

# Introducción a los Sistemas Operativos

## Administración de Memoria - I



Facultad de Informática  
UNIVERSIDAD NACIONAL DE LA PLATA

## I.S.O.

- ✓ Versión: Agosto 2013
- ✓ Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Particiones, Fragmentación

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



Facultad de Informática  
UNIVERSIDAD NACIONAL DE LA PLATA

## Administración de Memoria

- ✓ División Lógica de la Memoria para alojar múltiples procesos
- ✓ La Memoria debe ser asignada eficientemente para contener el mayor número de procesos como sea posible.
- ✓ Cuanto más procesos estén en memoria, más procesos competirán por la CPU. Mas probabilidad que la CPU no este ociosa.



## Requisitos

- ✓ Reubicación
  - ✓ El programador no debe ocuparse de conocer donde será colocado el programa para ser ejecutado.
  - ✓ Mientras un proceso se ejecuta, puede ser sacado y traído a la memoria (swap) y colocarse en diferentes lugares.
  - ✓ Las referencias a la memoria se deben traducir según dirección actual del proceso.



## Requisitos (cont).

### ☑ Protección

- ✓ Los procesos no deben ser capaces de hacer referencias a direcciones de memoria de otros procesos (salvo que tengan permiso)
- ✓ El chequeo se debe realizar durante la ejecución:
  - ♦ El SO no puede anticipar todas las referencias a memoria que un proceso puede realizar.



## Requisitos (cont).

### ☑ Compartición

- ✓ Permitir que varios procesos accedan a la misma porción de memoria.
  - ♦ Ej: Rutinas comunes, librerías, espacios explícitamente compartidos, etc.
- ✓ Lleva a un mejor uso de la memoria, evitando copias innecesarias de instrucciones

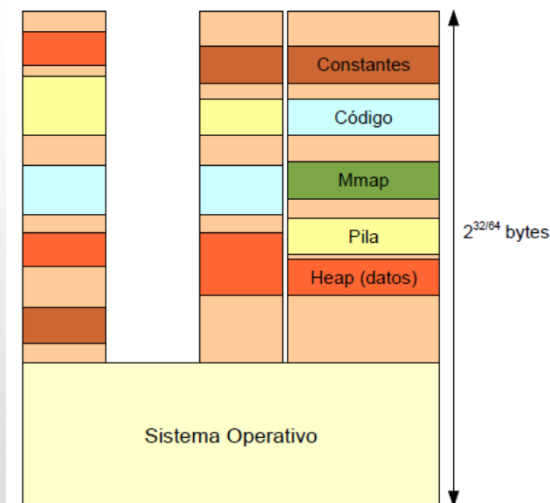


## Abstracción - Espacio de Direcciones

- ✓ Rango de direcciones (a memoria) posibles que un proceso puede utilizar para direccionar sus instrucciones y datos.
- ✓ El tamaño varía dependiendo de la arquitectura
  - ✓ 32 bits:  $0 \dots 2^{32} - 1$
  - ✓ 64 bits:  $0 \dots 2^{64} - 1$
- ✓ Debe ser independiente de la ubicación “real” del proceso en la memoria



## Abstracción -Espacio de Direcciones (cont.)



## Direcciones

### ☒ Lógicas

- ✓ 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.

### ☒ Físicas

- ✓ La dirección absoluta en la memoria principal.

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



## Conversión de Direcciones

Una forma simple de hacer esto es utilizando registros auxiliares

### ☒ Registro Base

- ✓ Dirección de comienzo del proceso.

### ☒ Registro Limite

- ✓ Dirección final del proceso o medida del proceso.

### ☒ Su valor se fija cuando el proceso es cargado a memoria (Context Switch)

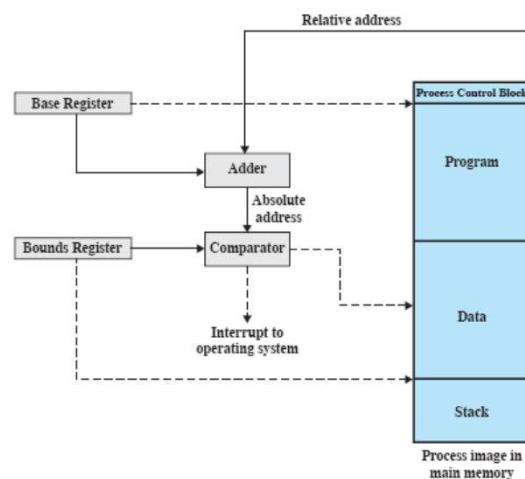


## *Direcciones - Registros utilizados (cont.)*

- ✓ 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 limite (según la técnica utilizada).
- ✓ Si la dirección generada es incorrecta, se genera una interrupción al SO.

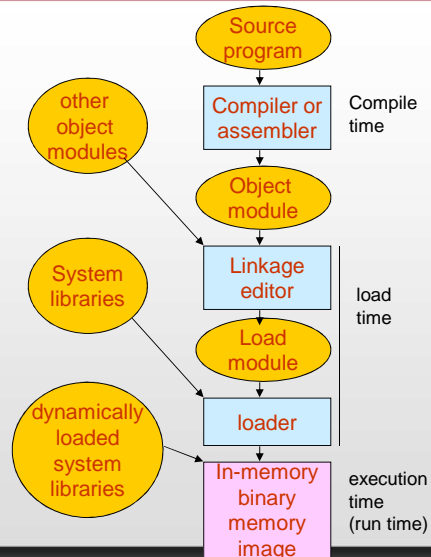


## *Direcciones (cont.)*



## Binding de direcciones

- ✓ Las direcciones en los programas fuentes son simbólicas
- ✓ Compiladores relacionan direcciones simbólicas a direcciones reubicables
- ✓ El linkeditor/cargador relaciona las direcciones reubicables en direcciones absolutas



## Dir. Lógicas vs. Físicas

- ✓ Si la CPU trabaja con direcciones lógicas, para acceder a memoria principal, se deben transformar en direcciones absolutas.
  - Resolución de direcciones (address-binding): transformar la dirección lógica en la dirección física correspondiente
- ✓ Resolución en momento de compilación (Archivos .com de DOS) y en tiempo de carga
  - ✓ Direcciones Lógicas y Físicas son idénticas
  - ✓ Para reubicar un proceso es necesario recompilarlo o recargarlo.



## *Dir. Lógicas vs. Físicas*

- ☑ Resolución en tiempo de ejecución
  - ✓ Direcciones Lógicas y Físicas son diferentes
  - ✓ Direcciones Lógicas son llamadas “Direcciones Virtuales”
  - ✓ La reubicación se puede realizar facilmente
  - ✓ El mapeo entre “Virtuales” y “Físicas” es realizado por hardware: Memory Management Unit (MMU)



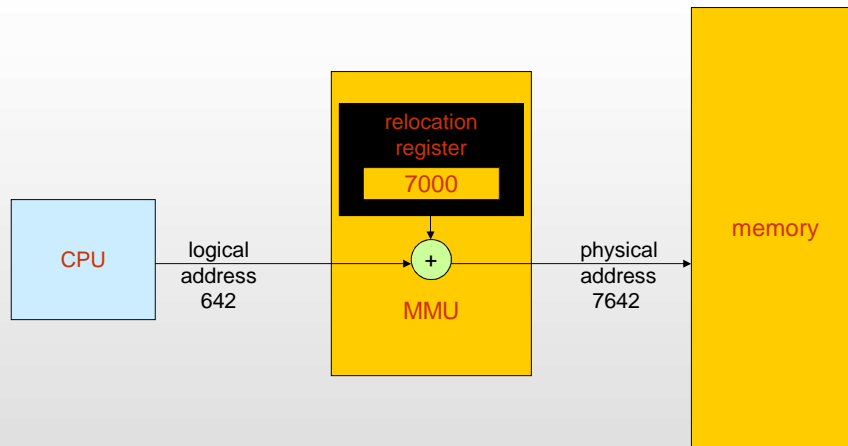
## *Memory Management Unit (MMU)*

- ☑ Dispositivo de Hardware que mapea direcciones virtuales a físicas
  - ✓ Es parte del Procesador
  - ✓ Re-programar el MMU es una operación privilegiada que solo puede ser realizada en Kernel Mode
- ☑ 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.
  - ✓ Los procesos nunca ver direcciones físicas





# MMU



## Carga Dinámica

- ☒ Mejora la utilización de la memoria.
- ☒ Una rutina no es cargada hasta que no es llamada.  
Rutinas no utilizadas nunca son cargadas
- ☒ Todas las rutinas se mantienen en el disco en un formato "reubicable"



## Carga Dinámica (cont)

- ☑ Cuando una rutina invoca a otra:
  - ✓ Se chequea que ya no haya sido cargada
  - ✓ Si no se cargo, se llama al “cargador” para que la cargue
  - ✓ El “cargador” actualiza el espacio de direcciones del proceso
  - ✓ El control es pasado a la nueva rutina



## Linkeo Dinámico (Dynamic linking)

- ☑ Similar a la carga dinámica
  - ✓ El linkeo es postergado hasta el momento de “run-time”
  - ✓ Las librerías dinámicas no son “adjuntadas estáticamente” al proceso, solo un pequeño “stub” es adjuntado.
  - ✓ El “stub” indica como cargar la rutina apropiada
- ☑ Mejora la utilización de la memoria al no tener código de librerías replicado en cada proceso.
- ☑ Mejora la utilización del disco, al no tener binarios con librerías estáticas.



## Linkeo Dinámico (Dynamic linking)

- ✓ Todos los programas pueden utilizar la misma copia de una librería (código) (DLLs)
- ✓ El Linkeo Dinámico requiere soporte del S.O.
  - ✓ S.O. es el único que puede localizar una librería en el espacio de direcciones de otro proceso



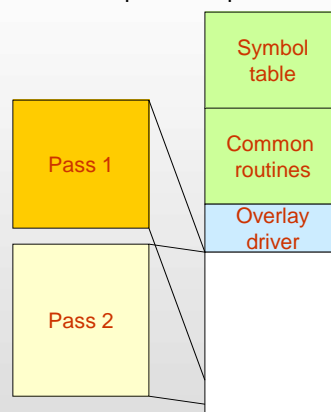
## Superposiciones (overlapping)

- ✓ El tamaño de los procesos y sus datos puede exceder el tamaño de la memoria

Conceptos:

- ✓ Separar el programa en módulos
- ✓ Cargar los módulos alternativamente
- ✓ El dispositivo de “superposiciones” localiza los módulos en el disco
- ✓ Los módulos superpuestos se guardan como imágenes absolutas de memoria
- ✓ Se requiere soporte de los compiladores y expertice del programador.
- ✓ No se requiere soporte del SO

Example:  
multi-pass compiler



# Swapping

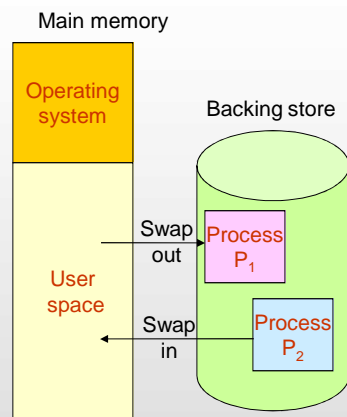
- ☑ Un proceso puede ser temporalmente sacado de la memoria (*swapped out*) a un disco de manera de permitir la ejecución de procesos.

Si se descarga considerando las direcciones físicas

- ☑ Al hacer *swapped in* se debe cargar en el mismo espacio de memoria que ocupaba antes

Si se descarga considerando las direcciones lógicas

- ☑ Al hacer *swapped in* se puede cargar en cualquier espacio de direcciones de memoria



# Asignación de Memoria

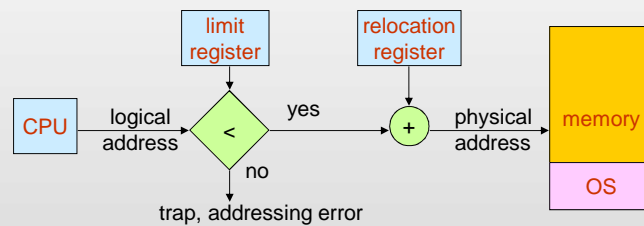
- ☑ La memoria principal debe ser organizada para contener el S.O. (rutinas, librerías, estructuras) y procesos
  - ✓ La memoria del S.O. debe protegerse del acceso a memoria de los procesos.
  - ✓ La memoria de un proceso debe protegerse del acceso a memoria de otros procesos.
  - ✓ El esquema utilizado es dependiente del diseño del hardware



## Esquemas de asignación de Memoria

### ☑ Única Partición:

- ✓ Los procesos ocupan una única partición de memoria
- ✓ La protección se implementa por un "límite" y un registro de "reubicación"



## Múltiples Particiones

- ☑ La memoria es dividida en varias regiones (particiones).
- ☑ Los procesos (su espacio de direcciones) se colocan en las particiones según su tamaño.
- ☑ Técnicas:
  - ✓ Particiones Fijas
  - ✓ Particiones Dinámicas



## Particiones Fijas Opción 1

☑ Regiones definidas con límites fijados

☑ Particiones de IGUAL tamaño:

- ♦ Cualquier proceso cuyo tamaño es menor igual que la partición puede ser colocado en una partición libre.
- ♦ Si todas están ocupadas → Swap
- ♦ Ineficiencia: Programa mas grandes que el tamaño de las particiones no podrán ejecutarse, por mas que la memoria total sea mas grande que el programa.
- ♦ Ineficiencia: Programa pequeños, ocuparan una partición desperdiciando mucho espacio



## Particiones Fijas Opción 2

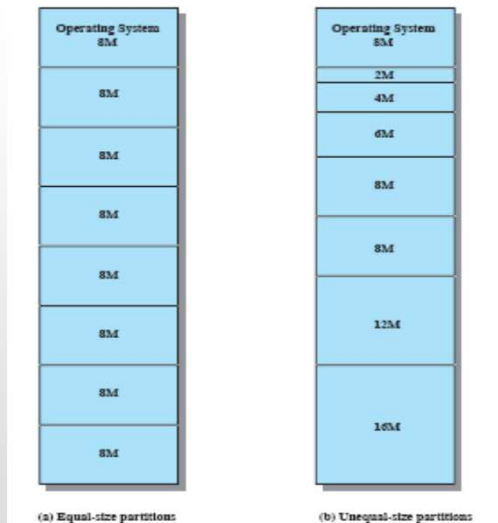
☑ Regiones definidas con límites fijados

☑ Particiones de DIFERENTE tamaño:

- ♦ Evita el problema de las particiones de igual tamaño:
  - Procesos pequeños pueden usar particiones pequeñas
  - Se pueden prever algunas particiones grandes para procesos grandes.
- ♦ Complejidad en el algoritmo de selección de partición para un proceso.



## Ejemplo de Particiones Fijas



## Múltiples Particiones

- ✓ La memoria es dividida en varias regiones (particiones).
- ✓ Los procesos (su espacio de direcciones) se colocan en las particiones según su tamaño.
- ✓ Técnicas:
  - ✓ Particiones Fijas
  - ✓ **Particiones Dinámicas**

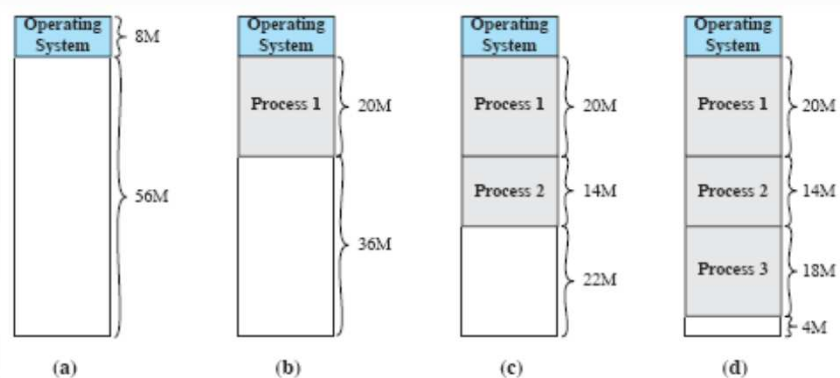


## Particiones Dinámicas

- ✓ Las particiones son de tamaño y número variable
- ✓ Los procesos son colocados exactamente en particiones de igual a su tamaño (generadas dinámicamente)
- ✓ Es necesario administrar la memoria para saber en cualquier momento, cuales porciones están siendo usadas y cuales libres.

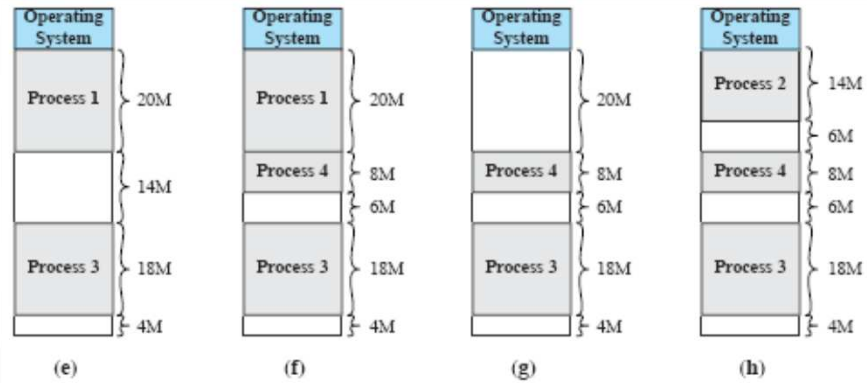


## Particiones Dinámicas (cont.)



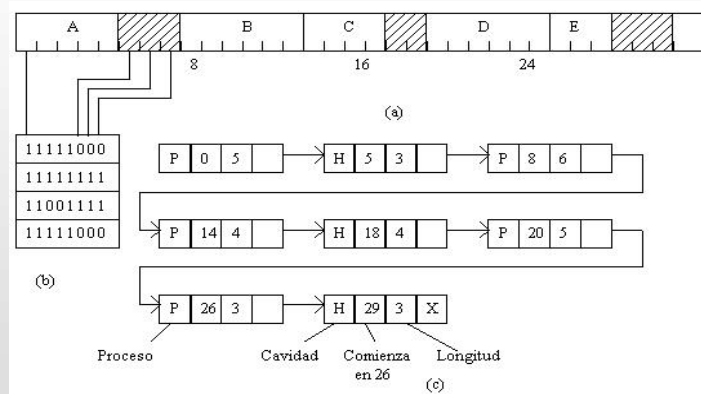


## Particiones Dinámicas (cont.)



## Particiones dinámicas - Manejo de los "huecos"

Dos formas de llevar el registro del uso de la memoria: mapas de bits y listas.



## Manejo de “huecos” con mapas de bits

- La memoria se divide en unidades de asignación de tamaño variable.
- Cuanto más pequeñas las unidades más grande será el mapa.
- Cada unidad de asignación hay un bit correspondiente en el mapa de bits, que es 0 si la unidad está libre y 1 si está ocupada (o viceversa)
- Malo para realizar búsquedas de bloques consecutivos



## Manejo de “huecos” con listas

- Lista ligada de segmentos de memoria asignados y libres: un segmento contiene un proceso o es un hueco vacío entre dos procesos.
- Cada entrada en la lista especifica un hueco (H) o un proceso (P), la dirección en la que inicia, la longitud y un apuntador a la siguiente entrada.
- Actualizar la lista es simple
- Se pueden utilizar múltiples algoritmos para realizar búsquedas y asignar memoria



## Fragmentación

- ☑ Espacio libre de la memoria que no puede ser utilizado
- ☑ Interna → Particiones Fijas
  - ✓ Espacio dentro de la partición sin utilizar
- ☑ Externa → Particiones Dinámicas
  - ✓ Cada vez que entra y sale un proceso se genera huecos en la memoria, en los que eventualmente un proceso no podría entrar, pero si entraría si unimos todos los huecos → COMPACTACION



## Particiones - Algoritmos de Ubicación

- ☑ Particiones Fijas de igual tamaño
  - ✓ No se necesita un algoritmo
- ☑ Particiones Fijas de diferente tamaño y Particiones Dinámicas:
  - ✓ First Fit
  - ✓ Best Fit
  - ✓ Worst Fit
  - ✓ Next Fit



## Particiones - Algoritmos de Ubicación (cont.)

### ☑ Best fit

- ✓ Selecciona la partición mas pequeña que contiene al proceso
- ✓ Mucho overhead en la búsqueda
- ✓ En particiones dinámicas: se generan muchos huecos pequeños de memoria libre → Fragmentación externa

### ☑ First Fit

- ✓ Recorre las particiones libres en orden, buscando la primera que pueda alojar al proceso.



## Particiones - Algoritmos de Ubicación (cont.)

### ☑ Next fit

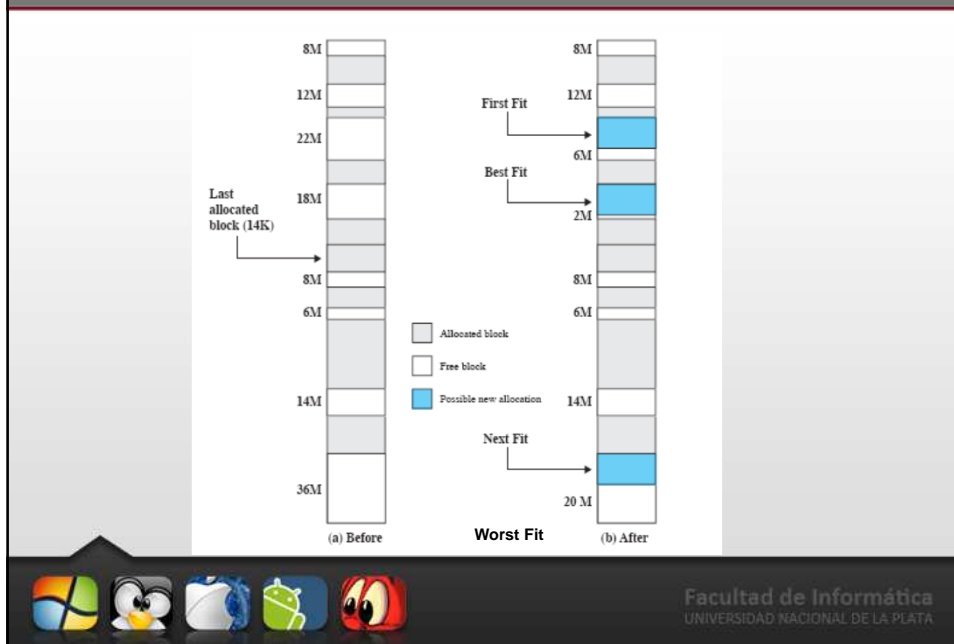
- ✓ Mantiene las particiones libres como una lista circular.
- ✓ Funciona como el First Fit solo que empieza desde la posición actual en la lista, no el principio.
- ✓ Selecciona la primer partición que encuentra que contenga el proceso

### ☑ Worst Fit

- ✓ Selecciona la partición libre mas grande que contenga el proceso.
- ✓ Mal uso en Particiones Fijas
- ✓ Buen uso en Particiones Dinámicas.



## Ejemplo de Algoritmos de Ubicación



## Repaso

¿Cuál es el grado de multiprogramación en los distintos esquemas de manejo de memoria?

- Única Partición
- Particiones Fijas de igual tamaño
- Particiones Fijas de distinto tamaño
- Particiones Dinámicas

## *Repaso (cont)*

¿Cuál es el tamaño máximo que podría tener un proceso en cada uno de los esquemas de manejo de memoria?

- Única Partición
- Particiones Fijas de igual tamaño
- Particiones Fijas de distinto tamaño
- Particiones Dinámicas

