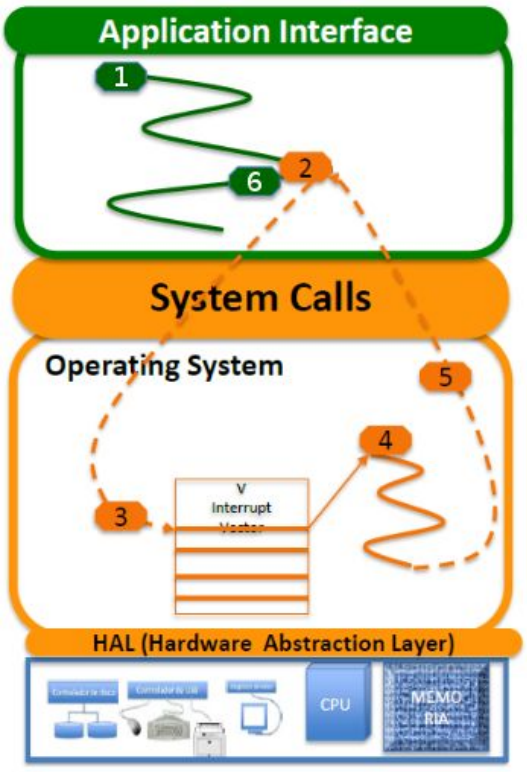
?

APELLIDOS		NOMBRE		Grupo
DNI		Firma		

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 9 cuestiones, cuya valoración se indica en cada una de ellas.**
- **Recuerde que debe justificar sus cálculos o respuestas para obtener buena calificación**

1. Describa los eventos numerados del 3 al 6 de la figura, relativos a la solicitud de una llamada al sistema por parte de un proceso de usuario, e indique en "Modo:" el modo de funcionamiento de la CPU que corresponde a la realización de cada uno de los eventos numerados.

(0,9 puntos)

1		**EVENTO 1** **Descripción: Programa de usuario en ejecución** **Modo:** **EVENTO 2** **Descripción: Llamada al sistema** **Modo:** **EVENTO 3** **Descripción:** **Modo:** **EVENTO 4** **Descripción:** **Modo:** **EVENTO 5** **Descripción:** **Modo:** **EVENTO 6** **Descripción:** **Modo:**

2. Dado el siguiente código cuyo fichero ejecutable es *example1*.

```
 1 /*** example1***/
 2 #include <all_needed.h>
 3
 4 int main()
 5 { int i=0, delay1=1, delay2=1;
 6   pid_t pid, pid_x;
 7
 8   for (i=0; i<2; i++)
 9   {  pid=fork()
10      if (pid==0)
11        { sleep(delay1);
12          if(execlp("wc","wc","-l","example1.c",NULL)<0)
13            {printf("Error running wc\n");
14              exit(0);}
15           pid_x=fork();
16      }
17   }
18   sleep(delay2);
19   while (wait(NULL) =! -1);
20   exit(0);
21 }
```

Suponga que *example1* se ejecuta sin errores e indique de forma justificada: **(1,0 puntos = 0,5 + 0,5)**

2	a) El número de procesos que se generan al ejecutarlo y dibuje el esquema de parentesco entre procesos.
	b) Indique de forma justificada la posibilidad de que se generen procesos zombies o huérfanos para cada uno de los siguientes pares de valores de *delay1* y *delay2*: *b1) delay1=1000; delay2=10*; b2) *delay1=10: delay2=1000*;

3. Sea un sistema de tiempo compartido con un planificador de procesos a corto plazo Round Robin, con quantum **q = 2 ut** y una única cola de preparados. Cuando se producen varios eventos en un mismo instante el orden de inserción en cola de preparados de los procesos es: nuevo, procedente de E/S y fin de quantum. La tabla muestra una planificación para los procesos, A, B y C, que llegan en los instantes t=0, t=3 y t=8 respectivamente y cuyas ráfagas son:

A (t=0)	4CPU+ 2E/S + 4CPU + 3E/S + 1CPU	B (t=3)	6CPU + 1E/S + 2CPU	C(t=8)	3CPU+2E/S+1CPU

Por un fallo de implementación, el planificador no funciona según las especificaciones propuestas y comete errores. Detecte en la tabla el instante de tiempo (T) en él que ocurre un error por primera vez, indique el motivo de dicho error y rehaga la planificación a partir de ese punto en las columnas paralelas de dicha tabla.

(1,2 puntos = 0,9 + 0,3)

T	Preparados	CPU	Cola E/S	E/S	Preparados	CPU	Cola E/S	E/S	Evento
0	(A)	A							Llega A
1		A							
2		A							
3	B	A							Llega B
4	(B)	B	(A)	A					
5		B		A					
6	B(A)	A							
7	B	A							
8	C A(B)	B							Llega C
9	C A	B							
10	B C(A)	A							
11	B C	A							
12	B(C)	C	(A)	A					
13	B	C		A					
14	C(B)	B		A					
15	A C	B							
16	A(C)	C	(B)	B					
17	B (A)	A	(C)	C					
18	(B)	B		C					Fin A
19	C	B							
20	(C)	C							Fin B
21									Fin C
22									
23									

3

a) Instante T del error =
Motivo:

b) Indique tiempo de Retorno y Tiempo de Espera para cada proceso

	A	B	C
Tiempo Retorno			
Tiempo Espera			

4. Dada la siguiente cadena de 52 caracteres correspondiente a un gen:
CACTCAGCACGAA GGGCAGAGGAATG CTTACCGTCCTGA GCCACCCACCAGC

Se desea buscar en qué posiciones se referencia al aminoácido CAG, con un diseño basado en 4 hilos concurrentes. Cada hilo analiza una única fracción de 13 caracteres del gen, de forma que un hilo analiza los 13 primeros caracteres, otro los 13 siguientes, etc. La función *"find_amino"* recibe un puntero a la posición del primer carácter de la fracción del hilo que tiene asignada y en caso de encontrar el aminoácido en esa fracción, marca el primer carácter donde aparece sobreescribiéndo "C" por el carácter '*'. El programa principal, tras asegurarse de que se han marcado todas las ocurrencias del aminoácido, localiza las posiciones marcadas y las envía a la salida estándar en orden, separadas por un espacio. **(1,0 puntos = 0,5 + 0,25 + 0,25)**

```
 1 #include <all_needed.h>
 2 #define NFRAC 4
 3 #define GENSIZE 52
 4
 5 char gen[] = "CACTCAGCACGAAGGGCAGAGGAATG
 6 CTTACCGTCCTGAGCCACCCACCAGC";
 7 char amino[] = "CAG";
 8
 9 void *find_amino(void *ptr){
10   char *pgen= (char*) ptr;
11   int i;
12   for (i=0; i<GENSIZE/NFRAC-2; i++) {
13     if (pgen[i] == amino[0] &&
14         pgen[i+1] == amino[1] &&
15         pgen[i+2] == amino[2])
16       pgen[i] = '*';
17 }
```

```
18 int main() {
19   int i;
20   char *pfrac;
21   pthread_attr_t attrib;
22   pthread_t thread[NFRAC];
23   pthread_attr_init(&attrib);
24
25   for (i = 0; i<NFRAC; i++) {
26     pfrac= gen + i * GENSIZE/NFRAC;
27
28 /**complete**/
...
     for (i = 0; i<GENSIZE; i++) {
       if (gen[i]=='*')
          printf("%d ", i);
  }
```

4

a) Utilice las variables ya declaradas en el programa y complete las líneas de código que faltan en el espacio marcado "**/**complete**/**", de forma que se realice la creación y espera de los 4 hilos "find_amino" para que se resuelva el problema de la forma propuesta.

NOTA. Definición de funciones: *int pthread_join(pthread_t thread, void **retval);*
*int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);*

```
25 for (i = 0; i<NFRAC; i++){
26   pfrac= gen + i * GENSIZE/NFRAC;
```

b) Justifique cuál es el máximo número de hilos pertenecientes a este proceso que llegarán a ejecutarse concurrentemente.

c) Razone si la ejecución de este programa puede dar lugar a alguna condición de carrera.

5. Sea el siguiente código de tres procesos concurrentes cuyos valores iniciales de la variable X y semáforos S1,S2 son los siguientes: X=1, S1=1, S2=1.

A	B	C
P(S1); X = X + 1; V(S1);	P(S1); V(S1); P(S2); X = X - 1; V(S2);	P(S1); P(S2); X = X + 1; V(S2);

Estos tres procesos se ejecutan en un sistema donde el orden en que se asigna por primera vez el procesador está SIEMPRE determinado por el orden de llegada a la cola de preparados. Tras conseguir la CPU por primera vez SIEMPRE se ejecuta la primera instrucción (P(S1)), pero a partir de ese punto pueden producirse cambios de contexto en cualquier momento. Indique de forma justificada: i) Si los tres procesos pueden finalizar;
ii) Si es posible que ocurra una condición de carrera y iii) El valor final de X, si los procesos acaban.
Considera las siguientes tres posibles secuencias de llegada:

(1,0 puntos = 0,5 + 0,5)

5

a) A, B, C

b) B, A, C

6. Un sistema con 4 GBytes de Memoria Principal utiliza paginación por demanda con un algoritmo **LRU** con reemplazo **LOCAL**. El espacio de direccionamiento lógico es de 4 GBytes y el tamaño de página de 4 KBytes. En un momento dado se inicia la ejecución de los procesos A y B, y el sistema asigna 4 marcos (numerados del 0 al 3) para A y B que se reparten según el esquema equitativo, marcos que inicialmente se encuentran vacíos.

(1,6 puntos = 0,2 + 0,8 + 0,6)

6

a) Obtenga la serie de referencias de la siguiente secuencia de direcciones lógicas (en hexadecimal): (A:1A407125) (A:1A4072C9) (A:4FB30190) (B:00400000) (A:1A407C41) (B:1000458F) (A:24B71AFF) (B:1E861100) (B:1E861ABC) (A:1A407642) (B:0040093C) (A:4FB30456)

b) Indique la evolución del contenido de memoria principal para la serie de referencias obtenida en el apartado a). Los marcos se asignan en orden creciente según demanda. Indique también los fallos de página producidos.

Marco										
0										
1										
2										
3										

FALLOS DE PÁGINA TOTALES :

c) Tras la serie de referencias del apartado b) se generan dos nuevas direcciones lógicas: (A:1A407FFF) (B:1E861008). Indique de forma justificada qué direcciones físicas generará la MMU para cada una de ellas:

7. Teniendo en cuenta el mecanismo de herencia de procesos en Unix y las llamadas POSIX, responda a los siguientes apartados: **(1,3 puntos = 0,6 + 0,7)**

7

a) Rellene la tabla de descriptores de archivos abiertos, indicando el contenido de los descriptores no vacíos, durante la ejecución de cada uno de los procesos que intervienen en la siguiente orden:

```
$ ls -l 2> err | tee f1 | grep ".c" > f2
```

La orden debe finalizar y ejecutarse correctamente. Recuerde que la orden **`tee`** escribe su entrada estándar tanto en el archivo que se le pasa como parámetro como en la salida estándar.

ls	
0	
1	
2	
3	
4	

tee	
0	
1	
2	
3	
4	

grep	
0	
1	
2	
3	
4	

Nota. Denomine al primer tubo "pipeA" y al segundo "pipeB"

b) Suponga que no se producen errores en las llamadas al sistema y complete el siguiente programa en C con las instrucciones y llamadas al sistema necesarias (una en cada línea con número subrayado) para que sea equivalente a la siguiente línea de órdenes: $ `ls -l 2> err | tee f1 | grep ".c" > f2`

```
 1. #include <all_needed.h>
 2. #define newfile (O_RDWR | O_CREAT | O_TRUNC)
 3. #define mode644 (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
 4. int main() {
 5.     int pipeA[2], pipeB[2], fd;
 6.
 7.     if (fork()) {
 8.         fd = open("err", newfile, mode644);
 9.
10.
11.         close(pipeA[0]); close(pipeA[1]); close(fd);
12.         execlp("ls", "-l", NULL);
13.      } else {
14.         pipe(pipeB);
15.         if (fork()) {
16.             dup2(pipeA[0], STDIN_FILENO);
17.             dup2(pipeB[1], STDOUT_FILENO);
18.             close(pipeA[0]); close(pipeA[1]);
19.             close(pipeB[0]); close(pipeB[1]);
20.             execlp(                                    ); /*complete*/
21.         } else {
22.             close(pipeA[0]); close(pipeA[1]);
23.             fd = open(                                  ); /*complete*/
24.
25.
26.             close(pipeB[0]); close(pipeB[1]); close(fd);
27.             execlp("grep", "grep", ".c", NULL);
28.         }
29.    }
30.    return 0;
31. }
```

8. Dado el siguiente listado del contenido de un directorio en un sistema POSIX:

```
 i-nodo     permisos enlaces usuario grupo    tamaño    fecha      nombre
32448485   drwxrwxr-x   2      pep   alumne     4096 ene  8  11:57  .
       1   drwxr-xr-x  11      pep   alumne   236871 ene  8  12:02  ..
32448793   -rw-rw----   1      pep   alumne      310 ene  9  10:37  f1
32448802   --w-rw-r--   3      pep   alumne      343 ene  9  11:15  f3
32448805   -rw----r--   3      pep   alumne      343 ene  9  10:33  f4
32448752   -rwxrwxrwx   1      pep   alumne     5824 ene  9  15:17  f5
33373385   -r-sr-xr-x   1      ana   alumne      706 ene  9  10:35  cp1
32448804   lrwxrwxrwx   1      ana   alumne        8 ene  9  10:36  cp2 ->cp1
32448803   lrwxrwxrwx   1      ana   profes        8 ene  9  10:40  f2 ->f1
```

Donde los programas `cp1 y cp2` requieren el nombre de dos archivos como argumentos. Las órdenes `cp1 y cp2 fich1 fich2`, tienen como resultado que el contenido de `fich1` se copia en `fich2` y si `fich2` no existe entonces se crea.

(1,0 puntos = 0,75 + 0,25)

8

a) Rellene la tabla e indique en caso de éxito cuales son los permisos que se comprueban y, en caso de error, cuál es el permiso que falla y porqué.

Usuario	Grupo	Orden	¿Funciona? ¿SI o NO?
ana	**profes**	**./cp1 f1 f3**	
Justifique			
pep	**alumne**	**./cp2 f3 f2**	
Justifique			
pep	**disca**	**./cp1 f4 f6**	
Justifique			

b) Justifica adecuadamente qué tipo de archivo es **f2** e indique el números de los nodos-i a los que será necesario acceder para leer el contenido de dicho archivo.

9. Un sistema de archivos Minix de 512MBytes tiene las siguiente características:

- Nodos-i con un tamaño de 32 Bytes con 7 punteros directos a zonas, 1 indirecto y 1 doblemente indirecto
- Punteros a zona de 16 bits (2Bytes)
- Entradas de directorio de 16 Bytes (14 Bytes para el nombre y 2 Bytes para el nodo-i)
- **1 Bloque = 1 Zona = 2 KBytes**

(1,0 puntos = 0,6 + 0,4)

9

a) Indique de forma justificada como quedarían los tamaños de cada elemento de la cabecera al formatearlo para almacenar 16384 nodos-i (16K nodos).

Arranque	Super bloque	Mapa de bits de Nodos-i	Mapa de bits de Zonas	Nodos- i	Zonas de datos

b) Calcule y justifique el tamaño mínimo de zona con que se debería formatear una partición de 512MBytes si se diseña el nodo-i con únicamente 8 punteros directos con punteros a zona de 32 bits y se quiere disponer de archivos de hasta 256 MBytes. Indique de forma justificada el tamaño de zona y qué ventajas y desventajas tiene este diseño para archivos grandes de 256MBytes y archivos pequeños de 1 Byte.