Sistemas Operativos

Curso 2018

Estructura de los sistemas operativos

Agenda

- Componentes de un sistema operativo
- Servicios del sistema operativo (*system services*).
- Llamados a sistema (*system calls*).
- Estructura del sistema.
- Máquinas virtuales.

Componentes de un sistema operativo

- Por su complejidad un sistema operativo debe ser, en su diseño, modularizado en varios componentes:
  - Administración de procesos.
  - Administración de memoria.
  - Subsistema de Entrada/Salida.
  - Administración de almacenamiento secundario.
  - Subsistema de archivos.
  - Subsistema de red.
  - Sistema de protección.

Servicios del SO

- El sistema brindará un entorno de ejecución de programas dónde se dispondrá de un conjunto de servicios que serán accesible mediante una interfase bien definida.
- Servicios básicos que debe brindar un sistema operativo:
  - Ejecución de programas y administración de procesos.
  - Operaciones de Entrada/Salida.
  - Manipulación de sistemas de archivos.
  - Comunicación entre procesos.
  - Detección y manipulación de errores (excepciones).

Ejecución de programas y administración de procesos

- El sistema debe ser capaz de cargar un programa a memoria y ejecutarlo. Es decir, crear un proceso.
- Cada proceso cuenta con un contador de programa (*PC, program counter*) que determina la próxima instrucción de código a ejecutar.
- El proceso necesita de ciertos recursos (CPU, memoria, archivos y dispositivos de E/S) para realizar su tarea.
- El sistema operativo es responsable de las siguientes tareas:
  - Mantener que partes de la memoria están siendo utilizadas y por quién.
  - Decidir cuales procesos serán cargados a memoria cuando exista espacio de memoria disponible.
  - Asignar y quitar espacio de memoria según sea necesario.

Ejecución de programas y administración de procesos

- El sistema albergará muchos procesos compitiendo por los recursos y será el responsable de proveer de medios o servicios para que realicen su tarea:
  - Crear y destruir procesos.
  - Suspensión y reanudación de procesos.
  - Proveer mecanismos para la cooperación (sincronización) y comunicación entre los procesos.
  - Proveer mecanismos para prevenir la generación de *dead-locks* o lograr salir de ellos (opcional).
- El proceso deberá poder en algún momento finalizar su ejecución, ya sea de forma normal o anormal (indicando un error).

Operaciones de entrada/salida

- Un programa en ejecución necesitará de operaciones de Entrada/Salida para acceder a un archivo o dispositivo.
- Por eficiencia y protección los usuarios no accederán directamente al dispositivo.
- El sistema operativo deberá encapsular y ocultar las características específicas de los diferentes dispositivos de almacenamiento y ofrecer mecanismos de acceso comunes para todos los medios de almacenamiento.
- Para ello proveerá de:
  - Un conjunto de servicios que provean la interfase con el subsistema e implementen técnicas de *cache*, *buffering* y *spooling*.
  - Una interfase cliente con el sistema operativo para los manejadores de dispositivos o *device drivers* que permitirá interactuar (mediante cargas dinámicas o no) con cualquier modelo de dispositivo.
  - *Device drivers* específicos.
  - Montaje y desmontaje (*Mount/Dismount*) de dispositivo.

Manipulación del sistema de archivos

- Se deberá proveer acceso al sistema de archivos.
- El sistema operativo es responsable de las siguientes actividades:
  - Administrar el espacio libre.
  - Asignación del lugar de la información.
  - Algoritmos de planificación de disco.
- Proporciona una vista uniforme de todas las formas de almacenamiento en los diferentes dispositivos implementando el concepto de archivo como una colección arbitraria de *bytes* u otras clases o organizaciones más sofisticadas.
- Implementará los métodos de:
  - Abrir, cerrar, extender y borrar archivos
  - Leer, escribir archivos
  - Crear y borrar directorios

Comunicación entre procesos

- Es deseable que los procesos puedan comunicarse.
- Se deberá proveer mecanismos de comunicación entre ellos ya sea que estén en el mismo computador (a través de memoria compartida), o en diferentes computadores (a través de transferencias de paquetes de red entre los sistemas operativos involucrados).
- También debe permitir la sincronización de procesos que trabajan en conjunto.
- En el caso de sistemas remotos se generaliza el concepto de dispositivo virtual implementando un manejador (*driver*) que encapsula el acceso a estos dispositivos.

Detección y manipulación de errores

- El sistema deberá tomar decisiones adecuadas ante eventuales errores que ocurran y proveer una interfaz para manejarlos.
- Ejemplos:
  - Fallo en un dispositivo de memoria.
  - Fallo en la fuente de energía.
  - Fallo en un programa.

Otros servicios del SO

- Otros servicios de propósito general que deberá brindar el sistema operativo son:
  - Asignación de recursos.
  - Contabilización.
  - Protección, manejo de usuarios y permisos.
- Una vez que están definidos los servicios que brindará el sistema operativo, se puede empezar a desarrollar la estructura del sistema.

Asignación de recursos

- Cuando hay varios procesos compitiendo por los recursos es importante tener algoritmos eficientes de asignación
- Cada recurso tiene necesidades y por lo tanto algoritmos diferentes

Contabilización

- Se quiere llevar un registro de que usuarios usan que recursos y en que cantidad
- Permiten dimensionar y evaluar el estado del sistema

Protección, manejo de usuarios y permisos

- En un sistema multiusuario donde se ejecutan procesos en forma concurrente se deben tomar medidas que garanticen la ausencia de interferencia entre ellos.
- Por protección nos referimos a los mecanismos por los que se controla al acceso de los procesos a los recursos.
- El mecanismo debe incorporar la posibilidad de definir reglas de acceso y asegurar su verificación en toda ocasión que corresponda.

Llamados al sistema

- Los **llamados al sistema** (*system calls*) son una interfaz, provista por el núcleo, para que los procesos de usuarios accedan a los diferentes **servicios** que brinda el sistema operativo.
- Al principio los *system calls* estaban desarrollados en lenguaje de la arquitectura de la máquina.
- En los sistemas modernos están programados en lenguajes de programación de alto nivel como C o C++.
- Los servicios son invocados por los procesos en modo usuario, cuando ejecutan lo hacen en modo monitor, y al retornar vuelven al modo usuario.
- Típicamente a los *system calls* se les asocia un número que los identifica (en Linux son aproximadamente 350).

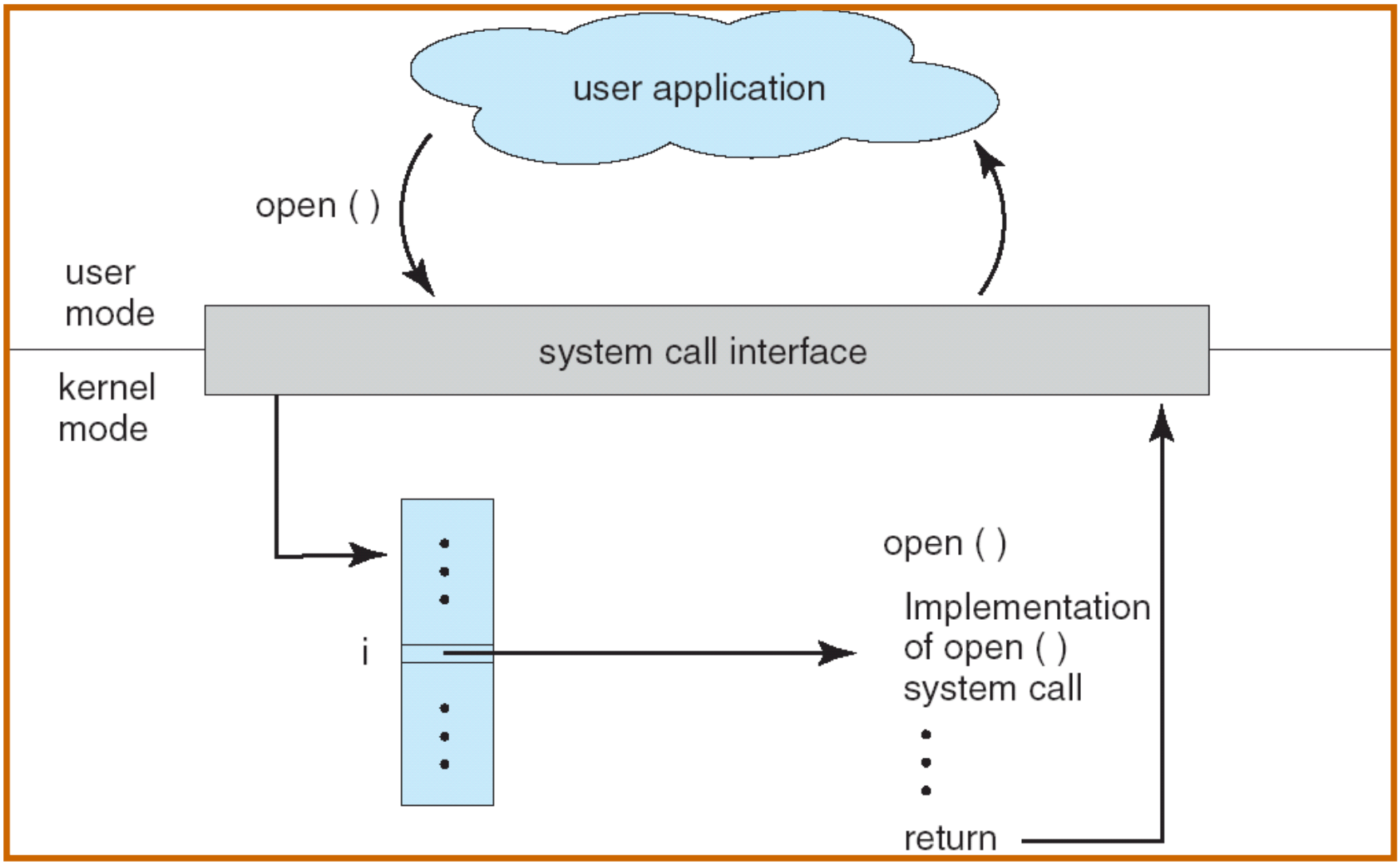
Llamados al sistema

- La llamada a un *system call* incluye las siguientes tareas:
  - Cargar los parámetros en el lugar adecuado (*stack* o registros).
  - Cargar el número de *system call* en algún registro específico (Ej: eax en Intel).
  - Invocar a la interrupción por software (*trap*) adecuada (*system call handler*).
  - El hardware cambia el bit de modo a monitor e invoca al manejador de la interrupción que controla que el número de *system call* pasado en el registro sea menor que el mayor del sistema y, finalmente, invoca al *system call* correspondiente.
  - El valor retornado por el *system call* es puesto en un registro específico (Ej.: eax en Intel).
  - Se vuelve el sistema a modo usuario y se retorna el control al proceso que invocó la *system call* (o eventualmente a otro)

Llamados al sistema

- Existen 3 formas de pasar los parámetros al sistema operativo:
  - A través de los registros: Se utilizan un conjunto de registros para pasar los parámetros. Tiene el problema de la cantidad de parámetros es fija y que restringe el tamaño del valor.
    - En Intel se utilizan 5 registros: ebx, ecx, edx, esi, y edi.
  - Un bloque de memoria apuntado a través de un registro.
  - En el *stack* del proceso que realiza el llamado. El proceso guarda los parámetros con operaciones *push* sobre el *stack* y el sistema operativo los saca con la operación *pop*.
- De la misma forma se pueden recibir los datos de respuesta

Llamados al sistema



Llamados al sistema

- Los *system calls* se clasifican en distintos tipos:
  - Control de procesos
    - Cargar, ejecutar, finalizar, abortar, obtener atributos, cargar atributos, esperar por tiempo, esperar por un evento o señal, obtener o liberar memoria, etc.
  - Gestión de archivos
    - Crear, borrar, abrir, cerrar, leer, escribir, obtener o cargar atributos, etc.
  - Gestión de dispositivos
    - Requerir o liberar un dispositivo, leer o escribir, buscar o cargar atributos de un dispositivo, etc.
  - Gestión de información del sistema
    - obtener o cargar la hora del sistema, datos del sistema, de procesos, etc.
  - Comunicaciones
    - Crear o destruir conexiones, enviar o recibir mensajes, etc.

Estructura del sistema

- Las estructuras interna de los sistemas operativos pueden ser muy diferentes.
- Se deben tener en cuenta:
  - Metas de los usuarios: ser amigable, intuitivo, confiable, seguro, rápido, etc.
  - Metas del sistema: fácil de diseñar, implementar y mantener, también flexible, confiable y eficiente.
- Diseño del sistema:
  - Sistema monolítico.
  - Sistema en capas.
  - Sistema con micronúcleo (microkernel).

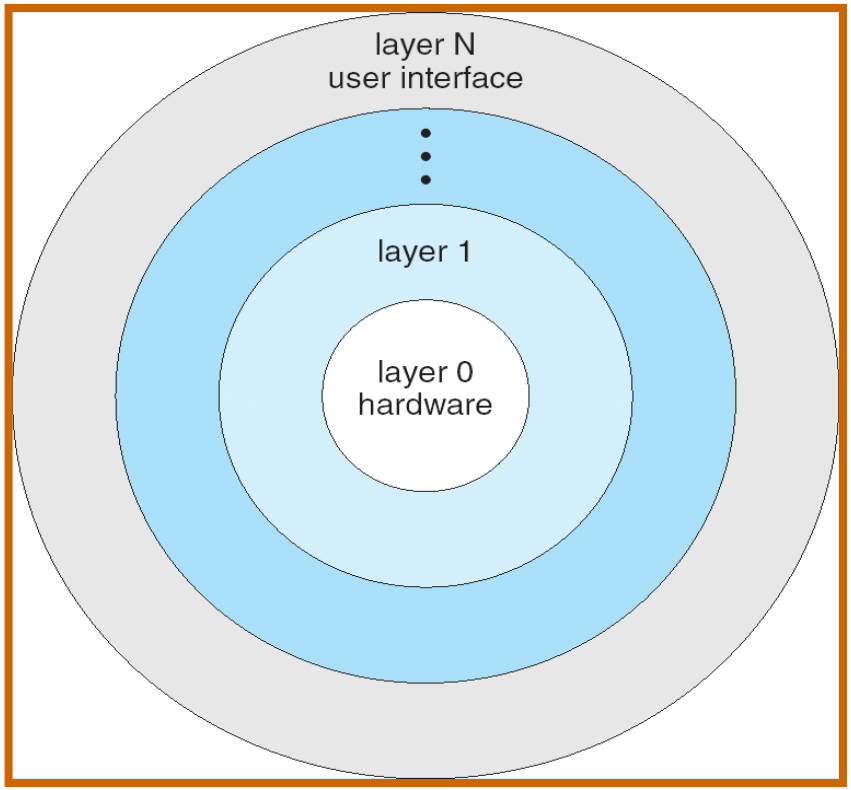
Sistema monolítico

- No se tiene una estructura definida.
- El sistema es escrito como una colección de procedimientos, que pueden ser invocados por cualquier otro.
- No existe “ocultación de información”, ya que cualquier procedimiento puede invocar a otro.
- Si bien todo procedimiento es público y accesible a cualquiera, es posible tener buenos diseños y lograr, de esa forma, buena eficiencia en el sistema.

- Ej.: MS-DOS.
  - Los componentes pueden invocar procedimientos de cualquiera.
- Ej.: Linux
  - Linux es un núcleo monolítico que a logrado un buen diseño orientado a objetos (sistema modular).

Sistema en capas

- Se organiza el diseño en una jerarquía de capas construidas una encima de la otra.
- Los servicios que brinda cada capa son expuestos en una interfase pública y son consumidos solamente por los de la capa de arriba.
- La capa 0 es el hardware y la N es la de procesos de usuario.

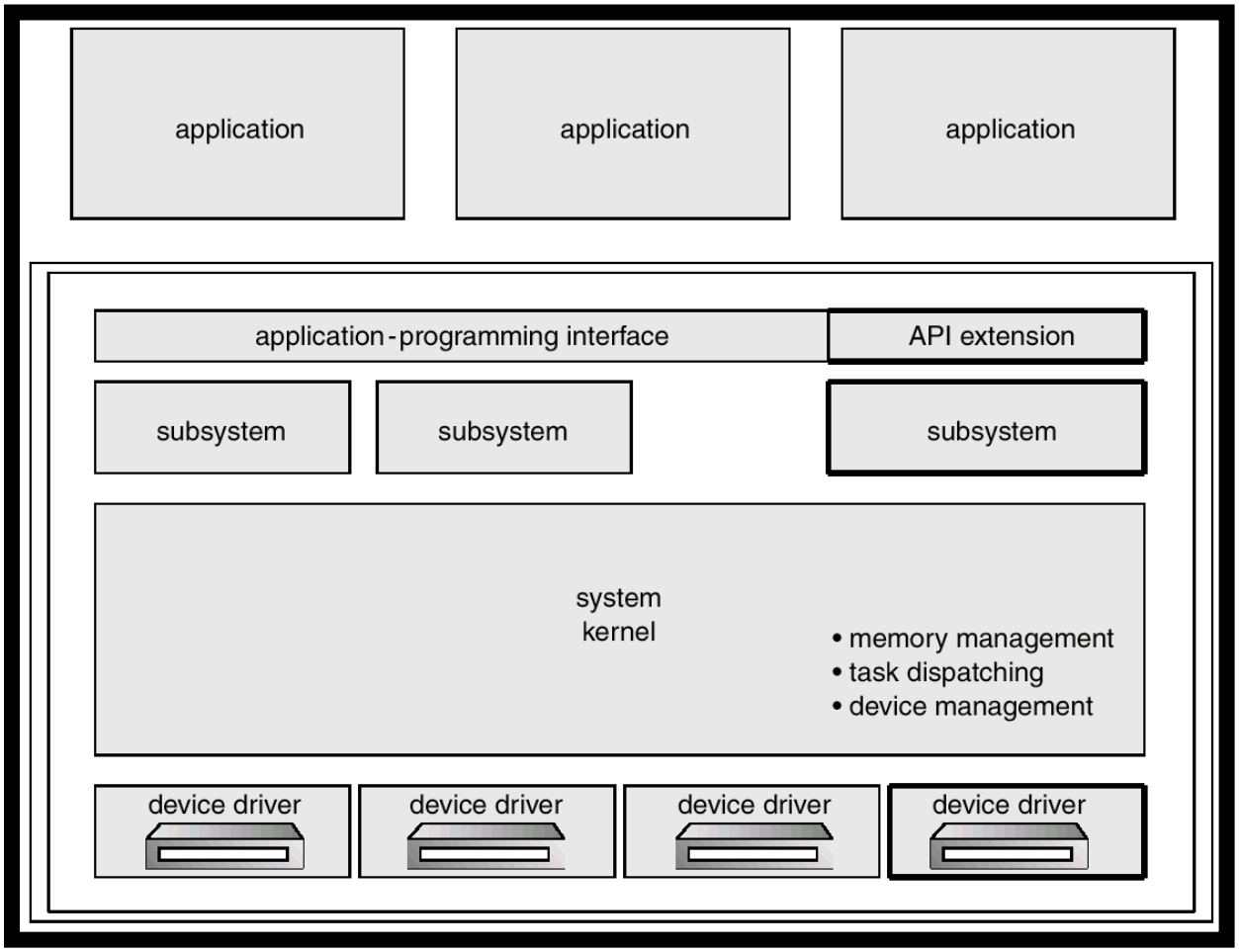


Sistema en capas

- Ventajas:
  - Modularidad.
  - Depuración y verificación de cada capa por separado.
- Desventajas:
  - Alto costo de definición de cada capa en la etapa de diseño.
  - Menos eficiente frente al sistema monolítico ya que sufre de *overhead* al pasar por cada capa.

Sistema en capas

- Ej.: en capas – OS/2.

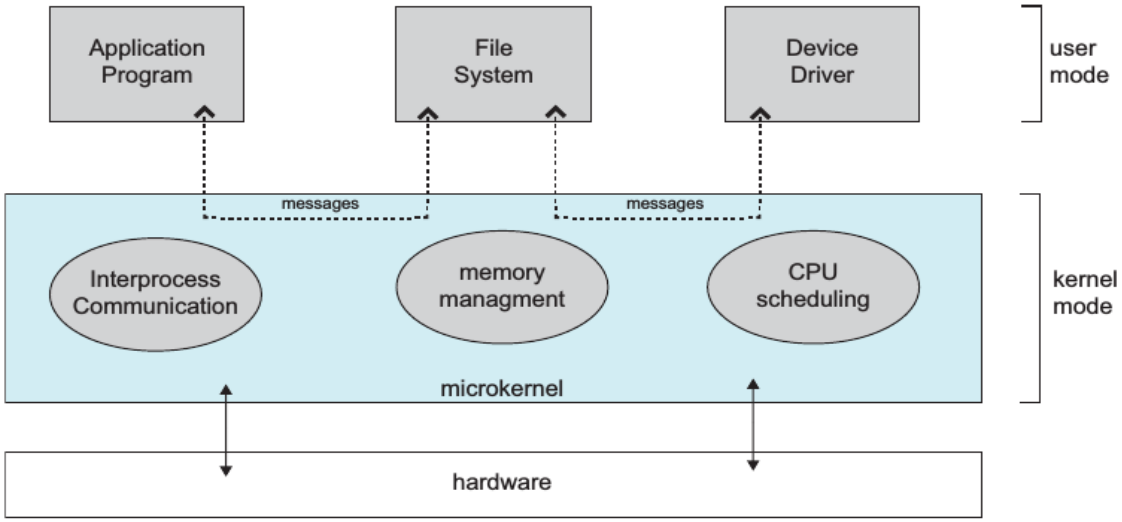


Sistema con micronúcleo (microkernel)

- Se constituye de un núcleo que brinde un manejo **mínimo de procesos, memoria** y, además, provea de una **capa de comunicación entre procesos**.
- La capa de comunicación es la funcionalidad principal del sistema.
- Los restantes servicios del sistema son construidos como procesos separados al micronúcleo que ejecutan en modo usuario.
- El acceso los servicios del sistema se realiza a través de pasaje de mensajes.

Sistema con micronúcleo

- Ventajas:
  - Aumenta la portabilidad y escalabilidad ya que encapsula las características físicas del sistema
  - Para incorporar un nuevo servicio no es necesario modificar el núcleo.
  - Es más seguro ya que los servicios corren en modo usuario.
  - El diseño simple y funcional típicamente resulta en un sistema más confiable.
- Ej:



Máquinas Virtuales

- Se puede ver como una extensión de los sistemas multiprogramados pero a más bajo nivel
- Los procesos no solamente trabajan sobre el sistema operativo como si fueran el único proceso en el sistema sino que tienen una copia virtual del hardware de la CPU
- Las máquinas virtuales corren como procesos a nivel de usuario y el administrador de MVs (*hypervisor*) implementa un modo usuario virtual y un modo administrador virtual
- También se implementan discos virtuales sobre los discos reales para las máquinas virtuales
- Dos modos básicos
  - Tipo 1: el administrador corre directamente sobre el hardware (ej. KVM)
  - Tipo 2: el administrador corre como un proceso sobre un sistema operativo normal (ej. VirtualBox)

Máquinas virtuales: beneficios

- Seguridad
  - Los procesos en cada máquina virtual son completamente independientes de los procesos en las otras
- Facilidad de desarrollo
  - Se pude correr un sistema operativo de test en una máquina virtual sin correr riesgos con el sistema real
- Flexibilidad
  - Correr un sistema operativo de una arquitectura en una máquina diferente
- Alta disponibilidad
  - En caso de falla de una MV se puede levantar otra rápidamente en otro hardware

Máquinas virtuales: desventajas

- Los tiempos de las operaciones pueden tardar más que en un sistema real
  - Tiempo adicional por traducir las operaciones
- Tiempo de respuesta de la máquina muy poco predecible por uso del sistema operativo de base u otras MVs
  - No apropiado para sistemas de tiempo real