Universidad Francisco Marroquin
Sistemas Operativos

Resumen Capítulo 8 Memoria

Introduccion

Como se ha mencionado en los capítulos anteriores, uno de los factores importantes para realizar más procesos en un tiempo determinado por parte del procesador es que los programas estén listos en la memoria principal, la cual es la única memoria que tiene al alcanze el CPU además de los registros. Es importante la organización de estos programas en la memoria principal para poder cargar los programas al CPU de la manera más eficiente. EL modo en que se cargan y organizan los programas en la memoria principal es dependiente de la aplicación del sistema operativo y el hardware.

Hardware

Ya que la única memoria que el CPU tiene acceso directamente es la memoria principal, aun el mover las instrucciones de la memoria al CPU es un proceso que toma varios ticks del reloj y si en dado caso no se tiene una instrucción en los registros del CPU, pues este tiene que detenerse a esperar a que se pasen las instrucciones de memoria al CPU, lo cual es caso muy costoso, por esto existe una memoria que se encuentra en medio del CPU y la Memoria principal y es la memoria cache la cual garantiza una mayor velocidad de carga de los procesos al CPU.

Ahora hablando de cómo garantizamos en un ambiente de multiprocesamiento que un programa no trate de modificar la región de memoria de otro programa? Bueno pues esto se podría hacer por medio de software, osea por medio del sistema operativo, pero esto sería extremadamente caro pues por cada dirección de memoria que se maneje, un proceso del sistem operativo debería de chequear si esta memoria realmente pertenece al proceso o no. La solución a esto es que el mismo hardware cuide los espacios de memoria de cada proceso esto por medio obtener almacenado en registros en donde empieza un programa y en otro registro almacenar dónde está el fin del espacio de memoria para dicho programa. Entonces se contaría con un registro llamado base y el registro límite respectivamente para indicar donde termina el espacio de un proceso.

Estos registros solamente el sistema operativo los puede cargar o modificar por medio de su ejecución en modo kernel, esto para garantizar que los procesos de los usuario modifique estos registros. Si un programa trata de modificar o saltar un registro fuera de su espacio de memoria, esto se trata como un error.

Ricardo Mendoza

Direccionamiento

Los programas al no almacenarse en memoria principal siempre, son almacenados en otro tipo de almacenamiento, como los discos duros, Al estar almacenados en discos, las direcciones de memoria que manejan los programas pues son direcciones relativas al espacio en donde será localizado el programa. Ahora la relación entre estas direccione relativas a las absolutas osea las ya fisicas se puede realizar en diferentes momentos del proceso desde que el programa es jalado del disco duro hasta que llega al CPU y es ejecutado. Los momentos en los que se puede hacer esto es:

- Durante el tiempo de compilación del programa, esto solo si el compilador ya sabe en qué espacio de la memoria va a ser localizado el programa, pues puede generar el código utilizando las direcciones físicas o absolutas de la memoria principal.
- Durante la carga del programa a la memoria principal, este se da cuando el compilador no conoce el espacio específico de memoria donde residirá el programa, entonces genera código para que las direcciones se recarguen al cargar el programa.
- Durante el tiempo de ejecución del programa, este se da cuando por ejemplo un programa se mueve de un espacio de memoria a otro por orden del sistema operativo, entonces las direcciones que contenía dicho programa deben de ser actualizadas a las nuevas absolutas en las que estará.

Las direcciones de memoria se puede decir que son lógicas o físicas y esto depende de quien esté manejando la dirección, como ya sabemos el CPU genera y ve direcciones lógicas que técnicamente son relativas al programa y las direcciones físicas son las direcciones absolutas de los segmentos o registros de la memoria principal. Por esto es que se mapean las direcciones lógicas a absolutas o físicas por medio de los anteriores 3 métodos mencionados.

Swapping

Swapping es una solución al siguiente problema, debido a que la memoria principal es limitada y por mucho más pequeña que nuestro medio de almacenamiento secundario como un disco duro, el tener muchos procesos cargados en memoria principal puede ser un problema, por lo que swapping permite que dichos procesos que están en espera por el CPU puedan almacenarse temporalmente en el almacenamiento secundario hasta que se necesiten y puedan ser cargados a memoria de nuevo. El almacenamiento secundario mantendrá el mismo tipo de almacenamiento que la memoria principal, osea que los segmentos serán del mismo tamaño y los procesos

serán localizados de forma contigua. Estos procesos se mantienen en una cola de listos, para cual el planificador los llame, entonces el despachador se encarga de cargarlos de nuevo a memoria. Ahora esto es una solución parcial pues el tiempo de transferencia de un almacenamiento secundario a memoria principal es exageradamente alto, y no se compara con la velocidad de la memoria principal al CPU.

Ahora un alternativa al swapping convencional es la que utilizan los dispositivos móviles, esta consiste en que en lugar de mover los procesos al almacenamiento secundario, el sistema pide a los procesos que dejen su espacio en la memoria principal para que los nuevos procesos hagan uso de ellos.

Fragmentación y Localización de Memoria

Como hemos visto, la forma en la que se colocan los procesos en la memoria principal es un factor que influye en el desempeño de estos, ya que lo ideal sería ubicar los procesos que están relacionados de forma contigua para una más fácil y rápida ubicación y traslado de ellos al CPU, pero esto como sabemos puede empezar bien, pero cuando los procesos son terminados, su espacio se queda ahora disponible para que otro proceso lo utilice, la forma en la que el sistema organiza los procesos en estos huecos puede ser de varias formas entre las cuales están el primer hoyo donde quepa el nuevo proceso, el hoyo más pequeño que mejor ajuste al proceso o el hoyo más grande para el proceso.

Bajo los anteriores escenarios se produce fragmentación externa e interna ya que al por ejemplo al tener huecos muy pequeños y procesos entrantes muy grandes, estos huecos se quedan disponibles, dejando así huecos de por medio entre los demás espacios. La fragmentación interna se daría también cuando el proceso es asignado en el primer hueco donde se pueda ajustar o cuando se inserta en el primer hueco grande, ya que si el proceso ocupa muy poco espacio, pues el restante al asignado se queda sin uso alguno. Una solución a esto es la compactación, la cual consiste en que los espacios de memoria utilizados por procesos se mueva o se organice a un segmento junto y contiguo, y el espacio vacío sea un bloque junto. Lógicamente la compactación es algo que no se puede hacer a cada momento que se libera un espacio.

Segmentacion

La segmentación es un esquema de administración de memoria, el cual brinda una solución al problema de; que pasa si el programador tiene que estar al tanto del

direccionamiento físico en su programa. Este esquema permite que el programador tenga una vista lógica de la memoria por medio de segmentos en vez de espacios físicos de memoria, entonces este mismo esquema permite que el usuario piense en una administración de memoria lógica, mientras en realidad los programas se carguen no de forma contigua o literal como la lógica.

Entonces bajo este esquema cada programa contiene mas o menos 5 segmentos, cada segmento puede tener una longitud diferente. Los segmentos son los siguientes:

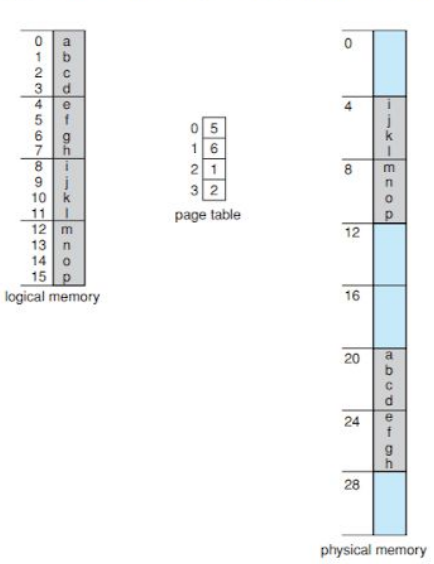
- Segmento de codigo
- Segmento de variables globales
- El heap
- El stack
- Libreria compartidas

Cada segmento desde el punto de vista lógica, su dirección contiene el número de segmento que es y su offset de cuanto es su longitud.
Para llevar control de donde se ubica cada segmento del programa se utiliza una tabla de segmentos, la cual mapea o relaciona por ejemplo el número de segmento de la dirección lógica a la dirección física donde empieza el segmento en la memoria física, así como si el offset es mayor al indicado por el registro de límite en la memoria física para verificar no se trate de accesar memoria fuera del segmento.

Pagineo

Pagineo al igual que segmentacion son esquemas de administración de memoria, ambos pueden colocar procesos de forma no contigua sin ningun problema, pero el pagineo soluciona el problema de la fragmentación externa, como habíamos visto esta fragmentación se refiere a los espacios o huecos que se quedan sin utilizarse cuando un proceso es terminado, La forma en como paginado soluciona esto es al separar la memoria física en pequeñas unidades básicas llamadas frames de una longitud fija y al separar la memoria lógica en unidades llamadas páginas. Entonces cuando un programa es cargado a memoria, según la cantidad de páginas que necesita así también se le asigna una cantidad de frames por cada pagina. Entonces por ejemplo si un programa es terminado y deja la memoria, el espacio vacío son frames que pueden fácilmente ser utilizados para otra pagina para otro proceso, entonces se soluciona la fragmentación externa, ahora con la fragmentación interna si se sufre pues puede que un programa ocupe una cantidad de páginas no exacta o entera entonces el números de frames será otorgado a esa última página que no será totalmente utilizada, por lo que una cantidad de memoria pequeña, una porción del último frame no será utilizada.

Bueno y ahora cómo se hace el mapeo de las páginas que son pedazos de memoria lógica a frames que son espacios de memoria físicos? Aquí al igual que en segmentación con su tabla, el paginado utiliza una tabla de páginas. Ya que las direcciones lógicas ahora están compuestas por el número de página y el offset de esta, entonces la tabla mapea el numero de pagina con el registro base de la página en la memoria física.
La división o la definición del tamaño del frame y de la pagina es decision del diseño del hardware.



La imagen anterior ayuda a visualizar como funciona esta tabla de páginas, por ejemplo si se da la direccion logica de la cuarta página que contiene m,n,o,p la cual tiene un índice de 3, se va buscar con dicho índice a la tabla de páginas y nos indica que esta pagina empieza en el índice 2 de la memoria física.

Ahora el tema de protección en el esquema de pagineo se cumple de la siguiente forma:
En la tabla de pagineo, se le asocia un bit a cada frame para indicar si el frame está en modo de lectura o escritura.

La estructura de la tabla de pagineo, pues en algún momento es un problema ya que mientras más grande es la memoria, pues la tabla es mucho más grande, entonces esto no solo es un problema de espacio sino también entra en juego la eficiencia en la que se puede acceder a un frame ya que recorrer un tabla que tiene asociado la dirección de una pagina a un frame es enorme. Para solucionar esto existen diferentes estructuras que ayudan:

- Tabla de pagina Hash

Universidad Francisco Marroquin
Sistemas Operativos

- Tabla de pagineo Jerárquica
- Tabla de pagineo invertida

Ricardo Mendoza