Sistemas Operativos
Grado en Ingeniería del Software
15 de septiembre 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:

```
1  #include <stdio.h>
2  #include <pthread.h>
3
4  #define N 1024
5  #define NHILOS 2
6  int A[N],B[N],C[N];
7
8  void* suma_vector(void* arg) {
9    int* begin=(int*) arg;
10   int end=(*begin)==0?N/2:N;
11   int i=(*begin);
12
13   for (;i<end;i++){
14     C[i]=A[i]+B[i];
15   }
16   return NULL;
17 }
18
19 void inicializa_vectores(void){ ... }
20
21 void imprime_resultado(void){ ... }
22
```

```
23 int main(void) {
24   pthread_t tid[NHILOS];
25   int i;
26   int start[NHILOS]={0,N/2};
27
28   inicializa_vectores();
29
30   for (i=0;i<NHILOS;i++){
31     pthread_create(&tid[i],NULL,suma_vector,
32           &start[i]);
33   }
34
35   printf("Hilos creados\n");
36
37   for (i=0;i<NHILOS;i++){
38     pthread_join(tid[i],NULL);
39   }
40
41   imprime_resultado();
42
43   return 0;
44 }
```

a) ¿Además de la cabecera, de qué secciones estará constituido el fichero ejecutable (ELF) de dicho programa?

b) Indique qué regiones están presentes en el mapa de memoria del proceso cuando el hilo principal del programa ejecuta la sentencia de la línea 35. Asumir que en este punto los hilos creados ya han comenzado a ejecutar la función `suma_vector()`.

C2.- (0,75 p.) Indique si son verdaderas o falsas las siguientes afirmaciones:

(I) Si un módulo del kernel hace una referencia a una zona de memoria no permitida, el kernel lo detectará y descargará el módulo automáticamente.

(II) Un módulo del kernel que desee realizar varias tareas concurrentemente puede invocar la llamada al sistema *fork()* para hacerlo.

(III) Siempre que un proceso de usuario desea hacer una operación de Entrada/Salida el sistema operativo bloquea a dicho proceso hasta que la operación ha finalizado.

(IV) Un usuario con privilegios de administrador puede descargar un módulo del kernel, aunque se encuentre en uso.

C3.- (1,25 p.) Responda de manera razonada a las siguientes cuestiones:

a) Describa qué son y cómo se implementan los enlaces simbólicos y físicos en un sistema de ficheros como el usado por Linux/UNIX (basado en nodos-i y tablas de punteros a bloques).

b) Se dispone de un fichero `/home/user/file.dat` en un sistema de ficheros tipo UNIX. ¿Qué modificaciones sufre el sistema de ficheros al crear un enlace simbólico y otro físico sobre este fichero?. ¿Qué ocurriría con cada enlace al borrar el fichero original?.

c) ¿Sería posible implementar los dos tipos de enlace en un sistema de ficheros tipo FAT?. En el caso de que no sea posible indique el motivo.

C4.- (1,25 p.) En un sistema tipo UNIX, un proceso ejecuta el siguiente programa, que pretende buscar el valor máximo en un vector de enteros utilizando dos procesos concurrentes:

```
1  #include <stdio.h>
2  #include <limits.h>
3
4  /* La función lee un vector del
5   * fichero cuya ruta se pasa como
6   * parámetro. Devuelve un puntero
7   * al vector y el # de elementos.
8  */
9  int* lee_vector(char* fichero,
10    int* n_elementos) { ... }
11
12 /* Devuelve el maximo del vector
13  en el rango [start_idx:end_idx) */
14 int busca_max(int* vector,
15    int start_idx,int end_idx) {
16  int max_value=-INT_MAX-1;
17  int i=0;
18
19  for (i=start_idx;i<end_idx;i++)
20   if (vector[i]>max_value)
21      max_value=vector[i];
22
23  return max_value;
24 }
25
26 int main(int argc, char *argv[]) {
27  int n_elem;
28  int* vector;
29  pid_t pid;
30  int max1=0,max2=0;
31
32  if (argc!=2) {
33   fprintf(stderr,"Uso: %s <fichero>
34      \n",argv[0]);
35   exit(1);
36  }
37
38 vector=lee_vector(argv[1],&n_elem);
39
40 if ((pid=fork())==0) {
41  /* Proceso hijo */
42    max1=busca_max(vector,0,n_elem/2);
43    exit(0);
44  }
45 else if (pid>0) {
46  /* Proceso padre */
47    max2=busca_max(vector,n_elem/2,
48       n_elem);
49 }
50 else {
51    fprintf(stderr,"Error en fork()\n");
52    exit(2);
53  }
54
55 /* Esperar al proceso hijo */
56 while(wait(NULL)!=pid) {}
57
58 if (max1>=max2)
59  printf("El máximo del vector es %d\n",max1);
60 else
61  printf("El máximo del vector es %d\n",max2);
62 }
63
64 free(vector);
65
66 return 0;
67 }
```

a) Explique razonadamente por qué este programa no funciona correctamente.

b) ¿Qué mecanismos de comunicación/sincronización del SO podrían utilizarse para conseguir que este programa multiproceso funcionara correctamente? (**Nota:** No es necesario proporcionar una versión corregida del código. Basta con describir una aproximación a la solución.)

Problemas

P1.- (1,5 p.) Un sistema de ficheros UNIX utiliza bloques de 2K bytes y direcciones de disco de 32 bits. Los nodos-i contienen 10 direcciones de disco para bloques de datos, una dirección de bloque índice indirecto simple y una dirección de bloque índice indirecto doble. Conteste de manera razonada a las siguientes cuestiones:

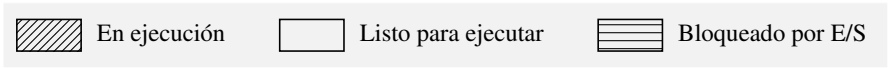
a) ¿Cuál será el tamaño máximo de un fichero en este sistema suponiendo despreciable el espacio ocupado por el superbloque y la tabla de nodos-i?

b) Un programa UNIX crea un fichero en este sistema e inmediatamente después escribe un byte de datos en la posición 32.000 y otro en la posición 2.000.000, habiéndose posicionado en dichos bytes con `lseek()`. Suponiendo que se usan punteros a NULL para representar los bloques de los huecos dejados por `lseek()`, ¿cuántos bloques de datos y de índices ocupa este nuevo fichero en disco?

P2.- (2 p.) Considere un sistema monoprocesador con una política de planificación de procesos round-robin con quanto 3. En este sistema, los trabajos descritos en la tabla entran al sistema en los instantes indicados. El desglose de los tiempos de los trabajos se refiere al tiempo requerido para ráfagas de uso de CPU y de E/S, alternativamente. Suponer que en el instante $t = 0$, P1 se encola antes que P3 en la cola del planificador.

Proceso	T. Llegada	CPU	E/S	CPU	E/S
P1	0	1	5	1	
P2	1	3	1	1	1
P3	0	5	4	1	
P4	3	3	2	1	1

a) Dibujar un diagrama de tiempos en el que se muestre el estado de los procesos hasta la finalización de todos ellos. 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 los tiempos de espera y ejecución (o retorno) de cada proceso, el porcentaje de utilización de la CPU y la productividad.

P3.- (2 p.) Proporcionar una solución al problema de los lectores/escritores de forma que sean los escritores los que tengan prioridad de acceso a la sección crítica. Un número arbitrario de hilos lectores y escritores se comportan del siguiente modo:

```
void Lector() {
   while(true) {
     EntraLector();
     <Sección Crítica del Lector>
     SaleLector();
   }
}
```

```
void Escritor() {
   while(true) {
     EntraEscritor();
     <Sección Crítica del Escritor>
     SaleEscritor();
   }
```

Implementar las funciones `EntraLector()`, `SaleLector()`, `EntraEscritor()` y `SaleEscritor()` empleando un mutex, dos variables condición y variables compartidas. Cada variable condición se usará para bloquear los hilos de un solo tipo, lectores o escritores.