NOMBRE Y APELLIDOS: _____ NIA: _____

SOLUCIÓN Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

NOTAS:
* Para la realización del presente examen se dispondrá de **30 minutos.**
* **No** se pueden utilizar libros **ni** apuntes, ni usar móvil (o similar).
* Marque en la tabla la respuesta correcta a cada pregunta.
* Cada pregunta bien contestada vale 1 punto. Cada error resta 0,25 puntos.

PREGUNTAS DE TEST (2 puntos)

Pregunta	1	2	3	4	5	6	7	8	9	10
Respuesta										

PREGUNTA 1. En qué situaciones se puede producir un cambio de contexto en un sistema con un planificador basado en el algoritmo SJF (Shortest Job First) además de cuando se bloquea o se termina el proceso?
A.- Cuando se pone en estado de listo un proceso con mayor prioridad.
B.- En ninguna más.
C.- Cuando se pone en estado de listo un proceso cuya próxima racha tiene una duración prevista menor que la del proceso que está ejecutando.
D.- Cuando se termina el cuanto del proceso.

PREGUNTA 2. ¿Qué significa poner un proceso en background?
A.- Cuando su padre muere sin hacer WAIT.
B.- Conectarlo con el padre en un PIPE y no esperar.
C.- Cuando es heredado por el proceso INIT.
D.- Cuando el padre ejecuta el proceso y no espera en un WAIT.

PREGUNTA 3. El fichero pepe tiene los permisos rwxr-xr-x. ¿Qué mandato debería usarse para que el fichero sólo pueda ser leído y ejecutado por el propietario y los miembros de su grupo y solo leído por el resto del mundo?
A.- chmod 766 pepe
B.- chmod 554 pepe
C.- chgrp +rx pepe
D.- chmod 556 pepe

PREGUNTA 4. El sistema operativo Windows NT está diseñado como un sistema:
A.- De máquinas virtuales.
B.- Monolítico.
C.- Jerárquico por capas.
D.- Cliente-Servidor.

PREGUNTA 5. Sea un programa concurrente con cuatro procesos iguales cuyo código consiste tan solo en incrementar en uno una variable V compartida entre ellos y protegida por un semáforo. ¿Cuál de las siguientes opciones podría ser correcta respecto al posible valor resultante de V después de la ejecución concurrente de los tres procesos si V vale 0 inicialmente?

```
wait(sem);
V++;
signal(sem);
```

A.- V tiene valor 1,5.
B.- V tiene valor 4.
C.- V tiene valor 3.
D.- V tiene valor 0.

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

PREGUNTA 6. ¿Qué caracteriza más a un Sistema Operativo?
A.- Servicios y llamadas al sistema.
B.- El intérprete de comandos.
C.- El shell.
D.- Los comandos.

PREGUNTA 7. La llamada al sistema "stat(…); " permite obtener:
A.- El estado del proceso que la ejecuta.
B.- Información del superbloque del sistema de ficheros indicado.
C.- Información del nodo-i del fichero indicado.
D.- El estado de todos los procesos que estén ejecutando en el sistema.

PREGUNTA 8. ¿Qué información, entre otras, comparten un proceso A y su hijo B después de ejecutar el siguiente código?

```
Proceso A
        if (fork()!=0)
                wait (&status);
        else
                execve (B, parámetros, 0);
```

A.- Segmento de pila.
B.- Descriptores de ficheros abiertos.
C.- Segmento de texto.
D.- Segmento de datos.

PREGUNTA 9. El proceso A abre el fichero F1 (descriptor fd1) de 19.000 bytes. Seguidamente hace un fork, generando el proceso hijo B. Suponiendo que el proceso A ejecuta la llamada

```
#define BEGIN 0
lseek(fd1,805,BEGIN);
```

y que, posteriormente, el proceso B ejecuta la llamada

```
#define BEGIN 0
lseek(fd1,975,BEGIN);
```

¿Cuál es el valor del puntero del fichero después de ejecutarse el segundo lseek?
A.- El puntero del fichero fd1 queda apuntando a la posición 805.
B.- El puntero del fichero fd1 queda apuntando a la posición 1.780.
C.- El puntero del fichero fd1 queda apuntando a la posición 975.
D.- El puntero del fichero fd1 del proceso A queda apuntando a la posición 805 y el puntero del fichero fd1 del proceso B queda apuntando a la posición 936.

PREGUNTA 10. Supóngase que /etc/bin/enlace es un enlace (link) simbólico que apunta a /usr/bin/cp de nodo-i 74 del dispositivo /dev/hd3 y que este último fichero no tiene ningún enlace real adicional. ¿Qué es cierto?
A.- Al borrar /etc/bin/enlace se decrementa el contador de enlaces del nodo-i 74 de /dev/hd3.
B.- Si se borra /usr/bin/cp, se puede seguir accediendo al fichero a través del nombre /etc/bin/enlace.
C.- Al borrar /usr/bin/cp se borra el fichero realmente y ya no se puede acceder al fichero utilizando el nombre /etc/bin/enlace,
D.- Aunque se desmonte /dev/hd3 se puede seguir accediendo al fichero a través del nombre /etc/bin/enlace.

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

NOTAS:
* Para la realización del presente examen se dispondrá de **2 horas y 30 minutos.**
* **No** se pueden utilizar libros **ni** apuntes, ni usar móvil (o similar).
* Responda cada ejercicio en hojas distintas.

EJERCICIO 1 (2,5 puntos)

a) Codificar un programa en lenguaje C que lea números de teclado y muestre la suma de los números introducidos. Se deben seguir las siguientes pautas:

- Debe haber **3 procesos**. Un proceso padre (P), un proceso hijo (H) y un proceso nieto (N).
- El proceso padre (P) crea el proceso hijo y espera a su finalización.
- El proceso hijo (H) crea el proceso nieto (N). Además, lee números de teclado y los envía a un **pipe**. Cuando se lea el número 0 de teclado, enviará una señal SIGUSR1 al proceso nieto (N) para que termine su ejecución.
- El proceso nieto (N) recibe los números del pipe y los va sumando en una variable. Cuando recibe la señal SIGUSR1, debe mostrar por pantalla el resultado de la suma y terminar su ejecución.

b) Indique el cambio o cambios que habría que añadir para que la ejecución de la lectura de números y la suma se realice en background.

c) ¿Podría el proceso hijo (H) enviar la señal SIGKILL para indicar al proceso nieto (N) que debe mostrar el resultado de la suma y terminar? Razone la respuesta.

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

EJERCICIO 2 (2,5 puntos)

Se desea realizar un programa que simule las colas de renovación del dni. Existe una zona en la que esperan los clientes hasta que exista una ventanilla libre. En ese momento pasan a la zona de atención y buscan cual de las ventanillas está libre para pasar a ser atendido en ella.

Se crea un thread por cada persona que llega y desea ser atendida, existiendo un tiempo de espera entre la creación de cada thread de entre 1 y 3 segundos elegido aleatoriamente.

Se crea también un thread por cada ventanilla abierta. El número de ventanillas será de 3.

El alumno no se preocupará por controlar que el primero que llega es el primero en ser atendidos. Simplemente se debe controlar que una persona sólo pasa a la zona de atención cuando hay alguna ventanilla libre y que después se busca la ventanilla libre en la que le atienden.

Cuando una persona pasa a la zona de atención buscará una ventanilla libre y será atendida en ella durante un tiempo aleatorio entre 2 y 7 segundos. Después el thread correspondiente a la persona finalizará y la ventanilla quedará libre para atender a otro cliente.

El alumno puede optar por realizar la aplicación por completo o añadir los mecanismos de control adecuados al siguiente código:

```
#define MAXPERSONAS        10
#define VENTANILLAS        3
pthread_mutex_t mtxparametro;     /* mutex para controlar el acceso al parámetro*/
pthread_cond_t esperaparametro;   /* controla que se haya tomado el valor del parámetro*/ int
transferido=FALSE;

pthread_mutex_t mutex; /* mutex para controlar el acceso a la zona de atención*/
pthread_mutex_t mutexventanilla; /* mutex para controlar el acceso a las ventanillas */
pthread_cond_t ventanillaespera[VENTANILLAS];   /* controla la espera del ventanilla*/
pthread_cond_t atencion;   /* controla la espera para acceder a la zona previa de atención*/
int ventlibres=VENTANILLAS;
int atendido =0;
int vventanillas[VENTANILLAS];   // una posición por ventanilla (0 indica libre y 1 ocupada)

void  *ventanilla(void *p)  {
  int i,segatencion,*aux,numventanilla;
```

-1-

```
/* ALMACENAR EL PARÁMETRO P EN LA VARIABLE LOCAL numventanilla */

  while (1) {
```

-2-

```
/*ESPERAR QUE LLEGUE UN CLIENTE A LA VENTANILLA vventanillas[numventanilla]==1 */


   segatencion=random () %6 + 2;  //simulo el tiempo de atención
   sleep (segatencion);
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

-3-

```
/*INDICAR QUE LA VENTANILLA ESTA LIBRE vventanillas[numventanilla]=0 Y ventlibres++ */
  }
  pthread_exit(0);
}

void  *persona(void *p)  {
  int i,*aux,usuario;
```

-4-

```
/* ALMACENAR EL PARÁMETRO P EN LA VARIABLE LOCAL usuario */
```

-5-

```
/* ESPERAR QUE HAYA VENTANILLAS LIBRES ventlibres > 0 */

printf ("Persona %d ha pasado a la zona de atención \n",usuario);
```

-6-

```
/* BUSCAR UNA VENTANILLA LIBRE ( CASILLA EN EL ARRAY vventanillas  CON VALOR 0 Y ACTIVAR ESA
VENTANILLA*/

  printf ("La persona %d ha sido atendida en la ventanilla %d\n", usuario, i);

  pthread_exit(0);
}

main(int argc, char *argv[]){
   int i,segllegadapersonas;
   pthread_t thp[MAXPERSONAS], thv[VENTANILLAS];
   pthread_attr_t attrpersona,attrventanilla;


   pthread_mutex_init(&mtxparametro, NULL);
   pthread_cond_init(&esperaparametro, NULL);

   pthread_mutex_init(&mutex, NULL);
   pthread_cond_init(&atencion, NULL);
   for (i=0;i<VENTANILLAS ;i++){
    vventanillas[i]=0;  // todas las ventanillas están libres
    pthread_cond_init(&ventanillaespera[i], NULL);
   }

   pthread_attr_init(&attrventanilla);
   pthread_attr_setdetachstate( &attrventanilla, PTHREAD_CREATE_DETACHED);
   for (i=0;i<VENTANILLAS;i++){
    pthread_mutex_lock (&mtxparametro);
    pthread_create(&thv[i], &attrventanilla, ventanilla, &i);
    while ( ! transferido)
      pthread_cond_wait(&esperaparametro, &mtxparametro); /* se bloquea */
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

```
    pthread_mutex_unlock (&mtxparametro);
    transferido=FALSE;
    pthread_mutex_unlock (&mtxparametro);
  }
printf ("Creadas las ventanillas\n");
  pthread_attr_init(&attrpersona);
  pthread_attr_setdetachstate( &attrpersona, PTHREAD_CREATE_DETACHED);
  srandom (getpid() );
  for (i=0;i<MAXPERSONAS;i++){
    segllegadapersonas=random () %3 + 1;
    sleep (segllegadapersonas);
    pthread_create(&thp[i], &attrpersona, persona, &i);
    pthread_mutex_lock (&mtxparametro);
    while ( ! transferido)
      pthread_cond_wait(&esperaparametro, &mtxparametro); /* se bloquea */
    pthread_mutex_unlock (&mtxparametro);
  }

  pause ();
  pthread_mutex_destroy(&mutex);
  pthread_cond_destroy(&atencion);
  for (i=0;i<VENTANILLAS ;i++){
    pthread_cond_destroy(&ventanillaespera[i]);
  }

  exit(0);
}
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

EJERCICIO 3 (3 puntos)

a) 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	Datos del Fichero f1		
	.. 1	.. 1			
	d 2	f1 3			

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 /d1

2- ln –s /d/f1 /d1/f2

3- rm /d/f1

b) Realizar un programa en C que lea de teclado el nombre de un directorio y muestre en pantalla el nombre y el tamaño de los ficheros que contiene.

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

SOLUCIÓN PREGUNTAS DE TEST (2 puntos)

Pregunta	1	2	3	4	5	6	7	8	9	10
Respuesta	*B*	*D*	*B*	*D*	*B o C*	*A*	*C*	*B*	*C*	*C*

SOLUCIÓN EJERCICIO 1 (2,5 puntos)

a) Apartado a.

```
int total=0;

void ImprimirResultado()
{
    printf("El total de los números es %d\n",total);
    exit(0);
}

int main()
{
    int fd[2];  //Descriptores para el pipe
    int pidHijo;
    int pidNieto;
    int num;
    int estado;
    struct sigaction a;

    pidHijo=fork();
    if(pidHijo!=0)  // Proceso padre P
    {
        wait(&estado);
    }
    else
    {
        pipe(fd);
        pidNieto=fork();
        if(pidNieto!=0)  // Proceso Hijo H
        {
            scanf("%d",&num);
            while(num!=0)
            {
                write(fd[1],&num,sizeof(num));
                scanf("%d",&num);
            }
            kill(pid,SIGUSR1);
            wait(&estado);
        }
        else
        {
            a.sa_handler=ImprimirResultado;
            a.sa_flags=0;
            sigaction(SIGUSR1,&a,NULL);
            while(1)
            {
                read(fd[0],&num,sizeof(num));
                total+=num;
            }
        }
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

```
    }
    return 0;
}
```

b) Apartado b.

```
int total=0;

void ImprimirResultado()
{
    printf("El total de los números es %d\n",total);
    exit(0);
}

int main()
{
    int fd[2];  //Descriptores para el pipe
    int pidHijo;
    int pidNieto;
    int num;
    int estado;
    struct sigaction a;

    pidHijo=fork();
    if(pidHijo!=0)  // Proceso padre P
    {
        wait(&estado); //En background el padre no espera por la finalización del hijo
    }
    else
    {
        pipe(fd);
        pidNieto=fork();
        if(pidNieto!=0)  // Proceso Hijo H
        {
            scanf("%d",&num);
            while(num!=0)
            {
                write(fd[1],&num,sizeof(num));
                scanf("%d",&num);
            }
            kill(pid,SIGUSR1);
            wait(&estado);
        }
        else
        {
            a.sa_handler=ImprimirResultado;
            a.sa_flags=0;
            sigaction(SIGUSR1,&a,NULL);
            while(1)
            {
                read(fd[0],&num,sizeof(num));
                total+=num;
            }
        }

    }
    return 0;
}
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

---

c) Apartado c.

Si el proceso hijo (H) envía SIGKILL al proceso nieto (N), N terminaría inmediatamente sin dar tiempo a que se muestre el resultado de la suma por pantalla. Además, por restricciones del sistema de señales, N no se puede armar para recibir la señal SIGKILL y por tanto tampoco podría cambiar el comportamiento por defecto. Por tanto, la respuesta es NO PODRÍA.

SOLUCIÓN EJERCICIO 2 (2,5 puntos)

```
#define MAXPERSONAS        10
#define VENTANILLAS        3
pthread_mutex_t mtxparametro;     /* mutex para controlar el acceso al parámetro*/
pthread_cond_t esperaparametro;   /* controla que se haya tomado el valor del parámetro*/ int
transferido=FALSE;

pthread_mutex_t mutex; /* mutex para controlar el acceso a la zona de atención*/
pthread_mutex_t mutexventanilla; /* mutex para controlar el acceso a las ventanillas */
pthread_cond_t ventanillaespera[VENTANILLAS];   /* controla la espera del ventanilla*/
pthread_cond_t atencion;   /* controla la espera para acceder a la zona previa de atención*/
int ventlibres=VENTANILLAS;
int atendido =0;
int vventanillas[VENTANILLAS];   // una posición por ventanilla 0 indica libre y 1 ocupada

void  *ventanilla(void *p)  {
  int i,segatencion,*aux,numventanilla;

  pthread_mutex_lock(&mtxparametro);       /* acceder al parámetro */
    aux=p;
    numventanilla=*aux;
    transferido=1;
    pthread_cond_signal(&esperaparametro);
  pthread_mutex_unlock(&mtxparametro);

  while (1) {
    pthread_mutex_lock(&mutexventanilla); /* para saber si hay un cliente o no en la ventanilla*/
      while (vventanillas[numventanilla] == 0) //ventanilla libre, espero que llegue una persona
        pthread_cond_wait(&ventanillaespera[numventanilla],&mutexventanilla);
    pthread_mutex_unlock (&mutexventanilla);

    segatencion=random () %6 + 2;  //simulo el tiempo de atención
    sleep (segatencion);

    pthread_mutex_lock (&mutexventanilla);  //indico que la ventanilla ya está libre
      vventanillas[numventanilla]=0;
    pthread_mutex_unlock (&mutexventanilla);
    pthread_mutex_lock (&mutex); // aumento el número de ventanillas libres
      ventlibres++;
      pthread_cond_signal (&atencion);   // despierto un cliente
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

```
    pthread_mutex_unlock (&mutex);
  }
  pthread_exit(0);
}

void  *persona(void *p)  {
  int i,*aux,usuario;

  pthread_mutex_lock(&mtxparametro);       /* acceder al parámetro */
    aux=p;
    usuario=*aux;
    transferido=TRUE;
    pthread_cond_signal(&esperaparametro);
  pthread_mutex_unlock(&mtxparametro);
printf ("Creada persona %d \n",usuario);

  pthread_mutex_lock(&mutex);  /* acceder al número de ventanillas libres */
    while (ventlibres ==0 )
      pthread_cond_wait(&atencion, &mutex); /* se bloquea */
    ventlibres--;
  pthread_mutex_unlock(&mutex); //libero el acceso a la zona de atención previa
printf ("Persona %d ha pasado a la zona de atención \n",usuario);

  // Busco una ventanilla libre (un valor 0 es que está libre)
  i=0;
  pthread_mutex_lock (&mutexventanilla);
    while ( vventanillas[i] == 1 && i<VENTANILLAS) i++;
    vventanillas[i]=1;
    pthread_cond_signal(&ventanillaespera[i]);  //despierto a la ventanilla elegida
  pthread_mutex_unlock (&mutexventanilla);

  printf ("La persona %d ha sido atendida en la ventanilla %d\n", usuario, i);

  pthread_exit(0);
}

main(int argc, char *argv[]){
   int i,segllegadapersonas;
   pthread_t thp[MAXPERSONAS], thv[VENTANILLAS];
   pthread_attr_t attrpersona,attrventanilla;


   pthread_mutex_init(&mtxparametro, NULL);
   pthread_cond_init(&esperaparametro, NULL);

   pthread_mutex_init(&mutex, NULL);
   pthread_cond_init(&atencion, NULL);
   for (i=0;i<VENTANILLAS ;i++){
     vventanillas[i]=0;  // todas las ventanillas están libres
     pthread_cond_init(&ventanillaespera[i], NULL);
   }
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

```
   pthread_attr_init(&attrventanilla);
   pthread_attr_setdetachstate( &attrventanilla, PTHREAD_CREATE_DETACHED);
   for (i=0;i<VENTANILLAS;i++){
    pthread_mutex_lock (&mtxparametro);
    pthread_create(&thv[i], &attrventanilla, ventanilla, &i);
    while ( ! transferido)
      pthread_cond_wait(&esperaparametro, &mtxparametro); /* se bloquea */
    pthread_mutex_unlock (&mtxparametro);
    transferido=FALSE;
    pthread_mutex_unlock (&mtxparametro);
   }
printf ("Creadas las ventanillas\n");
   pthread_attr_init(&attrpersona);
   pthread_attr_setdetachstate( &attrpersona, PTHREAD_CREATE_DETACHED);
   srandom (getpid() );
   for (i=0;i<MAXPERSONAS;i++){
    segllegadapersonas=random () %3 + 1;
    sleep (segllegadapersonas);
    pthread_create(&thp[i], &attrpersona, persona, &i);
    pthread_mutex_lock (&mtxparametro);
    while ( ! transferido)
      pthread_cond_wait(&esperaparametro, &mtxparametro); /* se bloquea */
    pthread_mutex_unlock (&mtxparametro);
   }

   pause ();
   pthread_mutex_destroy(&mutex);
   pthread_cond_destroy(&atencion);
   for (i=0;i<VENTANILLAS ;i++){
    pthread_cond_destroy(&ventanillaespera[i]);
   }

   exit(0);
}
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

SOLUCIÓN EJERCICIO 3 (3 puntos)

a) Tablas de I-nodos y Bloques de datos.

Tabla de I-nodos después de mkdir /d1**:**

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

Bloques de datos después de mkdir /d1**:**

Nº Bloque	11	12	13	14
Contenido	. 1	. 2	Datos del fichero	. 4
	.. 1	.. 1		. . 1
	d 2	f1 3		
	d1 4			

Tabla de I-nodos después de ln –s /d/f1 /d1/f2**:**

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

Bloques de datos después de ln –s /d/f1 /d1/f2**:**

Nº Bloque	11	12	13	14	15
Contenido	. 1	. 2	Datos del fichero	. 4	/d/f1
	.. 1	.. 1		. . 1	
	d 2	f1 3		f2 5	
	d1 4				

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen Extraordinario de Sistemas Operativos - 24 de Junio de 2010

Tabla de I-nodos después de rm /d/f1 **:**

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

Bloques de datos después de rm /d/f1**:**

Nº Bloque	11	12	13	14	15
Contenido	. 1	. 2	Bloque de datos Libre	. 4	/d/f1
	.. 1	.. 1		. . 1	
	d 2	~~f1 3~~		f2 5	
	d1 4				

b) Apartado b.

```
int main(){
DIR *d;
 char nomdir[90], nomfich[90];
struct stat datos;
 struct dirent *direc;

printf ("Introduzca el Nombre de un Directorio: ");
  fgets (nomdir,sizeof(nomdir),stdin);
nomdir[strlen(nomdir)-1]='\0'; /*Eliminamos el \n del Nombre del Fichero*/
  if ((d=opendir(nomdir))==NULL){
   printf ("El directorio no existe\n");
   return -1;
  }
   while ((direc=readdir(d)) !=NULL)   {
    strcpy(nomfich, nomdir);
    strcat(nomfich, "/");
    strcat(nomfich, direc->d_name );
    stat (nomfich, &datos);
    if (S_ISREG(datos.st_mode))
        printf ("Nombre: %s\t| Tamaño: %d\n",direc->d_name,datos.st_size);
   }/*Fin del While*/
   closedir(d);
}/*Fin del Main */
```