Resumen de Sistemas Operativos

Compilado a partir de los archivos provistos.  
Nivel Universitario.

# Tema 1 - Introducción a los Sistemas Operativos

## 1. Definición de Sistema Operativo

Un Sistema Operativo (SO) es un software que actúa como intermediario entre el usuario y el hardware de la computadora. Sus funciones principales son:  
- Gestionar los recursos de hardware.  
- Controlar la ejecución de los procesos.  
- Proveer servicios y abstracciones a los programas de aplicación.  
- Ofrecer una interfaz entre aplicaciones y hardware.

## 2. Perspectivas

Desde el usuario: proporciona comodidad y ocultamiento de la complejidad del hardware.  
Desde el sistema: administra eficientemente los recursos y permite ejecución simultánea.

## 3. Objetivos de los SO

- Comodidad: facilitar el uso del hardware.  
- Eficiencia: aprovechar mejor los recursos.  
- Evolución: permitir la incorporación de nuevas funciones sin afectar las existentes.

## 4. Componentes de un SO

- Kernel: núcleo que gestiona CPU, memoria, procesos, concurrencia y E/S.  
- Shell: interfaz (gráfica, textual o por línea de comandos).  
- Herramientas: compiladores, editores, librerías, etc.

## 5. Servicios de un SO

- Administración del procesador (planificación justa, manejo de prioridades).  
- Administración de memoria (física y virtual, protección de procesos).  
- Administración de almacenamiento y dispositivos (sistema de archivos, ocultar dependencias de HW).  
- Detección y manejo de errores.  
- Interacción con el usuario.  
- Contabilidad y monitoreo del sistema.

## 6. Modos de Ejecución

- Modo Kernel: ejecución privilegiada de rutinas críticas del SO.  
- Modo Usuario: ejecución de programas con acceso restringido al hardware.  
El cambio de modos se produce mediante interrupciones o traps.

## 7. Protección del Sistema

- Memoria: uso de registros base y límite para delimitar accesos.  
- CPU: interrupciones de reloj para evitar monopolización.  
- E/S: instrucciones privilegiadas solo en modo kernel.

## 8. System Calls

Las llamadas al sistema son la interfaz entre procesos de usuario y el SO. Ejemplos: fork, execve, read, write. Se clasifican en:  
- Control de procesos  
- Manejo de archivos  
- Manejo de dispositivos  
- Información del sistema  
- Comunicaciones

## 9. Arquitectura y Conceptos de Hardware

Elementos: CPU, memoria principal, dispositivos de E/S, bus del sistema.  
Ciclo de instrucción: fetch + execute.  
Interrupciones: permiten coordinación eficiente entre CPU y dispositivos.  
Tipos: enmascarables y no enmascarables, con distintos niveles de prioridad.

## 10. Evolución de los SO

- Procesamiento en serie: sin SO, interacción directa con la máquina.  
- Sistemas Batch: monitor residente, procesamiento por lotes.  
- Multiprogramación: concurrencia de tareas y aprovechamiento de CPU.  
- Tiempo compartido: multiprogramación interactiva con múltiples usuarios.  
- Evolución histórica: desde IBM 1401 hasta sistemas modernos como UNIX, Windows y Linux.

# Tema 2 - Procesos

## 1. Definición y Modelo de Proceso

Un proceso es un programa en ejecución. Diferencias con un programa:  
- Programa: estático, sin contador de programa.  
- Proceso: dinámico, tiene ciclo de vida, contador de programa y contexto.

## 2. Componentes de un Proceso

- Sección de código (texto).  
- Sección de datos (variables globales).  
- Stack(s).  
El Process Control Block (PCB) almacena información del proceso: PID, registros, estado, prioridad, etc.

## 3. Contexto y Cambio de Contexto

El contexto incluye toda la información necesaria para reanudar la ejecución. El cambio de contexto ocurre cuando la CPU alterna entre procesos.

## 4. Estados de un Proceso

- Nuevo, Listo, Ejecución, Espera, Terminado, Zombie.  
Los procesos alternan ráfagas de CPU y E/S.  
El SO usa colas y planificadores para administrar la ejecución.

## 5. Planificación de Procesos

Existen varios schedulers:  
- Largo plazo: controla la multiprogramación.  
- Corto plazo: elige qué proceso obtiene la CPU.  
- Mediano plazo: controla swapping.  
Algoritmos: FCFS, SJF, Round Robin, Prioridades, Colas multinivel, SRTF.

## 6. Políticas vs Mecanismos

El kernel implementa mecanismos de planificación; los usuarios/procesos definen políticas, por ejemplo, ajustando prioridades.

## 7. Creación y Terminación de Procesos

- Creación: en UNIX, fork() + execve(); en Windows, CreateProcess().  
- Terminación: exit() para finalizar, wait() para que el padre espere, kill() para forzar terminación.

## 8. Relación Padre-Hijo

Los procesos se organizan en árbol. El hijo puede ejecutar en paralelo o esperar al padre. En UNIX, el hijo hereda el espacio de direcciones del padre; en Windows, recibe uno nuevo.

## 9. Dispatcher

El dispatcher realiza el cambio de contexto, cambio de modo y salta a la instrucción a ejecutar. Es fundamental en la planificación de corto plazo.

# Ampliación con Bibliografía (Stallings, Silberschatz, Tanenbaum)

## 1. Procesos y PCB

Según Silberschatz, Galvin y Gagne, un proceso no es solo un programa en memoria, sino que incluye su contador de programa, registros, estado, memoria asignada y recursos. El Process Control Block (PCB) guarda toda esta información: PID, estado, contador de programa, registros CPU, prioridad, información de E/S y datos contables.

Tanenbaum enfatiza la importancia de los hilos (threads) como unidad ligera de ejecución, permitiendo concurrencia dentro de un proceso.

## 2. Planificación de CPU

Silberschatz clasifica algoritmos de planificación:  
- FCFS (First Come First Served): simple, pero puede generar largos tiempos de espera.  
- SJF (Shortest Job First) y SRTF: minimizan espera promedio, pero requieren estimación de ráfagas.  
- Round Robin (RR): ideal para sistemas interactivos, con quantum ajustable.  
- Por Prioridades: riesgo de inanición, mitigado con aging.

Stallings analiza el impacto del overhead del dispatcher, cambios de contexto y su efecto en la eficiencia. También cubre scheduling en multiprocesadores y sistemas en tiempo real.

## 3. Gestión de Memoria

Silberschatz distingue entre memoria física y virtual, y explica técnicas de paginación y segmentación.  
- Paginación: tablas de páginas, fallos de página y políticas de reemplazo (LRU, FIFO).  
- Segmentación: divisiones lógicas de código, datos y pila, con protección y compartición.

Tanenbaum explica mecanismos de protección (bits de acceso, límites, permisos) y estrategias como prefetching y swapping. Stallings detalla la importancia de la TLB, la jerarquía de memoria y cómo minimizar la fragmentación.

## 4. Entrada/Salida y Sistemas de Archivos

Los tres autores tratan:  
- Dispositivos de E/S: controladores, DMA, interrupciones, buffers.  
- Scheduling de discos: algoritmos SSTF, SCAN, C-SCAN.  
- Sistemas de archivos: estructuras como inodos, permisos, journaling y cachés de archivos.

## 5. Concurrencia y Deadlocks

Temas centrales:  
- Exclusión mutua: semáforos, mutex, monitores.  
- Problemas: race conditions, deadlocks, livelocks, inanición.  
- Deadlocks: condiciones necesarias (exclusión mutua, espera circular, no-preempción, retención y espera).  
- Métodos: prevención, detección, evitación (algoritmo del banquero), recuperación.

## 6. System Calls y Modos de Ejecución

Las system calls son la interfaz de los programas de usuario con el SO. El cambio de modo usuario a modo kernel ocurre mediante traps e interrupciones.  
Coste: cada system call implica cambio de contexto y protección de recursos.

## 7. Seguridad y Protección

Silberschatz y Stallings destacan mecanismos de control de acceso como ACLs y capacidades. Se busca garantizar integridad y confidencialidad con políticas de privilegios mínimos.  
También se incluyen técnicas contra ataques como buffer overflow.