**ATENCIÓN:**

* Lea atentamente todo el enunciado antes de comenzar a contestar.
* Dispone de **dos horas y media** para realizar la prueba.
* **No** se podrán **utilizar libros ni apuntes, ni calculadoras de ningún tipo**.
* **Los teléfonos móviles** deberán permanecer **desconectados** durante la prueba (apagados, no silenciados).
* **Solamente** se corregirán los ejercicios contestados **con bolígrafo**. Por favor no utilice lápiz.

NOMBRE:

APELLIDOS:

NIA:

**Ejercicio 1**: Codifique un programa que genere dos procesos hijos y envíe 100 números enteros (desde el 0 hasta el 100) a los dos procesos hijos **simultáneamente**. Estos procesos hijos mostrarán por pantalla un mensaje cuando el valor entero recibido sea múltiplo de 13, en el caso del primer hijo, y de 5, en el caso del segundo hijo. El padre una vez enviados los datos a los procesos hijos, debe esperar a su finalización.

**Ejercicio 2**: En un blog existen varios generadores de contenidos, al igual de numerosos visitantes que leen los contenidos del blog. Cuando los visitantes están viendo los contenidos del blog, ningún editor de contenidos puede incorporar nuevos contenidos en el blog hasta que no terminen todos los visitantes su acceso, debiéndose esperar el o los editores mientras tanto. Cuando un editor empieza a editar nuevos contenidos, ningún otro editor o visitante puede acceder al blog.

Programe este problema usando procesos ligeros de POSIX suponiendo que existen N procesos visitantes que visitan el blog 3 veces cada uno y M procesos editores que han de editar el blog 3 veces cada uno. Las funciones para acceder al blog son:

* **editar()** en el caso de los editores.
* **leer()** en el caso de los visitantes del blog.

**Ejercicio 3**: Se tiene un sistema de ficheros tipo Unix con la siguiente información:

**Tabla de I-nodos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Inodo** | 1 | 2 | 3 |  |  |
| **Tipo** | Directorio | Directorio | Fichero |  |  |
| **Contador Enlaces Fis.** | 3 | 2 | 1 |  |  |
| **Dirección Bloque Datos** | 11 | 12 | 13 |  |  |
| ……. |  |  |  |  |  |

**Bloques de datos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Bloque** | 11 | 12 | 13 |  |  |
| **Contenido** | . 1 | . 2 |  |  |  |
| .. 1 | .. 1 | Datos del |  |  |
| d 2 | f1 3 | Fichero f1 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Indica cómo quedan los i-nodos y los bloques de datos después de realizar cada una de las siguientes operaciones (Hacer una tabla de i-nodos y de bloques de datos por cada apartado):**

1. mkdir /d/d1
2. ln -s /d/f1 /d/f2
3. rm /d/f1
4. ln -s /d/f 1/d/f3

**SOLUCIÓN**

**Ejercicio 1:**

**#include<stdio.h>**

**#include <unistd.h>**

**#include <stdlib.h>**

**int p1[2],p2[2];**

**void proceso1(){**

**int entero;**

**close(0);**

**dup(p1[0]);**

**close(p1[0]);**

**close(p1[1]);**

**while(read(0, &entero, sizeof(int)) !=0){**

**if (entero%13 == 0)**

**printf("multiplo de 13 = %d\n",entero);**

**}**

**printf("Fin multiplo de 13\n");**

**exit(0);**

**}**

**void proceso2(){**

**int entero;**

**close(0);**

**dup(p2[0]);**

**close(p2[0]);**

**close(p2[1]);**

**while(read(0, &entero, sizeof(int)) != 0 ){**

**if (entero%5 == 0)**

**printf("multiplo de 5 = %d\n",entero);**

**}**

**printf("Fin multiplo de 5\n");**

**exit(0);**

**}**

**main()**

**{**

**int ret, pid;**

**int i;**

**pipe(p1);**

**pid = fork();**

**switch(pid){**

**case 0:**

**proceso1();**

**break;**

**case -1:**

**perror("error\n");**

**break;**

**default:**

**pipe(p2);**

**pid = fork();**

**switch(pid){**

**case 0:**

**close(p1[0]);**

**close(p1[1]);**

**proceso2();**

**break;**

**case -1:**

**perror("error\n");**

**default:**

**close(1);**

**dup(p1[1]);**

**close(2);**

**dup(p2[1]);**

**close(p1[0]);**

**close(p1[1]);**

**close(p2[0]);**

**close(p2[1]);**

**for(i=0;i<100;i++){**

**write(1, &i, sizeof(int));**

**write(2, &i, sizeof(int));**

**}**

**close(1);**

**close(2);**

**}**

**}**

**while( -1!=wait(&ret));**

**exit(0);**

**}**

**Ejercicio 2:**

**#include <stdio.h>**

**#include <pthread.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#define N 3**

**#define M 5**

**#define VECES\_L 3**

**#define VECES\_E 3**

**pthread\_t th\_l[N];**

**pthread\_t th\_e[M];**

**int n\_e=0, n\_l = 0;**

**pthread\_mutex\_t m;**

**pthread\_cond\_t c;**

**void leer(int id){**

**printf("[%d]leyendo\n",id);**

**sleep(rand()%3);**

**printf("[%d]fin leyendo\n",id);**

**}**

**void editar(int id){**

**printf("[%d]editando\n",id);**

**sleep(rand()%2);**

**printf("[%d]fin editando\n",id);**

**}**

**void\* func\_editor(void \*arg){**

**int id = (int)arg;**

**int i;**

**for (i=0;i<VECES\_E;i++){**

**pthread\_mutex\_lock(&m);**

**while((n\_l > 0)||(n\_e > 0))**

**pthread\_cond\_wait(&c, &m);**

**n\_e++;**

**editar(id);**

**n\_e--;**

**pthread\_cond\_broadcast(&c);**

**pthread\_mutex\_unlock(&m);**

**}**

**pthread\_exit(NULL);**

**}**

**void\* func\_lector(void \*arg){**

**int id = (int)arg;**

**int i;**

**for (i=0;i<VECES\_L;i++){**

**pthread\_mutex\_lock(&m);**

**while(n\_e > 0)**

**pthread\_cond\_wait(&c, &m);**

**n\_l++;**

**pthread\_mutex\_unlock(&m);**

**leer(id);**

**pthread\_mutex\_lock(&m);**

**n\_l--;**

**pthread\_cond\_broadcast(&c);**

**pthread\_mutex\_unlock(&m);**

**}**

**pthread\_exit(NULL);**

**}**

**int main(){**

**int i;**

**srand ( time(NULL) );**

**pthread\_mutex\_init (&m, NULL);**

**pthread\_cond\_init (&c, NULL);**

**for (i=0;i<N;i++)**

**pthread\_create( &th\_l[i], NULL, func\_lector, (void\*) i);**

**for (i=0;i<M;i++)**

**pthread\_create( &th\_e[i], NULL, func\_editor, (void\*) i);**

**for (i=0;i<N;i++)**

**pthread\_join(th\_l[i], NULL);**

**for (i=0;i<M;i++)**

**pthread\_join(th\_e[i], NULL);**

**}**

**Ejercicio 3:**

1. mkdir /d/d1

**Tabla de I-nodos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Inodo** | 1 | 2 | 3 | 4 |  |
| **Tipo** | Directorio | Directorio | Fichero | Directorio |  |
| **Contador Enlaces Fis.** | 3 | 3 | 1 | 2 |  |
| **Dirección Bloque Datos** | 11 | 12 | 13 | 14 |  |
| ……. |  |  |  |  |  |

**Bloques de datos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Bloque** | 11 | 12 | 13 | 14 |  |
| **Contenido** | . 1 | . 2 |  | . 4 |  |
| .. 1 | .. 1 | Datos del | .. 2 |  |
| d 2 | f1 3 | Fichero f1 |  |  |
|  | d1 4 |  |  |  |
|  |  |  |  |  |

1. ln -s /d/f1 /d/f2

**Tabla de I-nodos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Inodo** | 1 | 2 | 3 | 4 | 5 |
| **Tipo** | Directorio | Directorio | Fichero | Directorio | Simbólico |
| **Contador Enlaces Fis.** | 3 | 3 | 1 | 2 | 1 |
| **Dirección Bloque Datos** | 11 | 12 | 13 | 14 | 15 |
| ……. |  |  |  |  |  |

**Bloques de datos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Bloque** | 11 | 12 | 13 | 14 | 15 |
| **Contenido** | . 1 | . 2 |  | . 4 | /d/f1 |
| .. 1 | .. 1 | Datos del | .. 2 |
| d 2 | f1 3 | Fichero f1 |  |
|  | d1 4 |  |  |
|  |  |  |  |

1. rm /d/f1

**Tabla de I-nodos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Inodo** | 1 | 2 |  | 4 | 5 |
| **Tipo** | Directorio | Directorio |  | Directorio | Simbólico |
| **Contador Enlaces Fis.** | 3 | 3 |  | 2 | 1 |
| **Dirección Bloque Datos** | 11 | 12 |  | 14 | 15 |
| ……. |  |  |  |  |  |

**Bloques de datos:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nº Bloque** | 11 | 12 | 13 | 14 | 15 |
| **Contenido** | . 1 | . 2 |  | . 4 | /d/f1 |
| .. 1 | .. 1 | Libre | .. 2 |
| d 2 | d1 4 |  |  |
|  |  |  |  |
|  |  |  |  |

1. ln /d/f 1 /d/f3

No es posible, ya que no se puede realizar un enlace duro a un fichero eliminado.