

# Resumen sistemas operativos

## Temas parcial #1

### Estudiantes:

Juan Sebastian Loaiza – juans.loaiza@udea.edu.co

Sulay Martínez – sulay.martinez@udea.edu.co



## Clase 1 - Introducción a los Sistemas Operativos

Curso: Sistemas Operativos

Tema: Fundamentos del Sistema Operativo

Objetivo: Comprender el rol, estructura y diseño de los SO modernos.



## ¿Qué es un Sistema Operativo?

Un **Sistema Operativo (SO)** es el software que:

- Administra recursos de hardware y software.
- Controla la ejecución de programas.
- Proporciona abstracciones amigables del hardware.
- Se encarga de **evitar errores y proteger el sistema**.

## Componentes del Sistema de Cómputo

```
+-----+
| Usuario Final |
+-----+
| Aplicaciones |
+-----+
| Sistema Operativo |
+-----+
| Hardware |
+-----+
```

- **Usuarios:** Personas, máquinas, otras computadoras.
- **Aplicaciones:** Compiladores, navegadores, etc.
- **Sistema Operativo:** Administra y coordina el hardware.
- **Hardware:** CPU, memoria, dispositivos de entrada/salida.

## Virtualización de Recursos

El SO convierte recursos **físicos** en **virtuales**:

Recurso Físico	Forma Virtual	Ejemplo
CPU	Proceso / Hilo	<code>cpu.c</code>
Memoria	Espacio virtual	<code>mem.c</code>
Disco	Sistema de Archivos	<code>io.c</code>

### Conceptos clave:

- Virtualización de CPU \* Ilusión de múltiples CPUs.
- Virtualización de Memoria \* Cada proceso tiene su espacio.
- Persistencia \* Datos almacenados en disco (no volátil).
- Llamadas al sistema \* Permiten al usuario comunicarse con el SO.

## Concurrency and Recursion

 ¡Cuidado con los accesos simultáneos!

- Problema típico: Dos hilos modifican una misma variable.
- Se pueden generar errores si no hay sincronización.

 **Ejemplo:**

```
load R1, balance
add R1, amount
store R1, balance
```

 Solución: **sincronización** con semáforos, locks, etc.

## Operaciones del Sistema Operativo

 **Modos del procesador:**






Modo	Características
Usuario	Limitado, ejecuta apps
Kernel	Privilegiado, ejecuta el SO

 **Proceso de una llamada al sistema:**

1. App hace una solicitud.
2. El control pasa al SO.
3. Se cambia al **modo kernel**.
4. El SO ejecuta la operación.
5. Se regresa al **modo usuario**.

# **Objetivos de Diseño del SO**

 El SO debe...

1.  **Crear abstracciones útiles**
2.  **Tener buen desempeño** (mínimo overhead)
3.  **Proteger los procesos entre sí**
4.  **Ser confiable**
5.  **Otros:**
  - Seguridad
  - Eficiencia energética
  - Soporte de movilidad

# Semana 5: Virtualización de Memoria y Paginación



## Conceptos Clave

Concepto	Descripción breve
<b>Espacio de direcciones</b>	Abstracción que permite a cada proceso creer que tiene acceso a toda la memoria
<b>Memoria virtual</b>	Técnica que permite ejecutar procesos sin que estén completamente en la memoria física
<b>MMU (Unidad de Gestión de Memoria)</b>	Hardware responsable de la traducción de direcciones virtuales a físicas
<b>Paginación</b>	Divide la memoria virtual y física en bloques de tamaño fijo llamados páginas y marcos
<b>Segmentación</b>	Divide el espacio de direcciones en segmentos lógicos de tamaño variable



## ¿Por qué virtualizar la memoria?

- Permite **aislamiento** entre procesos.
- Proporciona un entorno coherente y uniforme para los programas.
- Permite usar más memoria de la disponible físicamente.
- Mejora la **utilización** del sistema.



## Mecanismo de Traducción de Direcciones



### ¿Cómo se logra?

Mediante **traducción de direcciones** que realiza la MMU:

Dirección Virtual \* Dirección Física

## Métodos:

- **Relocalización dinámica (base y límite)**
- **Segmentación**
- **Paginación**



## Segmentación

Segmento	Contenido	Crece hacia
Código	Instrucciones del programa	Fijo
Heap	Datos dinámicos (malloc)	Positivo
Stack	Variables locales y funciones	Negativo

Cada segmento tiene su propio par de registros base y límite.



## Ejemplo de Segmentación

Segmento: Código

Base: 32K, Límite: 2K

VA:  $100 * PA > 100 + 32K > 32868$



## Paginación

- **Unidad de traducción fija:** evita fragmentación externa.
- La dirección virtual se divide en:
  - **VPN** (Virtual Page Number)
  - **Offset**



## Ejemplo

Memoria física: 128 bytes

Tamaño de página: 16 bytes

VPN = bits altos

Offset = bits bajos

VA = 21 -> binario 010101

VPN = 01, Offset = 0101

## Page Table

Virtual Page (VPN)	Physical Frame (PFN)
0	3
1	7
2	5
3	2

Traducción:

$$VA = VPN + Offset \rightarrow PFN + Offset = PA$$

## Posibles Problemas

- **Fragmentación interna** (espacios no usados dentro de páginas).
- **Tamaño de tabla de páginas grande** (1 tabla por proceso).

## Soluciones:

- Tablas de páginas multinivel
- TLB (Translation Lookaside Buffer)

## Fórmulas útiles

- Número de páginas:  $\text{size(AS)} / \text{size(page)}$
- Número de marcos:  $\text{size(PM)} / \text{size(frame)}$
- Bits necesarios:
  - VPN:  $\log_2(\text{num\_pages})$
  - Offset:  $\log_2(\text{size(page)})$

## Semana 6: TLB y Mecanismos

Tema clave: Cómo mejorar el rendimiento en la traducción de direcciones virtuales a físicas usando estructuras como la TLB.



### ¿Por qué usar TLB?

- La traducción por paginación puede ser **lenta** debido a:
  - Necesidad de buscar en la tabla de páginas para cada acceso.
  - Dos accesos a memoria por operación (uno para traducir, otro para acceder).
- Solución: **TLB (Translation Lookaside Buffer)** — una caché de traducciones.



### Flujo de traducción con TLB

1. Extraer VPN del VA.
2. Buscar en la TLB:
  - a. Si hay HIT \* obtener PFN y traducir rápidamente.
  - b. Si hay MISS \* consultar la tabla de páginas.
3. Si la página es válida, insertar en la TLB y reintentar.

Una buena TLB reduce drásticamente los accesos a memoria.



## Problemas comunes y soluciones

Problema	Solución
TLB Miss frecuente	Aumentar la localidad (espacial o temporal)
TLB pequeña	Usar <b>páginas más grandes</b> o múltiples tamaños

Overhead en memoria por PTs	Emplear <b>tablas multinivel</b> o <b>invertidas</b>
-----------------------------	--

## Tamaño de Páginas

- **4KB** es estándar, pero se pueden usar **16KB, 4MB**, etc.
- Ventaja: menos entradas en la tabla de páginas.
- Desventaja: **fragmentación interna** mayor.

## Enfoques híbridos

- **Paginación + Segmentación**
  - Cada segmento (código, pila, heap) tiene su tabla de páginas.
  - Se usan registros base/límite por segmento.

VA: [Seg][VPN]<sub>x</sub>Offset]  
2b 10b 12b

## Tablas de Página Multinivel

- Estructura jerárquica para reducir el uso de memoria.
- Tabla raíz → subtablas → marcos de página.
- Útil para direcciones virtuales de 32 o 64 bits.

## Tablas de Página Invertidas

- Una sola tabla global.
- Cada entrada indica proceso + VPN \* marco.

- Ahorro de espacio, pero búsqueda más compleja.



## Swapping y mecanismos más allá de la RAM

### ¿Qué pasa si no cabe todo en RAM?

- Se usa espacio de **swap** en disco.

Una página puede estar:

- En memoria \*  $P > 1, V > 1$
- En disco \*  $P > 0, V > 1$
- No válida \*  $V > 0$

### Bit de presencia ( **present bit** )

- Indica si la página está en memoria.
- Si no: se lanza un **page fault**.



## Page Faults

1. TLB MISS \* buscar en tabla de páginas.
2. Página no está en RAM \* cargar desde disco.
3. Actualizar TLB y tabla.
4. Reintentar instrucción.



## Importancia de la localidad

Tipo de localidad	Ejemplo
Temporal	Acceder varias veces a <code>a[i]</code>
Espacial	Acceder a <code>a[i+1]</code> , <code>a[i+2]</code> , etc.



## Referencias

- ♦ [09-TLB-2024-1.pdf](#)
- ♦ [10-Advanced\\_Page\\_TablesTLB-2024-1.pdf](#)
- ♦ [11-Swapping\\_Mechanism-2024-1.pdf](#)
- ♦ [vm-beyondphys.pdf](#)
- ♦ [vm-smalltables.pdf](#)
- ♦ [vm-tlbs.pdf](#)

OSTEP: *Operating Systems: Three Easy Pieces*

---

**Resumen:** La TLB es una herramienta crítica para la eficiencia del sistema. Se complementa con estrategias como paginación jerárquica, swapping y segmentación para manejar eficazmente el espacio de direcciones virtuales.

# Semana 7: Políticas de Reemplazo en Memoria Virtual

## ¿Qué problema abordan las políticas de reemplazo?

Cuando no hay suficiente memoria libre, el sistema operativo debe **elegir qué página eliminar de la memoria** para hacer espacio. Esta decisión es crucial y se basa en la **política de reemplazo**.

## Conceptos clave

### Memoria como caché

- La memoria física funciona como una **caché** para las páginas virtuales.
- Se busca **minimizar los fallos de caché (cache misses)** para evitar costosos accesos a disco.

### Fórmula del Tiempo Medio de Acceso (AMAT):

$$AMAT = T_M + P_{Miss} \times T_D$$

- $T_M$  : tiempo de acceso a memoria.
- $T_D$  : tiempo de acceso a disco.
- $P_{Miss}$  : probabilidad de fallo.

## Políticas de reemplazo

Política	Descripción	Ventajas	Desventajas
<b>Óptima (MIN)</b>	Reemplaza la página que se usará más tarde en el futuro.	Menor número de fallos.	No es implementable en la práctica.
<b>FIFO</b> ♦	La primera página que entra es la primera en salir.	Simple de implementar.	Puede eliminar páginas aún en uso.

<b>Random</b>	Reemplaza una página al azar.	Fácil de implementar.	Poco eficiente, depende de la suerte.
<b>LRU</b>	Reemplaza la página menos recientemente usada.	Aprovecha la localidad temporal.	Costoso de implementar exactamente.
<b>Clock</b>	Versión eficiente de LRU usando bits de uso (reference bits).	Buena aproximación a LRU.	Aún requiere cierta complejidad.

## Tipos de fallos de caché

1. **Compulsorio**: primera vez que se accede a una página.
2. **De capacidad**: no hay espacio suficiente en memoria.
3. **De conflicto**: no aplicable en cachés totalmente asociativas como la de SO.

## Casos de prueba

### Sin localidad

- Accesos completamente aleatorios.
- Todas las políticas tienen rendimiento similar.

### 80-20 (Localidad fuerte)

- El 80% de los accesos van al 20% de las páginas.
- **LRU** sobresale, manteniendo las páginas “calientes”.

### Secuencia en bucle

- Acceso a 50 páginas en orden y repetición.
- FIFO y LRU tienen mal desempeño; **Random** puede hacerlo mejor en ciertos casos.

## ⚙️ Implementaciones

### ! Problemas con LRU

- Requiere actualizar estructuras en cada acceso.
- Costoso en sistemas grandes (millones de páginas!).

### 🕒 Algoritmo del reloj (Clock)

- Cada página tiene un **bit de uso**.
- Se recorre como si fuera un reloj, se evitan las páginas usadas recientemente.

Start -> ¿bit=1? -> Sí -> Limpiar bit y avanzar -> ¿bit=1?...  
↓ No  
Reemplazar página

### 💡 Extra: Páginas sucias

- Si una página fue modificada (bit de modificación = 1), **debe escribirse en disco**.
- Evitar reemplazar páginas sucias mejora el rendimiento.

## 🧠 Conclusión

- El sistema ideal se aproxima al óptimo pero con costos razonables.
- Políticas como **Clock** ofrecen un buen balance entre desempeño y complejidad.
- ¡El conocimiento de estas políticas permite entender por qué los programas a veces se ralentizan!

## 📚 Referencias

- [OSTEP - Operating Systems: Three Easy Pieces](#)
- [Operating System Concepts - Silberschatz](#)
- 🌐 [GitHub: ostep-code](#)