

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.

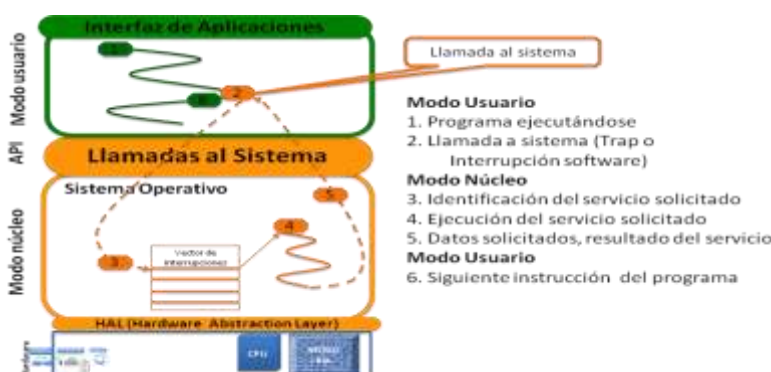
1. Suponga un puesto de trabajo donde debe atender a clientes que le hacen consultas por teléfono y a los que debe responder de forma casi inmediata. Para ello debe llevar a cabo acciones y consultas como usuario en un sistema informático. Escoja el tipo de sistema operativo con el que dotaría a su máquina entre un sistema por lotes multiprogramado, un sistema multiprogramado puro o un sistema de tiempo compartido, justifique su selección.

(0,75 puntos)

1	<p>Las necesidades de este puesto de trabajo conlleva que el usuario trabaje interactivamente con el computados, además será necesario la ejecución de varias aplicaciones concurrentemente y respuestas en breve espacio de tiempo.</p> <p>En los sistemas por lotes se encolarían las peticiones o solicitudes de ejecución y no se dispondrían de interacción usuario-máquina, ya que estos sistemas son manejados por un operador especializado. El tiempo de respuesta en un sistema por lotes depende de los trabajos encolados y por tanto no sería una buena opción.</p> <p>En los sistemas multiprogramados puros, se pueden ejecutar varias aplicaciones concurrentes pero no se considera la interacción usuario maquina.</p> <p>Debería ser dotado con un sistema de tiempo compartido. Estos sistemas pueden ejecutar varias aplicaciones a la vez y obtener respuestas de la maquina en un breve espacio de tiempo, ya que como usuario se interactúa directamente con la maquina.</p>
---	--

2. a) Indique y justifique las diferencias que existen entre órdenes del Shell y llamadas al sistema operativo, tanto desde el punto de vista conceptual como funcional.
- b) Enumere la secuencia de acciones que se lleva a cabo en un sistema cuando una aplicación de usuario solicita una llamada al sistema, indique cuales de ellas se realizan en modo núcleo.

(0,75 puntos)

2	<p>a) Las Llamadas al Sistema son el mecanismo mediante el cual las aplicaciones acceden a los servicios del Sistema Operativo y por tanto a todos los recursos de la máquina.</p> <p>Las órdenes del Shell son programas de usuario que utilizan llamadas al sistema para dar servicios a los usuarios. Las órdenes se invocan normalmente desde el terminal o de un Shell script, mientras que las llamadas forman parte del código de las aplicaciones, del código de las librerías y del código de las propias utilidades del Shell. El Shell forma parte de las herramientas básicas que se suministra junto con el s.o. en los sistemas UNIX.</p> <p>b)</p> 
---	--

3. Dado el siguiente código en C y POSIX correspondiente a un proceso que denominaremos Proc:

```

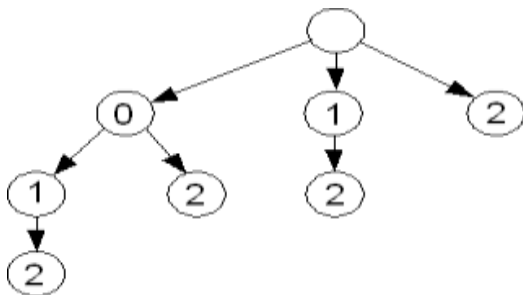
1 #include ... //los necesarios
2 #define N 3
3
4 main() {
5     int i = 0;
6     pid_t pid_a;
7
8     while (i<N)
9     { pid_a = fork();
10      switch (pid_a)
11      { case -1:
12          printf("Error creating child...\n");
13          break;
14          case 0:
15              printf("Message 1: i = %d \n", i);
16              if (i < N-1) break;
17              else exit(0);
18          default:
19              printf("Message 2: i = %d \n", i);
20              while (wait(NULL)!=-1);
21      }
22      i++;
23  }
24  printf("Message 3: i=%d\n",i);
25  exit(0);
26 }
27
```

- a) Represente el árbol de procesos generado al ejecutarlo e indique para cada proceso el valor de la variable “i” en el instante de su creación.
- b) Indique de forma justificada si existe o no la posibilidad de que los hijos creados queden huérfanos o zombies.

(1,0 punto)

3

- a) Se crean un total de 8 procesos con la estructura en árbol que se muestra en la figura, los números indican el valor de la variable i en el instante de su creación.



b)

No hay posibilidad de zombies ni de huérfanos al ejecutar este código ya que los procesos que actúan como padres siempre esperan a los procesos hijos con la sentencia:

```
while (wait(NULL) != -1);
```

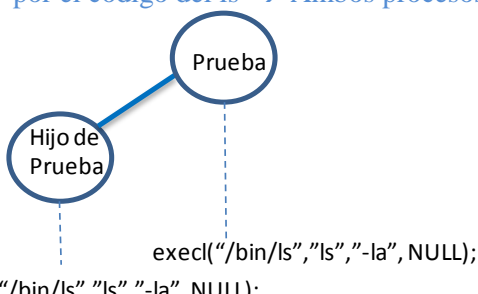
Esperando la finalización de todos los hijos se garantiza la ausencia de huérfanos. Dado que la llamada wait() se ejecuta inmediatamente después de crear a los procesos hijos se garantiza también la ausencia de zombies

4. Suponga que el siguiente c3digo en C y POSIX correspondiente a un proceso denominado Prueba que se ejecuta con 3xito:

```
1 #include ... //los necesarios
2
3 int main()
4 { pid_t val;
5   printf("Mensaje 1: antes de exec()\n");
6   fork();
7   execl("/bin/ls", "ls", "-la", NULL);
8   val = fork();
9   if (val==0)
10  {execl("/bin/ps", "ps", "-la", NULL);
11    printf("Mensaje 2: despu3s de exec()\n");
12    exit(1)
13  }
14
15  printf("Mensaje 3: antes del exit()\n");
16  exit(0);
17 }
```

- a) Indique de forma justificada el n3mero de procesos que se crean al ejecutar Prueba y el parentesco entre ellos.  
b) Indique de forma justificada que mensajes e informaci3n se muestra por pantalla como consecuencia de ejecutar Prueba.

(1,0 punto)

4	<p>a) Al ejecutar Prueba con 3xito se crean 3nicamente dos proceso, el proceso Prueba y un proceso hijo (creado en la l3nea 6 con el fork). Ambos procesos padre e hijo ejecutar3an la l3nea 7, llevando a cabo un cambio en su imagen de memoria. Con ello se cambiar3a, el c3digo mostrado por el c3digo del ls → Ambos procesos ejecutan el "ls -la"</p>  <p>c) Por pantalla aparece: Mensaje 1: antes de exec() El resultado de ejecutar la orden "ls -la" El resultado de ejecutar la orden "ls -la"</p>
---	--

5. Dado el siguiente c3digo del proceso Ejemplo:

```
1 #include ... //los necesarios
2
3 int main(void)
4 { int val;
5   printf("Message 1\n");
6   val=fork();
7   /** Aqu3 deben ir sus modificaciones **/
8   sleep(5);
9   printf("Message 2\n");
10  return 0;
11 }
```

- a) Indique qu3 modificaciones ser3an necesarias introducir en el c3digo anterior para que el proceso hijo se quede hu3rfano y sea adoptado por el proceso INIT (). (Nota: Utilice el sleep() e instrucciones en C).

- b) Indique qué modificaciones serían necesarias introducir en el código anterior para que el proceso hijo se quede zombie durante un tiempo. (Nota: Utilice el sleep() e instrucciones en C).
- c) Indique de forma justificada en qué instrucciones del código propuesto puede asegurarse que ocurrirá un cambio de contexto en la CPU y en cuáles se producirá un cambio de estado para el proceso Ejemplo.

(1,0 puntos)

5	<p>a)</p> <p>Un proceso queda huérfano cuando su padre finaliza antes que él. Para garantizar la finalización del proceso padre antes que su hijo, el código de la línea 6 podría ser similar al siguiente: Opción 1: if (val&gt;0) exit(0); /** Padre finaliza para dejar al hijo huérfano **/ Opción 2: if (val== 0) sleep(30); /** sleep() por un tiempo mucho mayor que el proceso padre**/</p> <p>b)</p> <p>Un proceso queda zombie cuando finaliza antes que su padre y su padre no está ejecutando wait(). Para garantizar la finalización del proceso hijo antes que su padre, el código de la línea 6 podría ser similar al siguiente: Opción 1: if (val==0) exit(0); /**hijo finaliza antes que el padre y se queda huérfano**/ Opción 2: if (val&gt; 0) sleep(30); /** sleep() por un tiempo mucho mayor que el proceso hijo**/</p> <p>c)</p> <p>En la Línea 8 cuando se ejecuta el sleep(5) el proceso se suspende voluntariamente con lo cual abandona la CPU y la CPU se queda ociosa. A continuación actuaría el sistema operativo y en concreto su planificador a corto plazo para seleccionar un proceso de la cola de preparados, lo cual generaría cambios de contexto. Siempre que un proceso de usuario necesita la intervención del sistema operativo ocurren cambios de contexto. En el código propuesto esto puede ocurrir en la línea 6 (llamada fork()) y en las líneas 5 y 9 ya que requiere el acceso a un dispositivo de E/S. Estas instrucciones de E/S podrían generar cambios de contexto, aunque dada el tipo de instrucción de que se trata (mostrar por pantalla) es probable que se atienda la solicitud del proceso sin llegar a suspenderlo.</p>
---	--

6. Dado el siguiente código que intenta solucionar el problema de la condición de carrera, y asumiendo que hay más de un hilo ejecutando concurrentemente el código de la función agrega, indique cuáles de las siguientes sentencias son verdaderas y cuáles falsas (Nota: Un error penaliza una respuesta correcta)

```

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

```

(1,0 puntos)

6	<table> <tr> <th>V/F</th><th></th></tr> <tr> <td>V</td><td>La línea 4 representa el protocolo de entrada a la sección crítica.</td></tr> <tr> <td>F</td><td>La solución planteada no garantiza que el código esté libre de condiciones de carrera</td></tr> <tr> <td>V</td><td>La solución planteada garantiza que el código está libre de condiciones de carrera, ya que, realmente, el acceso a la variable global V se realiza de forma secuencial</td></tr> <tr> <td>V</td><td>Si intercambiamos las líneas 4 y 5 , y las 7 y 8, la solución propuesta proporciona un código libre de condiciones de carrera</td></tr> <tr> <td>V</td><td>La función Test_and_Set(&amp;llave) consulta y cambia el valor de llave de forma atómica</td></tr> </table>	V/F		V	La línea 4 representa el protocolo de entrada a la sección crítica.	F	La solución planteada no garantiza que el código esté libre de condiciones de carrera	V	La solución planteada garantiza que el código está libre de condiciones de carrera, ya que, realmente, el acceso a la variable global V se realiza de forma secuencial	V	Si intercambiamos las líneas 4 y 5 , y las 7 y 8, la solución propuesta proporciona un código libre de condiciones de carrera	V	La función Test_and_Set(&llave) consulta y cambia el valor de llave de forma atómica
V/F													
V	La línea 4 representa el protocolo de entrada a la sección crítica.												
F	La solución planteada no garantiza que el código esté libre de condiciones de carrera												
V	La solución planteada garantiza que el código está libre de condiciones de carrera, ya que, realmente, el acceso a la variable global V se realiza de forma secuencial												
V	Si intercambiamos las líneas 4 y 5 , y las 7 y 8, la solución propuesta proporciona un código libre de condiciones de carrera												
V	La función Test_and_Set(&llave) consulta y cambia el valor de llave de forma atómica												



7. Dado el siguiente código cuyo archivo ejecutable ha sido generado con el nombre “Hilos1”.

```

1 #include <stdio.h>
2 #include <pthread.h>
3 #define DosSegons 2000000
4 pthread_t H1,H2;
5 pthread_attr_t atr;
6
7 void *FaenaB (void *P)
8 { char * texto =(char *) P;
9   usleep(DosSegons);
10  printf("%s\n",texto);
11 }
12
13 void *FaenaA (void *T)
14 { printf("Texto:\n");
15   pthread_create(&H2, &atr, FaenaB, T);
16   usleep(DosSegons);
17   pthread_join(H2, NULL);
18 }
19
20 int main()
21 { pthread_attr_init(&atr);
22   pthread_create(&H1, &atr, FaenaA, "Examen de FSO");
23   usleep(DosSegons);
24   pthread_join(H1, NULL);
25   pthread_join(H2, NULL); /*Hilo creado en FaenaA*/
26   return(0);
27 }
```

- Indique de forma justificada las cadenas que imprime el programa en la Terminal tras su ejecución.
- Indique de forma justificada el tiempo aproximado que tardará en ejecutarse el programa.
- Justifique si en la función main(), sería correcto o no esperar al hilo H2 que es creado al ejecutarse la función FaenaA.
- Durante la ejecución del código, ¿cuántos hilos hay activos cuando escribe "Texto:" en el terminal y cuantos cuando escribe "Examen de FSO"? Justifique su respuesta.

(1,0 puntos)

7	a) El programa imprime dos líneas: primera línea "Texto: ", y luego (después de esperar dos segundos) "Examen de FSO"
	b) Tiempo aproximado dos segundos. Los tres hilos (main, H1 y H2) hacen un poco de trabajo (llamadas al sistema para escribir un texto o para crear un hilo) y luego se suspenden durante dos segundos. Es bastante probable que los hilos se encuentren suspendidos a la vez, solapándose sus tiempos de suspensión.
	c) A diferencia de los procesos del sistema operativos, los hilos no tienen relación de parentesco padre-hijo. La función <i>main</i> puede esperar la terminación de un hilo que no ha creado directamente. De todas formas, la llamada a <i>pthread_join</i> de la línea 25 no tendrá efecto, dado que se espera previamente con una llamada bloqueante <i>pthread_join</i> al ejecutar FaenaA y por tanto H2 ya habrá finalizado cuando se ejecute la línea 25.
	d) Mientras se imprime “Texto:” hay dos hilos activos el hilo que ejecuta la función <i>main</i> y H1. Mientras se imprime “Mensaje de FSO” hay tres hilos activos H1, H2 y el que ejecuta <i>main</i> .



9. Dado el siguiente programa, donde se utilizan tres semáforos, con los hilos agrega y resta. Indique de forma justificada para cada una de los valores de  $x$  e  $y$  propuestos, si puede haber o no condición de carrera, el estado en que quedan los hilos creados y el valor final de  $V$  que se escribe en la línea 12:

<pre>#include ... //suponga los includes int V = 100; sem_t sem, sum, res;  void *agrega (void *argumento) {     int cont;     for (cont=0; cont&lt;100; cont++) {         sem_wait(&amp;sum);         sem_wait(&amp;sem);         V = V + 1;         sem_post(&amp;sem);         sem_post(&amp;res);     } }  void *resta (void *argumento) {     int cont;     for (cont=0; cont&lt;100; cont++) {         sem_wait(&amp;res);         sem_wait(&amp;sem);         V = V - 1;         sem_post(&amp;sem);     } }</pre>	
<pre>1. int main (void) { 2.     pthread_t hiloSuma, hiloResta, hiloInspeccion; 3.     pthread_attr_t attr; 4.     int x,y; 5.     pthread_attr_init(&amp;attr); 6.     sem_init(&amp;sem,0,1); 7.     sem_init(&amp;sum,0,x); 8.     sem_init(&amp;res,0,y); 9.     pthread_create(&amp;hiloSuma, &amp;attr, agrega, NULL); 10.    pthread_create(&amp;hiloResta, &amp;attr, resta, NULL); 11.    usleep(80000000); //suficiente para ejecutar agrega y resta 12.    fprintf(stderr, "-----&gt; VALOR FINAL: V = %d\n", V); 13.    exit(0); 14. }</pre>	

- a) Suponga que  $x=5$  e  $y=1$ .
- b) Suponga que  $x=500$  e  $y=1$ .
- c) Suponga que  $x=20$  e  $y=5$ .

(1,5 puntos = 0,9+0,3+0,3)

9	<p>a) <code>sem_init(&amp;sum,0,5);</code> El semáforo <i>sum</i> se inicializa a 5 (<math>\text{sum}=5</math>).  <code>sem_init(&amp;res,0,1);</code> El semáforo <i>res</i> se inicializa a 1 (<math>\text{res}=1</math>).  Este código sólo realiza operaciones de decremento sobre <i>sum</i> <math>\rightarrow P(\text{sum})</math> y no se realizan operaciones de incremento <math>V(\text{sum})</math>. Se hace una operación <math>P(\text{sum})</math>, en <i>agrega</i> antes de incrementar <math>V(V=V+1)</math>, con lo que el número de veces que se puede incrementar <math>V</math> será el valor inicial del semáforo <i>sum</i> <math>\rightarrow V</math> se incrementa en 5. Después de incrementar <math>V</math>, se hace una operación de incremento sobre el semáforo <i>res</i> <math>\rightarrow V(\text{res})</math>.  Se realiza una operación <math>P(\text{res})</math> antes de restar 1 a <math>V(V=V-1)</math>, en la función <i>resta</i>, por tanto se pondrán realizar tantas restas como sumas más el valor inicial de <i>res</i> que es 1 <math>\rightarrow V</math> se decrementa en 6.  Por tanto el valor final de <math>V</math> será 99 (<math>V=100+5-6</math>).  Además <i>hiloSuma</i> se quedarán suspendidos en el semáforo <i>sum</i>, mientras que <i>hiloResta</i> se quedará suspendido en el semáforo <i>res</i>.  El semáforo <i>sem</i> garantiza la exclusión mutua por lo que no hay condiciones de carrera sobre <math>V</math> y el resultado siempre será el mismo.</p>
	<p>b) El semáforo <i>sum</i> se inicializa a 500 (<math>\text{sum}=500</math>). El semáforo <i>res</i> se inicializa a 1 (<math>\text{res}=1</math>).  El bucle <i>for</i> está limitado a 100 vueltas, por tanto ahora no se bloquearán los hilos, mbos harán 100 operaciones y por tanto el valor final de <math>V</math> será 100.</p>
	<p>c) El semáforo <i>sum</i> se inicializa a 20 (<math>\text{sum}=20</math>). El semáforo <i>res</i> se inicializa a 5 (<math>\text{res}=5</math>).  Es el mismo caso que en el apartado a. Se realizaran un número de restas que superara en 5 al número de sumas <math>\rightarrow V=95</math></p>