

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre: \_\_\_\_\_

Grupo: \_\_\_\_\_

\* Para la realización del presente examen se dispondrá de 3 horas, incluido el test.

\* No se pueden utilizar libros ni apuntes. Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

---

Ejercicio 2 (2 puntos)

En un computador que utiliza política de planificación cíclica, se ejecutan los siguientes procesos de los que se conoce su tiempo de llegada y su tiempo total de ejecución:

Proceso	Tiempo de llegada	Duración
A	0	400
B	125	200
C	150	400
D	175	300

A) Rellene una tabla como la siguiente, asumiendo que la rodaja de tiempo es de 100 ms., indicando qué proceso está en ejecución y el estado de la cola de procesos listos en cada instante de tiempo. Indique en la columna eventos en qué momento del tiempo se produce para cada proceso, su llegada, su inicio y su finalización.

Instante	En ejecución	Cola de listos	Eventos
...	...	...	

B) Rellene una tabla como la siguiente, indicando para cada proceso los tiempos de llegada, servicio, inicio, fin, retorno, espera y espera normalizado.

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0						
B	125						
C	150						
D	175						

C) Repita el apartado A, considerando un rodaja de 200 ms.

Instante	En ejecución	Cola de listos	Eventos
...	...	...	

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

D) Repita el apartado B, para el caso de una rodaja de tiempo de 200 ms.

SOLUCIÓN

A)

Instante	En ejecución	Cola de listos	Eventos
0	A(400)		Llegada A, Inicio A
100	A(300)		
125	A(275)	B(200)	Llegada B
150	A(250)	B(200), C(400)	Llegada C
175	A(225)	B(200), C(400), D(300)	Llegada D
200	B(200)	C(400), D(300), A(200)	Inicio B
300	C(400)	D(300), A(200), B(100)	Inicio C
400	D(300)	A(200), B(100), C(300)	Inicio D
500	A(200)	B(100), C(300), D(200)	
600	B(100)	C(300), D(200), A(100)	
700	C(300)	D(200), A(100)	Fin B
800	D(200)	A(100), C(200)	
900	A(100)	C(200), D(100)	
1000	C(200)	D(100)	Fin A
1100	D(100)	C(100)	
1200	C(100)		Fin D
1300			Fin C

B)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0	400	0	1000	1000	600	1000/400
B	150	200	200	700	550	350	550/200
C	150	400	300	1300	1150	750	1150/400
D	175	300	400	1200	1025	725	1025/300

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

C)

Instante	En ejecución	Cola de listos	Eventos
0	A(400)		Llegada A, Inicio A
125	A(275)	B(200)	Llegada B
150	A(250)	B(200), C(400)	Llegada C
175	A(225)	B(200), C(400), D(300)	Llegada D
200	B(200)	C(400), D(300), A(200)	Inicio B
400	C(400)	D(300), A(200)	Inicio C, Fin B
600	D(300)	A(200), C(200)	Inicio D
800	A(200)	C(200), D(100)	
1000	C(200)	D(100)	Fin A
1200	D(100)		Fin C
1300			Fin D

D)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0	400	0	1000	1000	600	1000/400
B	150	200	200	400	250	50	50/200
C	150	400	400	1200	1050	650	650/400
D	175	300	600	1300	1125	825	825/300

### Ejercicio 3 (2 puntos)

Realizar un programa que cree 12 threads que simularán la entrada de 3 personas en una sala. En la sala solo puede haber o 0 o 3 personas por lo que los threads deberán esperar para entrar a que haya un grupo de 3. Estarán esperando en la sala durante un tiempo aleatorio entre 1 y 4 segundos y después saldrán de la sala las 3 personas a la vez. Hasta que no salgan las 3 no podrán entrar otras 3 en la sala. Nota: use la función `srandom` para la espera aleatoria.

Cada vez que se cree un thread, este debe escribir el mensaje "Thread creado: ID". Cuando estén dentro de la sala escribirán en pantalla el mensaje "Persona dentro: ID". Cuando salgan escribirán el mensaje "Persona fuera: ID"., ID serán número entre 0 y 11 asignado al crear el thread.

La aplicación se deberá realizar utilizando como mecanismo de sincronización entre los threads únicamente `mutex` y `variables condicionales`.

Ejemplo de ejecución:

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

Thread creado : 0  
Thread creado : 1  
Thread creado : 2  
Persona dentro : 2  
Persona dentro : 0  
Persona dentro : 1  
Thread creado : 3  
Thread creado : 4  
Thread creado : 5  
Thread creado : 6  
Thread creado : 7  
Thread creado : 8  
Thread creado : 9  
Thread creado : 10  
Thread creado : 11  
Persona fuera : 1  
Persona fuera : 2  
Persona fuera : 0  
Persona dentro : 5  
Persona dentro : 3  
Persona dentro : 4  
Persona fuera : 4  
Persona fuera : 5  
Persona fuera : 3  
Etc.

### Solución

```
pthread_attr_t attr;
pthread_t idth[12];
pthread_mutex_t mtx;
pthread_cond_t varcond;
int yacopiada=0;
int personasEntrada=0;
pthread_mutex_t mtxEntrada;
pthread_cond_t varcondEntrada;
int lleno=0;
int personasSalida=0;
pthread_mutex_t mtxSalida;
pthread_cond_t varcondSalida;

void *hilo(void *num) {
    int id,espera;
    srandom(time (NULL));
    //Copio el parámetro recibido en una variable local en una zona protegida por
    //mutex para tener acceso exclusivo
    //Después señalo que ha finalizado la operación para que se entere el main
    pthread_mutex_lock (&mtx);
    id=((int *)num);
    yacopiada=1;
```

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

```
pthread_cond_signal(&varcond);
pthread_mutex_unlock (&mtx);
printf("\tThread id creado : %d\n",id);

//Cola de espera para entrar, se van apuntando en personasEntrada;
pthread_mutex_lock (&mtxEntrada);
while (lleno==1 )
    pthread_cond_wait(&varcondEntrada, &mtxEntrada);
personasEntrada++;
if (lleno==0 && personasEntrada==3){
    lleno=1;
    pthread_cond_broadcast(&varcondEntrada);
}
while ( personasEntrada <3)
    pthread_cond_wait(&varcondEntrada, &mtxEntrada);
pthread_mutex_unlock (&mtxEntrada);
printf("\tPersona id dentro : %d\n",id);
espera=random()%4 +1;
sleep(espera);
pthread_mutex_lock (&mtxEntrada);
personasEntrada--; //Se prepara para salir
pthread_mutex_unlock (&mtxEntrada);

pthread_mutex_lock (&mtxSalida);
personasSalida++;
if (personasSalida==3)
    pthread_cond_broadcast(&varcondSalida);
if (personasSalida <3)
    pthread_cond_wait(&varcondSalida, &mtxSalida);
printf("\tPersona id fuera : %d\n",id);
personasSalida--;
if (personasSalida==0){ //Permito entrar a otras 3
    lleno=0;
    pthread_cond_broadcast(&varcondEntrada);
}
pthread_mutex_unlock (&mtxSalida);
pthread_exit(0);
}

int main(){
    int i;

    pthread_mutex_init (&mtx, NULL);
    pthread_attr_init(&attr);
    pthread_mutex_init (&mtxEntrada, NULL);

    for (i=0; i<12; i++) {
        pthread_create(&idth[i],&attr,hilo,&i);
        //Espero a que el thread creado utilice el valor de i
        pthread_mutex_lock (&mtx);
        while (yacopiada==0)
            pthread_cond_wait(&varcond, &mtx);
        yacopiada=0;
    }
}
```

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

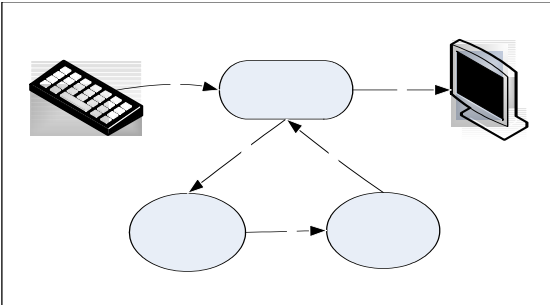
Nombre: \_\_\_\_\_

Grupo: \_\_\_\_\_

```
pthread_mutex_unlock (&mtx);
}
// Espero la finalización del thread
for (i=0; i<10; i++)
    pthread_join(idth[i],NULL);
return(0);
}
```

Ejercicio 4 (2 puntos)

El siguiente programa en C (incompleto y simplificado) pretende implementar la configuración de procesos de la figura y el correspondiente flujo de datos con tuberías.



El proceso A (padre) tomará datos del archivo “entrada.txt” que enviará al proceso hijo B; este, a su vez, le enviará los datos a su proceso “hermano”, hijo C, que se encargará de devolverlos al padre para que los imprima por pantalla. La ejecución de los procesos concluye cuando el proceso A (padre) recibe la cadena "exit".

Lea el código e incluya las definiciones y llamadas pertinentes en C en las zonas señaladas para implementar el comportamiento descrito en el texto.

<pre>void hijoB() {     char buf[4096];     int exit_flag = 0;     ... // OCHO     do {         read(...); // NUEVE         if (strncmp("exit\n", buf, strlen("exit\n")) == 0)             exit_flag = 1;         write(...); // DIEZ     } while (!exit_flag);     ... // ONCE }</pre>	<pre>int ...; // UNO int main(void) {     char buf[4096];     int pid, i;     int exit_flag = 0;     ... // DOS     for (i=0; i &lt; 2; i++) {         ... // TRES         if (pid == 0) {             switch(i) {                 case 0: /* tratamiento hijo B. */                     hijoB();                     exit(EXIT_SUCCESS);                 case 1: /* tratamiento hijo C. */                     hijoC();                     exit(EXIT_SUCCESS);             }         } else if (pid &gt; 0 &amp;&amp; i == 1) { /* Tratamiento Padre */</pre>
<pre>void hijoC() {     char buf[4096];</pre>	

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

<pre>int exit_flag = 0; ... // DOCE do {     read(...); // TRECE     if (strncmp("exit\n", buf, strlen("exit\n")) == 0)         exit_flag = 1;     write(...); // CATORCE } while (!exit_flag); ... // QUINCE }</pre>	<pre>... // CUATRO do {     read(STDIN_FILENO, buf, sizeof(buf));      if (strncmp("exit\n", buf, strlen("exit\n")) == 0)         exit_flag = 1;      write(...); // CINCO      read(...); // SEIS      write(STDOUT_FILENO, buf, strlen(buf));  } while (!exit_flag); ... // SIETE do {     pid = wait(NULL); } while (pid &gt; 0); /* Fin del Main */ }</pre>
---	---

### Solución

1. `int fds_pipe1[2], fds_pipe2[2], fds_pipe3[2]; // UNO`
- 2.
3. `void hijoB()`
4. `{`
5. `char buf[4096];`
6. `int exit_flag = 0;`
- 7.
8. `close(fds_pipe1[1]); // OCHO`
9. `close(fds_pipe2[0]);`
- 10.
11. `do {`
12. `read(fds_pipe1[0], buf, sizeof(buf)); // NUEVE`
- 13.
14. `if (strncmp("exit\n", buf, strlen("exit\n")) == 0)`
15. `exit_flag = 1;`
- 16.
17. `write(fds_pipe2[1], buf, strlen(buf)); // DIEZ`
- 18.
19. `} while ( !exit_flag);`
- 20.
21. `close(fds_pipe1[0]); // ONCE`
22. `close(fds_pipe2[1]);`
23. `}`

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre: \_\_\_\_\_

Grupo: \_\_\_\_\_

```
24.
25. void hijoC()
26. {
27.   char buf[4096];
28.   int exit_flag = 0;
29.
30.   close(fds_pipe2[1]); // DOCE
31.   close(fds_pipe3[0]);
32.
33.   do {
34.     numbytes = read(fds_pipe2[0], buf, sizeof(buf)); // TRECE
35.
36.     if (strcmp("exit\n", buf, strlen("exit\n")) == 0)
37.       exit_flag = 1;
38.
39.     write(fds_pipe3[1], buf, strlen(buf)); // CATORCE
40.
41.   } while ( !exit_flag);
42.   close(fds_pipe3[1]); // QUINCE
43.   close(fds_pipe2[0]);
44. }
45.
46.
47. int main(void) {
48.
49.   char buf[4096];
50.   int pid, i;
51.   int exit_flag = 0;
52.   int fd;
53.
54.   pipe(fds_pipe1); // DOS
55.   pipe(fds_pipe2);
56.   pipe(fds_pipe3);
57.
58.   for (i=0; i<2; i++) {
59.
60.     pid = fork(); // TRES
61.
62.     if (pid == 0) {
63.
64.       switch(i) {
65.
```



Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre: \_\_\_\_\_

Grupo: \_\_\_\_\_

```
66.     case 0:
67.
68.     /* tratamiento hijo B. */
69.     hijoB();
70.     exit(EXIT_SUCCESS);
71.
72.     case 1:
73.
74.     /* tratamiento hijo C. */
75.     hijoC();
76.     exit(EXIT_SUCCESS);
77.
78.     }
79.
80. } else if (pid > 0 && i == 1) {
81.
82.     /* Tratamiento del Padre */
83.
84.     close(fds_pipe1[0]); // CUATRO
85.     close(fds_pipe3[1]);
86.
87.     fd = open ("entrada.txt", O_RDONLY);
88.     if ( fd == -1 ) {
89.         printf ("error al abriir archivo entrada \n");
90.         exit();
91.     }
92.     do {
93.
94.         i = read(fd, buf, sizeof(buf));
95.
96.         // se sale si llega exit o archivo vacío
97.         if (I == 0) || (strncmp("exit\n", buf, strlen("exit\n")) == 0)
98.             exit_flag = 1;
99.
100.         write(fds_pipe1[1], buf, strlen(buf)); // CINCO
101.
102.         read(fds_pipe3[0], buf, sizeof(buf)); // SEIS
103.
104.         write(STDOUT_FILENO, buf, strlen(buf));
105.
106.     } while (numbytes > 0 && !exit_flag);
107.
```

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

```
108.     } else if (ret == -1) {
109.
110.         perror("fallo en fork");
111.         exit(EXIT_FAILURE);
112.     }
113. }
114.
115. // El padre cierra la tubería antes de esperar y salir
116.
117. close(fds_pipe1[1]); // SIETE
118. close(fds_pipe3[0]);
119.
120. do {
121.
122.     pid = wait(NULL);
123.
124. } while (pid > 0);
125.
126. }
```

Ejercicio 5 ( 2 puntos).

Escriba un programa `ls`. Dicho programa toma como parámetro el nombre del directorio de entrada e imprime por pantalla una lista de archivos en ese directorio, uno por línea, de forma similar al comando `"ls -l"`. La impresión de fichero no seguirá ningún orden concreto. Si el directorio no existe dará un error y termina. La función devolverá -1 en caso de error, y 0 si todo funciona correctamente.

Adicionalmente, se desea implementar un programa `cat`. Este programa recibe como argumento de entrada un nombre de fichero, comprueba si existe e imprime su contenido por pantalla. La función devolverá -1 en caso de error, y 0 si todo funciona correctamente.

Se pide:

- a) Escriba el pseudocódigo de la función `ls` definida arriba, de forma que se cumpla la interfaz y funcionamiento descritos.
- b) Desarrolle, en lenguaje C, el pseudocódigo del apartado anterior.
- c) Escriba el pseudocódigo de la función `cat` definida arriba, de forma que se cumpla la interfaz y funcionamiento descritos.
- d) Desarrolle, en lenguaje C, el pseudocódigo del apartado anterior.
- e) Si queremos compilar el programa `ls.c` y obtener como salida el ejecutable `ls`. ¿Qué comando debemos usar?

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

- f) Escriba un comando en Linux que: liste las entradas dentro del directorio actual , filtre si existe el archivo "kk.c" con el comando grep y se lo pase al al comando cat para que lo muestre por pantalla.

Solución: (explicación: la parte azul es obligatoria, la parte roja puede considerarse opcional)

a)

Programa ls

Abrir directorio\_entrada y comprobar error de apertura.

Devolver -1 si procede.

Mientras haya ficheros en el directorio:

Leer entrada de directorio.

Imprimir el nombre del fichero

b)

Programa ls

```
#include <sys/types.h> // opendir, readdir
```

```
#include <dirent.h> // opendir, readdir
```

```
#include <stdlib.h> // NULL
```

```
#include <unistd.h> // write
```

```
#include <string.h> // strlen
```

```
int main (int argc, char *argv[]) {
```

```
    struct dirent *entrada;
```

```
    DIR *fd1;
```

```
    fd1 = opendir(argv[1]);
```

```
    if(fd1 == NULL) {
```

```
        return -1;
```

```
    }
```

```
    int c, r1, r2;
```

```
    do{
```

```
        entrada = readdir(fd1);
```

```
        if(entrada != NULL) {
```

```
            write(STDOUT_FILENO, entrada->d_name, strlen(entrada->d_name));
```

```
            write(STDOUT_FILENO, "\n", 2);
```

```
            // printf("%s\n", entrada->d_name); // faltaría #include <stdio.h>
```

```
        }
```

```
    }while(entrada != NULL);
```

```
    closedir(fd1);
```

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

```
    return 0;  
}
```

c)

Abrir fichero\_entrada y comprobar error de apertura.

Devolver -1 si procede.

Mientras queden caracteres en fichero\_entrada:

Leer carácter de fichero\_entrada.

Escribir carácter en salida estándar.

Comprobar lectura/escritura.

Cerrar fichero y devolver -1 si procede.

Cerrar fichero.

Devolver 0.

d)

```
#include <sys/types.h> // open
```

```
#include <sys/stat.h> // open
```

```
#include <fcntl.h> // open
```

```
#include <unistd.h> //STDOUT_FILENO
```

```
int main (int argc, char *argv[]) {
```

```
    int fd1 = open(argv[1], O_RDONLY);
```

```
    if(fd1 < 0) {
```

```
        return -1;
```

Examen Extraordinario de Sistemas Operativos - Mayo de 2015  
Grado en Ingeniería Informática

Nombre:

Grupo:

```
}

char c;
int r1, r2;

do{
    r1 = read(fd1, &c, 1);
    if(r1 < 0) {
        close(fd1);
        return -1;
    }
    if(r1 != 0) {
        r2 = write(STDOUT_FILENO, &c, 1);
        if(r2 < 0) {
            close(fd1);
            return -1;
        }
    }
}while(r1 > 0);

close(fd1);

return 0;
}
```

e)

```
gcc ls.c -o ls
```

f)

```
cat < 'ls ./ | grep "kk.c"'
```