**Lectura 1:  
8.1 Fundamentos**

* **Separación de Procesos**:
  + Cada proceso tiene su propio espacio de memoria.
  + Se usan registros de base y límite para definir el rango de memoria permitido.
  + Accesos fuera del rango generan una interrupción que el sistema operativo trata como un error.
* **Protección de Memoria**:
  + La CPU compara cada dirección generada en modo usuario con los registros base y límite.
  + Esto previene accesos no autorizados a la memoria de otros procesos o del sistema operativo.

**8.1.1 Hardware básico**

* **Acceso a Memoria**:
  + La CPU accede directamente solo a la memoria principal y a los registros integrados.
  + Los registros son más rápidos que la memoria principal.
  + Se usa memoria caché para reducir el tiempo de acceso y mejorar el rendimiento.

**8.1.2 Reasignación de direcciones**

* **Reasignación en Diferentes Tiempos**:
  + Las direcciones pueden reasignarse en tiempo de compilación, carga o ejecución.
  + La reasignación dinámica en tiempo de ejecución usa la Unidad de Gestión de Memoria (MMU).
  + La MMU traduce direcciones lógicas a direcciones físicas.

**8.1.3 Espacios de direcciones lógico y físico**

* **Direcciones Lógicas y Físicas**:
  + Las direcciones lógicas son generadas por la CPU.
  + La MMU convierte las direcciones lógicas en físicas.
  + Esto asegura que cada proceso opera dentro de su espacio de memoria asignado.

**8.1.4 Carga dinámica**

* **Ventajas y Funcionamiento**:
  + Las rutinas de un programa se cargan en memoria solo cuando se necesitan.
  + Ahorra espacio de memoria, útil para grandes cantidades de código que se usan infrecuentemente.
  + No requiere soporte específico del sistema operativo, pero los programas deben estar diseñados para ello.

**8.1.5 Bibliotecas compartidas**

* **Optimización de Espacio**:
  + Múltiples programas usan el mismo código de biblioteca sin cargar múltiples copias.
  + Se usa un stub en cada programa para localizar y cargar las rutinas de biblioteca.
  + Solo una copia de la biblioteca está en memoria, compartida por todos los programas que la necesiten.
* **Actualización de Bibliotecas**:
  + Las bibliotecas se pueden actualizar sin recompilar todos los programas que las usan.
  + Los programas usan información de versión para asegurarse de que ejecutan la versión correcta de la biblioteca, evitando problemas de compatibilidad.

**Video:**

1. **Tipos de Asignaciones de Memoria**:
   * **Asignación Contigua**: Cada tarea debe estar en áreas contiguas de la memoria.
   * **Asignación No Contigua**: No se detalla en la transcripción.
2. **Tabla de Descripción de Particiones**:
   * Utilizada para escribir cada partición de la memoria, indicando base, tamaño y estado.
   * Ejemplo:
     + **Partición 0**: Sistema operativo, tamaño 100K (de 0 a 100).
     + **Partición 1**: Base 100, tamaño 200 (de 100 a 300), libre.
     + **Partición 2**: Base 300, tamaño 200 (de 300 a 500), libre.
3. **Algoritmos de Asignación de Memoria**:
   * **Primer Ajuste**: Asigna el primer hueco suficientemente grande.
   * **Mejor Ajuste**: Asigna el hueco más pequeño que sea suficiente.
   * **Peor Ajuste**: Asigna el hueco más grande.
4. **Asignación Dinámica de Memoria**:
   * Inicialización del sistema declarando toda la memoria libre.
   * Uso de algoritmos para asignar y particionar la memoria según tamaño de procesos y bloques libres.
   * Diferencia (DIFE) y constante (c) para decidir la partición.
5. **Paginación**:
   * Transferencia de páginas del almacenamiento secundario al primario en bloques llamados marcos de página.
   * Dirección real formada por la concatenación del número de página y el desplazamiento.
6. **Segmentación**:
   * Soporte directo del hardware a regiones de memoria, mapeadas en varios segmentos (subrutinas, pilas, símbolos, programa principal).
   * Traducción de direcciones lógicas a direcciones físicas mediante la tabla de segmentos.
   * Comparación de desplazamiento con límite, suma de base y asignación en memoria principal.
7. **Comparación entre Paginación y Segmentación**:
   * **Paginación**:
     + No hay visión del programador.
     + Espacio de direcciones es uno.
     + Fragmentación interna.
   * **Segmentación**:
     + Visión del programador.
     + Varios espacios de direcciones.
     + Fragmentación externa.
     + Protección de códigos y datos.
     + Crecimiento y compartición de memoria.

**Lectura 2**

**8.2 Intercambio**

1. **Definición de Intercambio**:
   * Un proceso debe estar en memoria para ser ejecutado, pero puede ser intercambiado temporalmente fuera de la memoria hacia un almacén de respaldo, y luego regresado a la memoria para continuar su ejecución.
2. **Uso en Entornos de Multiprogramación**:
   * En entornos con planificación de CPU basada en turnos, cuando un cuanto de tiempo termina, el gestor de memoria intercambia procesos para mantener algunos siempre listos para ejecutarse.
   * En algoritmos de planificación con prioridad, los procesos de menor prioridad pueden ser intercambiados cuando llegan procesos de mayor prioridad .
3. **Ubicación del Proceso Intercambiado**:
   * Generalmente, un proceso descargado se recarga en el mismo espacio de memoria para evitar problemas de reasignación de direcciones.
   * Con la reasignación en tiempo de ejecución, el proceso puede moverse a diferentes ubicaciones de memoria .
4. **Consideraciones de Tiempo de Intercambio**:
   * Ejemplo de cálculo del tiempo de intercambio: un proceso de 10 MB intercambiado con un disco duro que tiene una velocidad de transferencia de 40 MB/s tomaría aproximadamente 516 milisegundos en total (incluyendo carga y descarga).
   * Es crucial que el tiempo de ejecución de cada proceso sea largo en relación con el tiempo de intercambio para asegurar un uso eficiente de la CPU .

**8.3 Asignación de memoria contigua**

1. **Introducción**:
   * La memoria principal debe albergar tanto el sistema operativo como los procesos de usuario, y es necesario asignar la memoria de forma eficiente .
2. **División de la Memoria**:
   * La memoria se divide en dos particiones: una para el sistema operativo y otra para los procesos de usuario. Usualmente, el sistema operativo se ubica en la parte baja de la memoria .
3. **Múltiples Procesos en Memoria**:
   * Para tener varios procesos de usuario en memoria simultáneamente, cada proceso se contiene en una única sección contigua de memoria .

**8.3.1 Mapeo de memoria y protección**

1. **Registro de Reubicación y Límite**:
   * Se utilizan un registro de reubicación y un registro límite para manejar la protección y mapeo de memoria.
   * El registro de reubicación contiene la dirección física más pequeña, y el registro límite contiene el rango de direcciones lógicas .
2. **Cambio de Contexto**:
   * Cuando el planificador de la CPU selecciona un proceso, se cargan los valores correctos en los registros de reubicación y de límite como parte del cambio de contexto.
   * Este mecanismo protege tanto al sistema operativo como a otros programas y datos de las modificaciones por parte del proceso en ejecución .
3. **Flexibilidad del Sistema Operativo**:
   * Permite que el tamaño del sistema operativo cambie dinámicamente.
   * El código transitorio del sistema operativo se carga y descarga según sea necesario, lo que modifica el tamaño del sistema operativo durante la ejecución del programa .

**8.3.2 Asignación de memoria**

1. **Método de Particiones Múltiples**:
   * La memoria se divide en varias particiones de tamaño fijo, cada una albergando exactamente un proceso.
   * El grado de multiprogramación está limitado por el número de particiones disponibles .
2. **Método de Particiones Variables (MVT)**:
   * Es una generalización del esquema de particiones fijas y es utilizado principalmente en entornos de procesamiento por lotes.
   * Se mantiene una tabla que indica qué partes de la memoria están disponibles y cuáles están ocupadas. Se asigna sólo la memoria necesaria para un proceso, manteniendo el resto disponible para futuras solicitudes .

**8.3.3 Fragmentación**

* **Fragmentación externa:** Ocurre cuando el espacio de memoria libre se divide en fragmentos pequeños, dificultando la asignación de memoria.
* **Impacto de los algoritmos de asignación:** Tanto el primer ajuste como el mejor ajuste pueden afectar el grado de fragmentación según el sistema y el tamaño de los procesos.
* **Fragmentación interna:** Sucede cuando se asigna más memoria de la necesaria para un proceso, generando desperdicio de espacio dentro de una partición.
* **Compactación:** Solución que reorganiza la memoria para crear un único bloque contiguo de memoria libre, aunque puede ser costosa o no siempre posible.
* **Asignación no contigua:** Solución que permite asignar memoria física a procesos independientemente de su ubicación en la memoria, utilizando técnicas como la paginación y la segmentación.

**Lectura 3**

**Sistema por lotes:**

* **Particiones fijas en memoria:**
  + Proporciona una forma sencilla y efectiva de gestionar la memoria.
  + Cada partición tiene un tamaño predefinido y se asigna a un trabajo cuando llega al frente de la cola.
  + Los trabajos permanecen en la memoria hasta su finalización.

**Sistemas de tiempo compartido o computadoras personales orientadas a gráficos:**

* **Memoria insuficiente:**
  + En ocasiones, la memoria principal no es suficiente para contener todos los procesos activos simultáneamente.
  + Los procesos excedentes se mantienen en disco y se intercambian dinámicamente según sea necesario para su ejecución.

**Estrategias de administración de memoria:**

* **Intercambio:**
  + Cada proceso se carga completamente en la memoria, se ejecuta durante un tiempo y luego se coloca de nuevo en disco.
* **Memoria virtual:**
  + Permite que los programas se ejecuten, aunque solo partes estén en memoria principal.
  + Segmentos de programas se cargan y descargan dinámicamente según sea necesario.

**Administración de memoria con mapas de bits:**

* **Contabilización de la memoria:**
  + La memoria se divide en unidades de asignación con un mapa de bits asociado.
  + Cada bit en el mapa indica si una unidad está ocupada o libre.
* **Problemas:**
  + La búsqueda de series de bits libres puede ser lenta, especialmente para series largas que atraviesan fronteras de palabras en el mapa.

**Administración de memoria con listas enlazadas:**

* **Listas de segmentos:**
  + Mantienen una lista enlazada de segmentos de memoria libres y asignados.
  + Facilita la actualización cuando un proceso termina o se intercambia a disco.
* **Algoritmos de asignación:**
  + Primer ajuste, siguiente ajuste y mejor ajuste determinan cómo se asigna la memoria a los procesos.

**Memoria Virtual:**

* **Concepto:**
  + Permite que los programas se ejecuten, aunque el tamaño total exceda la memoria física disponible.
  + Segmentos del programa se mantienen en memoria principal y el resto en disco, intercambiándose según sea necesario.
* **Paginación:**
  + Una técnica común de memoria virtual que transforma direcciones virtuales en direcciones físicas mediante una unidad de administración de memoria (MMU).