Introducción a los Sistemas Operativos
Resumen para el primer parcial teórico

I. Introducción a los sistemas informáticos.

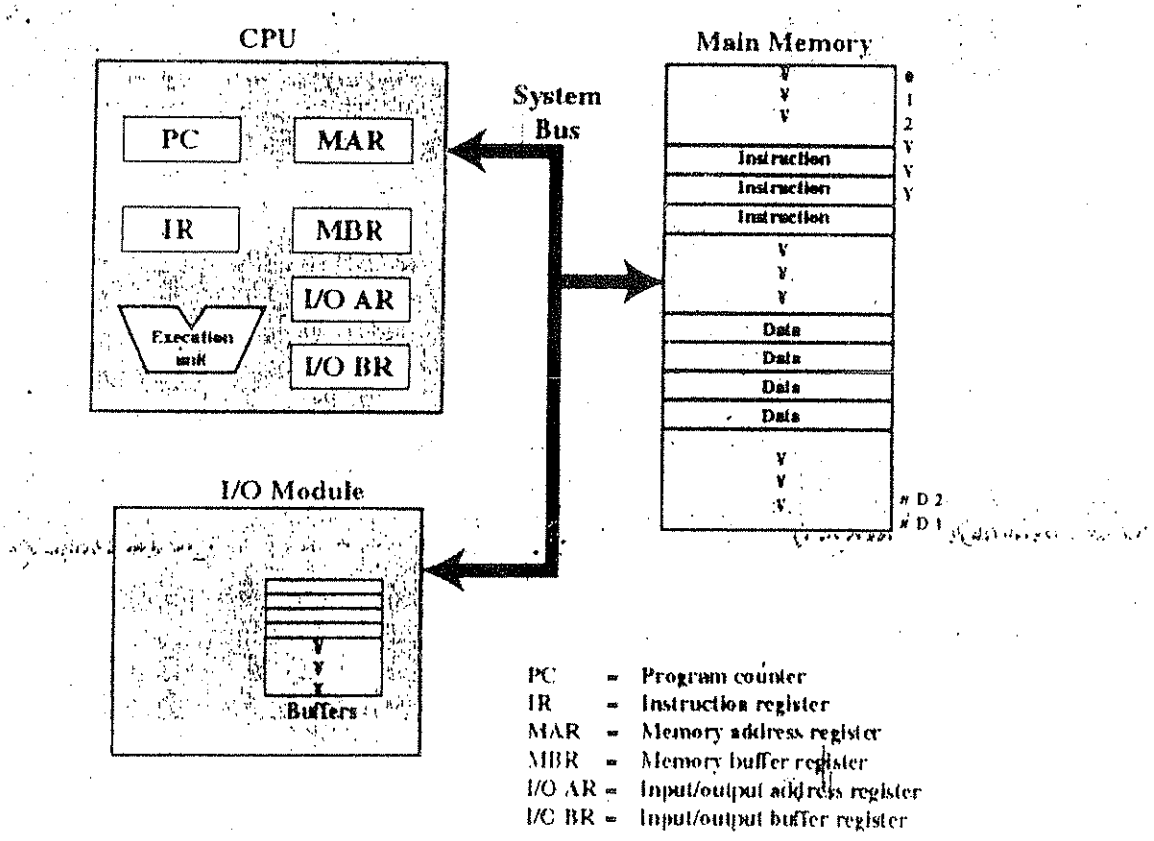
Antes de que podamos abordar el estudio de los sistemas operativos (SO) se necesita comprender el hardware del sistema informático ya que el SO se va a encargar de explotarlo para ofrecerles servicios a los usuarios, como así también, va a gestionar la memoria secundaria y dispositivos de E/S en nombre de los usuarios.

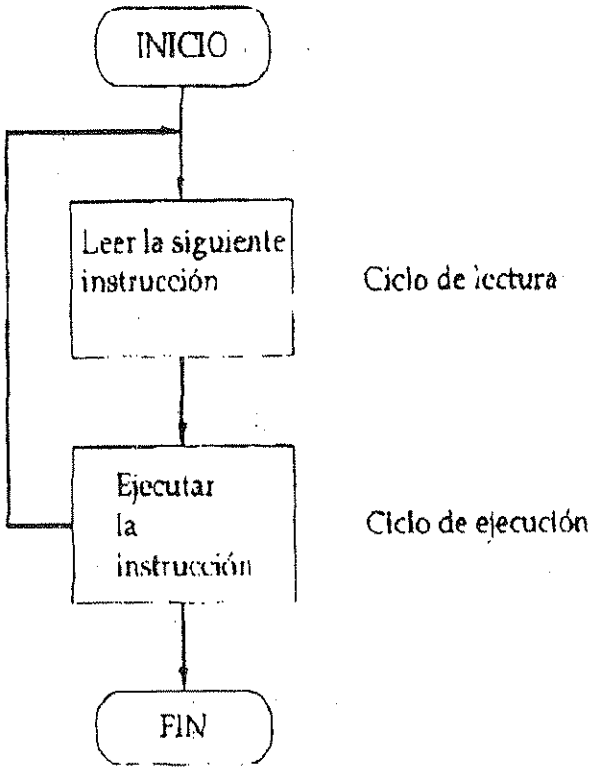
Elementos básicos de una computadora:

- En un alto nivel:
  - Procesador: controla la operación del computador (la computadora) y procesa los datos. Si hay un solo procesador se lo suele llamar CPU (Unidad Central de Procesamiento).
  - Memoria Principal: almacena datos y programas. Normalmente volátil (cambia su contenido).
  - Componentes de E/S.
  - Interconexión de sistemas: mecanismos y estructuras que permiten la comunicación entre procesadores, memoria principal y los módulos de E/S. Los tres componentes anteriores están interconectados para llevar a cabo la función principal del computador: ejecutar programas.

(Interconexión: conexión recíproca. Dan y reciben. Relacionado con el concepto de comunicación. Se establece una comunicación entre procesador, memoria y componentes de E/S)

Dibujo de los componentes de alto nivel:





(Ciclo de lectura = fetch)
La ejecución del programa se detiene si se apaga la máquina, por algún error irreparable o se encuentra una instrucción en el programa que detiene el computador.
El PC se incrementa después de cada fetch.
La instrucción buscada (Fetched instruction) se pone en el IR .
El PC me indica la próxima dirección de la instrucción a ser leída.
El IR contiene la instrucción leída. Esta instrucción está en binario. El procesador la interpreta y realiza la acción. Estas acciones pueden ser cuatro:

1. Procesador-Memoria: transferencia de datos de procesador a memoria o viceversa.
2. Procesador-E/S: transferencia de datos desde o hacia un dispositivo periférico, se realiza entre el procesador y un módulo de E/S.
3. Tratamiento de datos: el procesador realiza alguna operación aritmética o lógica sobre los datos.
4. Control: la instrucción pide que se altere la secuencia de ejecución. La instrucción modifica el PC para indicarle la próxima instrucción a leer. Si la instrucción que leí estaba en 149 y la instrucción dice que la próxima a leer es la 200 el PC que está en 150 cambia a 200.

Funciones de E/S: los módulos de E/S pueden intercambiar datos directamente con el procesador. Este puede leer o escribir datos en el módulo. El módulo puede escribir o leer en la memoria.
Acceso directo a memoria (DMA): a veces es conveniente que los intercambios de E/S se produzcan directamente con la memoria. El procesador autoriza al módulo de E/S para que escriba o lea en memoria, el procesador queda en segundo plano ya que se le libra de responsabilidad en el intercambio. La cosa es entre el módulo de E/S y la memoria, el procesador autoriza que se realice de manera directa el intercambio de información entre la memoria y el módulo de E/S y luego queda en segundo plano.

usuario final es el que usa la computadora. Simplemente se sienta y empieza a usar las aplicaciones.
Si los programadores tuvieran que desarrollar programas en el lenguaje de máquina sería una tarea abrumadora y compleja. Por eso, para facilitar esta tarea, se ofrecen programas de sistemas que se denominan utilidades que sirven para la creación de programas.
El programa de sistema más importante es el SO. Este le oculta al programador los detalles del hardware y le proporciona una interfaz cómoda para utilizar el sistema. Resumiendo el SO ofrece servicios en las áreas siguientes:

- Creación de programas.
- Ejecución de programas.
- Acceso a los dispositivos de E/S.
- Acceso controlado a archivos.
- Acceso al sistema.
- Detección y respuesta de errores: errores de hardware internos y externos y errores de software.
- Contabilidad: debe reunir estadísticas que servirán para mejorar el SO en un futuro.

Tareas de un SO:

- Administración y planificación del procesador
  - Imparcialidad, "justicia" en la ejecución (Fairness).
  - Que no haya bloqueos.
  - Manejo de Prioridades.
- Administración de Memoria
  - Memoria física versus memoria virtual. Jerarquías de memoria.
  - Protección de programas que compiten o se ejecutan concurrentemente.
- Administración del almacenamiento– Sistema de archivos
  - Acceso a medios de almacenamiento externos.
- Administración de dispositivos
  - Ocultamiento de dependencias de hardware.
  - Administración de accesos concurrentes.
- Batch processing
  - Definición de un orden de ejecución; maximización del rendimiento (throughput).

El SO funciona de la misma manera que el software normal de una computadora, es decir, es un programa ejecutado por el procesador. También, el SO abandona con frecuencia el control y debe depender del procesador para recuperarlo. El SO es simplemente un programa que da instrucciones al procesador como todos los demás programas. La diferencia radica en su propósito. El SO dirige al procesador en el empleo de otros recursos del sistema y en el control del tiempo de ejecución de otros programas. Pero para que el procesador puede hacer todo esto el SO le cede el control para que haga un trabajo "útil" y el SO cesa la ejecución hasta que el procesador termine de hacer el trabajo. Luego el SO retorna el control durante el tiempo suficiente para preparar el procesador para hacer otro trabajo.
Kernel: porción del SO que se encuentra en la memoria principal, contiene las funciones más frecuentemente usadas.

- Kernel monolítico: un SO con kernel monolítico concentra todas las funcionalidades posibles dentro de un gran programa.
- Microkernel: las partes funcionales están divididas en unidades, separadas con mecanismos de comunicación estrictos entre ellos.

nfiabilidad. Los sistemas paralelos son aquellos que tienen la capacidad de realizar varias raciones de manera simultánea.

emas operativos distribuidos: presenta el mismo aspecto a los usuarios que un sistema tradicional de un solo procesador, aunque en realidad se compone de múltiples procesadores. Los usuarios no deben enterarse de en dónde se están ejecutando sus programas o almacenando sus archivos; de todo eso se debe encargar el SO automática y eficientemente. El trabajo es distribuido a los largo de varios procesadores, cada uno de estos cuenta con su propia memoria local.

Tiempo real: un sistema en tiempo real (STR) debe aceptar y procesar en tiempos muy breves un gran número de sucesos. Si el sistema no respeta las restricciones de tiempo en las que las operaciones deben entregar su resultado se dice que ha fallado. El tiempo de respuesta debe servir para resolver el problema o hecho planteado. El procesamiento de archivos se hace de una forma continua, pues se procesa el archivo antes de que entre el siguiente.

Sistema portable: ejemplo de estos son los PADs, teléfonos celulares. Tienen memoria limitada, procesadores lentos y pantallas pequeñas.

Logros importantes: hay cuatro logros importantes en el desarrollo de los SO:

1. Los procesos.
2. El manejo de la memoria (administración o gestión de memoria).
3. La seguridad y la protección de la información.
4. La planificación y gestión de recursos.
5. Las estructuras del sistema.

Servicios del SO: un SO proporciona un entrono para la ejecución de programas. El sistema presta ciertos servicios a los programas y a los usuarios de dichos programas. Estos servicios se proporcionan para la comodidad del programador, con el fin de facilitar la tarea de desarrollo.

Conjunto de servicios del SO que proporcionan funciones útiles para el usuario:

- Interfaz de usuario (puede ser por línea de comandos o gráfica).
- Ejecutar programas.
- Operaciones de E/S.
- Manipulación del sistema de archivos: leer y escribir en archivos y directorios. Crearlos, borrarlos usando su nombre, buscarlos.
- Comunicaciones: un proceso necesita intercambiar información con otro.
- Detección de errores

Otras funciones que no están pensadas para ayudar al usuario, sino para garantizar la eficiencia del propio sistema son:

- Asignación de recursos.
- Responsabilidad: conviene hacer un seguimiento de qué usuario emplea qué clases de recursos de la computadora y en qué cantidad.
- Protección y seguridad.

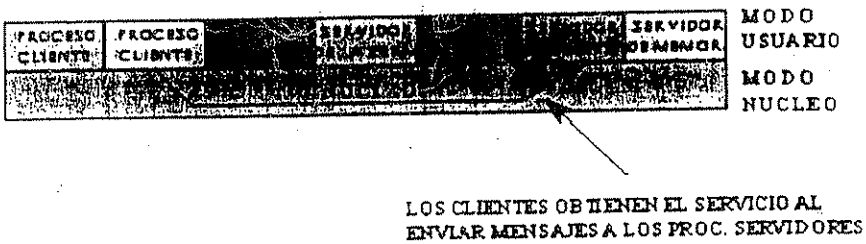
La interfaz de usuario: el usuario interactúa con el sistema mediante dos métodos fundamentales: línea de comandos o intérprete de comandos, que permite a los usuarios introducir directamente comandos que el SO pueda ejecutar. El otro método permite que el usuario interactúe con el SO a través de una interfaz gráfica (GUI).

Intérprete de comandos: cuando los sistemas tienen varios intérpretes, se los conoce como **shells**. La función principal del intérprete es obtener y ejecutar el siguiente comando especificado por el usuario.

Operaciones en modo dual: para asegurar la correcta ejecución del SO, tenemos que poder distinguir entre la ejecución del código del SO y del código definido por el usuario. El método que usan la mayoría de los sistemas informáticos consiste en proporcionar soporte hardware que nos permita diferenciar entre varios modos de ejecución.

Como mínimo, necesitamos dos modos diferentes de operación: modo usuario y modo kernel (núcleo, supervisor, del sistema o privilegiado).

vuelve la respuesta.



Lo que hace el kernel es manejar la comunicación entre los clientes y los servidores. Al dividir el SO en partes, cada una de las cuales sólo se encarga de una faceta del sistema, como el servidor de archivos, de procesos, etc. Cada parte puede ser pequeña y manejable. Además, dado que todos los servidores se ejecutan como procesos en modo usuario, y no en modo kernel, no tienen acceso directo al hardware. Por tanto, si se activa un error, en el servidor de archivos, es posible que el servicio de archivo se caiga, pero normalmente esto no hará que se caiga toda la máquina.

Las **llamadas al sistema** (system calls) proporcionan los medios para que un programa de usuario pida al SO que realice tareas reservadas del SO en nombre del programa de usuario.
Cada SO tiene sus propios nombres de llamadas al sistema y se ejecutan en modo kernel.
Los programadores diseñan sus programas usando una API (interfaz de programación de aplicación). La API especifica un conjunto de funciones que el programador de aplicaciones puede usar, indicándole los parámetros que hay que usar en cada función y los valores de retorno que el programador debe esperar. Los parámetros asociados a las llamadas pueden pasarse de varias maneras: por registro, bloques o tablas en memoria o pilas.
Quien realiza la llamada no tiene por qué saber nada acerca de cómo se implementa dicha llamada al sistema o qué es lo que ocurre durante la ejecución. Tan sólo necesita ajustarse a lo que la API especifica y entender lo que hará el SO como resultado de la ejecución de dicha llamada al sistema.
Por tanto, la API oculta al programador la mayor parte de los detalles de la interfaz del SO.
Categorías de llamadas al sistema:

- Control de procesos:
  - Terminar, abortar.
  - Cargar, ejecutar.
  - Crear procesos o terminar procesos.
  - Obtener o definir atributos del proceso.
  - Esperar para obtener tiempo.
  - Esperar o señalizar suceso.
  - Asignar o liberar memoria.

Un programa en ejecución puede interrumpir dicha ejecución bien de forma normal (end) o de forma anormal (abort).

- Administración de archivos
  - Crear o borrar archivos.
  - Abrir, cerrar.
  - Leer, escribir, reposicionar.
  - Obtener o definir atributos de archivo.
- Administración de dispositivos

- Stack (contiene datos temporarios como parámetros de subrutinas, variables temporales y direcciones de retorno).

Un proceso cuenta con dos stack: uno para modo usuario y otro para modo kernel.
Se crea automáticamente y su medida se ajusta en run-time.

Atributos de un proceso:

- Identificación del proceso, y del proceso padre.
- Identificación del usuario que lo "disparó".
- Si hay estructura de grupos, grupo que lo "disparó".
- En ambientes multiusuarios, de dónde disparó.

Bloque de control de Proceso (PCB): estructura asociada al proceso. Una por proceso. Contiene información asociada con cada proceso: estado, contenido del PC, contenidos de los registros de la CPU.
Es lo primero que se crea cuando se crea un proceso y lo último que se borra cuando termina.
El bloque de control (PCB) es la estructura de datos central y más importante de un SO. Cada bloque de control de proceso contiene toda la información de un proceso necesaria para el SO, incluyendo aquellos que tienen que ver con la planificación, la asignación de recursos, el tratamiento de interrupciones y el análisis y supervisión del rendimiento. Puede decirse que el conjunto de los bloques de control de procesos definen el estado del SO.

Estructuras asociadas al proceso:

- del proceso (PCB).
- De archivos que utiliza.
- De memoria que utiliza.

El espacio de dirección de un proceso es el conjunto de direcciones de memoria que ocupa el proceso. No incluye su PCB o tablas asociadas. Un proceso en modo usuario puede acceder sólo a su espacio de direcciones; en modo kernel, a estructuras del kernel o a espacios de direcciones de otros procesos.

Contexto de un proceso: incluye toda la información que el SO necesita para administrar el proceso, y que la CPU necesita para ejecutarlo correctamente. Son parte del contexto los registros de CPU, inclusive el contador del programa, prioridad del proceso, si tiene E/S pendientes, etcétera.

Cambio de contexto (context switch): se produce cuando la CPU cambia de un proceso a otro. Se debe resguardar la información del proceso saliente, que pasa a espera y retornará después la CPU. Se debe cargar la información asociada al nuevo proceso y comenzar desde la instrucción siguiente a la última ejecutada.
Cuando se produce una interrupción el sistema tiene que guardar el contexto actual del proceso que se está ejecutando en la CPU, de modo que pueda restaurar dicho contexto cuando su procesamiento concluya, suspendiendo el proceso y restaurándolo después. El contexto se almacena en la PCB del proceso e incluye el valor de los registros de la CPU, el estado del proceso y la información de gestión de memoria. Es decir, realizamos una salvaguarda del estado actual de la CPU, y una restauración del estado para reanudar las operaciones. La conmutación de la CPU a otro proceso requiere una salvaguarda del estado del proceso actual y una restauración del estado de otro proceso diferente. Esta tarea se conoce como cambio de contexto. Cuando se produce el kernel guarda el contexto del proceso antiguo en su PCB y carga el contexto almacenado del nuevo proceso que se ha decidido ejecutar. Mientras se hace el cambio de contexto la CPU no hace nada y es tiempo desperdiciado.

Colas en la planificación de procesos:

- De trabajos o procesos: de todos los procesos del sistema.
- De procesos listos: residente en memoria principal, están listo para ejecutarse, se encuentran en estado listo.
- De dispositivos: procesos esperando por un dispositivo E/S.

Estado del proceso: a medida que se ejecuta un proceso, este va cambiando de estado. Estos son:

continuamente y cada vez que un proceso tiene que esperar, otro proceso puede hacer uso de la CPU. Cuando la CPU queda inactiva, el SO debe seleccionar uno de los procesos que se encuentran en la cola de procesos preparados para ejecutar. El **planificador a corto plazo** lleva a cabo esa selección del proceso. El planificador elige uno de los procesos que están en memoria preparados para ejecutarse y asigna la CPU a dicho proceso.
La **planificación a largo plazo** 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.
Entonces el planificador a largo plazo decide añadir procesos a la reserva de procesos a ejecutar.
El **medium term scheduler** (swapping), si es necesario, reduce el grado de multiprogramación. Saca temporariamente de memoria los procesos que sean necesarios para mantener el equilibrio del sistema. Términos asociados: swap out (sacar de memoria) y swap in (volver a memoria).

Transiciones:

- New-ready: por elección del scheduler de largo plazo (carga en memoria).
- Ready-running: por elección del scheduler de corto plazo (asignación de CPU).
- Running-waiting: el proceso "se pone a dormir", esperando por un evento.
- Waiting-ready: terminó la espera y compite nuevamente por la CPU.
- Caso especial: running-ready: cuando el proceso termina su quantum (franja de tiempo), sin haber necesitado interrumpirlo por un evento, pasa al estado de ready, para competir por CPU, pues no está esperando por ningún evento.

La adecuada planificación de la CPU depende de una propiedad observada de los procesos: la ejecución de un proceso consta de un ciclo de ejecución en la CPU, seguida de una espera de E/S; los procesos alternan entre estos dos estados. La ejecución del proceso comienza con una ráfaga de CPU y va seguida una ráfaga de E/S, después una de CPU, luego de E/S, y así hasta que la ráfaga final de CPU concluya con una solicitud al sistema para terminar la ejecución.
CPU-bound: un proceso limitado por CPU (CPU-bound) es aquél que pasa más tiempo computando que haciendo E/S (tiene ráfagas de CPU largas).
I/O-bound: un proceso limitado por E/S (I/O-bound) es aquél que se pasa más tiempo haciendo E/S que usando la CPU (tiene ráfagas de CPU cortas).
La velocidad de CPU tiene un aumento considerable respecto a dispositivos de I/O. Entonces, necesidad de atender rápidamente procesos I/O-bound para mantener el dispositivo ocupado y aprovechar la CPU para procesos CPU-bound.
Algoritmos no apropiativos: una vez que el proceso pasa a estado de ejecución, continúa

- Protección: cada proceso debe protegerse contra interferencias no deseadas de otros procesos, tanto accidentales como intencionadas. Así pues, el código de un proceso no puede hacer referencia a posiciones de memoria de otros procesos, con fines de lectura o escritura, sin permiso. Hasta cierto punto, satisfacer las exigencias de reubicación aumenta la dificultad de satisfacción de las exigencias de protección. Es más, la mayoría de los lenguajes de programación permiten el cálculo dinámico de direcciones durante la ejecución, generando, por ejemplo, un índice de un vector o un puntero a una estructura de datos. Por tanto, todas las referencias a memoria generadas por un proceso deben comprobarse durante la ejecución para asegurar que sólo hacen referencia al espacio de memoria destinado a dicho proceso. Afortunadamente, los mecanismos que respaldan la reubicación también forman parte básica del cumplimiento de las necesidades de protección. El SO no puede anticipar todas las referencias a memoria que un proceso puede realizar. Normalmente, un proceso de usuario no puede acceder a ninguna parte del sistema operativo, tanto programa como datos. De nuevo, el programa de un proceso no puede en general bifurcar hacia una instrucción de otro proceso. Además, sin un acuerdo especial, el programa de un proceso no puede acceder al área de datos de otro proceso. El procesador debe ser capaz de abandonar tales instrucciones en el momento de la ejecución. Nótese que las exigencias de protección de memoria pueden ser satisfechas por el procesador (hardware) en vez de por el sistema operativo (software). Esto es debido a que el sistema operativo no puede anticiparse a todas las referencias a memoria que hará un programa. Incluso si tal anticipación fuera posible, sería prohibitivo en términos de tiempo consumido el proteger cada programa por adelantado de posibles violaciones de referencias a memoria. Así pues, sólo es posible evaluar la tolerancia de una referencia a memoria (acceso a datos o bifurcación) durante la ejecución de la instrucción que realiza la referencia. Para llevar esto a cabo, el hardware del procesador debe poseer dicha capacidad.
- Compartición: cualquier mecanismo de protección que se implemente debe tener la flexibilidad de permitir el acceso de varios procesos a la misma zona de memoria principal. Por ejemplo, si una serie de procesos están ejecutando el mismo programa, resultaría beneficioso permitir a cada proceso que acceda a la misma copia del programa, en lugar de tener cada uno su propia copia aparte. Los procesos que cooperan en una tarea pueden necesitar acceso compartido a la misma estructura de datos. El sistema de gestión de memoria debe, por tanto, permitir accesos controlados a las áreas compartidas de la memoria, sin comprometer la protección básica. Se verá que los mecanismos empleados para respaldar la reubicación forman parte básica de las capacidades de compartición.

Abstracción de la memoria:

- Espacio de direcciones: rango de direcciones posibles que un proceso puede utilizar para direccionar sus instrucciones y datos.
  Varía en un rango dependiendo la arquitectura:
  - 32 bits: 0..2^32-1
  - 64 bits: 0..2^64-1

  Debe ser independiente de la ubicación "real" del proceso en la memoria.
- 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.

  Registros utilizados:
  - Registro base: dirección de comienzo del proceso.

para formar un único bloque de gran tamaño.

Algoritmos de ubicación:

- En las particiones fijas de igual tamaño: no se necesitan un algoritmo.
- En las particiones fijas de diferente tamaño y particiones dinámicas: puesto que la compactación de memoria consume tiempo, atañe al diseñador del sistema operativo decidir adecuadamente cómo asignar un proceso a memoria (como llenar los huecos). Cuando llega el momento de cargar o traer un proceso a memoria principal y, si hay libre más de un bloque de memoria de tamaño suficiente, el sistema operativo debe decidir cuál asignar.
  - Algoritmos:
    - Best fit: selecciona la partición más 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 contenga el proceso.
    - Next fit: mantiene las particiones libres como una lista circular. Selecciona la primera partición que encuentra que contenga el proceso.
    - Worst Fit: selecciona la partición libre más grande que contenga el proceso. Mal uso en particiones fijas. Buen uso en particiones dinámicas.

Superposiciones (overlapping): el tamaño de los procesos y sus datos puede exceder el tamaño de la memoria. Entonces, se separa el programa en módulos, se cargan 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.

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.

V. Bibliografía.

- SISTEMAS OPERATIVOS (Segunda edición), William Stallings.
- FUNDAMENTOS DE SISTEMAS OPERATIVOS (Séptima edición), Galvin Silberschatz.
- SISTEMAS OPERATIVOS, DISEÑO E IMPLEMENTACIÓN (Segunda edición), Andrew S. Tanenbaum.
- http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SOF.htm
- http://es.wikipedia.org.

KUBRICK LIVES