Sistemas Operativos
Grado en Ingeniería del Software
13 de junio de 2014, Curso 2013-2014, Examen Tipo A
Facultad de Informática, UCM

Apellidos, Nombre: _____

DNI: _____

Cuestiones

C1.- (1,25 p.) En un sistema tipo UNIX, un proceso ejecuta el siguiente programa, que pretende llevar a cabo la copia de un fichero utilizando dos procesos concurrentes:

```
1  #include <stdio.h>
2  ...
3
4  int copia(int fdO, int fdD, int nB){
5    int i;
6    char dato;
7
8    for(i=0;i<nB;i++){
9      if(read(fdO,&dato,1) !=1 ||
10       write(fdD,&dato,1) !=1 ){
11     perror("Error");
12     return(-1);
13   }
14   }
15   return(0);
16 }
17
18 int main(int argc, char *argv[]) {
19   int pid, fdO, fdD, mid, ret;
20   struct stat stBuf;
21
22   if( (fdO=open("datos.txt",0))==-1)
23     perror("Error origen"), exit(-1);
24   if( (fdD=open("copia.txt",O_WRONLY|
25       O_TRUNC|O_CREAT,0666))==-1)
26     perror("Error destino"), exit(-1);
27
28   stat("datos.txt", &stBuf);
29   mid=stBuf.st_size/2;
30
31   pid=fork();
32   switch(pid){
33     case -1:
34       perror("Error fork");exit(-1);
35     case 0:
36       lseek(fdO, mid, SEEK_SET);
37       ret=copia(fdO, fdD, stBuf.st_size-mid);
38       printf("Hijo: ");
39       break;
40     default:
41       lseek(fdO, 0, SEEK_SET);
42       ret=copia(fdO, fdD, mid);
43       wait(NULL);
44       printf("Padre: ");
45       break;
46   }
47   if(!ret)
48     printf("copia correcta\n");
49   else
50     printf("la copia ha fallado!!\n");
51   return(ret);
52 }
```

a) ¿Los ficheros `copia.txt` y `origen.txt` serán idénticos en todos los casos? ¿Por qué?

b) ¿Qué modificaciones mínimas propone para arreglar el código y que la copia siga siendo concurrente?

C2.- (0,75 p.) En la práctica 3, el hilo principal del programa hace uso de la función `system()` de la biblioteca estándar de "C".

```
int system(const char *command);
```

Esta función crea un proceso shell que invoca el comando que se pasa como parámetro de la función (p.ej.: `"ls -l"`). Todo proceso de usuario que invoque `system()` se quedará bloqueado hasta que el comando finalice.

Implemente la función `system()` empleando llamadas al sistema `fork()`, `exec*()` y `wait()`. (**Pista**: Para ejecutar un comando se puede invocar al shell de la siguiente forma: `/bin/bash -c '<comando>'`. Ejemplo: `/bin/bash -c 'ls -l'`).

C3.- (0,75 p.) Indique cuáles de las siguientes afirmaciones son verdaderas y cuáles falsas. En el caso de que considere que la respuesta es falsa, indique el motivo.

a) Las operaciones de entrada/salida son, desde el punto de vista de un proceso de usuario, siempre bloqueantes.

b) Un SO tipo UNIX utiliza el `major number` de un dispositivo para identificar al driver que lo gestiona.

c) El programador de un driver en LINUX es el que especifica la interfaz de llamadas para realizar las operaciones de entrada/salida.

d) Una vez cargado un módulo del kernel, éste no se puede descargar.

C4.- (0,75 p.) Un proceso en UNIX ejecuta el siguiente código:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  #define CONSTANT 10
5
6  int numA = CONSTANT;
7  int numB;
8  char string[] = "numA";
9
10 int main(int argc, char *argv[]) {
11   int *i=NULL;
12   if( (i=(int*)malloc((sizeof(int))) == NULL)
13     return(-1);
14   numB=argc;
15   for(*i=0;*i<CONSTANT;*i=*i+1){
16     fprintf(stdout, "%s: %d, argc: %d\n", string, numA--, numB);
17   }
18   return(0);
19 }
```

Para las siguientes variables/macros indique cuáles ocupan espacio en memoria. En caso afirmativo, indique también la región del mapa de memoria (código, datos con valor inicial, datos sin valor inicial, pila o *heap*) en la que se encuentran.

Variable/Macro	Ocupa espacio (Sí/No)	Región del mapa de memoria
CONSTANT		
numA		
numB		
i		
*i		

Problemas

P1.- (1,5 p.) Un sistema de ficheros tipo UNIX utiliza bloques de disco de 2K bytes. Para el direccionamiento de estos bloques se utilizan punteros de 32 bits. Para indicar el tamaño del fichero y el desplazamiento (offset) de la posición en bytes en las operaciones read y write, se utilizan números de 32 bits. Cada nodo-i tiene 8 punteros de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble.

a) ¿Cuál será el tamaño máximo de un fichero en este sistema suponiendo despreciable el espacio ocupado por el superbloque y la tabla de nodos-i?

b) Si el nodo-i incluyese además un puntero indirecto triple ¿Cuál sería el tamaño máximo de un fichero en este sistema ?

P2.- (2 p.) Considere un sistema monoprocesador con una política de planificación de procesos round-robin con quanto 3. Inicialmente, hay 3 procesos en la cola del planificador: P1, P2 y P3 (en este orden). Estos procesos se ejecutan de forma indefinida siguiendo los siguientes patrones de CPU y E/S:

P1 (3-CPU, 5-E/S), P2 (7-CPU, 3-E/S) y P3 (4-CPU, 4-E/S)

a) Dibujar un diagrama de tiempos en el que se muestre el estado de los procesos durante las 20 primeras unidades de tiempo. Para representar los estados de los procesos en el diagrama se ha de emplear la siguiente notación:

b) Partiendo del diagrama del apartado anterior, calcular el tiempo de espera de cada proceso y el porcentaje de utilización de la CPU durante las 20 primeras unidades de tiempo.

P3.- (2 p.) En un buffet se sirven pizzas de dos tipos: *pepperoni* (0) y *margarita* (1). Cuando un cliente desea comer una pizza de un determinado tipo ha de ir a recogerla al buffet. Si en ese instante quedan pizzas del tipo elegido, el cliente se servirá una y comerá. En otro caso, el cliente deberá avisar al camarero del buffet, que estará bloqueado en ese momento. Al ser despertado, el camarero mirará qué tipo de pizzas hay que reponer, repondrá N unidades y volverá a bloquearse. Entonces el cliente podrá servirse la pizza y comer.

Un número arbitrario de clientes y el camarero (hilos del programa concurrente) se comportan del siguiente modo:

```
void Cliente(int tipoPizza){
  while (true) {
    conseguirPizza(tipoPizza);
    comer();
  }
}
```

```
void Camarero(){
  while (true) {
    reponerPizzas();
  }
}
```

Implementar las funciones `conseguirPizza()` y `reponerPizzas()` empleando un mutex, variables condicionales y variables compartidas.