Puntuación: para cada pregunta *• 100% si está bien contestada* *• -50% si no está bien contestada* *• 0 si se deja en blanco*	*UO:* *Nombre:*	*Modelo 1*

Cuestiones Verdadero / Falso

IMPORTANTE: Este examen consta de preguntas verdadero/falso agrupadas en grupos de cuatro. Sin embargo **el valor de cada pregunta es independiente de las otras tres**. Debes contestar a cada pregunta independientemente, con VERDADERO o FALSO.

[Arranque del Sistema]

1. El iniciador ROM es el encargado de crear las estructuras de datos del sistema Operativo.
2. El boot localizado en el sector de arranque es proporcionado por el fabricante del hardware.
3. *El iniciador ROM es proporcionado por el fabricante del hardware.
4. El test del Hardware lo realiza el boot.

[Activación del Sistema Operativo]

5. El SO se activa siempre que un proceso intenta acceder a una dirección de memoria principal.
6. *El SO se activa siempre que un proceso se sincroniza con otro proceso de la misma máquina.
7. *El SO se activa siempre que se produce un error de ejecución como “null pointer Exception”.
8. *El SO se activa siempre que el procesador recibe una interrupción externa.

[Activación del Sistema Operativo]

9. *Las llamadas al sistema son funciones de la API que contienen una instrucción que provoca el salto al Sistema Operativo.
10. *Cuando se activa el Sistema Operativo el procesador pasa a ejecución en modo núcleo.
11. *En una llamada al sistema el proceso que llama deja los parámetros en los registros del procesador para que los reciba el sistema operativo
12. Los dispositivos externos se comunican con el Sistema Operativo a través de llamadas al sistema.

[Interfaz con el Sistema Operativo]

13. Los programas se comunican con todos los servicios del Sistema Operativo a través de las órdenes del Shell.
14. *Las interfaces gráficas que proporcionan los sistemas son programas que usan llamadas al sistema para pedir servicios al Sistema Operativo.
15. *Las interfaces de línea de órdenes que proporcionan los sistemas son programas que usan llamadas al sistema para pedir servicios al Sistema Operativo.

16. Las interfaces de usuario final, como el intérprete de órdenes o la interfaz gráfica, se ejecutan en supervisor puesto que forman parte del sistema operativo.

[Evolución de los Sistemas Operativos]

17. Los primeros programas de control antecesores de los SSOO surgen en los años 60 y se denominan "monitor residente".
18. *La multiprogramación y el tiempo compartido se comenzó a desarrollar en los años 60
19. *Ken Thompson y Dennis Ritchie diseñaron UNIX y lo implementaron en lenguaje C.
20. Ada Lovelace diseñó un sistema operativo para las primeras máquinas de su época

[Multiprogramación]

21. *Uno de los fundamentos de la multitarea es la ejecución del procesador de manera simultánea con la E/S (el intercambio de información entre memoria y los dispositivos).
22. *Uno de los fundamentos de la multitarea es el reparto del uso del procesador entre varios procesos.
23. El bloque de control de proceso se guarda en memoria dentro de la imagen del proceso.
24. *El BCP de un proceso que está en estado listo contiene los valores de los registros del procesador de la última vez que abandonó la CPU.

[Ciclo de vida de un proceso] Sistema con modelo de 7 estados

25. Los procesos en estado listo pasarán a bloqueados si se produce algún evento para ello.
26. Hay una única cola que contiene todos los procesos bloqueados en espera de cualquier evento.
27. *Las colas de los procesos listos y bloqueados contienen PCBs (o enlaces a ellos)
28. Un proceso en estado bloqueado suspendido puede pasar a ejecución en cuanto lo carguen en memoria principal.

[Gestión de una interrupción] Supongamos un SO que se ejecuta en el Espacio de Direcciones del Proceso (dentro de los procesos)

29. Cuando se produce una interrupción el hardware guarda todos los registros del procesador en la pila.
30. *Cuando se produce una interrupción el hardware cambia el procesador de modo usuario a modo supervisor.
31. *Cuando se produce una interrupción el hardware sustituye los punteros a la pila del proceso de usuario por los punteros a la pila de control del sistema operativo.
32. *Cuando se produce una interrupción que implica un cambio de proceso el sistema guarda todas las entradas de la pila de control en el PCB del proceso que abandona el procesador.

[Cambio de contexto]

33. En un sistema operativo que se ejecuta como parte de los procesos de usuario, cada interrupción produce cambio de contexto
34. En un sistema operativo que se ejecuta como parte de los procesos de usuario, cada interrupción produce un cambio de proceso.
35. *En un sistema operativo que se ejecuta como parte de los procesos de usuario, el cambio de contexto sólo se realiza si hay cambio de proceso.

36. *En un sistema operativo que se ejecuta como parte de los procesos de usuario un cambio de contexto implica guardar los valores de los registros del procesador en el PCB del proceso que abandona y cargar los valores de éstos a partir del PCB del proceso que accede al procesador

[Hilos]

37. *Los hilos de un mismo proceso comparten todo el espacio de direcciones asignado al proceso.
38. *Los hilos de un mismo proceso tienen cada uno su propia pila de ejecución.
39. Los hilos de un mismo proceso comparten los valores de los registros del procesador guardados en el BCP.
40. La implementación de hilos a nivel de usuario permite la ejecución simultánea de los hilos en procesadores diferentes.

[Políticas de planificación]

41. *En un sistema operativo que soporta hilos, son los hilos de manera individual los que cambian de estado, no los procesos globalmente.
42. Las políticas de planificación con prioridades dinámicas con envejecimiento pueden producir inanición.
43. *En un sistema que use turno rotatorio para todos sus procesos favorece la igualdad en el uso del procesador
44. *En Windows se usa una política de prioridades dinámicas en la que se favorece a los procesos con mucha entrada/salida.

[Planificación de procesos]

45. La política de planificación apropiativa permite intercambiar procesos temporalmente a memoria secundaria.
46. *La política de planificación a corto plazo elegida pretende optimizar parámetros de rendimiento del sistema en cuanto a la velocidad de ejecución de los procesos
47. En un sistema con política de planificación con turno rotatorio, los procesos abandonan la CPU en dos únicos casos: cuando se acaba el cuanto de tiempo o bien cuando finalizan
48. En una política de planificación por prioridades con envejecimiento, se intenta primar a los procesos con más e/s que a los procesos con más CPU

[Planificación en Multiprocesadores]

49. *Los sistemas operativos actuales utilizan multiprocesamiento simétrico y se ejecutan en todos los procesadores.
50. *En un sistema multiprocesador con cola única se favorece el equilibrio de carga.
51. *En un sistema multiprocesador con colas múltiples se favorece la afinidad del procesador.
52. *El uso de cola única en sistemas con mucha carga puede provocar cuello de botella.

[Concurrencia]

53. Si se desea comunicar procesos en la misma máquina resulta más rápido usar tuberías que otros mecanismos.
54. *Si se desea comunicar procesos en diferentes máquinas se pueden utilizar sockets.
55. *Los procesos independientes concurrentes pueden competir por un recurso común

56. La programación concurrente es cada vez menos necesaria debida al incremento de la velocidad de los procesadores

[Concurrencia] Sean dos hilos que usan una variable común para almacenar información compartida y necesaria por ambos de forma que uno de ellos no debe leer la información de la variable hasta que el otro haya escrito en ella el dato adecuado

57. Este problema se clasifica como el problema de "los lectores y escritores".
58. *Este problema se clasifica como el problema de "el productor - consumidor".
59. Este problema se clasifica como el problema de "cliente - servidor".
60. Este problema se clasifica como el problema de "recursos compartidos o problema de los filósofos".

[Concurrencia]

61. *El problema de las condiciones de carrera que puede producirse en algunas situaciones de programación concurrente deberá evitarse mediante la sincronización adecuada de los hilos o procesos implicados.
62. *El problema del interbloqueo que pueda producirse en algunas situaciones de programación concurrente deberá prevenirse eliminando alguna de las 4 condiciones necesarias para que exista.
63. La sección crítica es un fragmento de código que comparten todos los procesos o hilos que colaboran entre sí
64. *La exclusión mutua implica la ejecución de manera atómica de la sección crítica de cada proceso o hilo

[Mecanismos de comunicación]

65. Tuberías y memoria compartida son mecanismos de comunicación para procesos situados en máquinas diferentes.
66. *Entre hilos del mismo proceso no es preciso un mecanismo de comunicación gestionado por el sistema operativo.
67. *Semáforos y mutex sirven para sincronizar pero no para comunicar.
68. *Las variables condicionales pueden evitar interbloqueos.

[Semáforos y mutex]

69. *Si un hilo ejecuta un lock(m) sobre un mutex que inicialmente está a 0, el sistema operativo pasa al hilo al estado bloqueado.
70. Si un proceso ejecuta un wait(s) sobre un semáforo que inicialmente está a 1, el sistema operativo pasa al proceso al estado bloqueado.
71. Un hilo bloqueado en una cola de un mutex puede ser desbloqueado por otro hilo de otro proceso
72. *Un proceso bloqueado en una cola de un semáforo, puede ser desbloqueado por otro proceso que se tenga acceso a dicho semáforo.

[Señales]

73. *El estándar POSIX define señales como mecanismo de sincronización pero no ocurre lo mismo con la interfaz API de Windows.
74. *Las señales pueden funcionar de manera síncrona o asíncrona.

75. *Kill es una llamada al sistema que envía una señal a un proceso.
76. Las señales sirven tanto para comunicación como para sincronización.

[Tuberías]

77. *Las tuberías son útiles para dar solución a problemas de productores / consumidores
78. Las tuberías sólo sirven para comunicar información, habrá que incluir también un mecanismo de sincronización adicional.
79. Las tuberías sirven para sincronizar procesos situados en máquinas diferentes
80. Las tuberías sirven para comunicar procesos situados en máquinas diferentes

[Políticas de planificación] Round Robin puro, Q=2, proceso A empieza en 0 y necesita 6, B 1/3, C 3/8, D 5/2, E 7/3 (nótese que los momentos de comienzo no son correlativos)

81. *Te para el proceso C es 19.
82. Tw para el proceso B es 6.
83. *Tw para el proceso E es 8.
84. *Te para el proceso A es 13.

[Cambio de contexto] Dada la situación en la hoja adjunta:

85. *Se está procesando una interrupción en este momento.
86. *El proceso 18721 se va a ejecutar.
87. *El proceso 377 fue interrumpido cuando su contador de programa estaba en la posición 003884
88. Este sistema operativo funciona en modo Núcleo independiente.

[Llamadas al Sistema para creación de procesos] Sea el siguiente código con llamadas fork y exec. Sea P el proceso que ejecuta este código

```
int cuenta = 5;
if (fork() == 0)
{
    exec("fichero ejecutable");
    printf("%d ", cuenta);
}
else
{
    for (int i=0; i<2; i++)
       {
       fork();
       cuenta ++;
       printf("%d ", cuenta);
       }

}
```

89. *A partir del proceso P que ejecuta este código se generan 4 procesos más (sin contar a P)
90. Una posible salida por pantalla es "5 6 6 7 7 7"
91. Todos los procesos que se generan son unos hijos de P
92. *Una posible salida por pantalla es "6 7 7 6 7 7"

[Concurrencia] Se desea sincronizar dos procesos de tal modo que, como resultado de su ejecución, uno escriba en un fichero "Hola que tal" y el otro lea este texto y lo muestre por pantalla.

```
fd file;
char buffer[128];

file = fopen("r+", "file.dat");
if (fork() != 0){
       wait(sem1);
       buffer = "hola que tal";
       write(file,buffer);
       signal(sem1);
       signal(sem2);
}
else{
       wait(sem2);
       read(file,buffer);
       printf(buffer);

}
```

93. *El semáforo sem2 debe inicializarse a 0 y sem1 a 1 para obtener el resultado esperado
94. * Si se elimina el semáforo sem1, y se inicializa sem2 a 0, se obtiene el resultado esperado
95. Para que la sincronización sea correcta falta que el proceso hijo finalice con un signal (sem1)
96. *Si se sustituye el fichero por una tubería, y la operación read por una de lectura en la tubería y write por una de escritura en la tubería, no sería necesario ningún semáforo para lograr la sincronización

[Concurrencia] Sean los siguientes procesos de una cadena de montaje para una fábrica de sidra en la que, se ejecutan estos dos métodos en hilos diferentes.

```
void rellenarBotella(){
       repeat{
              wait(carril);
              botella = new Botella();
              botella.rellenarContenido();
              carrilCorcho.put(botella);
              signal(rellena);
       }
}
void corcharBotella(){
       repeat{
              wait(rellena);
              botella = carrilCorcho.getBotella();
              botella.ponerCorcho();
              carrilEtiqueta.put(botella);
              signal(carril);
       }
}
```

97. *Si inicialmente ambos semáforos tienen valor 0 se produce interbloqueo
98. Si inicialmente ambos semáforos tienen valor 1 se produce interbloqueo
99. Si se inicializa carril a 10 y rellena a 0, hasta que las 10 primeras botellas no sean corchadas no se puede rellenar ninguna más.
100. *Si se inicializa carril a 10 y rellena a 0, como máximo habrá 10 botellas rellenas sin corchar