



Puntuación: para cada pregunta <ul style="list-style-type: none">• 100% si está bien contestada• -50% si no está bien contestada• 0 si se deja en blanco	UO: Nombre:	Modelo 0
---	------------------------------	-----------------

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.

[T1-Arranque del Sistema]

1. La primera instrucción que se ejecuta en la máquina corresponde a una instrucción del sistema operativo
2. *Desde el iniciador ROM se carga el sector de arranque
3. La parte residente del Sistema Operativo es la que se localiza en el disco duro
4. El iniciador ROM es el encargado de crear las estructuras de datos del sistema Operativo.

[T1-Activación del Sistema]

5. *Cada vez que se produce un tic de reloj se activa el sistema operativo
6. Cada vez que se realiza un acceso a una dirección de memoria se activa el sistema operativo

[T1-Activación del Sistema]

7. *Cuando se realiza una llamada al sistema se produce una interrupción software
8. Cuando se ejecuta una instrucción TRAP se produce una interrupción externa
9. *Cuando un proceso finaliza la ejecución de su código se produce una activación del sistema operativo

[T1-Cómo se activa el SO]

10. Cuando se activa el Sistema Operativo el hardware guarda todos los registros del procesador
11. *La gestión de la interrupción se realiza en modo núcleo (o supervisor)
12. *Las funciones de la API del Sistema Operativo contienen instrucciones tipo TRAP que provocan la activación del Sistema Operativo

[T1-Interfaz con el Usuario]

13. La interfaz del Sistema Operativo con el usuario forma parte del núcleo del Sistema Operativo
14. *La interfaz del Sistema Operativo con el usuario se ejecuta en modo usuario
15. *Win32 es una interfaz del Sistema Operativo con los programas
16. *POSIX es un estándar que define como debe ser una interfaz del Sistema Operativo con los programas

[T1-Historia de los Sistemas Operativos]

17. En los años 50 aparece la gestión de la multiprogramación
18. *MULTICS es el predecesor de Unix
19. *Unix es el primer sistema operativo portable a múltiples plataformas

[T2-Funciones de gestión de procesos]

20. *La creación de procesos es tarea del gestor de procesos
21. *La sincronización de procesos es tarea del gestor de procesos
22. *La planificación de procesos es tarea del gestor de procesos



23. *La creación de hilos es tarea del gestor de procesos

[T2-Multitarea]

- 24. Las operaciones de e/s se ejecutan en el procesador
- 25. Multitarea y multiprogramación son conceptos equivalentes
- 26. Multitarea y Tiempo compartido son conceptos equivalentes
- 27. El tiempo compartido se basa en la alternancia de los programas entre instrucciones de procesamiento y operaciones de e/s

[T2-Elementos de un proceso]

- 28. El Bloque de Control de un proceso guarda, entre otras cosas, código, datos y pila de ese proceso
- 29. *El PID, la prioridad y el estado del procesador se guardan en el BCP de un proceso

[T2-Ciclo de vida de un proceso]

- 30. *Los procesos en estado listo esperan por el uso del procesador
- 31. En un sistema que usa una política de planificación cíclica, un proceso en estado ejecutando que se le acaba el cuanto de tiempo pasa a estado bloqueado
- 32. En una política de planificación con prioridades expulsivas, si llega un proceso con mayor prioridad que el que se está ejecutando, el sistema lo expulsa y lo bloquea hasta que el de mayor prioridad finalice
- 33. Los procesos en estado bloqueado esperan por que quede el procesador libre

[T2-Gestión de Interrupción]

- 34. *Cuando hay una interrupción los registros del procesador se guardan en la pila del sistema
- 35. *Mientras se ejecuta el sistema operativo el procesador pasa a modo núcleo
- 36. Siempre que hay una interrupción se guarda el estado del procesador en el BCP del proceso interrumpido

[T2-Cambio de proceso]

- 37. Toda interrupción de reloj implica un cambio de proceso
- 38. Activar un proceso supone sacarlo de listos y pasarlo a ejecución
- 39. *La pila del sistema almacena el estado del procesador del proceso interrumpido en el momento de la interrupción

[T2-Hilos]

- 40. Hilos de la misma tarea comparten contador de programa
- 41. *Hilos de la misma tarea comparten código
- 42. Hilos de la misma tarea comparten pila de ejecución
- 43. *Hilos de la misma tarea comparten BCP

[Hilos]

- 44. *Hilos de la misma tarea comparten datos
- 45. El cambio entre hilos tiene menos coste si los hilos están gestionados dentro del sistema operativo que si están implementados a nivel de biblioteca
- 46. *El uso de hilos implementados a nivel de biblioteca no aprovecha las ventajas de disponer de varios procesadores

[T2-Planificación]

- 47. *La elección de un proceso de entre los procesos listos para pasar a ejecución constituye la planificación a corto plazo



- 48. La elección de un proceso de entre los procesos listos – suspendidos para pasar a ejecución constituye la planificación a medio plazo
- 49. Las prioridades dinámicas con envejecimiento sirven para favorecer a los procesos interactivos
- 50. *En una política de planificación de turno rotatorio o cíclica, si un proceso en estado ejecutando agota su cuanto es expulsado al final de la cola de listos
- 51. *Favorecer los procesos que realizan e/s incrementa el rendimiento del sistema

[T2-Ejercicio]

Sea un sistema en el que se quieren ejecutar 4 procesos, y se utiliza una planificación con prioridad expulsiva y cuanto de tiempo de 2 unidades. Sean los siguientes datos:

Proceso	Instante llegada	Prioridad Prioridad 2 > 1	Tiempo de CPU
A	0	1	7
B	3	1	4
C	6	2	6
D	9	2	3

- 52. *El tiempo de espera del proceso B es de 12 unidades
- 53. *El tiempo de retorno del proceso A es de 20 unidades
- 54. En el instante 8 obtiene el procesador el proceso D
- 55. *El proceso C finaliza en el instante 14
- 56. En el instante 13 la cola de listos tiene los procesos D B A, siendo D el primero de la cola y A el último
- 57. *En el instante 11 en la cola de listos están C, A , B siendo C el primero de la cola y B el último.

[T2- Creación de procesos en Posix]

Sea el siguiente pseudo-código, que **será ejecutado por un proceso llamado P.**

```
for (i = 1 to 4)
{
    pid = fork()
    If ( pid == 0)
        exec("ls");
    else
        printf(i);
}
```

- 58. *La ejecución de P genera 4 procesos
- 59. La ejecución de P genera 16 procesos
- 60. La llamada Exec crea un nuevo proceso que ejecuta el fichero ejecutable ls
- 61. *Todos los procesos que se generan son hijos de P

[T3-Concurrencia]

- 62. Las señales existen en Windows pero no existen en el estándar POSIX
- 63. *Ficheros, tuberías, memoria compartida y mensajes son mecanismos de comunicación
- 64. *Leer de una tubería vacía tiene el mismo efecto que hacer *wait* sobre un semáforo con valor 0
- 65. *Los sockets son implementaciones de mecanismos de comunicación entre procesos que están en máquinas diferentes.



66. *Los procesos que hacen wait sobre un semáforo con valor negativo pasan al estado bloqueado y entran en una cola

67. Para comunicar información entre dos tareas que se pueden ejecutar en la misma máquina, la forma más eficiente es crear dos procesos y utilizar memoria compartida

Dado el siguiente código que ejecutarán los **5 filósofos** que comparten mesa. Siendo los semáforos `palillo[k] = 1` en su valor inicial para todos los `k` filósofos:

```
Filósofo (K){  
  for(;;){  
    P(mesa);  
    P(palillo[k]);  
    P(palillo[k+1] % 5);  
  
    come();  
  
    V(palillo[k]);  
    V(palillo[k+1] % 5);  
    V(mesa);  
  
    piensa();  
  }  
}
```

68. *Si el valor inicial de mesa es 5 se puede producir interbloqueo

69. Si el valor inicial de mesa es 4 se puede producir interbloqueo

70. *Si el valor inicial de mesa es 1 los filósofos comen en exclusión mutua

Se desea realizar el módulo de un vector

$$| V | = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

Para ello se divide el problema en `n` tareas, cada una realiza el cuadrado de una de las `n` componentes del vector, y una tarea adicional que recibe los resultados y calcula la raíz cuadrada.

71. *Resulta más eficiente utilizar hilos que procesos para cada una de las tareas

72. *Serán necesarios `n` semáforos para sincronizar las tareas

73. *Se trata de un problema tipo productor/consumidor

74. Basta con un semáforo para sincronizar las tareas

75. Para que dos procesos sean concurrentes deben ejecutarse en la misma máquina

76. Siempre que se comparte un recurso es necesario un mecanismo de sincronización

77. *Para la resolución de problemas del tipo productor-consumidor son necesarios mecanismos de sincronización

78. El problema de los filósofos fue enunciado por Peter Denning

79. *Cuándo sólo hay accesos para lectura a un recurso común por parte de varios procesos, no se producen condiciones de carrera

80. *El problema del acceso de lectores y escritores a un mismo recurso debe ser sincronizado para evitar condiciones de carrera

81. *Los semáforos permiten el acceso a un recurso en exclusión mutua



Dado el siguiente código

[Sincronización]

Sean los siguientes procesos que se ejecutan concurrentemente y todos los semáforos toman un valor inicial igual a 0

Proceso 1	Proceso 2	Proceso 3
-----	-----	-----
<u>printf("3");</u>	<u>wait(s2);</u>	<u>wait(s1);</u>
<u>signal(s1);</u>	<u>printf("1");</u>	<u>printf("2");</u>
<u>wait(s3);</u>	<u>signal(s3);</u>	<u>signal(s2);</u>
<u>printf("0");</u>	<u>wait(s5);</u>	<u>signal(s4);</u>
<u>signal(s5);</u>	<u>printf("4");</u>	<u>printf("5");</u>

- 82. La ejecución concurrente puede producir interbloqueo
- 83. *La ejecución concurrente puede producir la salida 321054
- 84. *La ejecución concurrente puede producir la impresión 321045
- 85. *La ejecución concurrente puede producir la impresión 325104