APELLIDOS		NOMBRE		Grupo
DNI		Firma		

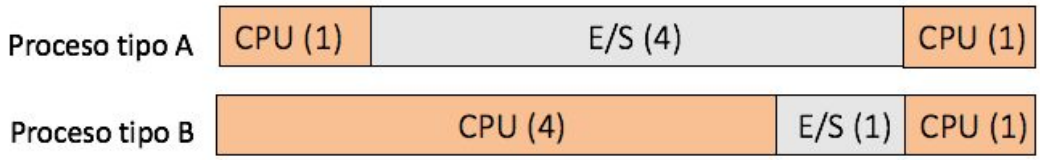
- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 7 cuestiones, cuya valoración se indica en cada una de ellas.**

1. De los siguientes datos almacenados en la PCB de un proceso, indique si pueden cambiar o no desde que empieza el proceso hasta que acaba, justificando la respuesta.

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

1a	**PID**
1b	**PPID**
1c	**Copia del contador de programa y registros de la CPU**
1d	**Estado**

2. Sea un sistema con una única CPU y la E/S constituida por un disco duro, que permite una **concurrencia máxima de 2 procesos**. A este sistema pueden llegar procesos de dos tipos que alternan ráfagas de CPU y de E/S (DISCO) de acuerdo a la siguiente figura (entre paréntesis se indica la duración de esas ráfagas):



Responde a las siguientes cuestiones brevemente:

(1,7 puntos=0,6+0,6+0,5)

2a	Describa brevemente qué es un proceso limitado por CPU o por E/S, e identifique entre los procesos anteriores de qué tipo son.
2b	Indique qué combinación y orden de llegada de procesos permite obtener el máximo rendimiento (productividad), y la máxima utilización de la CPU. Justifíquelo gráficamente y calcule la productividad y utilización de la CPU. Nota: los procesos pueden ser todos de un mismo tipo.
2c	En el sistema utilizado, se cambia el disco duro por uno de estado sólido (SSD), reduciendo el tiempo de E/S al 25% (es decir, lo que antes costaba 4, ahora cuesta 1). Indique en este caso, qué combinación de procesos obtiene un máximo rendimiento.

3. Considerad los dos códigos siguientes:

Código F.c

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  int main(int argc, char *argv[]){
4    int pid,i,status;
5    for(i=0;i<3;i++){
6        pid=fork();
7        if (pid==0){
8        printf("exit\n");
9        exit(i);
10       }
11   }
12   i=0;
13   while(wait(&status)>0) i++;
14   printf("wait(%d)\n",i);
15 }
```

Código E.c

```
1  #include <unistd.h>
2  int main(int argc, char *argv[]){
3        execl("./F","F",NULL);
4        execl("./F","F",NULL);
5  }
```

Suponed que ambos códigos están compilados como F y E en el directorio de trabajo.

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

3a	En total, ¿cuántos procesos crea la orden ./F? Dibuje el grafo que los relaciona.
3b	Describa la salida que produce la orden ./F

3c	De los procesos creados con la orden ./F, ¿cuántos quedan huérfanos? Justifique la respuesta
3d	¿Cuántos procesos crea la orden ./E? Justifique la respuesta

4. Queremos analizar diferentes planificadores a corto plazo para el sistema operativo de un computador. Los planificadores elegidos han sido SJF (Shortest-Job-First), SRTF (Shortest-Remaining-Time-First), y RR (Round-Robin) con un *quantum* de 1 u.t. (q=1). El análisis lo queremos basar en los tiempos medios de retorno y de espera de los siguiente procesos:

Proceso	Instante de llegada	Ráfagas de CPU y E/S
A	0	4CPU + 1E/S + 1CPU
B	1	2CPU + 2E/S + 1CPU
C	4	1CPU

Rellene los diagramas siguientes para cada unos de los planificadores y calcule los valores medios del tiempo de retorno y espera de los procesos, así como la utilización de la CPU. En caso de eventos simultáneos considerar la siguiente ordenación: 1) llegada de proceso nuevo, 2) terminación de proceso, 3) abandono estado de suspensión y 4) fin del quantum. ¿Cuál sería el mejor planificador considerando el tiempo medio de retorno?¿Y cuál considerando el tiempo medio de espera?¿Y si consideramos la utilización de la CPU?

(**1,7 puntos=0,5+ 0,5+0,5+0,2**)

4a) SJF (Shortest-Job-First)

T	Preparados	CPU	Cola E/S	E/S	Evento
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Tiempo Medio*
Tiempo Retorno				
Tiempo Espera				
Utilización de CPU				

4b) SRTF (Shortest-Remaining-Time-First)

T	Preparados	CPU	Cola E/S	E/S	Evento
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Tiempo Medio*
Tiempo Retorno				
Tiempo Espera				
Utilización de CPU				

4c) RR (Round-Robin) q=1 u.t.

T	Preparados	CPU	Cola E/S	E/S	Evento
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Tiempo Medio*
Tiempo Retorno				
Tiempo Espera				
Utilización de CPU				

4d) Selección de planificadores.

Mejor planificador desde el punto de vista de ...	Planificador	Tiempo Medio / Utilización
… tiempo de retorno:		
… tiempo de espera:		
… utilización de la CPU		

5. El programa ThreadsAdd.c de la práctica de hilos, cuyo listado aparece a continuación, realiza la suma de una matriz bidimensional `matrix`.

```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 #define DIMROW 1000000
 5 #define NUMROWS 20
 6
 7 typedef struct row{
 8   int vector[DIMROW];
 9   long addition;
10 } row;
11
12 struct row matrix[NUMROWS];
13 long total_addition=0;
14
15 void *AddRow( void *ptr )
16 {
17   int k;
18   row *fi;
19   fi = (row *)ptr;
20
21   fi->addition=0;
22   for(k=0;k<DIMROW;k++)
23     fi->addition +=
       exp((k*(fi->vector[k])+(k+1)*
       (fi->vector[k]))/
       (fi->vector[k]+2*k))/2;
24   //APDO.D
25 }
```

```
26 int main()
27 {
28   int i,j;
29   pthread_t  threads[NUMROWS];
30   pthread_attr_t atrib;
31
32   // Vector elements are initialized to 1
33   for(i=0;i<NUMROWS;i++)
34     for(j=0;j<DIMROW;j++)
35       matrix[i].vector[j]=1;
36
37   // Thread attributes initialization
38   pthread_attr_init( &atrib );
39
40   for(i=0;i<NUMROWS;i++){
41     pthread_create(&threads[i],&atrib,
                AddRow,(void *)&matrix[i]);
42     //APDO.C
43   }
44
45   for(i=0;i<NUMROWS;i++)
46     pthread_join(threads[i],NULL);
47
48   for(i=0;i<NUMROWS;i++)
49     total_addition += matrix[i].addition;
50   printf("Total addition is: %ld \n",
              total_addition);
51   pthread_exit(0);
52 }
```

Conteste, justificando sus respuestas, a las siguientes cuestiones acerca del programa ThreadsAdd.c. Todos los supuestos se refieren a variaciones independientes sobre el mismo código original del listado anterior y suponiendo en todos los casos que la ejecución se realiza con un procesador multinúcleo.

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

5a	¿Cuál sería el efecto sobre el resultado final obtenido si se eliminaran las líneas 45 y 46?
5b	Cuál sería el efecto sobre el resultado final obtenido y el tiempo de ejecución del programa si se eliminaran las líneas 45, 46 y 51.
5c	¿Cuál sería el efecto sobre el resultado final obtenido y el tiempo total de ejecución si se eliminaran las líneas 45 y 46 y se incluyera `pthread_join(threads[i],NULL)` en la línea 42?
5d	¿Qué implicaciones tendría sobre la corrección del resultado si se eliminaran las líneas 48 y 49 y se añadiera `total_addition += fi->addition;` en la línea 24 ?

6. Dadas las funciones agrega y resta vistas en prácticas, contesta de forma justificada a las siguientes cuestiones

NOTA: Asuma que previa la declaración de las funciones se definen las constantes y variables siguientes:

```
#define REPEAT  20000000
long int V = 100;
int llave = 0;
```

```
1  void *agrega (void *argumento) {
2   long int cont;
3   long int aux;
4
5   for (cont=0; cont<REPEAT; cont++) {
6
7     V = V + 1;
8
9   }
10
11  printf("--> Fin AGREGA (V=%ld) \n", V);
12  pthread_exit(0);
13 }
14
```

```
15 void *resta (void *argumento) {
16  long int cont;
17  long int aux;
18
19  for (cont=0; cont<REPEAT; cont++) {
20
21    V = V - 1;
22
23  }
24
25  printf("--> Fin RESTA(V=%ld)\n", V);
26  pthread_exit(0);
27 }
28
```

```
29 int main (void) {
30  pthread_t hiloSuma, hiloResta,
31  pthread_attr_t attr;
32
33  pthread_attr_init(&attr);
34  pthread_create(&hiloSuma,&attr, agrega,NULL);
35  pthread_create(&hiloResta,&attr,resta, NULL);
36
37  pthread_join(hiloSuma,NULL);
38  pthread_join(hiloResta,NULL);
39
40  fprintf(stderr, "-------> VALOR FINAL: V = %ld\n\n", V);
41  exit(0);
42 }
```

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

6a	¿Qué es una sección crítica? Identifique, si las hay, **las líneas del código (indique los números)** que representan las secciones críticas e indique en qué **líneas del código** se debería introducir el código del protocolo de entrada **(indique los números)** y del protocolo de salida **(indique los números)** para que la sección crítica esté protegida.

6b	Si queremos proteger las sección críticas utilizando la solución hardware (Test&Set), escriba el código que implemente el protocolo de entrada y el protocolo de salida.
6c	Justifique si la solución hardware que utiliza Test&Set, como la del código anterior, cumple las tres condiciones de los protocolos de acceso a las secciones críticas.
6d	Indique la relación entre solución básica basada en Test&Set y la espera activa. ¿En qué condiciones es adecuado utilizar espera activa en los protocolos de acceso a la sección crítica?

7. Sean tres semáforos: A, B y C inicializados del siguiente modo: A=0, B=1, C=0. Dados 3 procesos que se ejecutan concurrentemente y acceden a dichos semáforos de la siguiente forma:

Proceso 1	Proceso 2	Proceso 3
. . **P(A);** . . . **V(C);**	. . **P(B);** . . . **V(A);** **V(B);**	. . **P(B);** **P(C);** . . **V(B);**

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

7a Indique una secuencia de operaciones P y V de los tres procesos que les permita terminar su ejecución.

PROCESO	OPERACIÓN DE SEMÁFORO

7b Indique una secuencia de operaciones P y V de los tres procesos que les conduzca a una situación de interbloqueo.

PROCESO	OPERACIÓN DE SEMÁFORO