Preguntas y respuestas de Sistemas Operativos

Este compilado de preguntas y respuestas está hecho en base a un rejunte de resúmenes y otros compilados. Se trató de eliminar de los ya existentes, preguntas que ya no se toman más porque los temas ya no se encuentran en el temario de la materia y de actualizar con preguntas que no se encontraran en el mismo y que actualmente son preguntas habituales de los finales de la materia.

Al lado del número de cada pregunta es posible encontrar otro número; éste se corresponde con el número de pregunta dentro del ya famoso resumen preguntas_y_respuestas_de_so_2013.pdf, el cual se encuentra en la también famosa wiki.

Todas las preguntas corresponden a finales tomados en los años 2013, 2014, 2015 y 2016.

Por último, vale aclarar que hay preguntas que no se encuentran respondidas, pero, además de ser pocas, son de las menos habituales. Es por ello que les deseo la mejor de las suerte para que no les toque alguna de esas preguntas.

¡Espero que les sirva de ayuda!

1) 17. ¿Cuándo un proceso está en estado Zombie? Describa una forma de lograrlo

Un proceso está en estado zombie cuando el mismo ha finalizado, se ha liberado la memoria y los recursos utilizados, pero su entrada en la tabla de procesos permanece. Suele ser un breve período. Los procesos zombies que perduran denotan un error por parte del proceso padre al realizar una llamada al system call wait. (Se olvidó de llamar a dicha system call, ya que un proceso que termina no puede abandonar el sistema hasta que su padre acepte su código de retorno).

Los procesos zombies nunca recibieron la señal de finalización por parte del proceso padre que lo creo. Se pueden deber a errores de programación, a situaciones no contempladas por el programador y generalmente provocan lentitud y/o inestabilidad en el sistema.

```
int main () {
    pid_t child_pid;
    child_pid = fork();
    if (child_pid > 0) {
        sleep (60);
    }else {
        exit (0);
    }
    return 0;
}
```

Proceso huérfano: Si el proceso padre murió, es adoptado por el proceso “init” el cual siempre acepta los códigos de retorno de sus hijos.

2) Explique en un diagrama con valores ejemplo el proceso de traducción y carga de páginas usando páginas invertida para una dirección

1. cuyo mapeo se encuentra en la TLB.

2. el mapeo no se encuentra en la TLB pero la página está en memoria

3. la página no se encuentra en memoria.

1. Se busca en la TLB y se produce un hit, se obtiene la página requerida.

2. Se busca en la TLB y se produce un miss, entonces se debe ir a buscar la página en la lista invertida haciendo un recorrido de la lista entera (hasta que se encuentra la página), se encuentra la página y a partir de ella su frame en memoria, se carga en la TLB, luego obteniéndose la página requerida.

3. Se busca en la TLB y se produce un miss, entonces se debe ir a buscar la página en la lista invertida haciendo un recorrido de la lista entera, NO se encuentra la página y entonces se produce un *page fault*. A continuación se busca la página en disco, se carga en memoria, en la tabla invertida y en la TLB y se obtiene la página requerida.

3) 18. Describa la diferencia entre Storage Area Network y Network Area Storage (SAN y NAS). Dé ejemplo de los protocolos utilizados en cada caso.

NAS conecta un file system remoto a una red, proveyendo el acceso a clientes heterogéneos. Provee tanto almacenamiento como un file system. Aparece ante un sistema operativo cliente como un servidor. Los servicios que provee son orientados

al almacenamiento y no están diseñados para funcionar como un servidor multipropósito.

Provee servicios basados en archivos. Generalmente es una versión reducida empotrada de algún Sistema Operativo (Nexenta, FreeNAS, etc.). Usa protocolos como SMB/CIFS (Windows), NFS (Unix) o AFP (Mac).

SAN es una red dedicada que provee acceso a datos por bloques. Conecta dispositivos remotos que el SO ve como locales, e implementa el file system. No provee la abstracción de archivos, sólo operaciones en bloques. De esta manera, el SO ve al SAN como un disco que puede ser formateado con un file system y montado al sistema operativo.

Consolida las “islas de discos” con conexiones de red. Pueden ser discos, RAIDs o alguna arquitectura no RAID. Usan protocolos como SCSI, iSCSI, Fibre Channel, HyperSCSI, ATA over Ethernet (AoE) o InfiniBand. Requieren de un software de administración. Algunos proveen capacidades RAID.

4) 14. ¿Que son y cómo se usan los archivos mapeados a memoria? ¿Cual es su relación con la memoria virtual? ¿Tienen alguna ventaja respecto de los archivos tradicionales?

Los archivos mapeados a memoria se ven como parte de la memoria, se manejan junto con la memoria virtual y permiten compartir archivos.

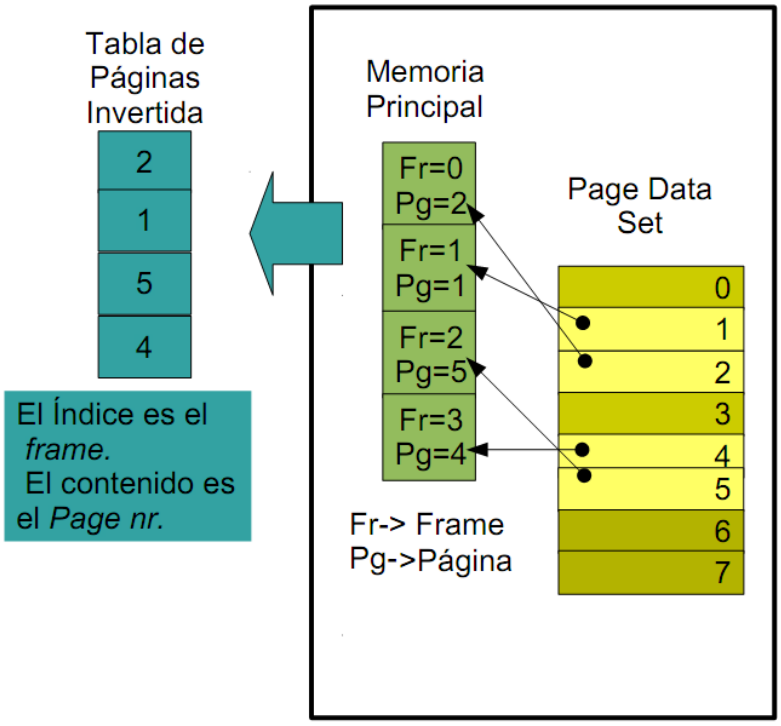
Tienen 3 propósitos los cuales son:

1. Cargar y ejecutar archivos .exe y bibliotecas de enlace dinámico (DLL).
2. Acceder a archivos en disco rápidamente sin necesidad de usar buffers.
3. Permitir a varios procesos compartir la misma información en memoria.

Un archivo mapeado en memoria es una porción de memoria virtual en la que se establece una correlación directa byte a byte con una parte de un archivo o un recurso similar. Este recurso es, normalmente, un archivo presente en el disco duro, o bien un objeto de memoria compartida u otro tipo de recurso al que el sistema operativo puede referirse por medio del descriptor de archivo. Una vez disponible esta correlación entre el archivo y el espacio de memoria, las aplicaciones pueden gestionar el acceso a ese recurso exactamente igual que si se tratara de memoria primaria.

Para su acceso se utilizan los comandos mmap y munmap.

5) 27. Explique en un diagrama el funcionamiento de la tabla de páginas invertida



6) 1. ¿Qué diferencias hay entre la implementación de threads a nivel de Kernel y como biblioteca de usuario? ¿Que ocurre en ambos casos con una llamada que debería bloquear al proceso como una lectura?

Implementación como biblioteca de usuario:

- Puede implementarse en sistemas operativos que no soporten threads.
- Cada proceso necesita su tabla de threads privada para poder monitorear a los threads de ese proceso.
- Se mantiene información acerca de cada thread individual: program counter, stack pointer, registros, estados y demás.
- Se puede hacer un switching de un thread a otro sin la necesidad de hacer un trap a nivel kernel.
- Cada proceso puede tener su propio algoritmo de scheduling para los threads.
- Si un thread comienza, ningún otro thread va a correr en CPU hasta que éste ceda voluntariamente la misma.

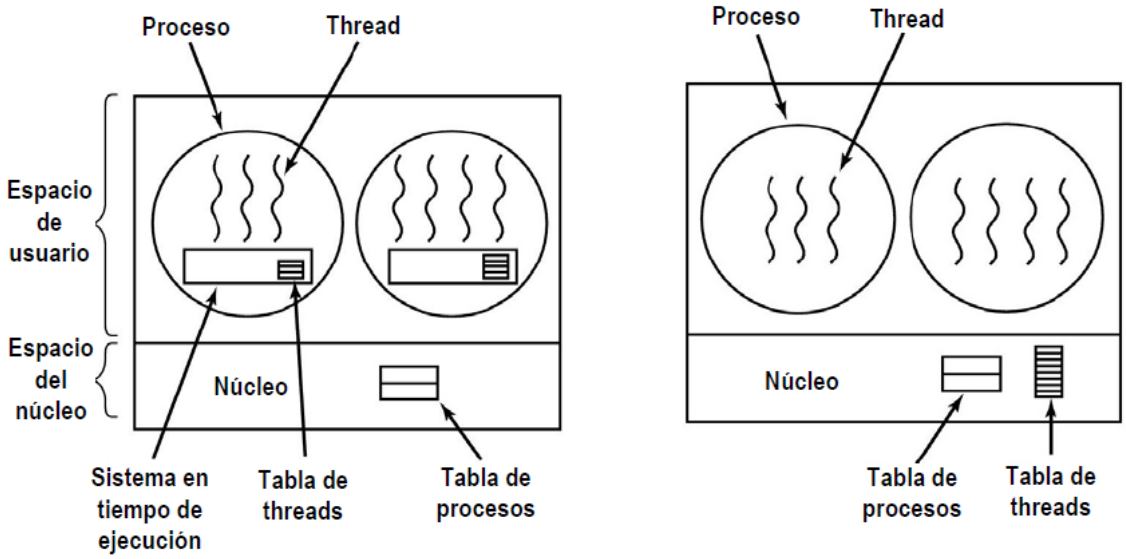
Tiene ciertos problemas cuando ocurre una system call bloqueante. Sería inaceptable dejar que el thread realice la system call, ya que bloquearía el proceso, afectando a otros threads del mismo. Una alternativa sería cambiar las system calls para que sean no bloqueantes, pero eso implicaría un cambio en el sistema operativo. Otra alternativa más viable sería realizar un wrapper para las llamadas bloqueantes, que sólo se realizarían cuando éstas son seguras.

Implementación a nivel kernel:

- No hay tabla de threads en cada proceso.

- El kernel tiene una tabla de threads que mantiene información de todos los threads del sistema.
- El kernel le puede sacar tiempo de CPU para otorgárselo a otro thread.
- Debido al alto costo de crear y destruir threads, éstos se reciclan.
- Cuando un thread causa una page fault, el kernel puede chequear si el proceso tiene otro thread y correr ese mientras espera que la página requerida sea traída desde el disco.
- Las señales son enviadas a los procesos, no a los threads.

Cuando un thread bloquea, el kernel puede correr otro thread, tanto del mismo proceso como de otro.



(a) Un paquete de threads a nivel de usuario.
(b) Un paquete de threads gestionado por el kernel.

7) 2. ¿Cumple el Intel Pentium con las condiciones de Popeck y Goldberg para virtualizar una plataforma?. ¿Cómo se realiza esta virtualización?

Condición de Popeck y Goldberg → Se puede construir VM si las instrucciones sensibles (las que dependen de los recursos o los configuran. Solo pueden ejecutarse en Modo Supervisor del procesador) son un subconjunto de las privilegiadas (las que pueden generar traps si esta en modo usuario. Son aquellas a capturar si el procesador está en modo usuario y que no se capturan si está en modo privilegiado).

Intel Pentium no cumple con esta condición; cuenta con instrucciones sensibles que no son privilegiadas, violando una de las condiciones de Popeck y Goldberg.

Para realizar la virtualización de esta máquina se requiere de una extensión que la permita, que en este caso es el Intel IVT, que genera containers en los que la ejecución de una instrucción sensible provoca un software trap. Luego, con la trampa se debe emular el comportamiento de la máquina autónoma del sistema operativo guest.

8) 16. En Android: ¿Como se separan los recursos de cada aplicación, como se evita que un error en una de ellas afecte al resto?

Cada aplicación corre en su propio proceso con su propia copia de Dalvik. Los procesos son provistos por el kernel y manejados por el Android Run Time. Para mantener la respuesta del sistema, el sistema Android puede matar sin aviso a procesos que considera que no están respondiendo (y las aplicaciones contenidas dentro del mismo).

9) 4. ¿Cuál es el principio de funcionamiento de los Hipervisores tipo II?

Corre bajo el control de un sistema operativo anfitrión. Puede virtualizar cualquier ambiente y modifica el programa que está corriendo.

Aplicación que se ejecuta sobre un sistema operativo convencional (Linux, Windows, MacOS) para virtualizar sistemas. De esta forma la virtualización se produce en una capa más alejada del hardware si lo comparamos con los hipervisores de tipo 1. Un ejemplo de software es Virtual Box.

10) ¿Que son los Log File Systems? ¿Que variantes conoce y que características tienen?

Los LFS surgen como una solución al tiempo que se tarda en escribir en disco. La idea básica de los mismos es estructurar al disco como un registro. De manera periódica, y cuando surge la necesidad especial de hacerlo, todas las escrituras pendientes que están en búferes en la memoria se reúnen en un solo segmento y se escriben en el disco como segmento contiguo único al final del registro. Además posee un *cleaner* de forma que recorre el registro en forma circular para compactarla (Eliminar información que ya no se use). Entonces, se van creando versiones del mismo archivo:

- Si éstas son accesibles, es un Versioning File System.
- Si se reconstruye el archivo, se lo llama Journaling File System

11) Describa el funcionamiento de la system call exec(), explicando su efecto sobre las estructuras de datos del proceso que la emite (Process Control Block, U-File Table, U-Area, BSS (área static), Txt, Stack).

La función exec() reemplaza la imagen del padre copiada en el hijo por la propia área static (BSS), TXT y STACK (Es decir, se pisa con la nueva información del hijo). En cuanto al PCB, sigue habiendo un solo PCB, con casi todos los campos iguales. Respecto del U_AREA, el proceso sigue siendo el mismo, sólo cambia su imagen, por lo que su U_AREA es la misma. Por último, la U_File Table no cambia.

Esta función se encarga de ejecutar el comando que corresponde al proceso creado.

12) Explique en un diagrama con valores ejemplo el proceso de traducción y carga de páginas usando páginas invertida para una dirección

Ver respuesta 2 y/o 5.

13) Describa los componentes y algunas funciones de EFI (Extended Firmware Interface).

Me mató.

14) 9. ¿Que es RAID, stripping y mirroring? Describa las diferencias entre RAID propiamente dicho y software raid o fake raid.

RAID es una tecnología de almacenamiento que combina múltiples discos en una sola unidad lógica. Los datos son distribuidos a través de los discos en una de las varias maneras, denominadas “niveles RAID”, dependiendo de la redundancia y performance requerida.

Data stripping es la técnica de segmentar datos secuencialmente lógicos, como un archivo, de manera que segmentos consecutivos son almacenados en dispositivos de almacenamiento distintos. Este mecanismo es útil cuando un proceso que accede a un dispositivo necesita acceso a datos de manera más veloz de la que este dispositivo puede proveer. Permite acceder a los datos de manera concurrente.

Disk mirroring es la replicación de volúmenes de discos en discos duros separados para asegurar la disponibilidad continua. La replicación se realiza a través de microcódigo en el controlador del disco o mediante software de servidor. Sirve como copia de seguridad de los datos ante alguna falla de hardware.

En los *Software RAID*, el procesador debe usar su tiempo para las operaciones RAID, que se realizan en una capa entre el File System y el Device Driver. El *Fake RAID* es un controlador de firmware que toma las funciones de RAID durante el boot. Una vez que el kernel de un sistema operativo está cargado, el control pasa al sistema operativo. Esto se debe a que Windows no puede bootear desde software RAID.

En los RAID originales, se requiere un controlador dedicado por parte de los discos. Este controlador debe tener un back end hacia los discos y un front end hacia el host. La ventaja es que no se gastan recursos del procesador para manejar operaciones RAID.

15) ¿Como se separan los recursos de cada aplicación, como se evita que un error en una de ellas afecte al resto?

a) en IOS

b) en Android

Para b), ver respuesta 8. Para a), LTA.

16) ¿Cuál es el problema de la relocación que se debe resolver en las bibliotecas dinámicas?. ¿Como resuelve este problema la generación de Position Independent Code/ Position Independent Executable (PIC/PIE) ?

El problema que surge para bibliotecas dinámicas es que al cargarse en forma dinámica, en cada proceso que la utilice se debe cargar en su correspondiente espacio de usuario con una dirección distinta. Para ello, generalmente se utiliza código PIC, en el cual las instrucciones que se refieren a posiciones de memoria específicas, son reemplazadas por instrucciones relativas. Así, un programa PIC puede ejecutarse en cualquier dirección de memoria sin necesidad de modificarlo. El problema es que se deben calcular las direcciones cada vez que se carga el programa o la biblioteca. El código PIC accede a datos y funciones a través de una tabla (Global Offset Table o GOT) que contiene sus direcciones de memoria.

17) ¿En qué casos una página puede estar siendo usada en forma simultánea por más de un proceso? ¿Varía esta respuesta si la página es de código o de datos?

Una posible situación podría ser, por ejemplo, si a un proceso le corresponden 1KB de memoria y las páginas son de 4KB. De esa forma, al estar el espacio de direcciones de ese proceso en una página (y no completarla), es posible que se le asigne memoria de esa página a otro proceso, con lo cual esa página estaría compartida por más de un proceso. En cuanto a si es de código o de datos, como el TXT area (código) de un proceso puede ser utilizada por más de uno a la vez, tranquilamente dos procesos podrían estar utilizando una misma página al mismo tiempo.

18) 10. Con base en un ejemplo sencillo de código, indique las diferencias entre un thread y un proceso en cuanto al tratamiento de los espacios de direcciones.

Los procesos agrupan los recursos usados. Cada proceso se ejecuta en un espacio de memoria separado.

Los threads son hilos de ejecución que comparten el agrupamiento de recursos, entre ellos, el espacio de memoria. Cada thread mantiene su propia información de estado.

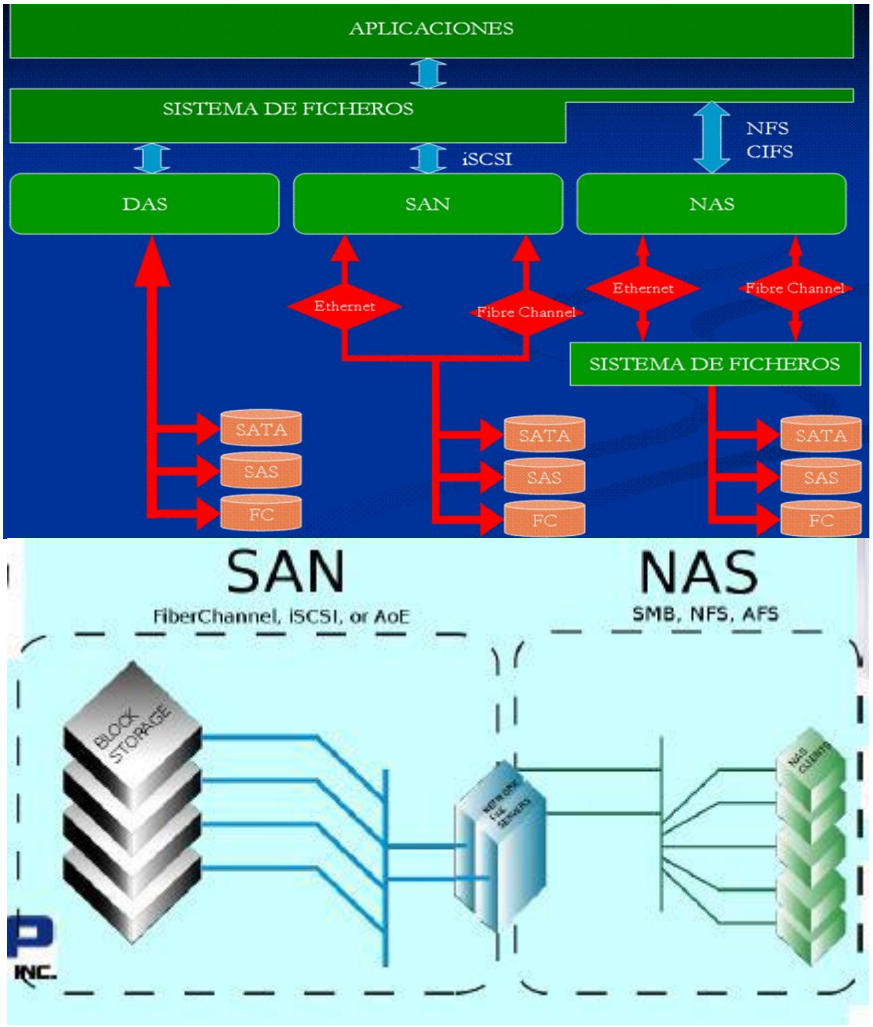
Por lo tanto, los threads son subconjuntos de los procesos.

Ver código ejemplo en la pregunta 10 de preguntas_y_respuestas_de_so_2013.pdf.

19) Describa en un gráfico un ejemplo completo de un sistema que use NAS, y uno que use SAN

1. en cada caso indique que protocolo está usando.

2. Indique como implementar RAID en cada uno de los casos. ¿Quien se encargaría de correr el software de raid?



20) 7. En Virtualización, ¿cuál es la diferencia entre paravirtualización y traducción binaria?

Paravirtualizar consiste en reemplazar, en el sistema operativo guest, las instrucciones delicadas por llamadas al hipervisor. En este caso, el sistema operativo huésped debe ser modificado para saber que va a correr en un entorno virtualizado. El huésped utiliza una API especial para comunicarse con la capa de virtualización e interactuar directamente con el hard. Si bien tenemos claros beneficios por el lado de la performance, se pierde compatibilidad ya que el SO huésped debe ser modificado. Requiere código fuente.

Traducción binaria, en cambio, consiste en examinar bloques básicos (código con un punto de entrada, uno de salida y sin jumps), tarea realizada por el hipervisor, y reemplazar las instrucciones delicadas por llamadas al hipervisor. El código traducido lo guarda en caché para aumentar la performance. Esto genera más sobrecarga que el enfoque de paravirtualización, pero no limita a usar sólo SO modificados, podemos usar cualquier SO. No requiere código fuente.

La traducción binaria es la emulación de un set de instrucciones por otro mediante la traducción del código. Mediante este método, secuencias de instrucciones son traducidas de la fuente al set de instrucciones destino.

21) 5. ¿Qué es y que cosas debe contener un Object File tanto para un ejecutable como para una Biblioteca (por ejemplo los object file formats pe, elf o coff)?

Es un formato de archivos de computadora usado para el almacenamiento de código objeto y datos relacionados.

Hay muchos formatos distintos. Originalmente, cada tipo de computadora tenía su propio formato único, pero con la venida de Unix y otros sistemas operativos portables, algunos formatos, como COFF y ELF fueron definidos y usados en distintos tipos de sistemas. Es posible usar el mismo formato como entrada y salida del linker y también como librería y formato del archivo ejecutable.

Un object file contiene 5 tipos de información:

- Header information: información general acerca del archivo, tales como el tamaño del código, nombre del archivo fuente del cual fue traducido y fecha de creación.
- Código objeto: instrucciones binarias y data generada por un compilador o ensamblador.
- Relocation: una lista de los lugares en el código objeto que tienen que ser modificador cuando el linker cambia la dirección del código objeto.
- Symbols: símbolos globales definidos en el módulo, símbolos a ser importador de otros módulos o definidos por el linker.
- Debugging information: otra información sobre el código objeto, no necesitada para el linkeo pero sí para el uso de un debugger. Esto incluye archivos fuentes e información sobre números de línea, símbolos locales, descripción de estructuras de datos usadas por el código objeto, tales como definiciones de estructuras de C.

No todos los formatos contienen todos los tipos mencionados y es posible tener formatos útiles con poca o ninguna información más allá del código objeto.

22) Describa brevemente dos algoritmos de scheduling que usen colas múltiples. Dé ejemplos de su funcionamiento.

Chupala.

23) 12. ¿Qué es la link-edición? En un ejemplo explique sus funciones principales.

Es la mezcla de las direcciones de cada módulo utilizado en una aplicación, en un único espacio de direcciones, produciendo como salida una biblioteca, un programa objeto o un programa ejecutable. La entrada también puede ser un ejecutable o una biblioteca estática.

24) Explique usando diagramas la diferencia entre Storage Area Network y Network Area Storage (SAN y NAS). Dé ejemplo de los protocolos utilizados en cada caso (ubíquelos en los diagramas)

Ver pregunta 19.

25) ¿Cuales son las ventajas y los inconvenientes de la Multiprogramación? ¿Cómo se implementa basándose en interrupciones?

En la multiprogramación hay un solo proceso corriendo a la vez en un sistema operativo, pero se cambia rápido de un proceso a otro. Se multiplexa la CPU en el tiempo mediante interrupciones de reloj. Cuando se produce una interrupción de reloj, el scheduler decide a qué proceso en estado *ready* darle el control. Para hacerlo, se mantiene una cola de PCBs correspondientes a procesos en estado *ready*.

Entonces, como ventaja principal vemos el aprovechamiento completo del tiempo del CPU, ya que mientras un proceso está en estado *blocked*, otro está haciendo uso del CPU, de forma que se optimiza su utilización. En cuanto a las desventajas, la coordinación necesaria es mucho más compleja que en la monoprogramación, pero vale la pena.

26) ¿Cómo pueden usarse las estructuras de paginado (MMU, Page tables, etc) para resolver los problemas de reubicación de código y cargar/compartir bibliotecas tradicionalmente resueltos por la link_edición?

Ni idea. ¿Irá por el lado de los archivos mapeados en memoria? (Ver respuesta 4).

27) En IOS: ¿Como se usa el almacenamiento en la nube (iCloud)? ¿Que debe hacer una aplicación para guardar sus datos en la nube?

Paja.

28) Discuta en que casos usaría linkeo

1. estático

2. dinámico en tiempo de carga

3. dinámico en tiempo de ejecución (donde la biblioteca se elige externamente)

1. Cuando sean bibliotecas que no estoy seguro que estén en todas las máquinas, o necesite una versión específica de alguna biblioteca, ya que en este tipo de linkeo el código de dicha librería va dentro del archivo ejecutable de nuestro programa y se carga en memoria al cargarse el programa y convertirse en proceso.

2. Cuando esté seguro que las versiones de las bibliotecas en otras máquinas son compatibles (Además de poseer dicha biblioteca), ya que en este tipo de linkeo la biblioteca se integra al ejecutable cuando se carga el programa a la memoria o se ejecuta por primera vez.

3. En caso de no contar con las bibliotecas a la hora de codificar, utilizaría este tipo de linkeo ya que me permite elegir la biblioteca a la hora de ejecutar el programa.

29) 11. Explique con un ejemplo las estructuras de las tablas de páginas directa, invertida y multinivel.

Tabla de páginas directa: Se indexa por número de página virtual, y el contenido de la misma es el número de frame. De la concatenación del frame junto con el offset de la página virtual (*n* bits menos significativos), se obtiene la dirección en memoria física. Desventaja: como se indexa por página virtual, la tabla puede ser muy grande.

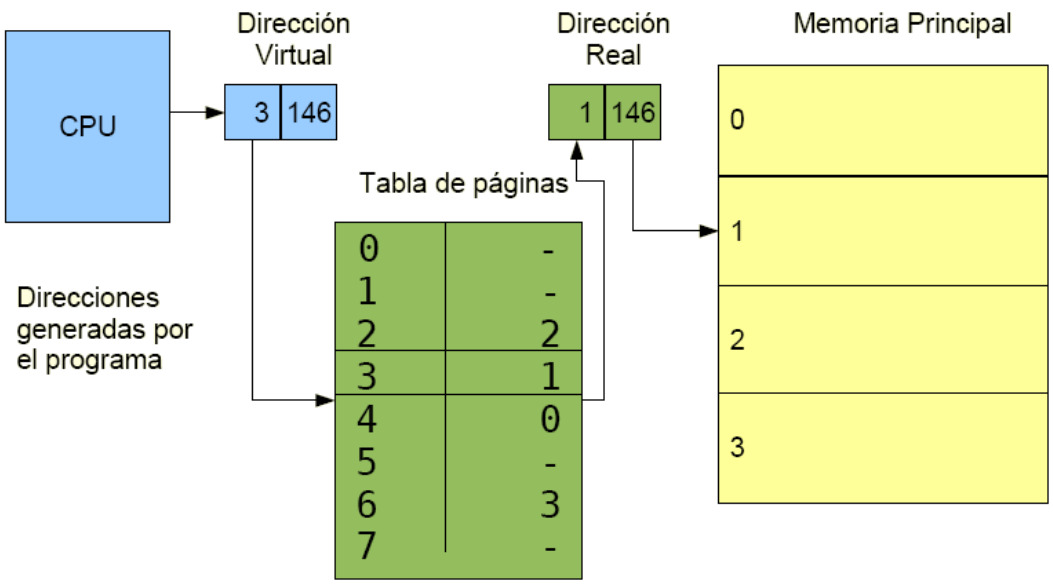


Tabla de páginas invertida: Se indexa por número de frame y el contenido es el número de página virtual. La búsqueda es secuencial hasta encontrar la página virtual buscada. De ahí se saca el índice (Que es número de frame), el cual, nuevamente, concatenado con el offset de la página virtual se obtiene la dirección en memoria física. Ventaja: tablas mucho más chicas. Desventaja: búsqueda secuencial. Para facilitar esta búsqueda se suele recurrir a una tabla de hash que maneja colisiones encadenando dichas páginas virtuales con su frame.

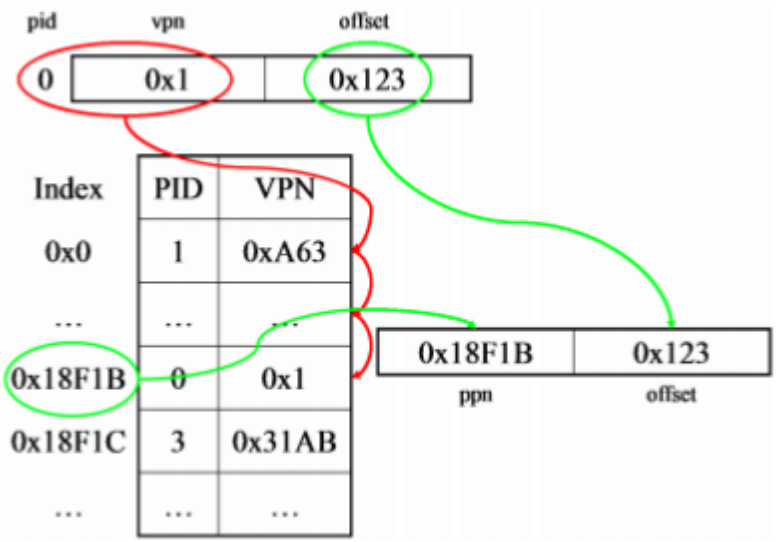
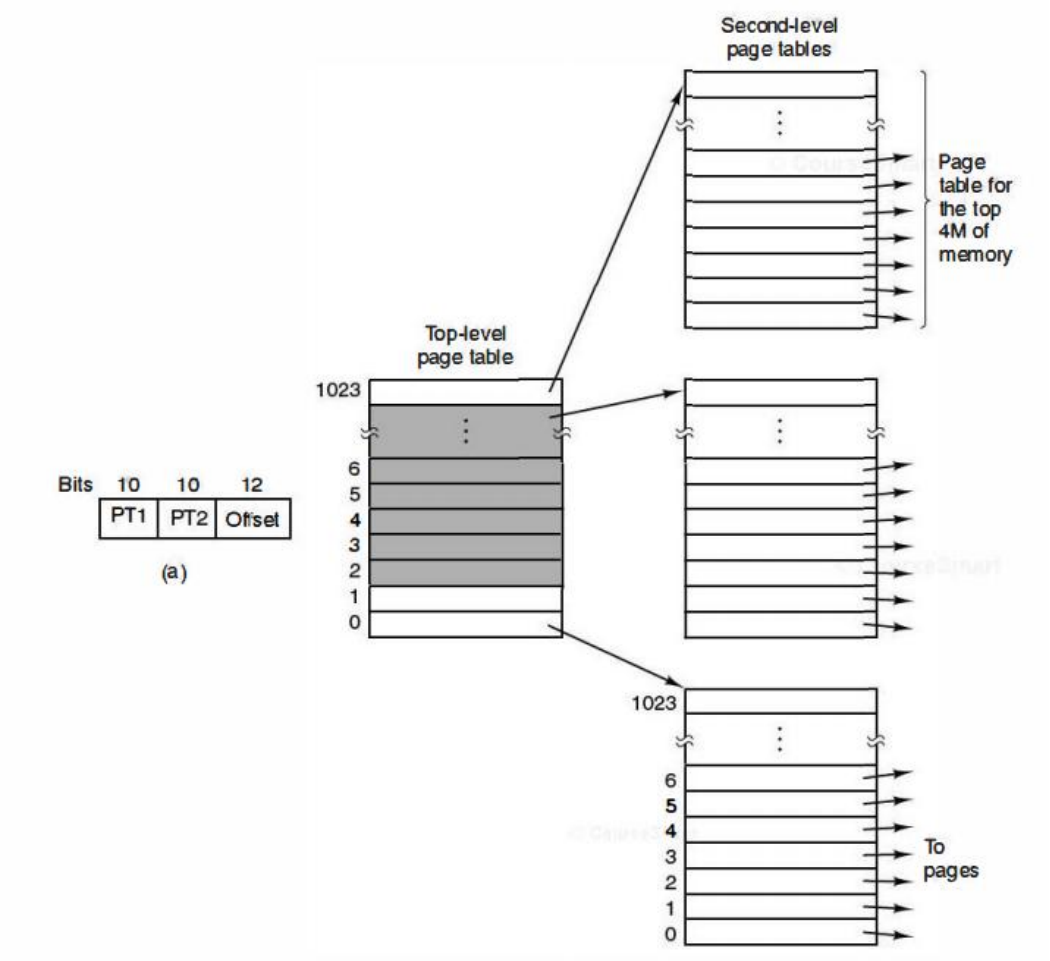


Tabla de páginas multinivel: De la dirección virtual en vez de partirla en VPN y offset, se parte en VPN1, VPN2, …, VPNn y offset. Como no se tienen todas las tablas en memoria todo el tiempo, se ahorra memoria dejándolas en disco.



30) 30. ¿Quien implementa el software de RAID y quien implementa el file system (el controlador o el Sistema Operativo) en?

1. **Un sistema con SAN**
2. **Un sistema con NAS**

En cada caso haga un diagrama indicando que protocolo puede usarse en cada comunicación.

NAS provee almacenamiento y un file system. Esto contrasta con SAN (Storage Area Network), el cual provee sólo almacenamiento basado en bloques y deja el manejo del file system en el lado del cliente.

Respecto al software RAID, cualquiera de las dos arquitecturas soporta dicha implementación.

31) 6. En seudocódigo muy simple indique como hace para llamar a una función de biblioteca func1( ) de la que no conoce el nombre de la biblioteca en que se encuentra hasta el momento de ejecución.

main /* Ejecucion principal */

…

lib_handle = dlopen(lib_name ,RTLD_LAZY) ; /* Se abre la biblioteca dinámica, cuyo nombre es conocido al momento de ejecución */

…

lib_func = dlsym(lib_handle , “func1”) ; /* Se obtiene la funcion pedida */

```
…
(*lib_func)( ) ; /* Se ejecuta la funcion adecuada */
```

32) 15. ¿Qué diferencia hay entre la system call fork() y las exec() (de cualquier tipo)?

1. Responda en base a un ejemplo de código.

2. Indique en su respuesta el contenido antes y después de la ejecución de las áreas PCB, TXT y U_AREA.

La función exec() reemplaza la imagen del padre copiada en el hijo por la propia área static (BSS), TXT y STACK (Es decir, se pisa con la nueva información del hijo). En cuanto al PCB, sigue habiendo un solo PCB, con casi todos los campos iguales. Respecto del U_AREA, el proceso sigue siendo el mismo, sólo cambia su imagen, por lo que su U_AREA es la misma. Esta función se encarga de ejecutar el comando que corresponde al proceso creado.

La función fork() crea un nuevo proceso (proceso hijo), y copia su U_AREA, TXT, stack, y BSS en el nuevo proceso creado. A su vez, el nuevo proceso cuenta con un PCB, con el PID con el que fue creado. La system call fork() devuelve 0 al proceso hijo creado (que puede conocer su PID mediante getpid()) y devuelve el PID del proceso hijo al proceso padre que lo creó.

La system call fork es la única manera de crear un proceso nuevo en POSIX. Crea un duplicado del proceso original, incluyendo todos los file descriptors y registros. La llamada exec reemplaza la imagen por el archivo pasado en el primer parámetro, es decir, se ejecuta el archivo especificado con los parámetros pasados.

Entonces, la diferencia es que en el fork, el proceso hijo es una copia exacta del padre. Se copian todos los datos (file descriptors, registros, etc.). El proceso padre se bloquea esperando a que termine el hijo (si se utiliza wait()). Por otro lado, el exec carga una imagen del programa desde un archivo, en el proceso actual. Se reemplazan todos los datos, excepto los file descriptors.

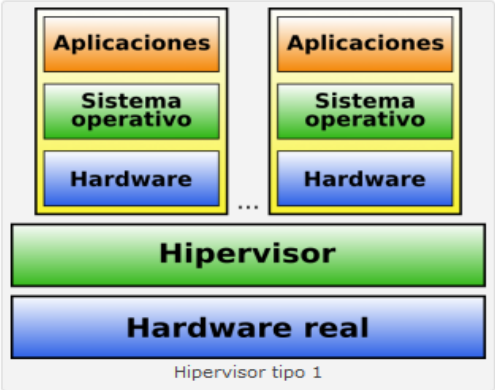
```
read_command(command, parameters);
if (fork() != 0) {
// Proceso padre
// Se bloquea este proceso esperando al hijo
     waitpid(-1, &status, 0);
} else {
// Proceso hijo
// User area distinta al padre
     execve(command, parameters, environment);
}
```

33) ¿Cuál es el principio de funcionamiento de los Hipervisores tipo I?

El hipervisor corre directamente en el hardware del anfitrión para controlar dicho hardware y administrar los sistemas operativos huéspedes. Los sistemas operativos huéspedes corren en un nivel superior a aquel del hipervisor.

El huésped debe tener una arquitectura virtualizable. El sistema operativo huésped corre en modo usuario. Su kernel cree haber pasado a modo supervisor, pero continúa en modo usuario. Al ejecutar una instrucción delicada,se produce una

software trap. Un software ejemplo es el Microsoft Hyper-V. No siempre son más rápidos que los tipo II (Por las software traps).



Hipervisor tipo 1

34) 8. ¿Cuál es la arquitectura general de Android? 3. ¿Qué es una Activity de Android y cuál es su ciclo de vida?

La arquitectura del sistema operativo está dividida en 4 capas:

- Aplicaciones.
- Frameworks.
- Bibliotecas y Android Run Time (Dalvik).
- Kernel Linux.

Una activity es una aplicación que se comunica por medio de una pantalla con el usuario. Generalmente, esta comunicación se realiza fullscreen, pero puede ser una pantalla flotante. Una aplicación está compuesta por una o más activities, pero sólo una puede estar activa a la vez. El resto de las activities se guarda en un stack.

Ciclo de vida:

- Active: está al tope del stack e interactuando con el usuario.
- Paused: visible pero sin foco.
- Stopped: queda en memoria pero ya terminó. Candidata al kill.
- Inactive: fuera de la memoria. Debe lanzarse nuevamente.

35) 22. ¿Como se realiza un write en un Log File System?

Las escrituras se hacen siempre en la cabeza de un log circular como transacciones.

- Se crean entonces versiones del mismo archivo.
- Si éstas son accesibles, es un Versioning File System.
- Si se reconstruye el archivo, se lo llama Journaling File System
- Ext3 o Ext4 son Journaling File System.
- JFFS se usa en las memorias flash para compensar su corta vida útil.

36) 26. ¿Cómo se acceden los archivos mapeados a memoria? Ejemplifique con un pseudocódigo parecido a C.

Ver código en la pregunta 26 de preguntas_y_respuestas_de_so_2013.pdf.