

# Preguntas-Examenes-Anos-Anterior...



xexu65



Sistemas Operativos



2º Grado en Ingeniería Informática

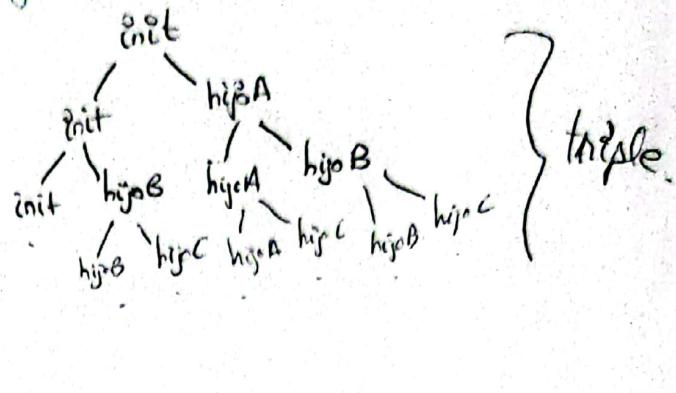


Escuela Técnica Superior de Ingeniería Informática  
Universidad de Málaga

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos  
No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo  
el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primera se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes posiciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará

El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convoy effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

ped->wait = waitpid(pgid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer parámetro de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para lanzares desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización no es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre no tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

(El manejador de señales SI debe recorrer la lista completa)

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

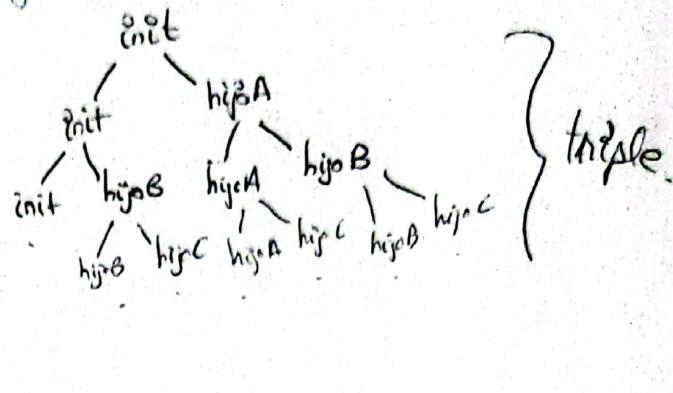
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer

Los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primero se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes porciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convoy effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

pid\_wait = waitpid(pid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer parámetro de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para lanzares desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre no tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

El manejador de señales SI debe recorrer la lista completa

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

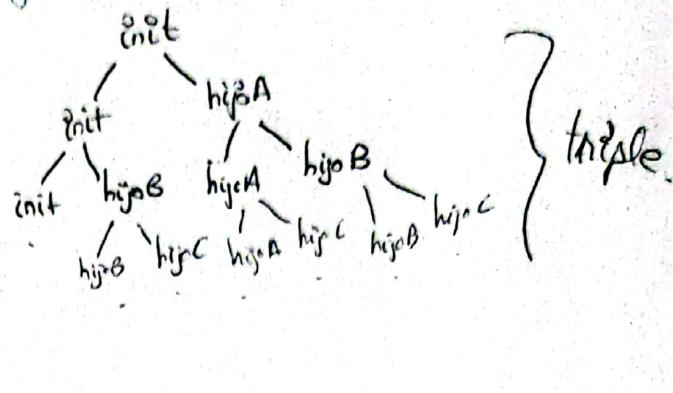
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primera se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes posiciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convoy effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

pid\_wait = waitpid(pid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer parámetro de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrarse el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para borrar desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto el S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización no es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre no tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

(El manejador de señales SI debe recorrer la lista completa)

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

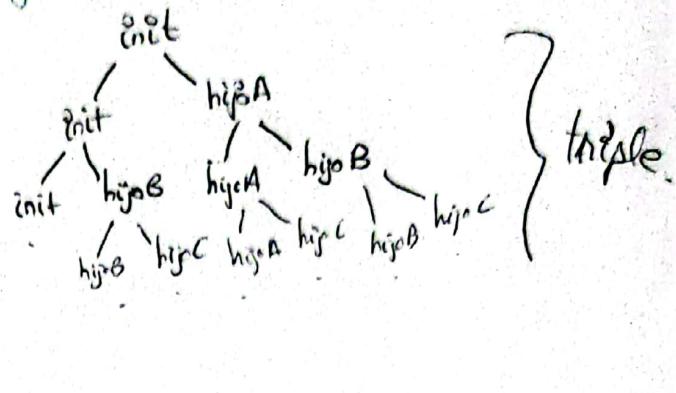
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos  
No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo  
el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer

Los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable}$   
a nivel de byte.  
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primero se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes porciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convey effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

ped->wait = waitpid(pgid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer parámetro de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para lanzares desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre no tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

(El manejador de señales SI debe recorrer la lista completa)

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

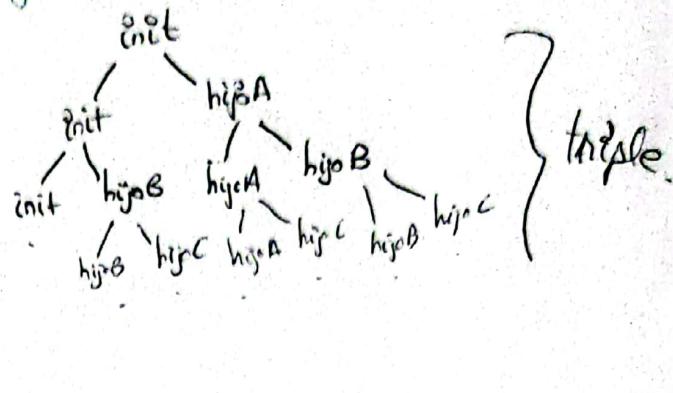
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.  
La implementación del Shell NO requiere modificar el código del
5. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La cuenta.
6. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
7. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
8. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
9. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
10. ¿Cuál de las siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNOHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer

Los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primero se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes porciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convey effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

ped->wait = waitpid(pgid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer paráel de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para borrar desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre No tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

El manejador de señales SI debe recorrer la lista completa

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

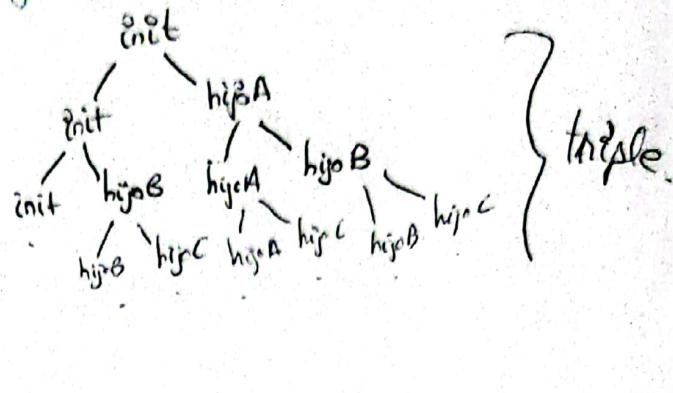
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer

Los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primera se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes posiciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convoy effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

pid\_wait = waitpid(pid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer parámetro de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para lanzares desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre No tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

El manejador de señales SI debe recorrer la lista completa

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

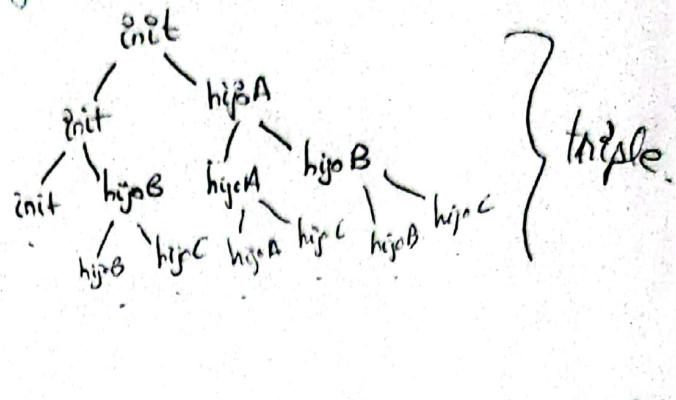
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer

Los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primera se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes posiciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará

El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convey effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

pid\_wait = waitpid(pid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer paráel de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para borrar desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre No tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

(El manejador de señales SI debe recorrer la lista completa)

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

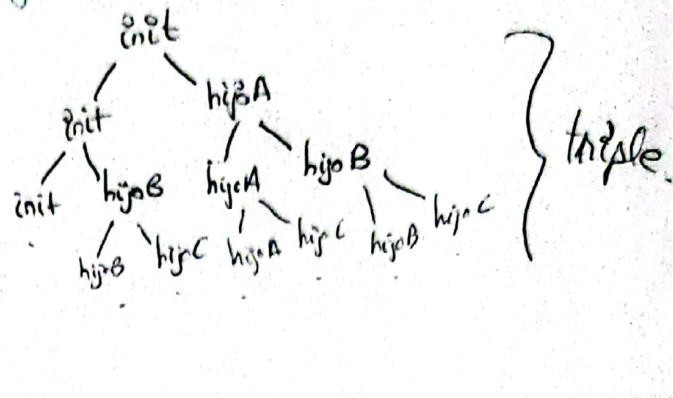
Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);

# PRIMER PARCIAL S.O.

1. Cuantos procesos se crean en el siguiente código? (Indicar el proceso init)

```
int main() {  
    pid_t pid;  
    pid = fork();  
    pid = fork();  
    if (pid == pid)  
        fork();  
}
```



Siete.

2. ¿Cómo se consigue que un algoritmo de planificación de procesos por prioridades se asemeje lo máximo posible a un algoritmo SJF?

Asignando las prioridades más altas (números más bajos) a los procesos más cortos.

3. Sea un sistema por prioridad expropiativa en el que comienzan a ejecutarse dos procesos: P1 con prioridad 1 (más alta) y P2 con prioridad 2 (más baja) y ambos necesitan el mismo tiempo de CPU para concluir su ejecución.

¿En qué situación puede acabar P2 antes que P1?

Si P1 realiza una operación de entrada/salida y P2 NC.

4. ¿A quién beneficia más el algoritmo SJF?

A los procesos que menos tiempo utilizan la CPU.

5. Si un proceso NO crea al menos 4 hilos

No podrá aprovechar el solo toda la potencia de una CPU de 4 cores.

6. Cuando un proceso padre crea un proceso hijo

el hijo hereda el espacio de direcciones del padre, copiándolo.

7. ¿Sabemos cuánto tiempo va a estar suspendido un proceso durante su ejecución?

NO.

8. A diferencia de un proceso, un hilo (thread)

No necesita apropiarse de recursos, ya que hereda los que utiliza el proceso que lo creó.

9. Indica cuál es la función del sistema operativo.

Establecer el control y la coordinación entre usuarios y aplicaciones para el uso adecuado de los recursos físicos de la máquina.

10. En la planificación de procesos por prioridad NO expropiativa, cuando un proceso abandona la CPU para completar ciclos de E/S, retorna inmediatamente al uso de la CPU al concluir dichos ciclos de E/S si tiene una prioridad. No tiene opción de ocupar la CPU hasta que no termine de usarla el proceso que ocupa la CPU en ese momento.

11. El planificador a corto plazo es planificador de la CPU

Seleccióna el proceso que se ejecuta a continuación y le asigna la CPU.

12. ¿Qué tipo de algoritmo de planificación predomina más en los S.O. Linux y Windows?

Expropiativos, esto es, permiten desalojar un proceso de la CPU.

13. El algoritmo de planificación FCFS busca como objetivo primordial favorecer los procesos que antes solicitan el uso de la CPU.

14. El sistema operativo...

- Alaja recursos
- Controla actividades
- Gestiona los dispositivos.

15. En un sistema de planificación por colas multivel con el mismo "quantum" de tiempo en las dos colas, se procede de la siguiente forma:  
Se concede un quantum a cada proceso de la cola 1, y luego se sigue repitiendo quantum en los procesos que sigan quedando en ella. No se pasa a la cola 2 hasta que NO esté vacía la cola 1.

## SEGUNDO PARCIAL S.O.

1. En el proceso de traducción de dirección virtual a física, ¿puede la TLB tener una anchura inferior a la tabla de páginas?  
No, en ningún caso.
2. ¿Qué estrategia para asignar los huecos libres a los procesos que solicitan memoria minimiza la existencia de huecos pequeños que son más difíciles de reasignar?  
Worst-fit.
3. Un sistema de memoria virtual tiene 1024 páginas de 8Kbytes mapeadas sobre una memoria física de 1Mbyte direccionable a nivel de byte.  
¿Qué anchura tiene la tabla de páginas y qué anchura tiene la TLB (Translation Look-Ahead Buffer)?  
 $1024 = 2^{\text{p}} \text{ páginas de } 2^{13} \text{ B sobre una memoria física de } 2^{20} \text{ B direccionable a nivel de byte.}$   
La tabla de páginas tiene una anchura de  $\text{p} = \frac{2^{20} \text{ B}}{2^{13} \text{ B}} = 2^7 \Rightarrow \text{p} = 7$   
La TLB tiene una anchura de  $\text{p} + \text{f} = \frac{2^{20} \text{ B}}{2^{11} \text{ B}} = 2^9 \Rightarrow \text{p} + \text{f} = 10 + 7 = 17$   
4. ¿Dónde tiene lugar el fenómeno de la fragmentación externa?  
Entre los distintos procesos que ocupan el espacio de memoria.
5. ¿Qué espacio de direcciones es más pequeño en un PC?  
El físico.
6. A medida que la memoria DRAM es más actual (DDR2, DDR3, DDR4...), ¿cómo han evolucionado sus principales parámetros de rendimiento?  
Mayor latencia y mayor ancho de banda.
7. ¿Qué indican las propiedades de localidad espacial y temporal que cumplen la mayoría de programas al ejecutarse?  
Que la próxima referencia a memoria se sitúa en una dirección cercana a la actual y que se ha solicitado antes con gran probabilidad.
8. ¿Puede tener la dirección lógica de memoria virtual una longitud inferior a la dirección física? Si, aunque es muy poco habitual.
9. ¿Cuántas entradas (o número de filas) tiene una TLB (Translation Look-Ahead Buffer)?  
El límite está impuesto por el coste de la implementación.

10. Durante el proceso de traducción de dirección virtual a física  
Primera se consulta a la TLB y luego a la tabla de páginas.

11. ¿Qué es el fenómeno de thrashing que sufre un Sistema Operativo?  
Acontece cuando se colapsa el proceso paginado del sistema operativo, al que se termina dedicando más tiempo que el progreso del usuario.

12. ¿Qué permiten las librerías de enlace dinámico o DLLs?

- Postergar hasta el tiempo de ejecución del programa la carga de las librerías que utiliza.
- Disminuir las rutinas de una librería que utiliza un programa de las que no.
- Optimizar el tamaño del código objeto que genere el compilador.

13. ¿Dónde es necesario implementar algoritmos de reemplazo?  
En los marcos de memoria física y en las entradas de la TLB.

14. ¿Cuál es la única memoria volátil de las 4 que se presentan?  
La DRAM (memoria dinámica).

15. Una FAT (File Allocated Table) de disco aloja los sectores de forma Enlazada.

16. Sea un sistema de ficheros formateado en dos particiones, una para Linux y otra para Windows. ¿Qué código ejecutará en su inicialización hasta dejarle el PC listo para que puedas ejecutar el comando ./Shell que lanza tu Shell de prácticas?

Una vez el MBR (Master Boot Record) y una vez el bloque de arranque (boot block).

17. ¿Cuántos i-nodos tiene un archivo en Linux?  
Uno.

18. ¿Cuántos super bloques, tablas de particiones e i-nodos tiene un sistema de ficheros con 3 particiones y 100.000 archivos en total?

Tres super bloques, una tabla de particiones y 100.000 i-nodos.

19. ¿Quién lleva a cabo la gestión del espacio libre en el disco?  
El sistema operativo.

20. En un disco duro magnético, el sistema de posicionamiento sobre su superficie se define por las siguientes coordenadas:  
Cabezal, plato, cilindro y sector.

Un sistema RAID (Redundant Array of Independent Disks) apunta al sistema de ficheros de un disco magnético.

Una mayor fiabilidad y/o velocidad a las operaciones de lectura y escritura en disco, según el esquema RAID elegido (RAID 0, 1, 2, ...).

22. Sectores de disco grandes favorecen

Tasas de transferencia (ancho de banda) elevadas en las operaciones de lectura y escritura del disco.

23. Un disco duro magnético del año 2021 gira a una velocidad angular aproximada de 120 vueltas por segundo.

24. Un disco de estado sólido (SSD) del año 2021 gira a una velocidad angular aproximada de NO GIRA (oneo) → Jesos.

25. ¿Qué método de asignación de espacio en disco puede provocar fragmentación interna?  
Asignación indexada, enlazada y contigua.

26. El número de bytes por entrada de la tabla de asignación de archivos en FAT16 es 2 bytes.

27. En un sistema que utiliza segmentación para la gestión de memoria, un proceso se divide en un número de segmentos variable que no tienen que ser del mismo tamaño.

28. La política de reemplazo que escoge solo entre las páginas residentes del proceso que generó el fallo de página, para decidir cuál es la página que va a ser reemplazada se denomina:  
Política de reemplazo local.

29. ¿Qué formato implica un menor grado de fragmentación interna?  
FAT32 con tamaño de cluster 16KB.

30. Cuando la entrada de la tabla de páginas solicitada no se encuentra en la TLB se dice que se produce un fallo de TLB.

31. ¿Qué método de asignación de espacio en disco puede provocar fragmentación externa?  
Asignación contigua.

32. La mínima cantidad de información a la que un controlador de disco puede acceder se denomina sector.

33. La dirección física de una palabra en memoria traduce a partir de las siguientes porciones de una dirección virtual  
Número de página y desplazamiento.

34. En un sistema de memoria paginado, si se disminuye el tamaño de página, manteniendo igual los tamaños de los espacios físico y lógico, aumentará  
El número de entradas de la tabla de páginas, y también el tamaño de cada entrada.

35. En un sistema de asignación contigua de espacio en disco los tipos posibles de acceso a los ficheros son secuencial y aleatorio.
36. ¿Qué son los atributos de un fichero?  
Metadatos asociados a cada fichero.
37. Con respecto a FAT12, el formato FAT16 permite  
clusters de menor tamaño.
38. El uso de tablas de páginas multível provoca  
Ahorro de espacio de memoria consumido por la tabla de páginas.
39. Un i-nodo clásico de Unix (4.1) contiene índices indirectos  
hasta de tres niveles.
40. En un instante dado, la coherencia (nº elementos) del conjunto activo (working set)  
de un proceso depende de la localidad del proceso.
41. En un sistema que combine la paginación y la segmentación, el espacio de direcciones del usuario  
se descompone en una serie de  
segmentos de tamaño variable, que se dividen a su vez en páginas de tamaño fijo.
42. El denominado "cilíndro" de un disco ya formateado contiene  
tantas pistas como cabezas.
43. En un sistema de ficheros tipo Unix, una estructura directaria correspondiente a un  
fichero regular apunta a el i-nodo del fichero.
44. El hardware de traducción (MMU)  
traduce las direcciones del espacio lógico de un proceso a direcciones físicas en memoria principal.
45. El acceso aleatorio a un fichero en disco  
es más rápido en asignación contigua que en enlazada.
46. ¿Qué sistema de ficheros es el más habitual en S.O.s tipo Unix?  
Basado en i-nodos.
47. El tipo de memoria que permite una multitarea muy efectiva, liberando al  
usuario de las restricciones ocasionadas por el tamaño de la memoria, se denomina:  
Memoria virtual.
48. La tabla de páginas mantiene, para cada proceso  
La localización del marco para cada página del proceso.
49. El cluster (también denominado bloque ó unidad de asignación) es un múltiplo de  
El sector.

El cilindro de un disco magnético consta de

todas las pistas ubicadas a diferentes alturas que haya como consecuencia de los múltiples cabezales que posee el brazo del disco.

51. ¿Dónde se almacena la tabla de páginas del SO?

En el espacio de memoria principal (DRAM)

52. El modelo del conjunto de trabajo (working set) persigue como objetivo prioritario mantener en memoria física un nº de páginas críticas para cada proceso que evite su congestión en el uso de la memoria.

53. El espacio de direcciones de un proceso se compone de áreas o segmentos destinados a almacenar el código de su programa, los datos de su programa, la pila y el heap.

54. La tabla de páginas invertida del SO en un sistema de memoria virtual tiene una entrada por cada marco de página (o frame) de memoria principal.

55. El bloque de control de un proceso (PCB) es una estructura de datos que crea y mantiene el sistema operativo.

56. La elección de un tamaño pequeño para el sector de disco favorece un mayor aprovechamiento del espacio de disco.

57. La jerarquía organizativa de un disco contempla, sucesivamente, particiones, directorios, ficheros y sectores.

58. Las señales en el S.O. Unix se envían desde el sistema operativo a sus procesos y desde un proceso a otro proceso.

59. Una FAT32 nos indica que el disco podrá tener como máxima 4 Giga-clusters.

60. La función de la TLB (Translation Look-Ahead Buffer) consiste en acelerar la traducción de las direcciones lógicas de memoria físicas, manteniendo en una caché las traducciones que se han realizado más recientemente.

61. El swapping se produce cuando un proceso requiere más memoria física de la que hay disponible.

62. La tabla de páginas del SO en un sistema de memoria virtual nos dice la página física que contiene una página lógica.

63. El tiempo que el S.O. dedica al cambio de contexto de un proceso

aumenta con la complejidad del sistema operativo y con el tamaño del PCB de los procesos involucrados en dicho cambio de contexto.

64. ¿Cuál es el tamaño máximo ocupado en disco para una FAT32?

16 GB

65. En la gestión de memoria, el proceso de compactación de memoria se realiza para reducir la fragmentación externa.

66. ¿Puede acceder un usuario a los metadatos que contiene un disco magnético?  
Sí, a través de llamadas al sistema operativo

67. ¿Qué apunta la multiprogramación en S.O.?

Rendimiento.

68. En un esquema de memoria virtual, ¿cuándo se genera la dirección física que corresponde una dirección lógica?

Durante la ejecución del programa.

69. Señala una situación que refleje el "convoy effect" que puede producirse en el contexto del uso compartido de la CPU por parte de los procesos.

Los procesos que más necesitan la CPU no dejan suficientes turnos disponibles a otros que la necesitan menos.

# PARCIAL SHELL

1. ¿Cuál de los códigos es correcto?

ped->wait = waitpid(pgid, &estado, WNOHANG);

enum status mi\_estado = analyze\_status(estado, &info);

2. ¿Qué secuencia de acciones debe realizar un usuario de nuestro Shell de prácticas para que un comando ya lanzado en segundo plano pueda pasar a primer plano y posteriormente regresar a segundo plano?

(1) Emplean el comando fg

(2) pulsar Ctrl+Z

(3) Emplean el comando bg.

3. ¿Qué comando(s) de tu Shell necesita(n) realizar una llamada killpg(1) para su correcta implementación?

'bg' y 'fg'

4. ¿Qué conflicto evita el uso de la pareja de funciones de apoyo 'block-SIGCHLD()' y 'unblock-SIGCHLD()' en la implementación de tu Shell de prácticas?

Que el manejador de la señal SIGCHLD pueda modificar la lista de procesos

el mismo tiempo que el proceso padre.

5. ¿Cuántos procesos hijo puede crear el padre (proceso main) durante una correcta ejecución del Shell de prácticas?

Muchos en primer plano y muchos en segundo plano, aunque en todo momento sólo puede haber a la suma un proceso vigente en primer plano.

6. ¿Qué llamada a waitpid() permite al padre esperar la finalización de su hijo y/o suspenderlo?

waitpid(pid, <status-argument>, WUNTRACED);

7. Cuando se activa el manejador de la señal SIGCHLD, ¿es necesario que este recorra la lista entera de procesos para verificar los estados de todos los procesos que están en ella?

Sí, porque durante el intervalo de tiempo en el que se bloquea la atención de señales

(desde 'block-SIGCHLD' hasta 'unblock-SIGCHLD') pueden sucederse varias señales que alteren el estado de varios procesos, y por lo tanto, tras un desbloqueo puede haber varios

procesos afectados por un cambio de estado.

8. Señala qué protocolo tiene la pareja de funciones 'ignone-terminal-signals()' y 'de\_nestore-terminal-signals()' en el primer organigrama que se presenta en la práctica.
- Retrasar el tratamiento de ciertas combinaciones de teclas, que por tanto, sufren una demora mientras el proceso padre habilita a su hijo para que sea este el receptor de dichas combinaciones de teclas.
9. ¿Por qué no utilizamos en nuestro Shell la llamada al sistema 'wait()' con la funcionalidad básica que viene en el primer paráel de la asignatura y es necesario incorporar 'waitpid()', que incluye "flags" para cubrir comportamientos más diversos?
- Porque en cuanto un proceso hijo queda suspendido, el padre se quedará esperándole, lo que le impide atender nuevos comandos introducidos desde teclado.
10. ¿Por qué se utilizan dos variables, 'status' y 'ground', para definir la situación de cada proceso dentro de nuestro Shell de prácticas?
- Porque los cambios sobre una de ellas puede registrárselas el S.O., pero NO es suficiente para completar la funcionalidad que exigen nuestras prácticas, lo que nos lleva a un seguimiento manual por parte del programador de la segunda variable.
11. ¿Qué proceso de tu Shell de prácticas se encarga de eliminar comandos de la estructura de datos "lista de comandos" realizando llamadas a 'delete-job()'? Tanto el padre como el manejador de señales.
12. ¿Qué proceso de tu Shell de prácticas se encarga de introducir comandos en la estructura de datos "lista de comandos" realizando llamadas a 'add-job()'? El padre.
13. ¿Qué proceso(s) de tu Shell de prácticas pueden ser suspendidos? Los hijos que van asociados a los comandos.
14. Si ejecutas tu Shell de prácticas para borrar desde dentro el comando "ls &", una vez el proceso padre ha ejecutado la llamada 'signal()' y la primera 'fork()', ¿cuantos procesos quedan definidos dentro del S.O.? Dos: El padre y el hijo.
15. ¿Qué pulsación de teclas debes contemplar explícitamente desde tu código del Shell y te exigen incorporar líneas de código para considerar funcionalidad adicional a la que ya proporciona por defecto al S.O. Linux? Control+Z.
16. ¿Qué ocurre en nuestro Shell de prácticas cuando el usuario pulsa Ctrl+C? El S.O. emite una señal de Interrupción.
17. IDEM 16 Ctrl+D → El teclado emite el código End-Of-File que da por concluida la entrada desde ese dispositivo.
18. IDEM 16 Ctrl+Z → El S.O. emite la señal SIGSTOP.

19. ¿Qué ocurre en tu Shell de prácticas cuando un proceso suspende su ejecución?

Se detiene temporalmente, en espera de reanudar su ejecución más adelante.

20. Se ejecuta tu Shell de prácticas para lanzar el comando "ls &". ¿Qué proceso controla la correcta finalización del proceso que se crea para ejecutar ese comando? El proceso padre, desde el código correspondiente al manejador de la señal SIGCHLD.

21. En una correcta ejecución del Shell de prácticas según se indica en el primer organigrama correspondiente a la implementación de las fases 1, 2 y 3.

¿Qué proceso hijo recoge el padre desde su llamada a 'waitpid'?

Todos los que ha generado en su llamada a 'fork()' para delegar el tratamiento de los comandos en primer plano.

22. Para garantizar la correcta secuenciación de los mensajes informativos en pantalla emitidos desde nuestro Shell y que éstos no se intercalen y/o solapan desordenadamente con la sucesiva impresión del prompt ("COMANDO -->"). ¿A qué debemos exponer cada tarea? El padre imprime todo.

23. ¿Quién se encarga en tu Shell de prácticas de reanudar la ejecución de un proceso suspendido para que pueda concluir la ejecución del correspondiente comando lanzado por el usuario?

El manejador de señales

24. ¿Cuáles son los procesos hijo cuya finalización NO es correctamente controlada por el proceso padre en la implementación de las primeras fases del Shell, y que posteriormente se subsana gracias a la incorporación del manejador de señales en una fase posterior?

Los procesos en segundo plano.

25. ¿Cómo se delega la aceptación de comandos por el terminal desde el proceso padre al proceso hijo?

El padre No tiene que hacer nada. El hijo toma el terminal con 'set-terminal'.

1. ¿Qué valores de las variables 'status' y 'ground' guardan una menor jerarquía (es decir, un proceso tiene una menor probabilidad de coincidir en ellas)?

status = FINALIZADO y ground = DETENIDO.

2. Se quiere imprimir en pantalla un mensaje que dice "Señal recibida" cada vez que se envía SIGHLD. ¿Qué parte del código del shell NO debe cambiar?

- El código del proceso hijo tras el fork()
- El código del proceso padre tras el fork()
- El código del main(), entre su inicio y la llamada a fork()

3. Los comandos internos del shell son aquellos interpretados por el shell y que NO corresponden a comandos ejecutables. Entre ellos, se encontraría jobs.

4. ¿Por qué el manejador de señales NO debe recorrer la lista de procesos completa y termina cuando encuentra el primer proceso que ha cambiado de estado?

El manejador de señales SI debe recorrer la lista completa

5. ¿Puede un proceso zombié que ha terminado irregularmente en tu Shell reanudar su ejecución? NO

Si queremos cambiar el aspecto de mi Shell para la recepción de comandos, "COMANDO->"

¿que parte del código NO debe cambiar?

- El manejador de señales.
- El proceso padre.
- El proceso hijo.

Fíjate en los 3 argumentos de las llamadas a waitpid (pid, &status, flags);

¿por qué pid y flags se pasan por valor y status se pasa por referencia?

Porque waitpid() devuelve información en la variable status, mientras que los otros dos parámetros le suministran información.

3. Queremos implementar un comando 'verboso' en nuestro Shell que nos desanchele de la terminal de entrada. Desde que teclea "verboso" hasta que teclea "normal" se deberá ignorar cualquier comando que se introduzca desde la terminal. ¿En qué parte del código de tu Shell desarrollarías la implementación para ese comando 'verboso'?
- En el código del main(), entre su inicio y la llamada a fork().
4. ¿Qué parte(s) del código del Shell utiliza(n) la llamada a killpg() para enviar una señal de reanudación a un proceso detenido?
- La implementación de los comandos 'fg' y 'bg'.
5. La implementación de la construcción del Shell NO requiere modificar el código del
6. ¿Qué fase de la construcción del Shell no requiere modificar el código del proceso padre?
- La quinta.
7. ¿Cuál es la diferencia entre el comando jobs y el comando ps?
- 'jobs' lista sólo trabajos lanzados desde el shell y 'ps' lista procesos en general, incluyendo a los de jobs.
8. En la primera fase de nuestro proyecto Shell, si pulsamos desde el teclado CTRL+C se mata tanto al proceso hijo lanzado por el Shell como a este último. ¿Qué debemos hacer en las fases siguientes para que esto NO suceda, y que las señales asociadas a las combinaciones de teclas como CTRL-Z y CTRL+C le lleguen sólo al proceso lanzado en primer plano?
- Independizar al nuevo proceso o su propia grupo de procesos y asignarle a terminal.
9. ¿Qué comando interno de tu Shell se implementa de manera más fácil?
- logout.
10. En un S.C. Unix, un proceso cuya señal SIGCHLD ha sido atendida por el padre finaliza su ejecución de forma normal.
- Finaliza su ejecución de forma normal.
11. ¿Cuál de los siguientes llamadas a la función waitpid() NO bloquea al proceso que la ejecuta?
- mypid = waitpid(pid, &status, WUNTRACED | WNCHANG);