Sistemas Operativos

Dr. Daniel Ochoa Donoso

Presentaciones

¿Quién soy yo?

- Daniel Ochoa Donoso PhD.
  - Ugent, Bélgica
  - Director Centro de Visión y Robotica
  - Director de la Maestria en Ciencias Computación
  - Miembro de N comites y grupos.
- Investigación
  - Analisis de imagenes Hyperspectrales ←-BigData?, y “robótica” ←- SO
- La lectura que le doy a mis ayudantes:
  - El mensaje a García

Syllabus

Textos guía

Sistemas Operativos, Silberschatz, Galvin

Sistemas Operativos, Andrew Tanenbaum

Sistemas Operativos, William Stallings

Contenido del Curso

- Introducción
- Estructuras del sistema operativo
- Procesos
- Hilos
- Planificación del CPU
- Sincronización entre procesos
- Interbloqueo
- Administración de memoria principal
- Administración de memoria virtual
- Sistemas de archivos
- Otros: Virtualización, Tiempo Real, Sistemas Embebidos, etc.

Políticas del Curso

Notas

- Parcial
  - Examen: 60%
  - Proyecto: 25%
  - Otros (lecciones, presentaciones): 15%
- Final
  - Examen: 60%
  - Proyecto: 25%
  - Otros (lecciones, presentaciones): 15%
- Mejoramiento
  - Examen: 100%

Capítulo 1
Conceptos y Estructura de un Sistema Operativo

Agenda

- ¿Qué es un sistema operativo?
  - Conceptos y funciones
- ¿Por qué estudiar Sistemas Operativos?
- Evolución de los sistemas operativos
- Kernel y modo dual de operaciones
- Organización de un sistema operativo
  - Componentes y Arquitectura

¿Qué es un sistema operativo?

El programa fundamental de todos los programas de sistemas es el sistema operativo, que controla todos los recursos de la computadora y proporciona la base sobre la cual pueden implementarse los programas de aplicación.

A. Tanenbaum

tl;dr

- Sistema operativo:
  - Interfaz entre aplicaciones y hardware
  - Administra los recursos del sistema
  - Programa que controla la ejecución de los programas de aplicación

Funciones de un Sistema Operativo

- Desarrollo de programas
- Ejecución de programas
- Acceso a dispositivos de E/S
- Acceso controlado a archivos
- Acceso al sistema
- Detección de errores y respuestas
  - Errores de software, de hardware, y acciones no permitidas
- Estadísticas
  - Uso de recursos, monitoreo

¿Por qué estudiar Sistemas Operativos?

- Para aprender cómo funcionan las computadoras
- Para aprender a manejar complejidad a través de abstracciones adecuadas: CPU infinito, memoria infinita, archivos, semáforos
- Para aprender sobre diseño de “sistemas”:
  - Rendimiento vs. simplicidad, HW vs. SW, etc.
- Necesario para desarrollar sistemas:
  - Eficientes
  - Tolerantes a fallos
  - Escalables
- Ayuda a encontrar y evitar “bugs”
- “Sistemas Operativos” entre las 5 materias más importantes de CS
  - ¿Cuál es la primera?
- Diferencia a un Computer Scientist de un Programador

¿Dónde hay sistemas operativos? ¡En todos lados!

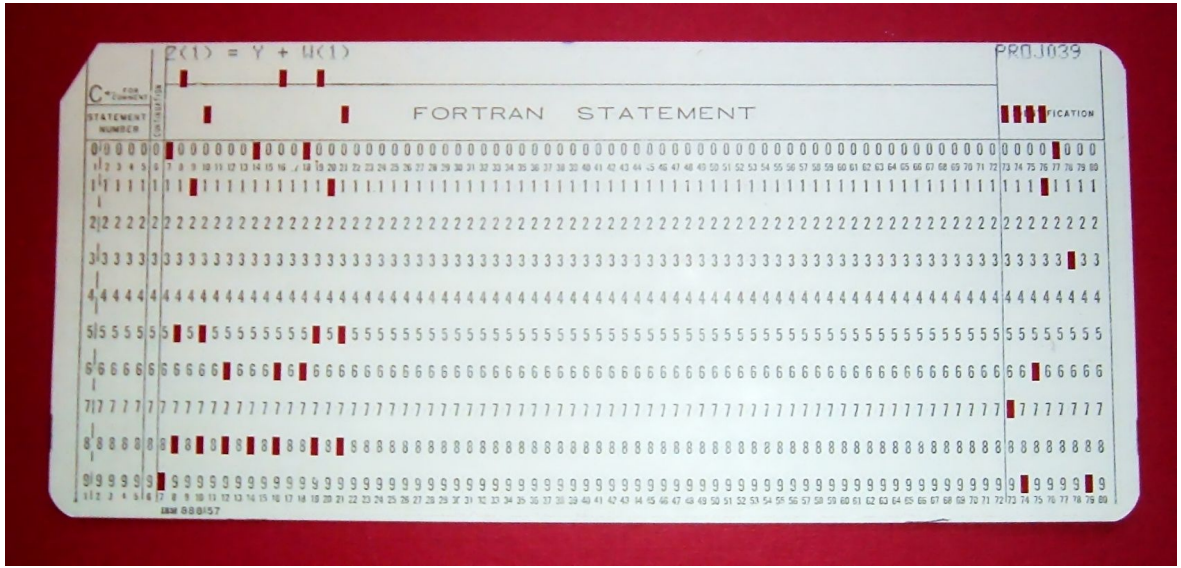
Evolución de los Sistemas Operativos

Algunos hitos

- Sistemas por lotes
- Sistemas con multi-programación
- Sistemas de tiempo compartido

Sistemas por lotes (batch)

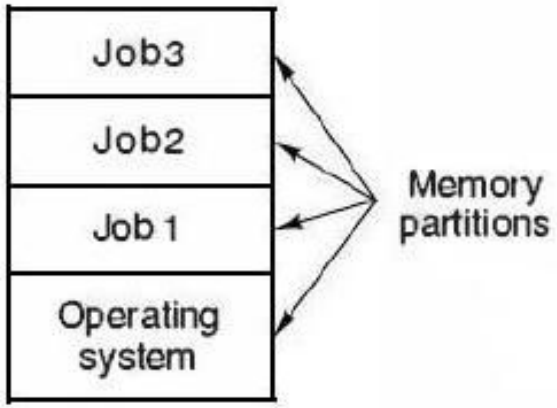
- Leían “streams” de trabajos (‘jobs”) de una lectora de tarjetas perforadas
- Salida impresa
- Sin interacción con el usuario durante ejecución de trabajo



¿Problemas?

Multiprogramación

- Diferentes trabajos se encuentran en la memoria principal
- Memoria se divide en varias partes, con un trabajo distinto en cada una
- **Inicialmente, sin paralelismo ni concurrencia**



A multiprogramming system with three jobs in memory.

¿Qué ventaja(s) traía la multiprogramación sobre el procesamiento por lotes?

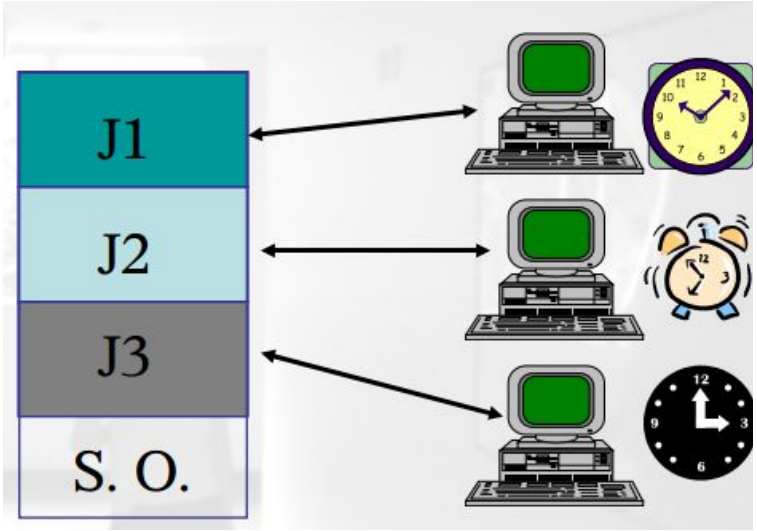
Spooling

- Simultaneous Peripherical Operation On Line
- Permitió acelerar E/S a través del uso del disco duro
  - Leer tarjetas perforadas a disco (y luego a memoria)
  - Imprimir a disco (y luego directamente de disco a la impresora)

- ¿Qué hacemos ahora cuándo el CPU pasa “tanto” tiempo libre?

Tiempo compartido

- Es una variante del concepto de multiprogramación
- Proveer a cada usuario de una terminal en línea
- Ahora, los usuarios podían compartir un CPU
  - ¿Cómo?
- Pseudo-paralelismo → Concurrencia



Tipos de sistemas operativos (terminología)

- multiusuario (multi-user):
  - permite que dos o más usuarios ejecuten programas al mismo tiempo
- multiprocesamiento (multiprocessing)
  - soporta el poder ejecutar un programa en más de un CPU
- multitareas (multitasking)
  - permite que más de un programa corra concurrentemente
- multihilos (multithreading)
  - permite que diferentes partes de un solo programa se ejecuten concurrentemente
- tiempo real (real time)
  - reponde a la entrada de forma instantánea

¿Qué desafíos introduce la multiprogramación?

¿Y qué es el multitasking?

Es tiempo compartido para sistemas modernos.

El timesharing (o multitasking) es una extensión multiprogramación, en la cual el CPU se cambia a otro trabajo y los usuarios pueden interactuar con cada trabajo mientras este se ejecuta, permitiendo tener una **computación interactiva**.

Nuevos desafíos (al tener multitasking)

- Tiempos de respuesta deben ser cortos (< 1 segundo)
  - Optimizamos diferentes métricas que en sistemas por lotes
- ¿Cómo determinamos a qué proceso le toca el turno para ejecutarse?
  - **CPU scheduling**
- ¿Y si todos los procesos no entran en memoria?
  - **Swapping**
- ¿Y si un proceso es muy grande y no queremos tenerlo TODO en memoria?
  - **Memoria virtual**

Kernel y Modo Dual de Operaciones

Kernel

- Porción del sistema operativo que se carga siempre en memoria
- Contiene las funciones más usadas
- También llamado **núcleo** del sistema operativo
- Concepto similar: **monitor**
  - Aquella porción del sistema operativo que está siempre en memoria

¿Cómo puede el S.O. tener el control del sistema?

Por ejemplo,

¿Cómo puede el sistema operativo implementar pseudo-paralelismo (tiempo compartido) en un sistema uniprocesador?

Operaciones del sistema operativo

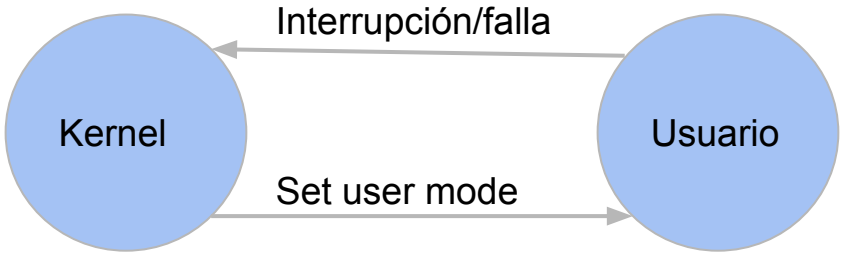
- Dirigido por interrupciones de hardware
- Un error de SW o un pedido, genera una **excepción** o **trampa** (**trap**)
  - División por cero
  - Pedido de un servicio al S.O.
  - Lazos infinitos
  - Procesos que intentan modificar a otros procesos o al sistema operativo
- Operaciones en **modo-dual** permiten al S.O. protegerse a sí mismo y a otros componentes del sistema
  - **Modo de usuario** y **modo de kernel**
  - **Bit de modo** proporcionado por HW
    - Permite saber cuándo el sistema está ejecutando código de usuario o de kernel
    - Algunas instrucciones son **privilegiadas**, solo ejecutables en modo de kernel
    - **Llamadas al sistema** cambian a modo de kernel, retorno de la llamada lo retorna a modo de usuario

Modo dual de operaciones

- Programa de usuario
  - Uso indebido de recursos
  - Atacar otros programas
- Se necesita protección contra programas de usuario no confiables
- Existen estructuras de hardware para diferencias entre al menos dos modos de operación
  - User mode
    - Ejecución de programas de usuario
    - No confiable
    - No se permite acceso directo/completo a recursos de hardware
  - Kernel mode (monitor mode)
    - Ejecución del sistema operativo
    - Acceso completo/directo a recursos de hardware

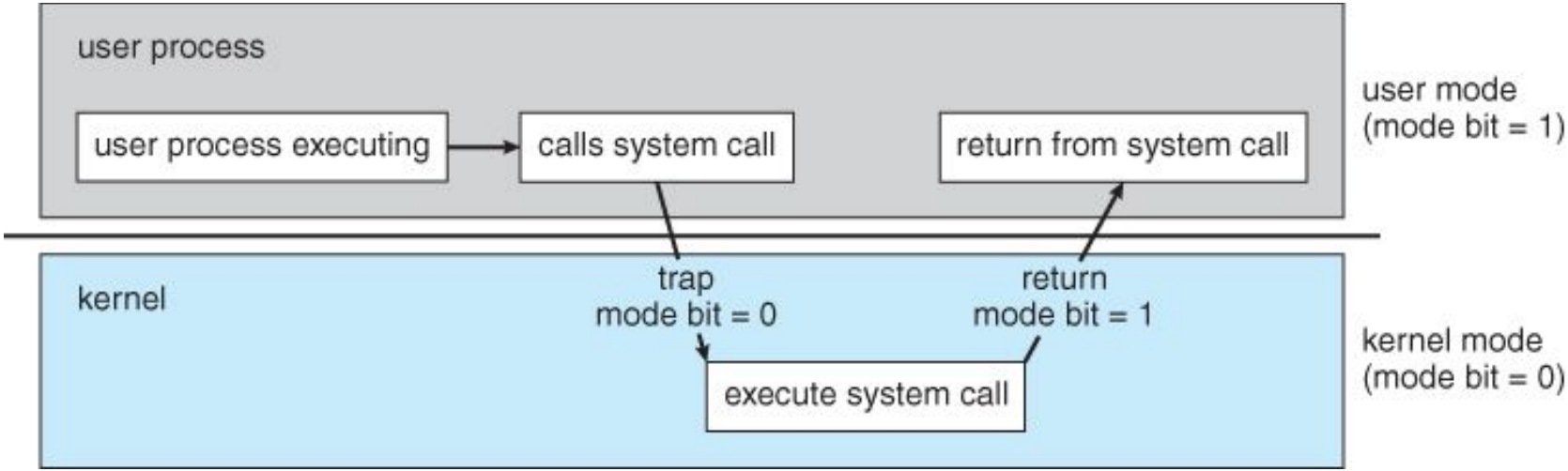
Modo dual de operaciones

- Mode bit Añadido al hardware de la computadora para indicar el modo actual de operación : monitor (0) o usuario (1)
- En interrupciones o fallas, el hardware cambia a modo monitor

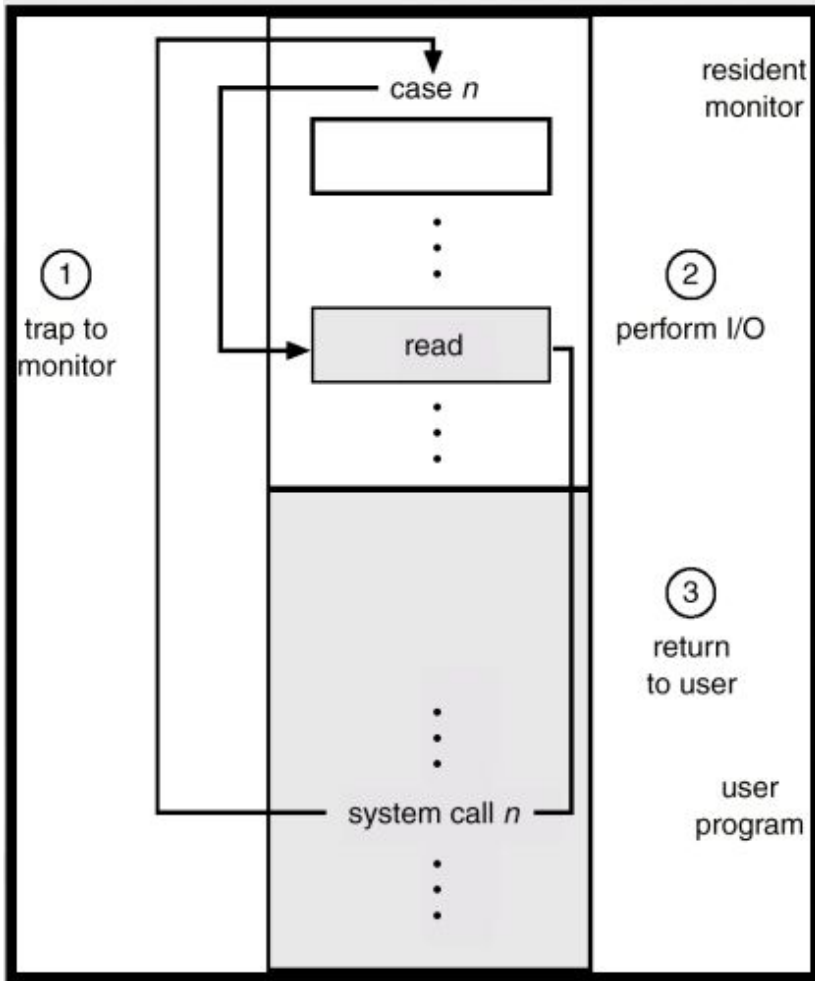


- *Instrucciones privilegiadas se pueden ejecutar solamente en kernel mode*

Modo dual de operaciones



Ejemplo:
Realizar una operación de E/S



¿Trampas, excepciones, interrupciones, fallas?

- Hacen que CPU cambie a modo de kernel y le pase el control al S.O.

- **Fault/Falla:** Por ejemplo, un page fault o alguna excepción ocasionada por una instrucción que se está ejecutando
- **Interrupción:** Por ejemplo, una interrupción del teclado; una operación de E/S que termina
- **Trampa:** Por ejemplo, una llamada al sistema

¿Solamente dos modos de operaciones?

- No, pero con dos modos podemos entender esta materia :-)

- La arquitectura Intel x86 tiene 4 anillos (rings)
  - Sin virtualización:
    - 3: modo de usuario
    - 0: modo de kernel
  - Con virtualización:
    - 0: Hypervisor
    - 1/2: Guest OS
    - 3: Aplicaciones

Ejemplos: Uso del modo dual de operaciones

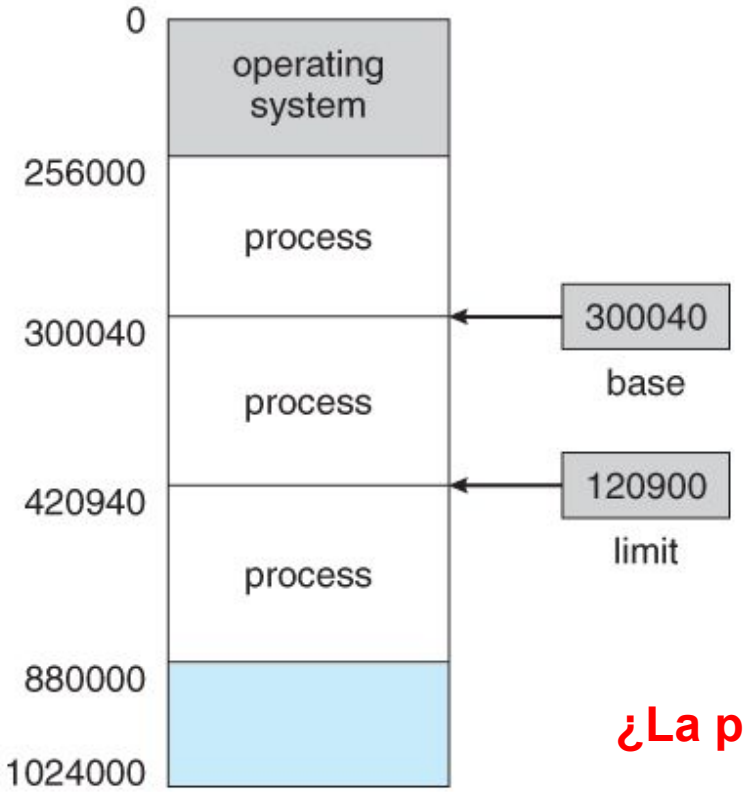
1. Protección de memoria
2. Protección del CPU

Ejemplo 1: Protección de memoria

Protección de memoria

- Objetivo:
  - Usuario no puede usar cantidades de memoria arbitrarias
  - Usuario no debe acceder a posiciones de memoria
    - Pertenecientes a otros usuarios o
    - Pertenecientes al sistema operativo

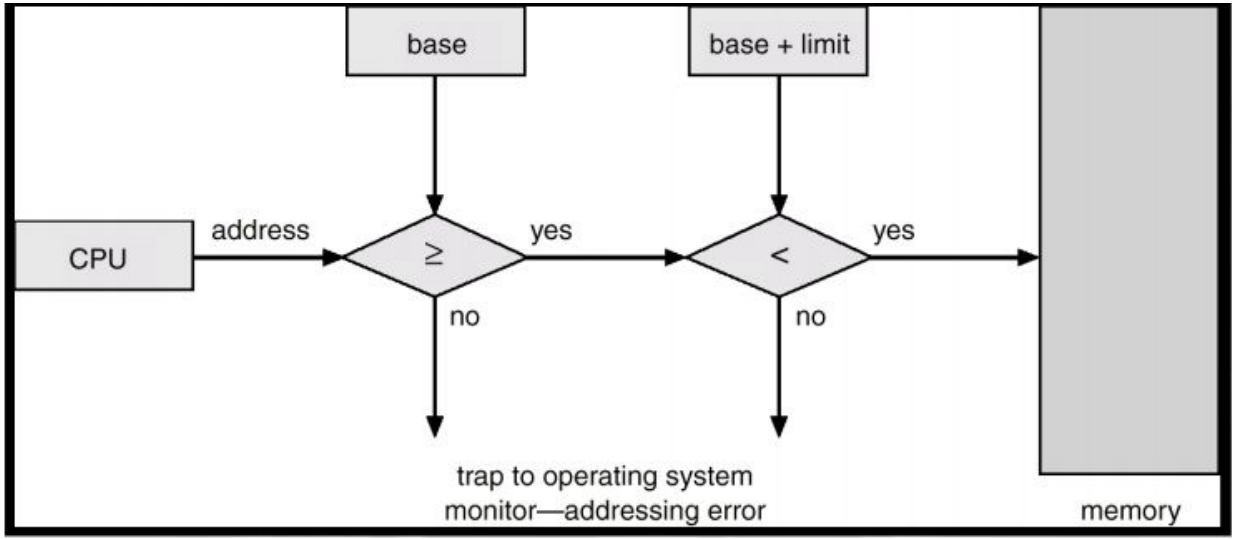
¿Cómo implementar la protección de memoria?



- Acceso indirecto
  - Uso de direcciones virtuales (traducción)
- Uso de registros para control de rango legal de direcciones
  - Registro **Base**, la dirección de memoria mínima
  - Registro **Límite**, tamaño del rango
  - Memoria fuera del rango es protegida
    - Cada dirección es verificada contra los registros
    - Trampa al OS si cae fuera del rango

¿La protección se la hace en el HW o en el SW?

Protección en Hardware



- En modo de kernel, S.O. tiene acceso ilimitado a memoria
- Instrucciones para manejo de base y límite son privilegiadas

Ejemplo 2: Protección del CPU

Protección del CPU

- Para implementar tiempo compartido (multitasking), dividimos el tiempo en pequeños “slices” y alternamos la ejecución de un proceso

- ¿Cómo podemos evitar que un usuario se apodere del CPU?
  - (Análisis se facilita si asumimos que tenemos un solo procesador)

Protección del CPU

- Un usuario no debe poder adueñarse del CPU
- Timer
  - HW interrumpe la computadora después de un periodo específico de tiempo para asegurar que el sistema operativo mantiene el control
  - Timer se decrementa cada tick del reloj
  - Cuando alcanza cero → Interrupción
- Usado para implementar time sharing
- Load-timer es una instrucción privilegiada

Organización del Sistema Operativo

Organización del Sistema Operativo

- Componentes
- Arquitectura

Componentes

- Administración de:
  - Procesos
  - Memoria
  - E/S
  - Archivos y almacenamiento
  - Redes
  - ...

Administración de procesos

- Un proceso es un programa en ejecución
  - Necesita recursos
- Responsabilidades del SO:
  - Crear y eliminar de procesos
  - Suspender y reanudar procesos
  - Sincronizar y manejar la comunicación entre procesos (IPC)

Administración de memoria

- Memoria
  - Un arreglo inmenso de bytes
  - Datos compartidos entre CPU y E/S
- Responsabilidades del SO:
  - Asignar y recuperar memoria
  - Mantener registro de partes ocupadas de la memoria
  - Utilización eficiente

Administración de E/S

- E/S para interactuar con el usuario
  - Terminal de una consola
  - Redes
  - ....
- Responsabilidades del SO:
  - Sistema de buffers
  - Interfase para drivers de dispositivo
  - Drivers para dispositivos específicos de hardware
  - . . .

Administración del Almacenamiento

- Archivo
  - Colección de información definida por su propietario
  - Datos y programas son almacenados como archivos
- Responsabilidades del SO:
  - Archivos:
    - Manipulación de archivos y directorios
    - Mapear archivos en almacenamiento secundario
  - Disco
    - Planificación del disco

Arquitectura

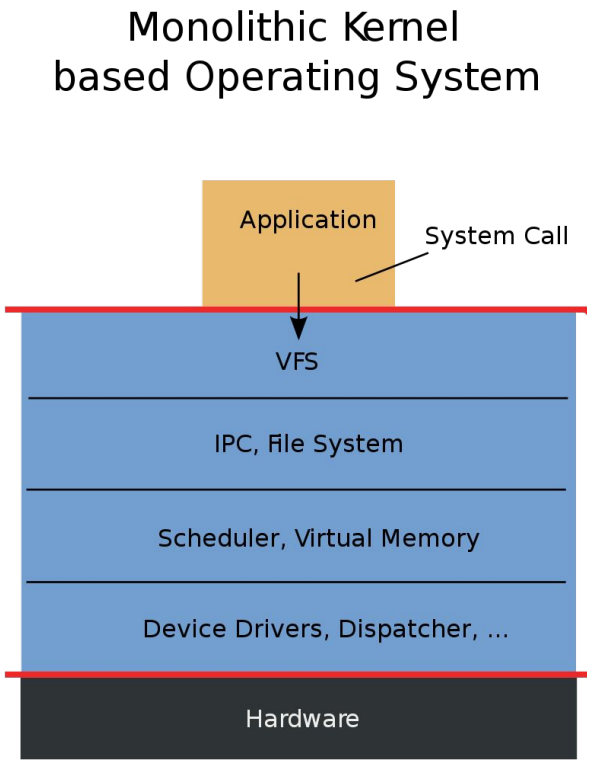
- Monolítico
- Microkernel
- Capas
- Modular
- Virtual machine

- Otros: Exokernel, Nanokernels, etc.

Sistemas monolíticos

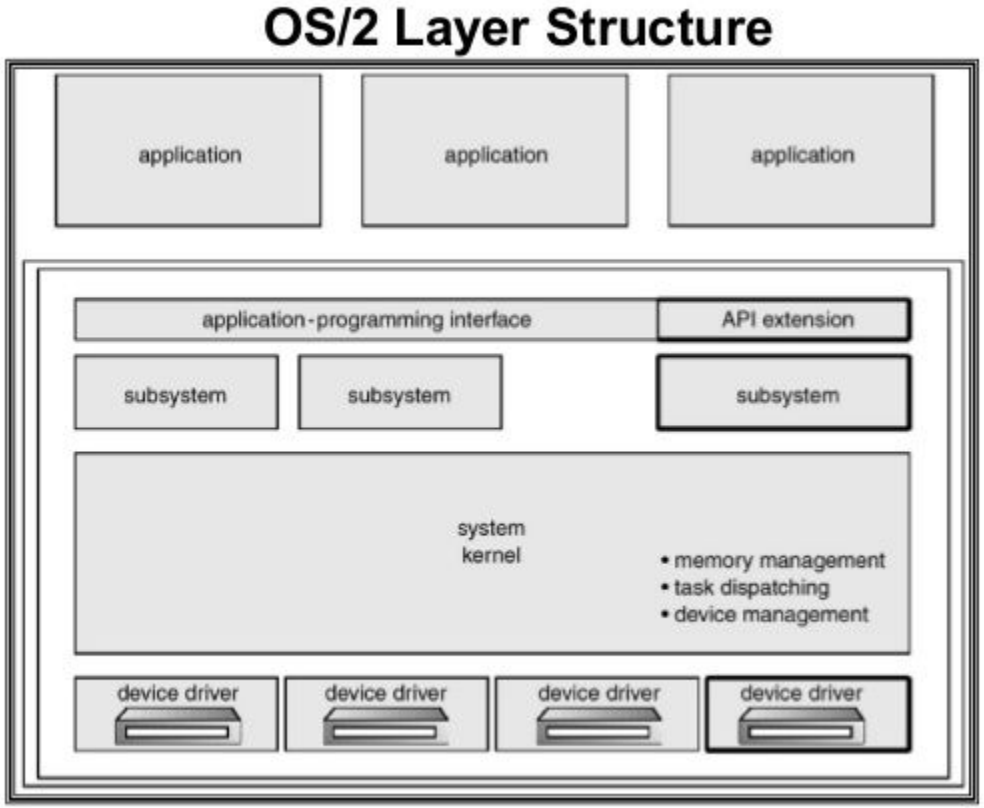
- No se tiene una estructura
- El sistema es una colección de procedimientos
- No existe encapsulamiento de información (módulos, packages, clases)
- Se tiene un procedimiento Principal y un conjunto de procedimientos para servicios varios
- Ejemplos: MS-DOS
  - Primeras versiones de Unix tenían un kernel monolítico

¿Desventajas?



Sistemas por capas

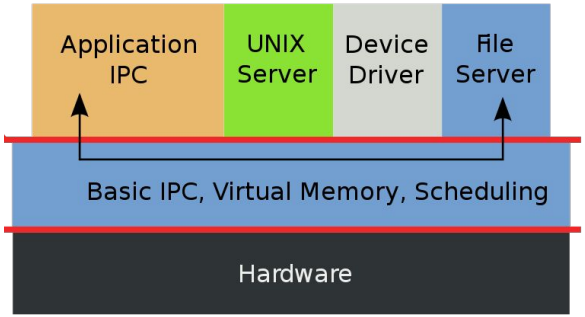
- Jerarquía de capas
- Capa *n*+1 usa (exclusivamente) servicios provistos por capa *n*
- Optimización: capa *n*+1 puede acceder a capas *n-k* directamente
- Fácil de extender



Microkernel

- Mueve la mayor parte de la funcionalidad del kernel al espacio de usuario
  - Comunicación → paso de mensajes
- Funciones esenciales (van en el kernel)
  - Manejo de procesos
  - Espacio de direcciones
  - Seguridad
  - Interprocess communication (IPC)
  - Manejo de hilos
  - Planificación
- Beneficios
  - Más seguro
  - Más confiable
  - Extensible

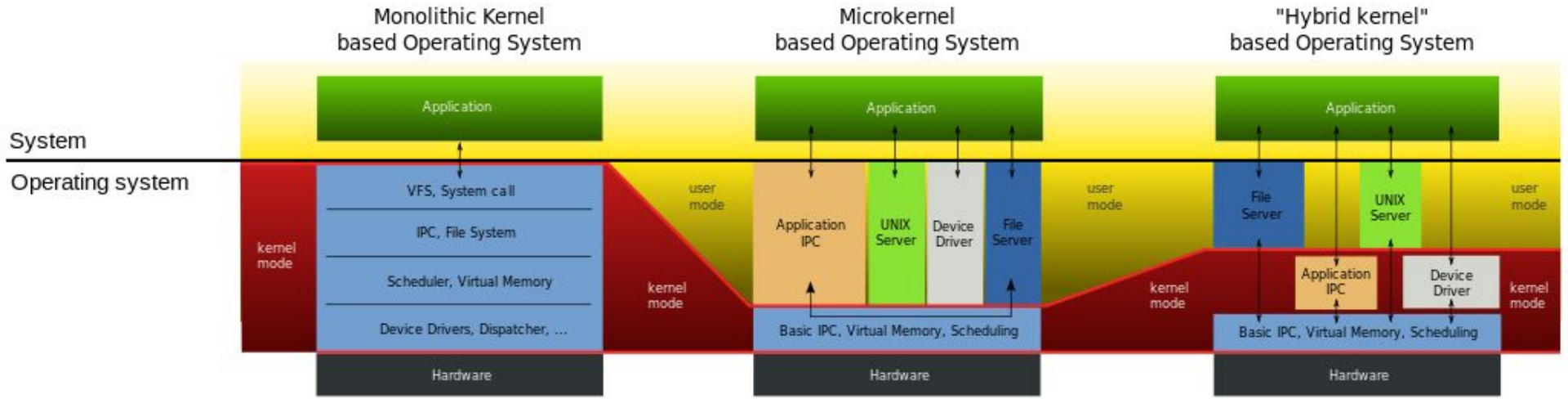
Microkernel
based Operating System



Sistemas modulares (híbridos)

- El kernel está conformado por los sistemas principales
- Servicios adicionales son módulos cargados de forma dinámica (filesytems)
- Similar a capas, pero la comunicación es entre cualquier módulo
- Similar a microkernel, pero no hay paso de mensajes

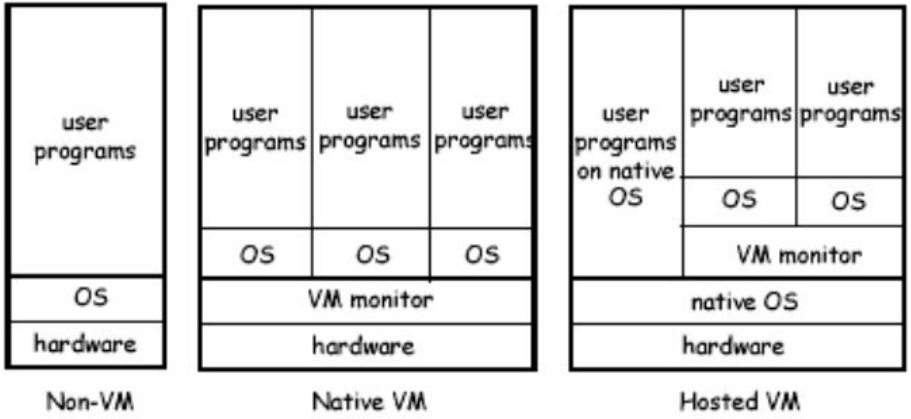
Monolítico vs. microkernel vs. híbridos



- Monolíticos: MSDOS y Windows (pre-NT)
  - Linux y Solaris son monolíticos con módulos que se pueden cargar dinámicamente
- Híbridos: OS X (XNU) y Windows (desde NT, Hyru)
- Microkernel: Mach y L4

Máquinas virtuales

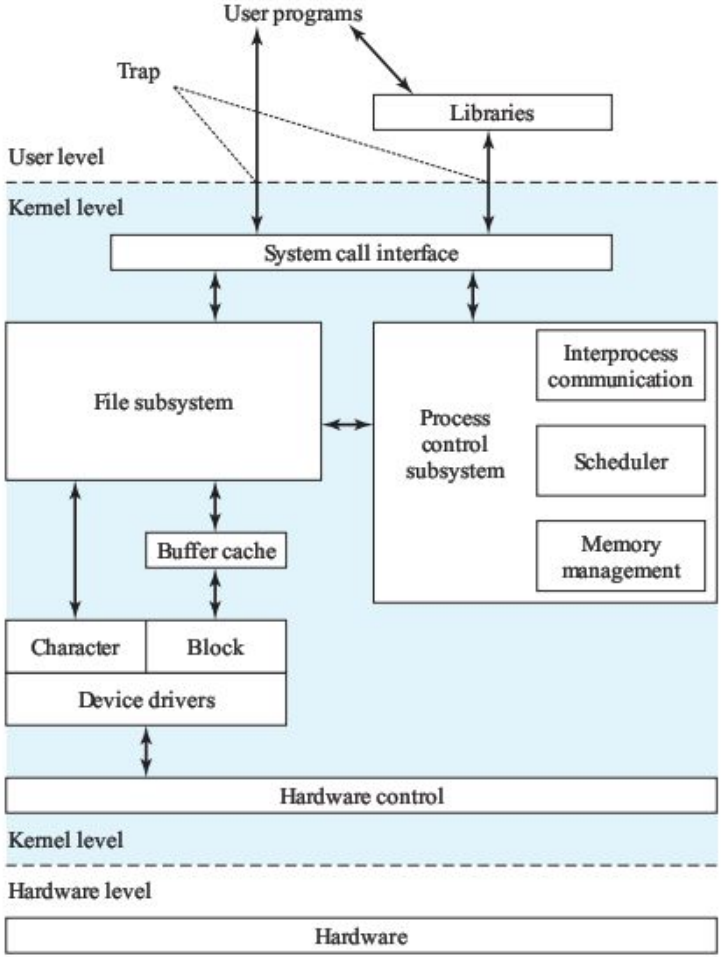
- Hardware es simulado en software
- Todos los recursos se manejan como recursos virtuales
- Sistema operativo individual corriendo con recursos virtuales
- Provee una interfaz que esconde el hardware
- Los recursos físicos son compartidos para crear las máquinas virtuales



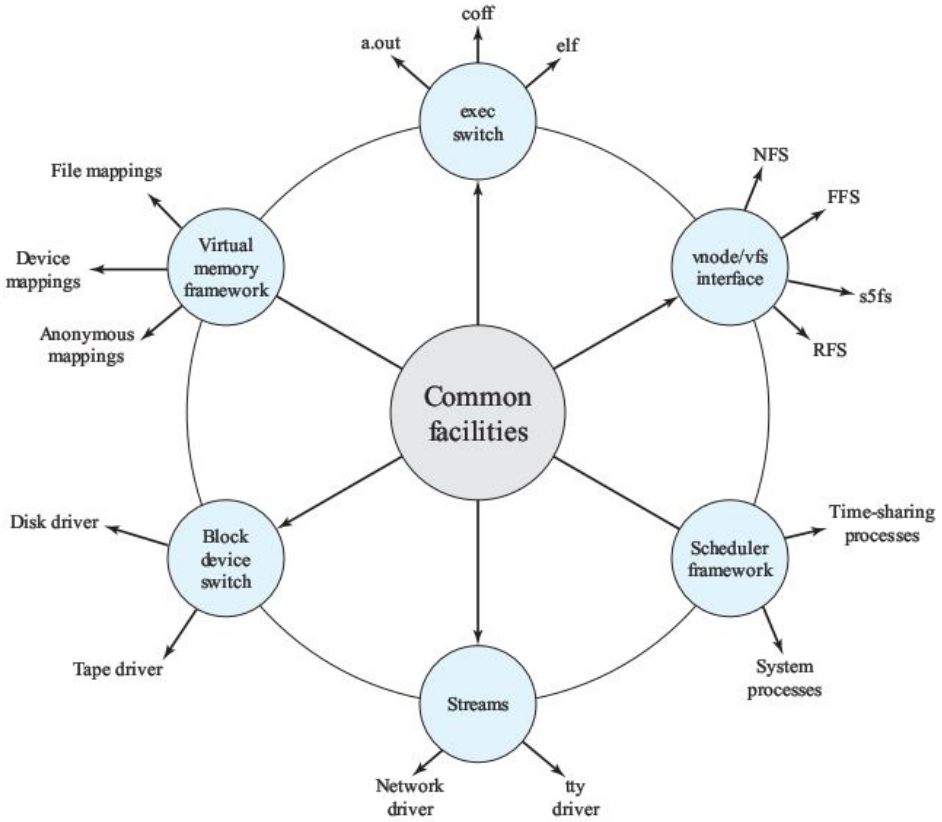
Máquinas virtuales

- La MV provee protección completa
  - Cada mv está aislada de las demás
  - Sin embargo, esto no permite compartir recursos
- Estupendo para investigación
  - No se afecta el funcionamiento normal del sistema operativo del computador
- Difícil de implementar
  - Copia exacta del hardware

Unix



1986



1996 (Solaris)

Trivia: Evolución de sistemas tipo Unix