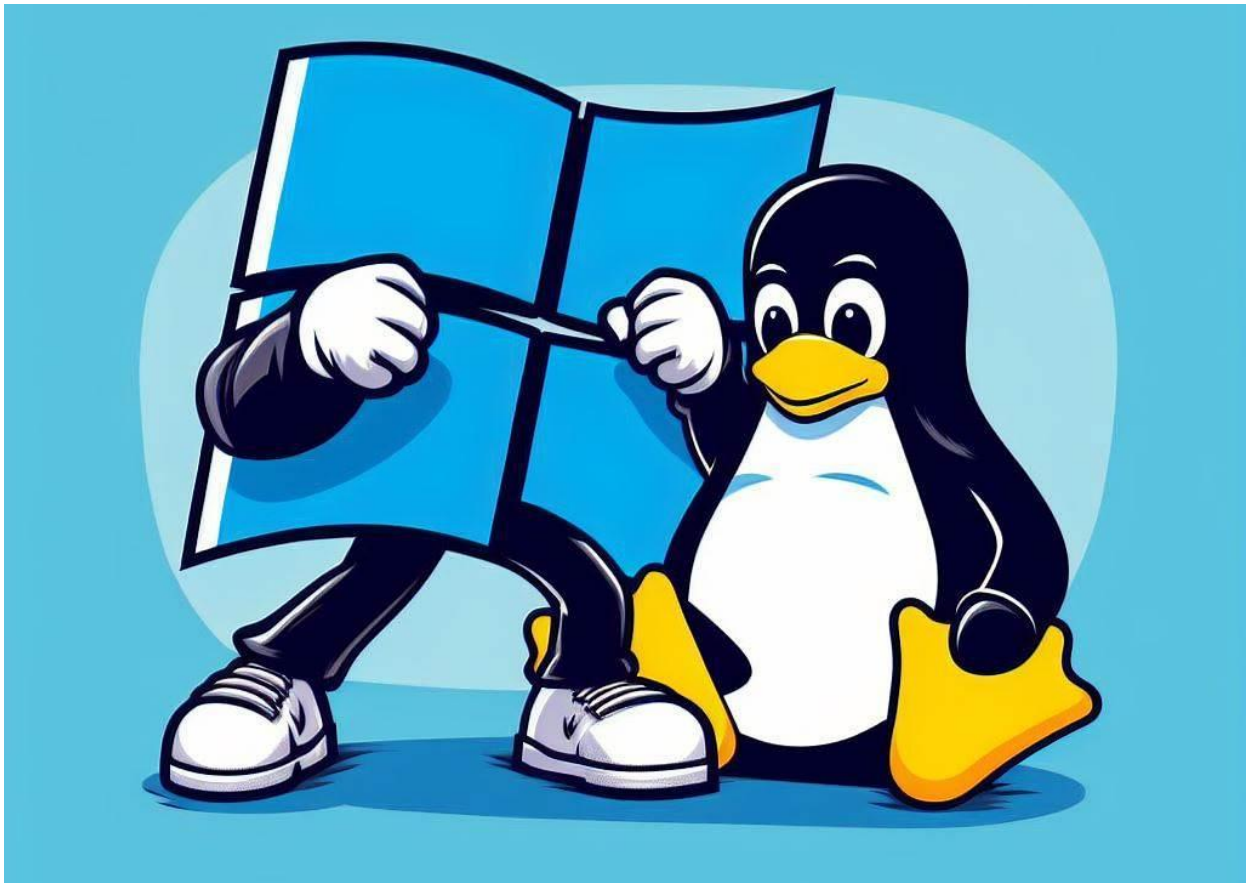
INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

MEMORIA

Resumen teoría



Caporal Nicolás

Facultad de Informática

UNLP

2023

ÍNDICE

Clase 1

MEMORIA

La organización y administración de la memoria principal es uno de los factores más importantes en el diseño de los Sistemas Operativos.

Los programas y datos deben estar en memoria principal para:

- Poderlos ejecutar
- Referenciarlos directamente

TAREAS DEL SISTEMA OPERATIVO

En lo referido a memoria, el Sistema Operativo debe:

- Llevar un registro de las partes de memoria que se están utilizando y de aquellas que no. Y conocer el contenido (Si es código, si son datos, etc).
- Asignar espacio de memoria principal a procesos cuando estos la necesitan.
- Liberar el espacio de memoria asignada a los procesos que han terminado.
- Lograr que el programador se abstraiga de la alocación de programas.
- Brindar seguridad entre los procesos para que no accedan a secciones privadas de otros.
- Brindar la posibilidad de acceso compartido a determinadas secciones de la memoria. (Librerías, código común, etc).

  Por ejemplo, si tengo 2 instancias abiertas de Chrome, en lugar de tener el código el código répetido 2 veces, una para cada proceso, tenerlo una vez para ambos, y así hacer un uso más eficiente del sistema.
- Garantizar la performance del sistema.

Se espera de un Sistema Operativo un uso eficiente de la memoria con el fin de alojar el mayor número de procesos (tener un alto grado de multiprogramación, maximizando el tiempo productivo de CPU).

ADMINISTRACIÓN DE MEMORIA

Se realiza una división lógica de la Memoria Física (real) para alojar multiples procesos, y se debe garantizar la protección (que los procesos no se “pisen” entre ellos).

Esto depende de un mecanismo provisto por el **Hardware**. El diseño de los SO se acomoda para funcionar con la forma de trabajar del HW, y no al revés.

Se debe realizar una asignación eficiente (intentar contener el mayor número de procesos para garantizar el mayor uso de la CPU por los mismos)

REQUISITOS PARA LA ADMINISTRACIÓN DE MEMORIA PRINCIPAL

REUBICACIÓN:

Referido a la independencia del programa respecto de la administración de la memoria física. Se logra gracias al Hardware.

- El programador no debe ocuparse de conocer donde será colocado en la Memoria RAM
- Mientras un proceso se ejecuta, puede ser sacado y traído a la memoria (swap) y, posiblemente, colocarse en diferentes direcciones.
- Las referencias a la memoria se deben “traducir” según ubicación actual del proceso.

PROTECCIÓN:

También se logra gracias al hardware (Registro base y registro límite).

- Los procesos NO deben referenciar (acceder) a direcciones de memoria de otros procesos
  Salvo que tengan permiso
- El chequeo se debe realizar durante la ejecución
  NO es posible anticipar todas las referencias a memoria que un proceso puede realizar, ya que el proceso es dinámico

COMPARTICIÓN:

Ligado al manejo eficiente (no repetir datos) así como a los procesos cooperativos que deseen compartir datos entre sí para lograr un objetivo común.

- Permitir que varios procesos accedan a la misma porción de memoria.
- Permite un mejor uso/aprovechamiento de la memoria RAM, evitando copias innecesarias (repetidas) de instrucciones.

ABSTRACCIÓN - ESPACIO DE DIRECCIONES

El Espacio de Direcciones es el rango de direcciones (a memoria) posibles que un proceso puede utilizar para direccionar sus instrucciones y datos durante su ejecución.

El tamaño del espacio de direcciones depende de la Arquitectura del Procesador

- Con 32 bits: 0 .. 2^{32} - 1
- Con 64 bits: 0 .. 2^{64} – 1

Todo esto pensado desde el lado lógico.

Es independiente de la ubicación “real” del proceso en la Memoria RAM
(La dirección 100 del espacio de direcciones del proceso 1, no necesariamente va a la dirección física 100)

DIRECCIONES (LÓGICAS VS FÍSICA)

DIRECCIONES LÓGICAS:

Los procesos trabajan y generan direcciones lógicas.

- Referencia a una localidad de memoria independiente de la asignación actual de los datos en la memoria real (dirección física).
- Representa una dirección en el “Espacio de Direcciones del Proceso”

En caso de usar direcciones Lógicas, es necesaria algún tipo de conversión a direcciones Físicas.

DIRECCIONES FÍSICAS

- Referencia una localidad en la Memoria Física (RAM) (Dirección absoluta)

CONVERSIÓN DE DIRECCIONES

Cada dirección lógica que se genera, debe ser convertida a la dirección física.

Una forma simple de realizarlo, es considerar que el espacio de direcciones (ED) de un proceso está en memoria RAM de forma contigua.

Para marcarlo se utilizan 2 registros auxiliares de la CPU, el Registro Base (dirección de comienzo del ED) y el Registro Límite (dirección final del ED).
Entonces, la dirección física se calcula haciendo Registro Base + un desplazamiento. Y se revisa que no este fuera del registro limite.

Ambos valores se fijan cuando el espacio de direcciones del proceso es cargado en memoria, y el kernel se lo indica a la CPU en sus 2 registros (siendo la CPU quien se encarga de controlar en tiempo de ejecución que el proceso no se salga de los límites).

El contenido de estos registros varía entre procesos, por lo que es otra cosa que el SO debe cambiar ante un Context Switch.

DIR. LÓGICAS VS FÍSICAS

La CPU trabaja con direcciones **lógicas** (recordemos que en los registros base y límite se carga según el espacio de direcciones, y el espacio de direcciones es una abstracción).
Por tanto, para acceder a memoria principal, las debe transformar en direcciones físicas.

Esa resolución se realiza normalmente en tiempo de ejecución:

- Direcciones físicas y Lógicas son diferentes
- El mapeo entre Virtuales y Físicas es realizado por Hardware (el MMU de la CPU)

MEMORY MANAGEMENT UNIT (MMU)

La MMU es un dispositivo de hardware (ligado a la CPU, o incluso dentro de la CPU) que se encarga de mapear (traducir) direcciones virtuales a físicas.

Re-programar el MMU es una operación privilegiada que solo se puede realizar en modo Kernel.

Cada vez que hay un Context Switch, recibe el Registro Base y el Registro Límite para hacer el cálculo. El valor en el “registro de realocación” (contiene el mismo valor del Registro Base) es sumado a cada dirección lógica generada por el proceso de usuario al momento de acceder a la memoria.

MECANISMOS DE ASIGNACIÓN DE MEMORIA (PARTICIONES)

Este modelo planteado donde un proceso se almacena de manera contigua a partir de una dirección y hasta un límite, se denomina Particiones.

Cuando se utilizaba este modelo (ya no se utiliza) existían dos formas de particionar la memoria:

PARTICIONES FIJAS:

De antemano, el SO realiza una división lógica de la memoria RAM. Y cada espacio de direcciones (cada proceso) entra justo en una partición,

- La memoria se divide en particiones de tamaño fijo (pueden ser todas del mismo tamaño o variar)
- Alojan a un proceso cada una
- Cada proceso se coloca de acuerdo a un criterio en alguna partición.
- Genera fragmentación interna (espacio desperdiciado).

PARTICIONES DINÁMICAS:

No existe una división previa de la RAM. A medida que los procesos van apareciendo, la memoria se va particionando de acorde al tamaño de los espacio de direcciones que van llegando

- Las particiones varían en tamaño y en número
- Alojan un proceso cada una
- Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso
- Genera fragmentación externa.

FRAGMENTACIÓN

La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua.

FRAGMENTACIÓN INTERNA:

- Se produce en el esquema de particiones Fijas y en Paginación
- Es la porción de la partición que queda sin utilizar
- No tiene solución

FRAGMENTACIÓN EXTERNA:

- Se produce en el esquema de Particiones Dinámicas y en la tecnica de Segmentación
- Son huecos que van quedando en la memoria a medida que los procesos finalizan
- Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero que no la podamos utilizar
- Para solucionar el problema se puede acudir a la compactación, pero es muy costosa en cuanto a tiempo de Memoria Secundaria (HDD)

PROBLEMAS DEL ESQUEMA REGISTRO BASE + LÍMITE

El esquema de Registro Base + Limite se utilizaba ya que los SO operativos estaban limitados por el HW. Recordemos que el diseño de los SO se acomodó para funcionar con la forma de trabajar del HW, y no al revés. Este esquema presentaba problemas:

- Necesidad de almacenar el Espacio de Direcciones de forma continua en la Memoria Física
- Los primeros SO definían particiones fijas de memoria, luego evolucionaron a particiones dinámicas
- Genera fragmentación
- Mantiene “partes” del proceso que no son necesarias
- Los esquemas de particiones fijas y dinámicas no se usan hoy en día

Solución: Ante la evolución del hardware, surgieron 2 técnicas de administración

- Paginación
- Segmentación

PAGINACIÓN

La Memoria Física es dividida lógicamente en pequeños trozos de igual tamaño llamados “Marcos”

La Memoria Lógica (espacio de direcciones) es dividido en trozos de igual tamaño que los marcos, llamadas “Páginas”

En esta técnica, cualquier página del espacio de direcciones, puede ir a cualquier marco.
De esta manera se rompe la continuidad, ya no es necesario que el espacio de direcciones vaya de forma continua.

Puede generar fragmentación interna solo en la última página.

El SO debe mantener una **tabla de páginas** por cada proceso, donde cada entrada contiene el Marco a dónde fue a parar esa página.

Ahora el MMU utiliza la tabla de páginas (en vez del RB y RL) para realizar la conversión.
La dirección física se calcula como: un número de página + un desplazamiento dentro de la misma.

SEGMENTACIÓN

Esquema que se asemeja a la “visión del usuario”.

El programa se divide en partes/secciones. Un programa es una colección de segmentos.

Un segmento es una unidad lógica como: Programa Principal, Procedimientos y Funciones, variables locales y globales, stack, etc.
Y cada uno de estos segmentos se pueden cargar de manera no continua al espacio de direcciones.

Todos los segmentos de un programa pueden no tener el mismo tamaño (código, datos, rutinas). Por lo tanto, la Segmentación puede causar **Fragmentación Externa**. Al los segmentos de programa de distintos tamaños, quedan espacios pequeños en el medio sin utilizar.

Las direcciones Lógicas consisten en 2 partes:

> Selector de Segmento + Desplazamiento dentro del segmento

El SO debe mantener una **Tabla de Segmentos** que permite mapear la dirección lógica en física.

Cada entrada contiene:

- Base: Dirección física de comienzo del segmento
- Limit: Longitud del Segmento
- Segment-table base register (STBR): apunta a la ubicación de la tabla de segmentos.
- Segment-table length register (STLR) : cantidad de segmentos de un programa

SEGMENTACIÓN VS PAGINACIÓN

Se utiliza más la paginación por sobre la Segmentación, ya que la fragmentación es menor. Y la Externa es costosa de resolver.

Sin embargo, la ventaja de Segmentación es que es más efectiva para Compartir y Proteger que la Paginación. Si tengo dos instancias de Chrome, el código es el mismo, por tanto el **segmento** que contiene el código se puede compartir facilmente, con una sola entrada en la tabla de segmentos (colocando la misma dirección base).

SEGMENTACIÓN PAGINADA

Es la combinación de las dos anteriores.

Cada segmento es dividido en páginas de tamaño fijo.

El cálculo de la memoria física, se obtiene con Segmentación, obteniendo las ventajas ya mencionadas en cuanto a compartir y proteger.

Para subir los datos a memoria principal, se utiliza Paginación. Se suben **páginas** a la RAM, eliminando la fragmentación Externa y sus problemas.

Para cada **segmento** hay una **tabla de página** que contiene a qué dirección de la memoria física fue el segmento.

La paginación

- Transparente al programador
- Elimina Fragmentación externa.

Segmentación

- Es visible al programador
- Facilita modularidad, estructuras de datos grandes y da mejor soporte a la compartición y protección