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. Un computador gestiona su espacio lógico de memoria de 1 MB mediante paginación, con páginas de 512 Bytes. El mapa de memoria de cierto proceso reserva espacio para tres regiones: una para 1100 instrucciones de 4 Bytes (todas contiguas), otra para 130 variables globales de 8 Bytes (también contiguas) y una tercera para la pila. El sistema asigna dinámicamente el mínimo número posible de páginas para la pila.

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

1

a) ¿Cuántos bits se reservan para número de página en la dirección lógica?

b) ¿Cuántas páginas se necesitan para almacenar las instrucciones de este proceso? ¿y para sus variables globales?

c) En el momento en que la pila ha crecido hasta ocupar 1800 bytes, ¿cuántas entradas de la tabla de páginas serán válidas para el proceso?

d) Calcule (en Bytes) la fragmentación interna en el mismo momento que en el apartado **c)**

2. Sea un sistema de tiempo compartido con planificador a corto plazo y una única cola de procesos preparados. En el instante t=0 llegan a su cola de preparados los procesos A, B y C en este orden. Se trata de tres procesos orientados a CPU con idéntico perfil de ejecución:

1000 CPU	10 E/S	1000 CPU	10 E/S	10 CPU

Determine de forma justificada el tiempo de retorno y el tiempo espera de cada proceso (A, B y C) para:

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

2

a) Un algoritmo de planificación FCFS.

Tiempo de retorno			Tiempo de Espera		
A=	B=	C=	A=	B=	C=

b) Un algoritmo de planificación RR con q=10ut.

Tiempo de retorno			Tiempo de Espera		
A=	B=	C=	A=	B=	C=

c) Asuma un coste para los cambios de contexto y justifique entre RR y FCFS cuál sería el algoritmo con mejor rendimiento para esta situación.

3. Dado el siguiente código cuyo fichero ejecutable ha sido generado con el nombre “Ejemplo1”.

```
1  /*** Ejemplo1.c ***/
2  #include "todas_las_cabeceras_necesarias.h"
3  #define N 2
4
5  main() {
6    int i=0, j=0;
7    pid_t pid;
8    while (i<N) {
9      pid = fork();
10     if (pid ==0) {
11       for (j=0; j<i ; j++) {
12          fork();
13          printf(“Child i=%d, j=%d \n”, i, j);
14          exit(0);
15       }
16     } else {
17       printf(“Parent i=%d, j=%d \n”, i, j);
18       while (wait(NULL) != -1);
19     }
20     i++;
21   }
22   exit(0);
23 }
```

Indique de forma justificada:

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

3	**a)** El número de procesos que se generan al ejecutarlo y dibuje el esquema de parentesco entre procesos.
	b) Indique los mensajes que se imprimirán por pantalla al ejecutar el código “Ejemplo1.c”

4. Dado el siguiente programa `hilos.c`, en el que la función `rever()` invierte la cadena que se le pasa como argumento:

```
 1 #include "Los_necesarios.h"
 2 #define ARG_MAX 20
 3 int done=0;
 4
 5 void *rever( void *ptr ){
 6  int i;
 7  char temp;
 8  char *pfirst= (char*) ptr;
 9  char *plast= pfirst+strlen(pfirst)-1;
10  while (plast > pfirst){
11    temp= *pfirst;
12    *pfirst= *plast;  *plast= temp;
13    pfirst++; plast--;
14  }
15  done++;
16 }
```

```
17 int main(int argc, char* argv[]){
18  int i;
19  pthread_t thr[ARG_MAX];
20  pthread_attr_t atrib;
21  pthread_attr_init(&atrib);
22
23  for(i=1;i<argc;i++)
24    pthread_create(&thr[i],&atrib,
                rever,(void *)argv[i]);
25
26  for(i=argc-1;i>0;i--){
27    pthread_join(thr[i],NULL);
28    printf("%s ",argv[i]);
29  }
30 }
```

Conteste, justificando sus respuestas, suponiendo que se compila y ejecuta de la siguiente forma:

```
gcc hilos.c -o hilos -lpthread
./hilos once upon a time
```

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

4

a) ¿Cuál es el número máximo de hilos que pueden llegar a ejecutarse concurrentemente?

b) ¿Qué se muestra en la salida estándar tras la ejecución del programa?

c) ¿De qué manera podría afectar a la salida generada por el programa la eliminación de la línea 27?

d) En el programa original, si se eliminara la línea 27 y se inserta en la línea 25 este código:

```
while (done < argc-1);
```

¿Funcionaría correctamente, obteniéndose la misma salida del programa original?

5. Se desea gestionar el aforo de un parque de atracciones limitado a 5000 personas. Para ello se dispone de un proceso que implementa las tareas de entrada y salida del parque mediante dos tipos de hilos que realizan las funciones "func_entrar" y "func_salir" respectivamente. Estas dos funciones deben controlar que no se produzcan más entradas cuando el aforo está lleno y que no se contabilicen salidas cuando el recinto está vacío. Además se desea conocer en todo momento el aforo real del recinto, para lo cual se utilizará una variable llamada "aforo" que los hilos irán modificando adecuadamente.

Para resolver los problemas de sincronización que surgen se utilizan tres semáforos:

- mutex : controla el acceso a la sección crítica
- plazas_libres : gestiona el número de plazas libres en el recinto
- personas : gestiona el número de personas que han entrado en el recinto

a) Complete el código del programa principal con la inicialización de los semáforos definidos

b) Complete el código de las funciones "func_entrar" y "func_salir", realizando las operaciones necesarias sobre los semáforos definidos.

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

5

a)

```
#include <semaphore.h>
#define N 5000
int aforo;
sem_t mutex, plazas_libres, personas;
pthread_t entrar, salir;
void main () {
  // Inicialización de semáforos




  //
  pthread_create(&entrar, NULL , func_entrar, NULL);
  pthread_create(&salir, NULL , func_entrar, NULL);
  ...
}
```

b)

```
void *func_entrar(void *p) {
  while (1) {



    aforo=aforo+1;



  }
}
```

```
void *fun_salir(void *p) {
  while (1) {



    aforo=aforo-1;



  }
}
```

6. Sea un sistema con **memoria virtual, paginación por demanda** y algoritmo de **reemplazo LRU GLOBAL**. Las páginas son de 4KB, la memoria principal es de 40 KB (10 marcos) y el tamaño máximo de un proceso de 16KB (4 páginas). El algoritmo LRU se implementa mediante contadores, anotando el instante de tiempo en que se referencia la página. La asignación de marcos en memoria principal es en orden creciente. En el instante t=10, existen 3 procesos cuyo contenido de sus tablas de páginas:

Tabla de páginas del proceso A			
Pág.	**Marco**	**Contador**	**Bit Val.**
0	2	4	v
1	5	1	v
2			i
3	1	5	v

Tabla de páginas del proceso B			
Pág.	**Marco**	**Contador**	**Bit Val.**
0	0	2	v
1	3	3	v
2			i
3			i

Tabla de páginas del proceso C			
Pág.	**Marco**	**Contador**	**Bit.Val.**
0	4	10	v
1	8	7	v
2	9	8	v
3			i

Considere el estado del sistema en t = 10 y conteste a las siguientes preguntas:

a) Indique el proceso y número de página que ocupa cada marco de la memoria principal. Rellene para ello la tabla propuesta con el siguiente formato: A0 corresponde a la página 0 del proceso A.

b) A partir de t=10, se referencian las siguientes direcciones lógicas (**direcciones en hexa**): (A, 2F2C), (C, 3001), (A, 11C1), (B, 3152). Rellene el contenido de la memoria principal y las tablas de páginas al finalizar dichas referencias..

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

6 **a)**

(TABLA apartado a))	
Nº marco	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(TABLA apartado b))	
Nº marco	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

TABLAS DE PÁGINAS DE LOS PROCESOS A, B y C (apartado b)

Tabla de páginas de A			
Pág	**Mar.**	**Cont.**	**Bit Val.**

Tabla de páginas de B			
Pág	**Mar.**	**Cont.**	**Bit Val.**

Tabla de páginas de C			
Pág	**Mar.**	**Cont.**	**Bit Val.**

7. Suponga que se ejecuta sin errores el siguiente código C, que crea 3 procesos que se comunican mediante 3 tubos, conteste de forma razonada a los apartados siguientes:

```
   /*** pipes.c   ***/
 1 #include "todas_cabeceras_necesarias.h"
 2 #define nproc 3
 3 #define nt 2
 4
 5 int main(int argc,char* argv[]) {
 6   int pipes[nproc][2];
 7   int i, j, pid, i_1;
 8
 9   for (i=0; i<nproc; i++)
10 pipe(pipes[i]);
11   for (i=0; i<nproc; i++) {
12     pid = fork();
13     if (pid == 0) {
14      dup2(pipes[i][1],STDOUT_FILENO);
15      i_1 = i_anterior(i, nproc);
16      dup2(pipes[i_1][0],STDIN_FILENO);
17      for (j=0; j<nproc; j++) {
18        close(pipes[j][0]);
19        close(pipes[j][1]);
20       }
21      comp_and_com(i, nt);
22      exit(0);
23     }
24   }
25   for (i=0; i<nproc; i++) {
26     close(pipes[i][0]);
27     close(pipes[i][1]);
28    }
29   while (wait(NULL) != -1);
30   exit(0);
   }
```

```
31 int comp_and_com(id, nturns) {
32   int v, n;
33   n = 0;
34   while (n < nturns) {
35     if (id == 0 && n == 0) {
36       v = 0;
37       n++;
38       write(STDOUT_FILENO,  &v, sizeof(v));
39     } else {
40       read(STDIN_FILENO,  &v, sizeof(v));
41       v = v + 1;
42       n++;
43       write(STDOUT_FILENO, &v, sizeof(v));
44     }
45   }
46   if (id == 0) {
47     read(STDIN_FILENO,  &v, sizeof(v));
48     fprintf(stderr,"V FINAL VALUE: %d\n", v);
49   }
50   return 0;
51 }
52
53 int i_anterior(int i, int np) {
54   if (i > 0) return (i-1)
55   else if (i == 0) return (np-1);
56 }
57
```

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

7

a) Dibuje el esquema de comunicación que se genera entre los procesos creados con fork()

b) ¿Explique que imprimirá la sentencia `fprintf` de la línea 48?

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

```
drwxr-xr-x   2 pepe     alumno         4096 ene  8     2013  .
drwxr-xr-x  11 pepe     alumno         4096 ene 10     14:39 ..
-rwsr-xr-x   1 pepe     alumno      1139706 ene  9     2013  copia
-rwxr-sr--   1 pepe     alumno      1139706 ene  9     2013  anyade
-rw-------   1 pepe     alumno       634310 ene  9     2013  f1
-rw----rw-   1 pepe     alumno       104157 ene  9     2013  f2
-rw-rw----   1 pepe     alumno       634310 ene  9     2013  f3
```

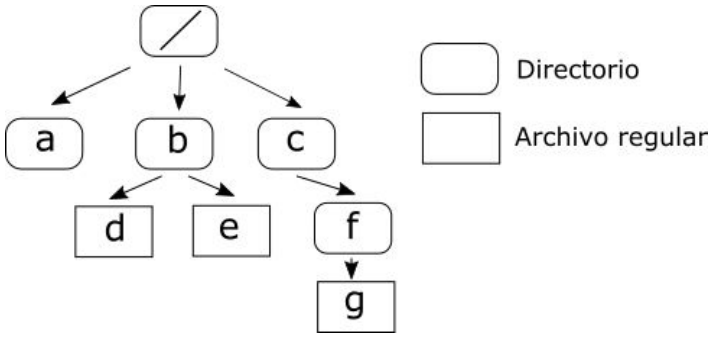
Donde los programas `anyade y copia` son programas que requieren el nombre de dos archivos como argumentos. El programa añade al final del archivo pasado como segundo argumento el contenido del primero, mientras que `copia` simplemente sustituye el contenido del archivo del segundo argumento por el del primero. 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.

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

8

Usuario	**Grupo**	**Orden**	**¿Funciona?**
ana	**profes**	**./anyade f1 f2**	
Justifique			
ana	**profes**	**./copia f2 f4**	
Justifique			
pau	**alumno**	**./anyade f2 f3**	
Justifique			
pepe	**alumno**	**./copia f3 f1**	
Justifique			

9. En un sistema de archivos minix se contiene el siguiente árbol de archivos:



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

9

a) Indique el contenido de cada directorio de este sistema de archivos, suponiendo que los i-nodos ocupados por cada elemento son: a(4), b(5), c(8), d(12), e(22), f(10) y g(16)

/		a		b		c		f	

b) Considere zonas de 1 KByte, punteros a zona de 32 bits, e i-nodos con 7 punteros directos, 1 puntero simple indirecto y 1 puntero doble indirecto, obtener el número de zonas ocupadas por el archivo *d* si su tamaño es de 2 MBytes.