Hola Lia como le va? queria inscribirme en el redictado este primer semestro pero yo no cumplo con las condiciones necesarias para esto (Alumnos que figuren con nota DESAPROBADO en el SIU-GUARANI en la
cursada de Introducción a los Sistemas Operativos del 2do semestre 2019) , ya que a mi se me vencio la materia. Queria saber si existe la posibilidad de incribirme.

Hola Alejandra como estas? me gustaria coordinar para poder hablar y conocer un poco mas sobre NanLab y sus proyectos.
Yo el viernes me tomo una semanita de vacaciones y voy a estar de vuelta en la plata la primer semana marzo, si te parece nos podemos juntar el jueves 5 de marzo.
Saludos, Mariano.

Mayo -2015

1) System Calls:
    a) Definición. Explicar como pueden ser implementadas, para ello tenga en cuenta: participación apoyo del Hardware, Modos de Ejecución, stacks (pilas).
    b) Suponiendo que un proceso X ejecuta la System Call ”fork()” . Describa eventos y actividades que se suceden hasta que el nuevo proceso comienza su ejecución (Tener en cuenta planificadores , dispatcher ,etc)
2) Memoria virtual con paginación por Demanda:
    a) Funcionamiento. Estructuras utilizadas, participación del HW, actividades desarrolladas por el SO.
    b) Fallos de Paginas: Explicar como se resuelven y qué estructuras de las descriptas en el punto anterior son utilizadas (Tanto modificadas, creadas y/o consultadas).
3) Sistema de Archivos.
    a) Describa la estructura del sistema de archivos de UNIX System V.
    b) ¿Si quisiera crear un nuevo archivo, por ejemplo /home/usuario/final.txt qué estructuras de las descriptas en el punto anterior son utilizadas (tanto modificadas, creadas y/O consultadas)?

Marzo - 2015

1) Se quiere implementar el concepto de “Proceso” junto con el “Algoritmo de planificación de CPU Round Robin” en un SO. ¿Como los implementaria? Tenga en cuenta apoyo del HW que necesita y con qué fin (interrupciones, modos de ejecución, etc) así como las estructuras de datos que necesitaría el mantenidas por el SO así como tareas deberá realizar el mismo, implementación del Quantum, tareas realizadas cada vez que un Quantum se consume, etc.

2) Memoria Virtual con Paginación por Demanda: Funcionamiento. Estructuras utilizadas, participación del HW, y actividades desarrolladas por el SO. Fallos de páginas. Definición.

3) **Sistemas de Archivos. Buffer Cache: Relacione los 2 subsistemas de un SO y su interacción. Para un desarrollo ordenado utilice la SysCall "read(fs,buff,count)" indicando las tareas que se suceden desde que un proceso la ejecuta hasta que le es devuelto el control al mismo.**

Septiembre - 2015

1. Procesos y syscall

a. Describa las estructuras de datos y actividades llevadas a cabo para implementar el manejo de procesos. Tener en cuenta: planificación, estados, módulos, etc.

Un proceso está compuesto por:

- Codigo: basicamente la serie de instrucciones que componen la tarea que va a realizar el mismo.
- Datos: variables globales.
- Stack: un proceso puede contar con 1 o más stacks. Se crea automáticamente y su medida se ajusta en run-time. Está formado por stack frames que son pushed (al llamar a una rutina) y popped (cuando se retorna de ella). El stack frame tiene los parámetros de la rutina(variables locales), y datos necesarios para recuperar el stack frame anterior (el contador de programa y el valor del stack pointer en el momento del llamado)

Además se requiere una estructura que contenga toda la información asociada al proceso, llamada **PCB (Process Control Block).** Cada proceso se representa en el sistema operativo mediante un bloque de control de proceso. Cada bloque contiene muchos elementos de información asociados a un proceso específico, entre los que se incluyen:

- **Estado de proceso:** el estado puede ser: nuevo, preparado, en ejecución, en espera , detenido,etc.
- **Contador de programa:** El contador indica la dirección de la siguiente instrucción que va a ejecutar dicho proceso.
- **Registro de la CPU:** los registros varían en cuanto a número y tipo, dependiendo de la arquitectura de la computadora. Incluyen los acumuladores, registros de índice, punteros de pila y registros de propósito general, además de toda la información de los indicadores de estado. Esta información de estado debe guardarse junto con el contador de programa cuando se produce una interrupción,para que luego el proceso pueda continuar ejecutándose correctamente.
- **Información de planificación de la CPU:** Esta información incluye la prioridad del proceso, los punteros a las colas de planificación y cualesquiera otros parámetros de planificación que se requieran.
- **Información de gestión de memoria:** Incluye información acerca del valor de los registros base y limite, las tablas de páginas o las tablas de segmentos, dependiendo del mecanismo de gestión de memoria por el sistema operativo.
- **Información contable:** Esta información incluye la cantidad de CPU y de tiempo real

empleados, los límites de tiempo asignados, los números de cuenta, el número de trabajo o de proceso, etc.

- <u>Información de estado de E/S:</u> incluye la lista de los dispositivos de E/S asignados al proceso, una lista de los archivos abiertos,etc.

Los procesos a lo largo de su paso por la memoria pasan por distintos **<u>estados</u>** como se nombraron anteriormente estos son: nuevo, listo, ejecutando, bloqueado y saliente.

Figura 3.6. Modelo de proceso de cinco estados.

- Nuevo: Un proceso que se acaba de crear y que aún no ha sido admitido en el grupo de los procesos ejecutables.
- LIsto: Un proceso que se prepara para ejecutar cuando tenga la oportunidad.
- Ejecutando: el proceso está actualmente en ejecución.
- Bloqueado: Un proceso que no puede ejecutar hasta que se cumpla un evento determinado o se complete una operación de E/S.
- Saliente: Un proceso que ha sido liberado del grupo de procesos ejecutables por el sistema operativo debido a que ha sido detenido o qué ha sido abortado por alguna razón.

Otros elementos a tener en cuenta son los módulos de planificación. Son módulos de software del SO que realizan distintas tareas asociadas a la planificación. Como por ejemplo creación/terminación de un proceso, eventos de sincronización o de E/S, finalización de lapso de tiempo (quantum),etc .
Los cuales son nombrados en relación a su frecuencia de ejecución:
<u>Long term:</u> Controla el grado de multiprogramación, es decir, la cantidad de procesos en memoria. Puede no existir este scheduler y absorber esta tarea el de short term.
<u>Short term:</u> Decide cuál de los procesos en la cola de listos se elige para que use la CPU.
<u>Medium term(Swaping):</u> Si es necesario, reduce el grado de multiprogramación. Saca temporariamente de memoria los procesos que sea necesario para mantener el equilibrio del sistema.
Otros módulos necesarios son dispatcher y loader. Dispatcher hace el cambio de

contexto, cambio de modo de ejecución y Loader carga en memoria el proceso elegido por el long term.

b. Syscall. Definición. Cómo trabaja fork() en Linux, indique el apoyo por parte de HW.
Cuando utilizamos la System Call Fork(), el proceso hijo creado es una copia exacta del padre excepto por su PID y la memoria que ocupa. Al proceso hijo se le facilita las variables del proceso padre y los descriptores de fichero. Las variables son una copia del padre y no una referencia , por lo que si son modificadas , esto no será reflejado directamente al padre. Para la creación de este nuevo proceso se debe crear una nueva PCB. Asignarle memoria para stack , texto (código) y datos. Normalmente después de la ejecución de fork , se utiliza la system call execve para cargar un nuevo programa en ese espacio de direcciones generado por la sys anterior.
El apoyo de hardware recibido para poder realizar esta tarea es principalmente el necesario para manejar el contexto ya qué se utiliza mucha información como por ejemplo la requerida en un cambio de contexto. Para esto se utiliza el procesador y muchos de sus registros. Además el llamado a una System Call como así también el retorno a la aplicación requiere de un cambio de contexto y de modo donde también es partícipe el procesador.

c. indicar que estructuras de a) son utilizadas y/o modificadas a causa de b)
Primero un proceso usuario hace la llamada a la Syscall fork.
Se guarda en el stack del usuario los parámetros de la syscall y se lanza una interrupción.
Se cambia a modo Kernel.
Se guarda el PC (Program counter: indica la dirección de la siguiente instrucción a ejecutar) y el PSW (Program Status Word: indica el estado del programa actual) en la pila del usuario.
Se carga en el PC la dirección de la rutina que atiende la interrupción.
Se sacan los parámetros de la SysCall de la pila del usuario para ver qué llamada tiene que atender.
Se guarda la dirección de la pila del usuario en el PCB del proceso y se pasa a utilizar la pila del kernel.
Se guardan los parámetros de la Syscall en la pila del kernel y se ejecuta la SysCall
Si la SysCall bloquea el proceso se ejecuta el short term scheduler para que un nuevo proceso tome la cpu y esta no quede ociosa.

a. Se da el context switch
b. se guarda en la pila la dirección que el HW dejo previo a que el proceso seleccionado sea suspendido.

Más allá de que se haya bloqueado o no el proceso, quien ahora tenga el control de la CPU va a cambiar el modo a User Mode.
Ejecutar la sentencia RET para obtener de la pila la dirección de retorno a ejecutar.
Se obtienen de la pila el PSW y PC y se continúa la ejecución del proceso actual.

2. Memoria virtual. Paginación con demanda

a. Cómo y quién realiza la traducción lógica en física. Tener en cuenta HW,actividades del SO, estructuras de datos, etc.

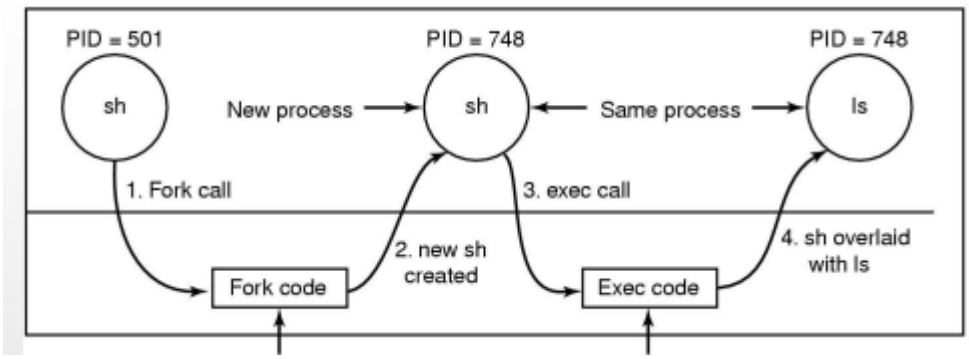
Paginación por demanda, bajo esta técnica la memoria es dividida en partes iguales llamadas “marco”. El espacio de direcciones de los procesos es dividido en partes del mismo tamaño que los marcos llamados “páginas”. El SO mantiene una tabla de páginas por cada proceso. Cada entrada de la tabla contiene en qué marco se encuentra cada página.
Esta técnica utiliza un intercambiador de procesos que se encarga de nunca intercambiar una página a menos que sea necesario. Cuando hay que cargar un proceso el paginador realiza una verificación de qué páginas serán utilizadas antes de cargarlo. En lugar de cargar el proceso completo, solo se cargan las páginas necesarias. Reduciendo el tiempo de carga y la memoria física ocupada.
La tabla de páginas puede marcar a cada entrada con un bit V que indica si la página se encuentra en memoria y con un bit M que indica si la página fue modificada. Si se modifico el cambio se debe reflejar en memoria secundaria.
El hardware para soportar paginación por demanda es el mismo que para paginación y swaping. Se requiere un área de intercambio (EJ: partición de swap). Es el espacio que se requiere para mover un apagia de memoria principal a secundaria. De esa manera poder liberar memoria principal para poder cargar páginas nuevas que sean requeridas. Y en el caso que sean necesarias nuevamente retornarlas a memoria principal.
TLB: caché que cuenta con la entrada de las tablas de página que fueron referenciadas recientemente. Es administrada por la MMU para reducir lecturas en memoria principal.
MMU: Se encarga de traducir direcciones virtuales a físicas. Genera una interrupción cuando hace referencia a una página que no se encuentra en memoria.

Dirección lógica: Se referencia a una localidad de memoria independiente de la asignación actual de los datos en la memoria. Se debe realizar una traducción a una dirección física.
Dirección física: La dirección absoluta en la memoria principal.
Conversión de direcciones:
Una forma simple de utilizar esto es utilizando registros auxiliares.

Registro base: Dirección de comienzo del proceso

Registro Límite: Direccion final del proceso o medida del proceso.

Su valor se fija cuando el proceso es cargado en memoria (Context Switch). Se utiliza la dirección lógica junto con el registro base para obtener una dirección física (según la técnica



utilizada). El resultado se compara con el valor del registro límite (según la técnica utilizada). Si la dirección generada es incorrecta se genera una interrupción del SO.
Las direcciones en los programas fuentes son simbólicas, el compilador las convierte en direcciones reubicables, y el linkeditor en direcciones absolutas, este proceso se conoce como binding de direcciones.
Al hacer Swap out y Swap in de un proceso, si trabajamos con direcciones físicas se debe asignar al mismo espacio de memoria que ocupaba antes, en cambio con las lógicas se puede cargar en cualquier lado.
El mapeo de direcciones lógicas y físicas es realizado por el dispositivo de hardware **MMU** (Memory Managment Unit), se encuentra en el cpu y es reprogramado por el kernel.
El valor en el "registro de realocación" es sumado a cada dirección generada por el proceso de usuario al momento de acceder a la memoria.

b. Hiperpaginación. Definición. Cómo puede ser tratada? Relación con working set.
Un sistema se encuentra en trashing (Hiperpaginación) cuando pasa más tiempo paginando qué ejecutando procesos. En consecuencia, hay una baja importante de performance en el sistema. Si un proceso cuenta con todos los frames que necesita, no habría trashing.
Se puede limitar el trashing usando algoritmos de reemplazo local. Con este algoritmo, si un proceso entra en trashing no roba frames a otros procesos. Si bien perjudica la performance del sistema, es controlable.
La forma de tratar la hiperpaginación es mediante la utilización de algoritmos de reemplazo de páginas de manera local. Es decir que si un proceso produce un fallo de página no robara frame a otro proceso sino que se limitará a utilizar sus marcos reemplazando sus propias páginas para evitar que otros procesos generen un fallo de página. Los algoritmos de reemplazo tanto locales como globales compartiendo o no frames están basados en los principios de localidad.
Localidad temporal: Si en un momento una porción de memoria es referenciada es muy probable que la misma posición sea referenciada nuevamente en un futuro cercano.
Localidad espacial: Si una porción de memoria es referenciada en un momento concreto es probable que las localidades cercanas a ella también serán referenciadas pronto.
Algoritmos de reemplazo
FIFO:(First in first out): Como su nombre lo indica, la primera página que fue cargada a memoria es la primera en salir de esta.

Segunda chance: Es una pequeña modificación de FIFO. Cuando se saca una pagina se toma la primera de la cola. Y en vez de sacarla se consulta el valor de un bit de referencia. Si está en 1 se pone en 0 y se vuelve a poner en la cola. Si es 0 se saca. Cada vez que la MMU accede a una página se fija su bit de referencia en 1. Para esto es necesario soporte de bit de referencia por hardware.

LRU: (Least recently used): Plantea quitar de memoria las páginas menos usadas recientemente, para ello, ordena las páginas poniendo arriba las que fueron usadas recientemente y va reemplazando por las páginas que se sitúan abajo.

NRU: (no recientemente usada) favorece a las páginas que fueron usadas recientemente. Cuando una página es referenciada se fija el bit de referencia para esta página. Del mismo modo ocurre cuando esta pagina es modificada (se fija el bit de modificada). Usualmente estas acciones son realizadas por el hardware. Aunque también pueden ser realizadas por

software.
cuando una página debe ser reemplazada el SO divide las páginas en 4 categorías:

- categoría 0: no referenciada , no modificada
- categoría 1: no referenciada modificada
- categoría 2: referenciada , no modificada
- categoría 3: referenciada modificada

Se desaloja una página al azar de la categoría más baja que no esté vacía.

ÓPTIMO: Tiene como finalidad retirar la pagina que sea referenciada más tarde. Para ello se necesita predecir la página lo cual es imposible. Se utiliza este modelo para poder compararla con un algoritmo implementable. El que más se acerque al óptimo será el mejor.

Working set: es el conjunto de páginas que tienen las más recientes referencias a páginas. Si el working set es demasiado chico y se necesitan muchas páginas disponibles , estas deberán ser cargadas por otras , si esta situación se mantiene se genera trashing . Si se puede aumentar el working set probablemente se puedan reemplazar las páginas por otras que no sean requeridas en un futuro cercano (utilizando alguno de los algoritmos recién nombrados).

3. Sistema de E/S.

Describa el sistema de archivos en Unix System V. describa los pasos necesarios para abrir un archivo open('/home/iso/finales.doc').

2017 - “The Mouseros”

1. **Identifique qué características debe tener el hardware (apoyo de hardware) para que el SO pueda brindar a los procesos protección del uso de la CPU. Para cada una de las características, justifique por qué es necesario contar con ella.**

   Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos, sean accidentales o intencionadas. Otros programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura. Por un lado, lograr los requisitos de la reubicación incrementa la dificultad de satisfacer los requisitos de protección. Todas las referencias de memoria generadas por un proceso deben comprobarse en tiempo de ejecución para poder asegurar que se refieren sólo al espacio de memoria asignado a dicho proceso.

   Normalmente, un proceso usuario no puede acceder a cualquier porción del sistema operativo, ni al código ni a los datos. De nuevo, un programa de un proceso no puede saltar a una instrucción de otro proceso. Sin un trato especial, un programa de un proceso no puede acceder al área de datos de otro proceso. El procesador debe ser capaz de abortar tales instrucciones en el punto de ejecución.

   Los requisitos de memoria deben ser satisfechos por el procesador (Hardware) en lugar del sistema operativo (software). Esto es debido a que el sistema operativo no puede anticipar todas las referencias de memoria que un programa hará. Incluso si tal anticipación fuera posible, llevaría demasiado tiempo de ejecución de la instrucción que realiza dicha referencia. Para llevar a cabo esto el hardware del procesador debe tener esta capacidad.

2. **Relacione y explique cómo, con el apoyo del hardware indicado en el punto 1, el SO realiza una planificación de CPU Round Robin. Tenga en cuenta interrupciones, modos de ejecución, módulos de planificación, estados de los procesos, etc.**

   Round-robin es un algoritmo de planificación de procesos simple de implementar. Dentro de un sistema operativo se asigna a cada proceso una porción de tiempo equitativa y ordenada, tratando a todos los procesos con la misma prioridad. En Sistemas operativos, la planificación Round-robin da un tiempo máximo de uso de CPU a cada proceso, pasado el cual es desalojado y retornado al estado de listo, la lista de procesos se planifica por FIFO, del inglés "First In, First Out" (primero en entrar, primero en salir o primero llegado, primero atendido).

   Este algoritmo de planificación, conocido por Round robin, está diseñado especialmente para sistemas de tiempo compartido. Se define un intervalo de tiempo denominado "Quantum", cuya duración varía según el sistema. La cola de procesos se estructura como una cola circular. El planificador la recorre asignando un cuanto de tiempo a cada proceso. La organización de la cola es FIFO. El Quantum se suele implementar mediante un temporizador que genera una interrupción cuando se agota el Quantum de tiempo. Si el proceso agota su ráfaga de CPU antes de finalizar el Quantum, él planificador

asigna la CPU inmediatamente a otro proceso. Este algoritmo tiene un tiempo de espera relativamente grande.

// agregar cambio de contexto

3. **Explique qué es la PCB. Indique en cuales momentos su contenido cambia. Tenga en cuenta estados de un proceso, cambio de modo, cambio de contexto, etc.**

Process Control Block (PCB):

Cada proceso se representa en el sistema operativo mediante un bloque de control de proceso (PCB, process control block), también denominado bloque de control de tarea. Un bloque de control de proceso contiene muchos elementos de información asociados con un proceso específico, entre los que se incluyen:

- **Estado del proceso:** el estado puede ser: nuevo, preparado, en ejecución, en espera, detenido, etc.
- **Contador de programa:** El contador indica la dirección de la siguiente instrucción que va a ejecutar dicho proceso.
- **Registro de la CPU:** los registro varían en cuanto a número y tipo, dependiendo de la arquitectura de la computadora. Incluyen los acumuladores, registros de índice, punteros de pila y registros de propósito general, además de toda la información de los indicadores de estado. Esta información de estado debe guardarse junto con el contador de programa cuando se produce una interrupción,para que luego el proceso pueda continuar ejecutándose correctamente.
- **Información de planificación de la CPU:** Esta información incluye la prioridad del proceso, los punteros a las colas de planificación y cualesquiera otros parámetros de planificación que se requieran.
- **Información de gestión de memoria:** Incluye información acerca del valor de los registros base y limite, las tablas de páginas o las tablas de segmentos, dependiendo del mecanismo de gestión de memoria por el sistema operativo.
- **Información contable:** Esta información incluye la cantidad de CPU y de tiempo real empleados, los límites de tiempo asignados, los límites de tiempo asignados, los números de cuenta, el número de trabajo o de proceso, etc.
- **Información de estado de E/S:** incluye la lista de los dispositivos de E/S asignados al proceso, una lista de los archivos abiertos,etc.

4. **Identifique qué características debe tener el hardware (apoyo de hardware) se deben agregar para pasar de un SO que administra memoria con paginación pura a un SO que utiliza paginación por demanda. Para cada una de las características, justifique por qué es necesario contar con ella. Indique y justifique también los cambios necesarios en la tabla de páginas.**

5. **Relacione y explique cómo, con el apoyo el apoyo del hardware indicado en el punto 4, se realiza la traducción de una dirección que provoca un TLB hit y otra que provoca un TLB miss pero no un page fault. Tenga en cuenta indicar estructuras utilizadas y actualizaciones necesarias tanto en estructuras como en elementos de hardware.**
   Cada referencia en el espacio virtual puede causar 2 (o más) accesos a la memoria física. Uno (o más) para obtener la entrada en la tabla de páginas. Y otro para obtener los datos.
   Para solucionar este problema, una memoria caché de alta velocidad es usada para almacenar entradas de páginas **(TLB - Translation Lookaside Buffer)**.
   La TLB contiene las entradas de la tabla de páginas que fueron usadas más recientemente. Dada una dirección virtual, el procesador examina la TLB. Si la entrada de la tabla de páginas se encuentra en la TLB (hit). es obtenido el frame y armada la dirección física. Si la entrada no es encontrada en la TLB (miss), el numero de pagina es usado como índice en la tabla de páginas del proceso. Se controla si la página está en memoria. Si no esta, se genera un Page Fault. La TLB es actualizada para incluir la nueva entrada.

   - en caso del page fault hay cambio de contexto por la interrupción/trap generado
   - MMU
   ----O--- // página 196 Tanenbaum
   El dispositivo, llamado TLB (Translation Lookaside Buffer, Búfer de traducción adelantada) o algunas veces memoria asociativa. Por lo general se encuentra dentro de la MMU y consiste en un pequeño número de entradas. Cada entrada contiene información acerca de una página, incluyendo el número de página virtual, un bit que se establece cuando se modifica la página, el código de protección (permisos de lectura/escritura/ejecución) y el marco de página físico en el que se encuentra la página. Estos campos tienen una correspondencia de uno a uno con los campos en la tabla de páginas, excepto por el número de página virtual, que no se necesita en la tabla de páginas. Otro bit indica si la entrada es válida (es decir, si está en uso) o no.

Válida	Página virtual	Modificada	Protección	Marco de página
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Cuando se presenta una dirección virtual a la MMU para que la traduzca, el hardware primero comprueba si su número de página virtual está presente en el TLB al compararla con todas las entradas en forma simultánea (es decir en paralelo). Si se encuentra una coincidencia válida y el acceso no viola los bits de protección, el marco de página se toma directamente del TLB, sin pasar por la tabla de páginas. Si el número de página virtual está presente en el TLB, pero la instrucción está tratando de escribir en una pagina de solo lectura, se genera un fallo por protección.
El caso interesante es lo que ocurre cuando el número de página virtual no está en el TLB. La MMU detecta que no está y realiza una búsqueda ordinaria en la tabla de páginas. Después desalo- ja una de las entradas del TLB y la reemplaza con la entrada en la tabla de páginas que acaba de buscar. De esta forma, si esa página se utiliza pronto otra vez, la segunda vez se producirá un acier- to en el TLB en vez de un fracaso. Cuando se purga una entrada del TLB, el bit modificado se co- pia de vuelta a la entrada en la tabla de páginas en memoria. Los otros valores ya están ahí, excepto el bit de referencia. Cuando el TLB se carga de la tabla de páginas, todos los campos se toman de la memoria.

6. **Relacione y explique cómo, con el apoyo del hardware indicado en el punto 4, se realiza la traducción de una dirección que provoca page fault. Tenga en cuenta indicar estructuras utilizadas, interrupciones o traps, modos de ejecución, módulos de planificación, estados de los procesos, manejo de interrupciones, E/S necesarias, etc.**

7. **Describa la estructura del volumen del sistema de archivos de UNIX System V (visto en la teoría).**

System V UNIX
UNIX - Estructura del Volumen
- **Boot Block:** Ocupa la parte del principio del sistema de archivos,

normalmente en el primer sector, y puede contener el código de arranque. Este codigo es un pequeño programa que se encarga de buscar el sistema operativo y cargarlo en memoria.

- **Superblock:** Describe el estado de un sistema de archivos. Contiene información de su tamaño total del archivo que puede contener, qué espacio libre queda, etc.
- **I-NODE Table:** Tabla que contiene todos los I-NODOS
- **I-NODO:** Estructura de control que contiene la información clave de un archivo
- **Data Blocks:** empiezan a continuación de la lista de inodos y ocupa el resto del sistema de archivos. En esta zona es donde se encuentra situado el contenido de los archivos a los que hace referencia la lista de inodos. Cada uno de los bloques destinados a datos sólo puede ser asignado a un archivo, tanto si lo ocupa totalmente como si no.

8. **¿Con cuál de los tipos de asignación de memoria asocia el concepto de i-nodo?¿preasignación, asignación continua, asignación encadenada o asignación indexada? Justifique**

Tipos de asignación de memoria(almacenamiento Ej: disco):

Pre-asignación: Se necesita saber cuánto espacio va a ocupar el archivo en el momento de su creación. Se tiende a definir espacios mucho más grandes qué lo necesario. Posibilidad de utilizar sectores contiguos para almacenar los datos de un archivo.

Dinámica: El espacio se solicita a medida que se necesita. Los bloques de datos pueden quedar de manera no contigua.

Continúa: Conjunto continuo de bloques son utilizados. Se requiere una pre-asignación. Se debe conocer el tamaño del archivo durante su creación. File Allocation Table (FAT) es simple. Solo una entrada que incluye bloque de inicio y longitud. El archivo puede ser leído con una unica operacion. Puede existir fragmentación externa. Compactación .

Problemas de la técnica. Encontrar bloques libres continuos en el disco. Incremento del tamaño de un archivo.

Encadenada: Asignación en base a bloques individuales . Cada bloque tiene un puntero al próximo bloque del archivo. File allocation table: Única entrada por archivo. Bloque de inicio y tamaño del archivo. No hay fragmentación externa. Útil para acceso secuencial (no random). Los archivos pueden crecer bajo demanda. No se requieren bloques contiguos. Se pueden consolidar los bloques para garantizar cercanía de los bloques de un mismo archivo.

Indexada: Asignación de bloques en base a bloques individuales. No se produce fragmentación externa. El acceso random a un archivo es eficiente. File allocation Table: Única entrada con la dirección del bloque de índices (index node / i-node). Puede tener niveles de indirección.

Respuesta:

El tipo de asignación de memoria más adecuado para trabajar con i-nodos es el indexado. Ya que se acceden a los mismos de manera aleatoria, y esta asignación es eficiente en este caso. Además proporciona mecanismos de acceso indirecto permitiendonos trabajar mejor con una amplia cantidad de direcciones.

9. **Explique y ejemplifique como en el sistema de archivos de UNIX System V, a partir del path absoluto de un archivo, por ejemplo "/home/documentos/final.txt", se localizan los bloques de datos del contenido de dicho archivo.**

2016 - Febrero

1. **System Calls:**

   a. ¿Por qué son necesarias? ¿Por qué no pueden ejecutarse en modo usuario?

   Las System call's son necesarias ya que proporcionan una interfaz con la que poder invocar los servicios que el SO ofrece. Los desarrolladores de aplicaciones diseñan sus programas utilizando una **API** (application programming interface), esta especifica un conjunto de funciones que el programador de aplicaciones puede usar, indicando los parámetros que hay que pasar a cada función y los valores de retorno que el programador debe esperar. Es decir, que una API es un ambiente de programación provisto por el SO, que nos da un interfaz para la system call.

   Ya qué se tratan de operaciones proporcionadas por el núcleo del sistema, las System call se ejecutan en modo supervisor/privilegiado . Esto es así por varios motivos, pero fundamentalmente por seguridad. De este modo se protege el funcionamiento del kernel, si no fuera así, se dejaría en manos en muchos casos en desarrollo de la aplicación. Ya que las System calls tiene acceso a funciones muy delicadas como control de procesos, manejo de archivos, mantenimiento de información del sistema etc.

   Si esto no fuera así el sistema sería muy vulnerable a usuarios malintencionados.

   b. Se quiere ejecutar la instrucción que implica una lectura de una fila de una base de datos qué es en disco. Considerando que el controlador no sabe de base de datos … Como se transforma esa instrucción en una orden para el controlador de disco? Cual seria ese comando? Load o store?

   Tema 3 - Memoria 1

   Swapped in - Load

   SysCall read () ??

2. **Memoria Virtual con Paginación por Demanda:**

   a. ¿Cómo Funciona?. Estructuras utilizadas, participación del HW, y actividades

desarrolladas por el SO. Fallos de páginas.

Consiste en dividir los procesos en páginas con el fin de cargar en memoria principal solo las partes que sean necesarias. Con la memoria virtual basada en paginación bajo demanda solo se cargan las paginas cuando se solicitan durante la ejecución del programa, de este modo las páginas a las que nunca se acceda nunca serán cargadas en memoria. Cada proceso tiene una tabla de páginas. Cada entrada en la tabla, referencia al frame o marco en el que se encuentra la página en la memoria principal. Cada entrada en la tabla de páginas tiene bits de control (entre otros):

- Bit V: Indica si la página está en memoria
- Bit M: Indica si la página fue modificada. Si se modifica en algún momento, se deben reflejar los cambios en memoria secundaria.

Los fallos de página ocurren cuando el proceso intenta usar una dirección que está en una página que no se encuentra en memoria principal. Bit V = 0. La página no se encuentra en su "working set". El bit V es controlado por hardware. El hardware detecta la situación y genera un trap al S.O. El S.O podrá colocar al proceso en estado "Blocked" (espera) mientras gestiona que la página que se necesite se cargue. El S.O busca un "Frame o Marco" en la memoria y genera una operación de E/S al disco para copiar en dicho Frame la pagina del proceso que necesita utilizar. El S.O puede asignarle la CPU a otro proceso mientras se completa la E/S. La E/S se realizará y avisará mediante interrupción su finalización.

Cuando la operación de E/S finaliza, se notifica al SO y este:

Actualiza la tabla de páginas del proceso

- Coloca el Bit V en 1 en la página en cuestión
- Coloca la direccion base del Frame donde se coloco la pagina

El proceso que generó el Fallo de página vuelve a estado de Ready (listo). Cuando el proceso se ejecute, se volverá a ejecutar la instrucción que antes generó el fallo de página.

b. Hiperpaginación (Thrashing). Definición, formas de detectarla y tratarla.

Un sistema está en trashing cuando pasa más tiempo paginando qué ejecutando procesos. Como consecuencia hay una baja importante de performance en el sistema.

Ciclo de trashing:

1) El SO monitorea el uso de la CPU.
2) Si hay baja utilización => aumenta el grado de multiprogramación.
3) Si el algoritmo de reemplazo es global, pueden sacarse frames a otros procesos.
4) Un proceso necesita más frames. Comienzan los page-faults y robo de frames a otros procesos.
5) Por swapping de páginas, y encolamiento en dispositivos, baja el uso de la CPU.

6) Vuelve a 1)

El scheduler de CPU y el trashing

Cuando se decrementa el uso de la CPU, el scheduler long term aumenta el grado de multiprogramación.

El nuevo proceso inicia nuevos page-fault, y por lo tanto, más actividad de paginado.

Se decrementa el uso de la CPU.

Vuelve a 1).

Control de trashing:

Se puede limitar el trashing usando algoritmos de reemplazo local. Con este algoritmo, si un proceso entra en trashing no roba frames a otros procesos. Si bien perjudica la performance del sistema, es confortable.

Conclusión sobre trashing:

Si un proceso cuenta con todos los frames que necesita, no habría más trashing.

3. Sistemas de Archivos.

Describir la estructura del sistema de archivos de UNIX System V. ¿Si quisiera borrar un archivo, por ejemplo /home/usuario/final.txt, qué estructuras se verían afectadas? ¿Como?

Un sistema de archivos permite realizar una abstracción de los dispositivos físicos de almacenamiento de la información para que sean tratados a nivel lógico, como una estructura de más alto nivel y más sencilla que la estructura de su arquitectura hardware particular.

Se caracteriza por:

- Poseer una estructura jerárquica
- Realizar un tratamiento consiste de los datos de los archivos
- Poder crear y borrar archivos
- Permitir un crecimiento dinámico de los archivos
- Proteger los datos de los archivos
- Tratar a los dispositivos y periféricos (terminales, unidades de disco, etc).

Estructura:

- Bloque de arranque (boot): Ocupa la parte del principio del sistema de archivos, normalmente en el primer sector, y puede contener el código de arranque. Este codigo es un pequeño programa que se encarga de buscar el sistema operativo y cargarlo.

- Superbloque: describe el estado de un sistema de archivos. Contiene información acerca de su tamaño, total de archivos que puede contener, qué espacio libre le queda, etc.

- Lista de inodos: Se encuentra a continuación del superbloque. Esta lista tiene una entrada por cada archivo, donde se guarda una descripción del mismo: situación del archivo en el disco, propietario, permisos de acceso,

fecha de actualización, etc. El administrador del sistema es el encargado de especificar el tamaño de la lista de inodos al configurar el sistema.
- Bloques de datos: empiezan a continuación de la lista de inodos y ocupa el resto del sistema de archivos. En esta zona es donde se encuentra situado el contenido de los archivos a los que hace referencia la lista de inodos. Cada uno de los bloques destinados a datos sólo puede ser asignado a un archivo, tanto si lo ocupa totalmente como si no.

Nov - 2016

1) **System calls y Procesos**
   a) Suponiendo que un proceso X utiliza una System call qué ocasiona que el mismo quede bloqueado, por ejemplo read(). Analice y describa qué eventos , actividades, se suceden hasta que otro proceso Y (que se encontraba en el estado de listo) comienza con su ejecución (Tener en cuenta: las estructuras y eventos relacionados a planificación, apoyo del HW, modos de ejecución, Stacks, módulos de planificación, etc).

2) **Administración de Memoria**
   a) **Paginación y Segmentación. Cómo funcionan (Tenga en cuenta estructuras necesarias, participación del HW, etc). Cite ventajas y desventajas de un esquema respecto al otro.**
   La **paginación** consiste en dividir la toda memoria en porciones del mismo tamaño fijo, conocidos como marcos. Los procesos son divididos en porciones llamadas páginas. Este mecanismo no presenta fragmentación externa , pero si fragmentación interna en el espacio desperdiciado por la última página de cada proceso.
   El SO mantiene una lista de marcos libres. No es necesario asignar memoria a un proceso de manera contigua. Existe el concepto de dirección lógica. El SO mantiene una tabla de páginas por cada proceso. La tabla de páginas muestra la ubicación del marco por cada pagina del proceso. Cada dirección lógica está formada por un número de página y un desplazamiento dentro de la página. La traducción de direcciones lógicas a físicas las realiza el hardware del procesador. El procesador debe saber cómo acceder a la tabla de páginas del proceso actual. Presentando como una dirección lógica (número de página, desplazamiento), el procesador utiliza la tabla para producir la dirección física (número de marco, desplazamiento).
   Por ejemplo una dirección de n + m bits, donde los n bits de la izquierda corresponden al numero de pagina y los m bits de la derecha corresponden al desplazamiento
      - Extraer el número de página como los n bits de la izquierda de la direccion logica.

- Utilizar el número de página como un índice a la tabla de páginas del proceso para encontrar el número de marco, k.
- La dirección física inicial del marco es k * 2^m , y a dirección física del byte referenciado es dicho numero más el desplazamiento. Esta dirección física no necesita calcularse; es fácilmente construida concatenando el número de marco al desplazamiento.

La **segmentación** consiste en subdividir el programa y sus datos en segmentos. Los cuales no tienen que ser de igual tamaño en contraste de lo que son las páginas en paginación. En este caso una dirección lógica está compuesta por un número de segmento y un desplazamiento. La fragmentación sufre la partición externa pero no fragmentación interna. La paginación es invisible al programador en cambio la segmentación no lo es. Normalmente el programador o compilador asignara programas y datos a diferentes segmentos. A su vez los programas y datos se pueden dividir en múltiples segmentos. Por ejemplo librerías por un lado, programa principal en otro etc.

Al igual que en paginación se utiliza una tabla. Por cada entrada a la tabla, se obtiene la dirección inicial de cada segmento, además debe proporcionar la longitud del segmento, para asegurar que no se utilizan direcciones inválidas. Cuando un proceso entra en el estado Ejecutando, la direccion de su tabla de segmentos se carga en un registro especial utilizado por el hardware de gestión de memoria.

Considérese una dirección de n+m bits, donde los nbits de la izquierda corresponden al número de segmento y los m bits de la derecha corresponden al desplazamiento.Se necesita llevar a cabo los siguientes pasos para la traducción de decisiones:

- Extraer el número de segmento como los n bits de la izquierda de la dirección lógica.
- Utilizar el número de segmento como un índice a la tabla de segmentos del proceso para encontrar la dirección física inicial del segmento.
- Comparar el desplazamiento, expresado como los m bits de la derecha, y la longitud del segmento. Si el desplazamiento es mayor o igual que la longitud, de la dirección no es válida.
- La dirección física deseada es la suma de la dirección física inicial y el desplazamiento.

b) Hiperpaginación (Trashing): Definición , formas de detectarla y tratarla.

3) Sistemas de Archivos

a) Describe la estructura del sistema de archivos de UNIX System V . ¿Si quisiera crear un nuevo archivo. por ejemplo /home/usuario/final.txt, qué estructuras de las indicadas son utilizadas (modificadas, creadas y/o utilizadas)?

CREAR

Para crear un nuevo archivo en el sistema de archivos. Tiene la misma funcionalidad que open. Si el archivo no existe , se crea uno nuevo. Si ya existía, trunca su contenido.
El algoritmo intenta obtener el i-nodo a partir del nombre (/home/usuario/final.txt) .
Si el archivo no existía. Se asigna el inodo libre y se crea la entrada en el directorio padre si tenemos permisos de escritura en el directorio padre.
Si ya estia:

Si el acceso no está permitido retorna error. Si tiene acceso, se trunca el archivo si tenemos permiso de escritura sobre el archivo. Liberar todos sus bloques de datos.

En cualquier caso: Asignar una entrada en la tabla de archivos, cuenta de referencia =1.
Asignar una entrada en la tabla de descriptores de archivos.

LEER

Se obtiene la entrada en la tabla de archivos a partir del descriptor del archivo. Se comprueba tener los permisos de lectura. Se debe obtener el inodo siguiendo los punteros.
Hasta el fin de la lectura, error o no quedan bytes en el archivo:

- Desplazamiento → nº de bloque de disco (bmap)
- Calcula el desplazamiento en el bloque para comenzar la E/S
- Calcular el nº de bytes a leer dentro del bloque
- Si el nº de bytes a leer = 0 → fin de la busqueda
- Sino:
    - Leer el bloque en un buffer (buffer cache).
    - Copiar los datos del buffer → dirección destino del proceso usuario.
    - Actualiza el desplazamiento en el archivo, direccion en el proceso de usuario, nº de bytes qué quedan por leer.

actualizar el desplazamiento en la tabla de archivos.

GLOSARIO:

ESPACIO VIRTUAL / DIRECCIÓN VIRTUAL: