* Lee del df STDIN_FILENO MAX bytes y los almacena en buffer */
read (STDIN_FILENO, buffer, MAX);

/* Escribe en el df STDOUT_FILENO MAX bytes de buffer */
write(STDOUT_FILENO, buffer, MAX);

/* Esta función concatena a la variable cadena lo contenido en
 * bufferResultado */
concatenar(char *bufferResultado, char *cadena);
```

```
#define       NUM_PROCESOS  3      /* Numero de procesos para crear */
#define       MAX           256    /* Máximo del buffer a leer */

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

int main()
{

       int PID;
       int i;

       int pipe1[2];
       int pipe2[2];
       int pipe3[2];

       char buffer[MAX];


       /* Rellene con las variables que necesite */
        int finfich;




       bzero(buffer, MAX);


       /* Rellene con la creación de tuberías */
       if (pipe(pipe1)==-1)
       {
              printf("Error al crear el pipe1\n");
              exit(-1);
       }
       if (pipe(pipe2)==-1)
       {
              printf("Error al crear el pipe2\n");
              exit(-1);
       }
       if (pipe(pipe3)==-1)
       {
              printf("Error al crear el pipe3\n");
              exit(-1);
       }
```

```
for (i=0;i<NUM_PROCESOS-1;i++)
{
        PID=fork();
        switch(PID)
        {
                case -1:
                        printf("Error en el fork\n");
                        exit(-1);
                case 0:
                {

                        if (i==0)
                        {

                                /* El proceso P2 cierra los
                                   descriptores que no va a usar */
                                close(pipe1[1]);
                                close(pipe2[0]);
                                close(pipe3[0]);
                                close(pipe3[1]);


                                /* El proceso P2 redirige
                                   su lectura al pipe1 */
                                close(STDIN_FILENO);
                                dup(pipe1[0]);
                                close(pipe1[0]);


                                /* El proceso P2 redirige
                                   su escritura al pipe2 */
                                close(STDOUT_FILENO);
                                dup(pipe2[1]);
                                close(pipe2[1]);


                                /* El proceso P2 añade “Juan”
                                   a todo lo que le llega por
                                   su entrada, y lo manda a su salida */
                                finfich=0;
                                while (!finfich) {
                                   finfich=read(STDIN_FILENO, buffer, MAX);
                                   if (finfich) {
                                          strcat(buffer, " Juan");
                                          write(STDOUT_FILENO, buffer, MAX);
                                          finfich=0 ;
                                   }
                                }

                                /* Finalizar el proceso si fin de
                                   fichero por la entrada */
                                exit(0);

                        }
```

```
                        if (i==1)
                        {
                                /* El proceso P3 cierra los
                                   descriptores que no va a usar */
                                close(pipe1[0]);
                                close(pipe1[1]);
                                close(pipe2[1]);
                                close(pipe3[0]);


                                /* El proceso P3 redirige
                                   su lectura al pipe2 */
                                close(STDIN_FILENO);
                                dup(pipe2[0]);
                                close(pipe2[0]);


                                /* El proceso P3 redirige
                                   su escritura al pipe3 */
                                close(STDOUT_FILENO);
                                dup(pipe3[1]);
                                close(pipe3[1]);


                                /* El proceso P3 añade ", ¿qué tal?"
                                   a todo lo que le llega por
                                   su entrada, y lo manda a su salida */
                                finfich=0;
                                while (!finfich) {
                                  read(STDIN_FILENO, buffer, MAX);
                                  strcat(buffer, ", ¿que tal?");
                                  write(STDOUT_FILENO, buffer, MAX);
                                }


                                /* Finalizar el proceso si fin de
                                   fichero por la entrada */
                                exit(0);

                        }


                }


        }/*switch*/
}/*for*/
```

```
        /* Enviar "Hola" al proceso P2,
           Recepción del mensaje de P3 e
           Impresión por la salida estándar */

        write (pipe1[1], "Hola", MAX) ;
        read  (pipe3[0], buffer, MAX);
        printf("::::PROCESO P1 -- Leo del pipe2:%s::::::\n",buffer);



        /* Cerrar los descriptores de no usados ya */

        close(pipe1[1]);
        close(pipe1[0]);
        close(pipe2[0]);
        close(pipe2[1]);
        close(pipe3[0]);
        close(pipe3[1]);



        /* Esperar por la finalización de los hijos */

        while(wait(&status)!=PID);
        exit(0);

} /* main */
```

Ejercicio 3 (3,5 puntos)

Se dispone de un disco de 2 megabytes de capacidad sin formatear (lo que suman todos los bloques se usen luego para lo que se usen). Se pretende realizar sobre él un formateo para que adopte el sistema de ficheros tipo UNIX descrito a continuación:

Sector de arranque	Superbloque	Mapas de bits	Lista de inodos	Bloques de datos

Tamaño de bloque: 512 bytes.
Tamaño de dirección de bloques: 2 bytes (permite direccionar hasta 65.536 bloques distintos).

Datos de control del sistema de ficheros:
- Sector de arranque: 1 bloque de tamaño
- Superbloque: 1 bloque.
- Mapa de bits: 1 bit por bloque direccionable en todo el disco físico
- Lista de inodos:
  - Nº de inodos: 500
  - Campos de un inodo:
    - U.I.D. y permisos: 10 bytes
    - Tamaño del fichero: 4 bytes
    - Fecha de última actualización: 4 bytes.
    - 4 entradas de referencia directa a bloques del fichero: 4 x 2 bytes.
    - 2 entradas de referencia indirecta de primer nivel: 2 x 2 bytes.
    - 1 entrada de referencia indirecta de segundo nivel: 2 bytes.

Nota: Los inodos deben poder identificarse mediante un número de bloque, aunque se desperdice espacio.

Se pide:
A) Calcular el tamaño máximo efectivo del disco, una vez formateado, que puede ser usado para datos de usuario.
B) Número máximo de directorios que pueden ser almacenados en este disco.
C) Tamaño máximo teórico de un fichero usando este formato de sistema de ficheros. (Sin contar con la limitación de la capacidad de este disco concreto).
D) Decir si es posible y cómo, la realización de enlaces duros y simbólicos en este sistema de ficheros. En caso contrario indicar que cambios serian necesarios para que fuera posible realizarlos.

NOTA: 1K = 1024 (algunos fabricantes últimamente cuando hablan de los GB multiplican por 1000).

a) (2 * 1024 * 1024) / (1024 / 2) = 4 K bloques
Los bloques usados por el sistema de ficheros son:

Arranque:	**1 bloque**
Superbloque	**1 bloque**
mapas de bits:	**4Kbits (tantos como bloques) dividido entre 8bits/byte * 512bytes/bloque 1 * 2 (recordar que hay dos mapas de bits, uno de bloques libres y otro de bloques ocupados) = 2 bloques**
i-nodos:	**un bloque por cada i-nodo 500 bloques**

Luego el sistema emplea 504 bloques y quedan libres 4*1024 -504 = 3592 bloques

b) Si todo el disco son ficheros de tipo directorio, tantos como i-nodos: 500

c) Depende de las referencias, que son:
4 bloques directos 4
2 indirectos, cada uno a un bloque donde caben 512/2 referencias a bloques 512
1 de dos niveles: 256 entradas a bloques con 256 referencias a bloques 64K
Sumando el número de bloques cada uno de ½ K salen 32M + 256K + 2K
d) Duros: No se pueden hacer enlaces duros, para hacerlos basta con que el mismo número de i-nodo aparezca en más de un directorio, y se necesita un campo contador de enlaces para poder llevar la cuenta del número de enlaces y saber cuándo liberar el i-nodo.
Simbólicos: Si se pueden hacer ya que basta con poner como contenido el path referenciado, y marcarlo en el inodo para que el sistema operativo lo trate de forma especial (y desreferencie el camino).