Actividad 2: Gestión de Entrada/Salida y de Memoria

1. Gestion de Entrada/Salida

La gestión de memoria en los sistemas operativos no solo se ocupa de optimizar el uso de la RAM, sino también de la correcta comunicación entre el software y el hardware a través de los dispositivos de Entrada/Salida (E/S). La jerarquía de E/S, tal como la describe Tanenbaum, establece una estructura que organiza el flujo de datos entre los procesos de usuario y los dispositivos físicos, al mismo tiempo que asegura la protección de la memoria y los recursos del sistema.

1.1 Jerarquía de Entrada/Salida

La jerarquía de E/S es un modelo que describe cómo el sistema operativo maneja las operaciones de entrada y salida a través de diferentes capas. Estas capas incluyen el hardware físico, los controladores, el sistema de archivos, y las bibliotecas de E/S, hasta llegar a las aplicaciones de usuario que realizan las operaciones finales.

- **Hardware**

  Incluye los dispositivos físicos, como discos, teclados, pantallas, etc., que están directamente conectados al sistema operativo a través de los controladores y manejadores de interrupciones.

- **Manejadores de Interrupciones**

  Son parte del sistema operativo y se encargan de gestionar las interrupciones generadas por los dispositivos hardware, como teclas pulsadas, paquetes de red recibidos o finalización de una operación de disco. Están diseñados para ser rápidos y eficientes, ya que deben gestionar eventos en tiempo real.

- **Controladores de Dispositivos (Drivers)**

  En el contexto de los sistemas operativos, los controladores de

dispositivos se refieren a programas de software diseñados para facilitar la comunicación entre el sistema operativo y el hardware. Estos drivers actúan como interfaces que traducen las solicitudes de entrada/salida (E/S) de alto nivel realizadas por el sistema operativo o las aplicaciones en instrucciones específicas que el hardware puede entender y ejecutar. Por ejemplo, un controlador de disco convierte una solicitud como "leer archivo" en comandos específicos que el hardware del disco puede interpretar.

Es importante no confundir este uso del término "controlador de dispositivo" con su aplicación en el ámbito del hardware. En ese contexto, un controlador de dispositivo se refiere a un componente físico, como una **controladora de disco**, que es un hardware encargado de gestionar la transferencia de datos entre el sistema y el disco, traduciendo las instrucciones enviadas por el software (driver) en señales físicas que el disco puede procesar.

Por lo tanto, en este caso, cuando hablamos de "controladores de dispositivos", nos referimos exclusivamente al software (drivers).

- **Sistema de E/S Genérico**

  Esta capa del sistema operativo abstrae los detalles de los controladores de dispositivos y gestiona aspectos como la cola de solicitudes de E/S, buffering y caching, lo cual optimiza el acceso a los dispositivos.

- **Sistema de Archivos**

  El sistema de archivos organiza y gestiona los datos en los dispositivos de almacenamiento. Proporciona una interfaz para manipular archivos y directorios de manera independiente al hardware subyacente.

- **Bibliotecas de Entrada/Salida**

  Son software independiente del sistema operativo que proporciona

funciones estándar para realizar operaciones de E/S, como `fopen()`, `fclose()` o `printf()` en C, sin que el programador tenga que preocuparse por los detalles del hardware específico.

- **Aplicaciones de Usuario**
  En la capa más alta, los programas de usuario realizan operaciones de E/S utilizando las bibliotecas de entrada/salida. Estas aplicaciones se comunican con los dispositivos sin interactuar directamente con el hardware.

1.2 Anillos de Protección y Gestión de Recursos Críticos

¿Los controladores de dispositivos deben ejecutarse en modo kernel o en modo usuario?

La ejecución de controladores de dispositivos plantea una cuestión importante sobre el nivel de privilegios más adecuado para garantizar tanto su funcionalidad como la seguridad del sistema. Recordemos que, además de los modos kernel y usuario, los procesadores modernos cuentan con un sistema de **anillos de protección**. Estos anillos organizan los niveles de privilegio en el acceso a los recursos del sistema, desde el **Anillo 0** (modo kernel), que tiene acceso total al hardware y todos los recursos críticos, hasta el **Anillo 3** (modo usuario), donde los procesos tienen privilegios limitados para prevenir posibles daños al sistema.

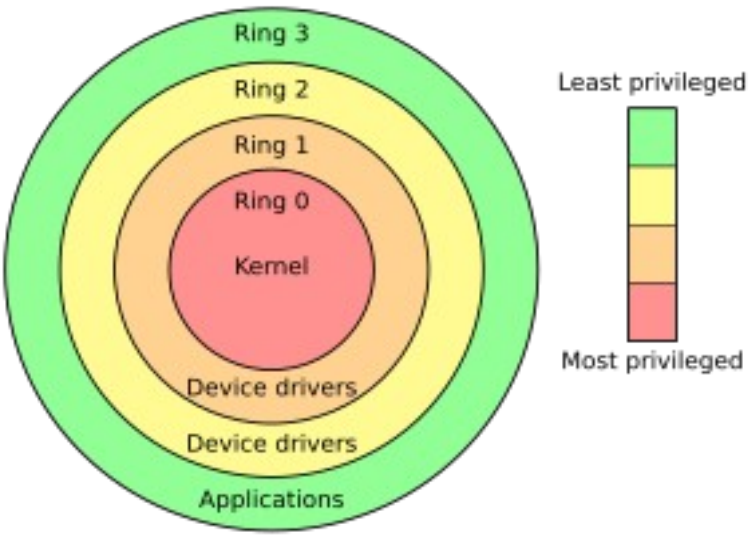
Los controladores de dispositivos necesitan privilegios suficientes para interactuar directamente con el hardware, lo que normalmente justificaría su ejecución en el **Anillo 0**. Sin embargo, otorgarles acceso total al sistema puede ser un riesgo, ya que podrían acceder a memoria o recursos críticos no relacionados con su tarea.

Por esta razón, lo más adecuado sería que los controladores de dispositivos se ejecuten en un **anillo de protección intermedio**. Este nivel les permitiría comunicarse eficazmente con el hardware, pero sin comprometer la seguridad

del sistema al restringir su acceso a memoria y recursos ajenos a su función específica. De este modo, se logra un equilibrio entre funcionalidad y seguridad, minimizando los riesgos asociados con el uso de controladores en niveles de privilegio demasiado altos.

En resumen, la organización en anillos de protección no solo optimiza la gestión de recursos críticos, sino que también refuerza la seguridad al controlar cómo y quién accede a ellos, proporcionando una arquitectura más robusta y confiable.

F



***igura** 1: Anillos de protección*

2. Gestión de Memoria

2.1 Problemas de Protección y Reubicación en la Gestión de Memoria

La **protección** y la **reubicación** son dos problemas fundamentales que un sistema operativo debe resolver al gestionar la memoria de un sistema. Ambos problemas están intrínsecamente relacionados con la capacidad del procesador para

traducir las direcciones virtuales (o lógicas) generadas por un programa en direcciones reales (o físicas) en la memoria principal.

2.1.1. Protección

La protección implica garantizar que un proceso no pueda acceder al espacio de memoria de otro proceso ni a las áreas reservadas para el sistema operativo. Esto es fundamental para evitar conflictos, corrupción de datos o comportamientos inesperados entre procesos.

2.1.2 Reubicación

La reubicación implica que el sistema operativo debe ser capaz de asignar un proceso a cualquier área libre de la memoria física en el momento de su ejecución. Esto permite aprovechar la memoria disponible de manera eficiente, ya que los procesos no dependen de estar ubicados en un lugar específico.

Traducción de Direcciones Virtuales a Reales

Resolver estos problemas requiere que el procesador traduzca las **direcciones virtuales** generadas por las instrucciones de código máquina del programa en **direcciones reales** en la memoria física. Esta traducción es posible gracias a mecanismos específicos del procesador, que trabajan con información proporcionada por el sistema operativo. Los métodos más comunes para realizar esta traducción son el uso de **registros base y límite** (en el caso del intercambio) o **tablas de páginas** (en el caso de la paginación).

2.2 Intercambio (Swapping)

El intercambio (swapping) es un método básico para gestionar la memoria y resolver los problemas de protección y reubicación. Este mecanismo mueve procesos completos entre la memoria principal (RAM) y el almacenamiento secundario (como discos duros o SSD) para liberar espacio y permitir la multitarea.

2.2.1 Traducción de Dirección Virtual a Real en el Intercambio

El procesador utiliza dos registros clave:

- **Registro Base**: Contiene la dirección de memoria física donde comienza el proceso en la RAM.
- **Registro Límite**: Define el tamaño máximo de memoria asignada al proceso.

Cuando una instrucción del programa genera una dirección virtual, el procesador realiza los siguientes pasos:

- **Comprobación de Protección**: La dirección virtual se compara con el valor en el registro límite. Si la dirección virtual es mayor que el límite, el procesador genera una excepción, evitando accesos no autorizados.
- **Traducción**: Si la dirección virtual es válida, se suma al valor del registro base para obtener la dirección física real.

2.2.2. Protección en el Intercambio

- Los registros base y límite garantizan que el proceso solo pueda acceder a su espacio asignado.
- Si un proceso intenta acceder a una dirección fuera de su rango, el procesador bloquea el acceso y genera una interrupción.

2.2.3. Limitaciones del Intercambio

- El intercambio requiere mover procesos completos entre RAM y almacenamiento secundario, lo que puede ser lento y consume recursos.
- Es ineficiente para manejar procesos grandes o un gran número de procesos.

2.3. Paginación

La paginación es un método más avanzado que divide tanto la memoria virtual como la memoria física en bloques de tamaño fijo llamados **páginas** y **marcos**, respectivamente. Este método resuelve los problemas de protección y reubicación de manera más eficiente y flexible que el intercambio.

2.3.1. Traducción de Dirección Virtual a Real en la Paginación

El procesador utiliza una estructura llamada **tabla de páginas**, que mapea las páginas virtuales del proceso a los marcos físicos en la RAM.

1. **Estructura de la Dirección Virtual**:
    - Una dirección virtual se divide en dos partes:
        - **Número de página**: Identifica la página en el espacio virtual del proceso.
        - **Desplazamiento dentro de la página**: Especifica la ubicación exacta dentro de esa página.
2. **Proceso de Traducción**:
    - El procesador consulta la **tabla de páginas** del proceso, que está almacenada en la memoria.
    - Utiliza el número de página para encontrar el marco correspondiente en la memoria física.
    - Combina la dirección base del marco con el desplazamiento dentro de la página para obtener la dirección física real.
3. **Comprobación de Protección**:
    - Cada entrada en la tabla de páginas incluye permisos (lectura, escritura, ejecución) que el procesador verifica antes de acceder a una dirección.
    - Si el proceso intenta acceder a una página para la que no tiene permisos, el procesador genera una excepción.

2.3.2. Ventajas de la Paginación sobre el Intercambio

1. **Flexibilidad en la Ubicación de los Procesos**:
    - En el intercambio, un proceso debe ocupar un bloque contiguo de memoria física, lo que puede fragmentar la memoria.

- En la paginación, las páginas de un proceso pueden distribuirse en diferentes marcos de la memoria física, eliminando la necesidad de bloques contiguos.

2. **Mejor Uso de la Memoria**:
    - La paginación permite cargar solo las páginas necesarias de un proceso, mientras que el intercambio requiere cargar el proceso completo.
    - Esto optimiza el uso de la RAM y permite que más procesos se ejecuten simultáneamente.
3. **Eficiencia en la Gestión de Memoria Virtual**:
    - La paginación admite **memoria virtual**, donde el espacio de direcciones virtuales es mayor que la RAM disponible, gracias a la carga y descarga dinámica de páginas.

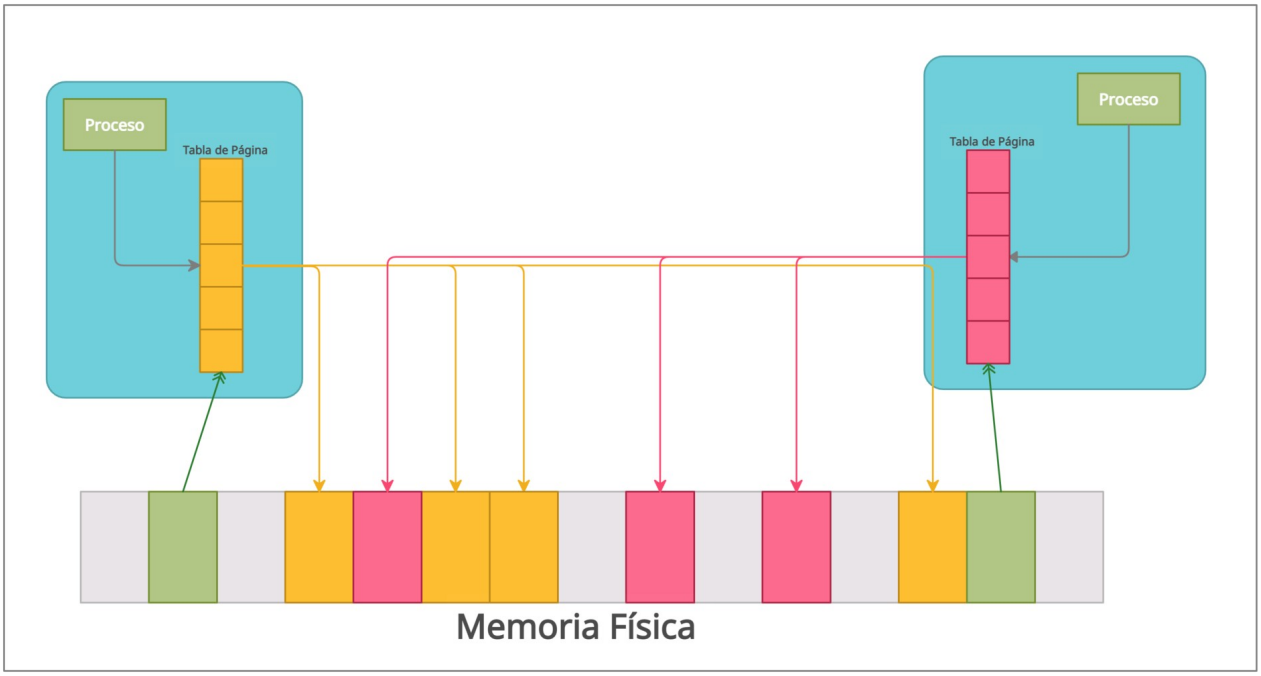


Figura 2: Conversión de dirección virtual a física con paginación

2.4. Conclusión

Tanto el intercambio como la paginación resuelven los problemas de protección y reubicación al permitir que los procesos se alojen en cualquier lugar de la memoria y evitar accesos no autorizados. Sin embargo, la **paginación es una solución superior** porque:

- Reduce la fragmentación.
- Optimiza el uso de la memoria física.
- Facilita la ejecución simultánea de múltiples procesos con menos requisitos de espacio en la RAM.

Esto la convierte en el estándar para la gestión de memoria en los sistemas modernos.