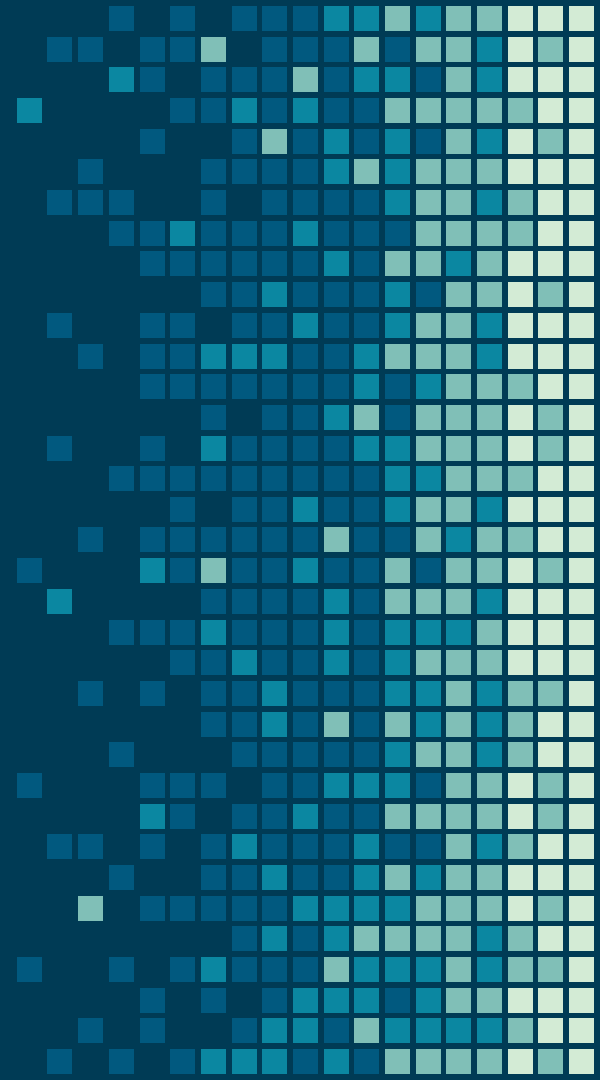

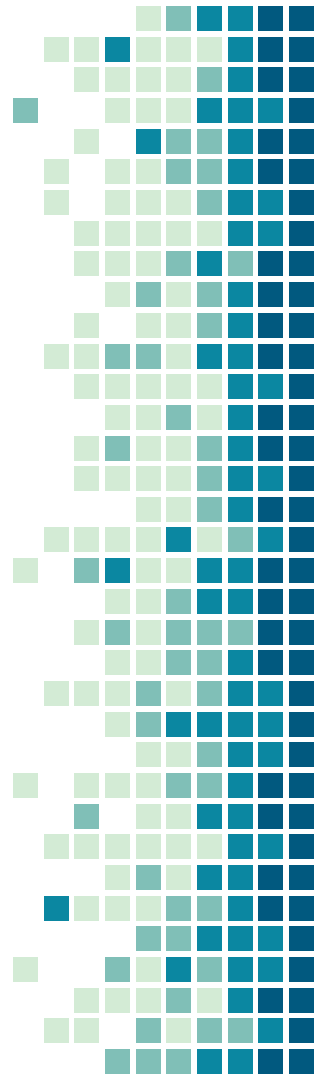


Servicios de un Sistema Operativo

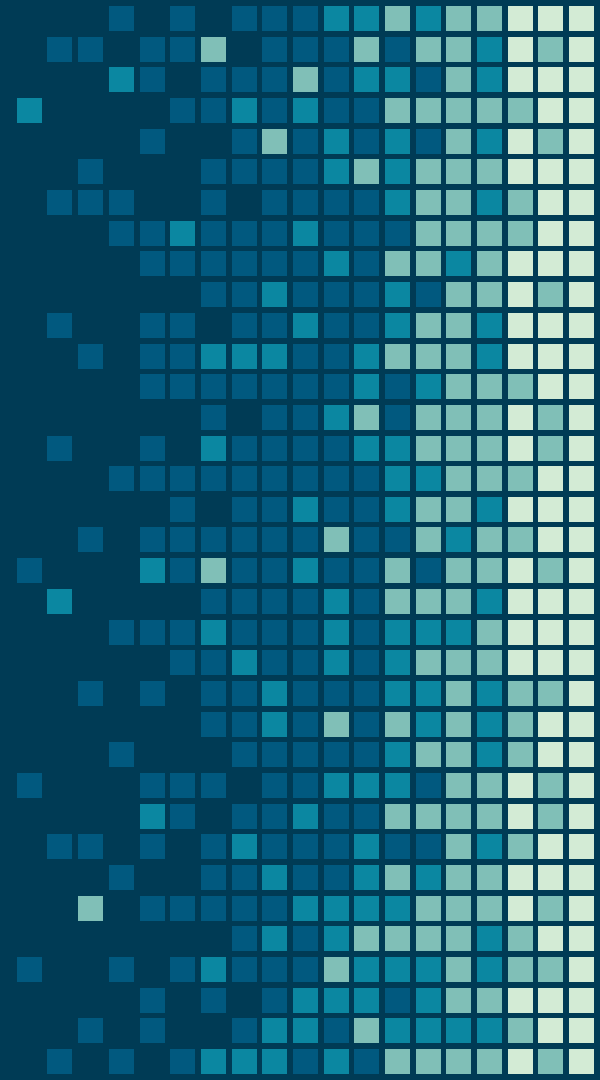


Agenda

- 
- A vertical line with a blue circle at the top and a blue arrowhead at the bottom, pointing downwards.
- **Estructura de los Sistemas Operativos**
 - **Tipos de sistemas operativos**
 - **Interrupciones por hardware**
 - **Llamadas al sistema**
 - **Funciones específicas del SO**
 - **Proceso de arranque**



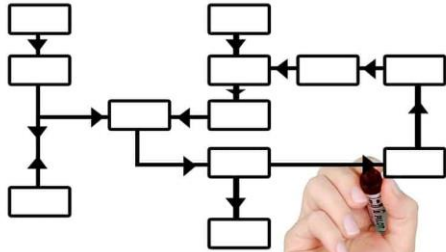
Estructuras de los sistemas operativos



Sistemas monolíticos

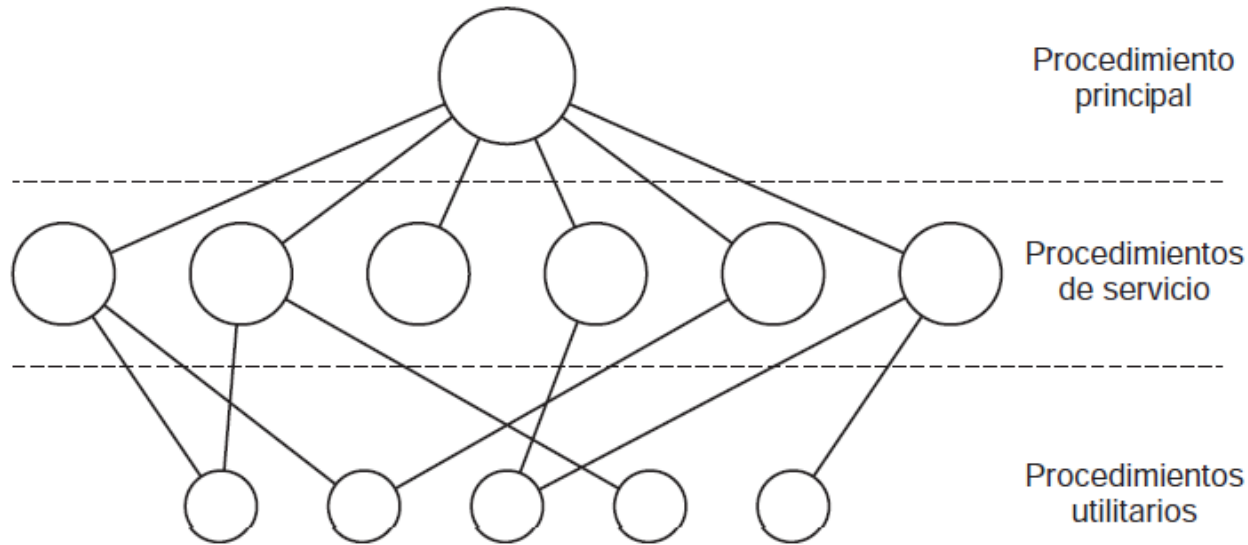
- El sistema operativo se ejecuta como un solo programa en modo kernel.
- Se escribe como una colección de procedimientos, enlazados en un solo archivo binario.
- Cada procedimiento en el sistema tiene la libertad de llamar a cualquier otro.
- No se oculta información entre procesos, todo proceso es visible.

Organización de los sistemas monolíticos



- 1 Un programa principal que invoca el procedimiento de servicio solicitado.
- 2 Un conjunto de procedimiento de servicio que llevan a cabo las llamadas al sistema.
- 3 Un conjunto de procedimientos utilitarios que ayudan a los procedimientos de servicio.
- 4 Simplifica la cantidad de mecanismos de comunicación.

Modelo de estructuración del sistema monolítico



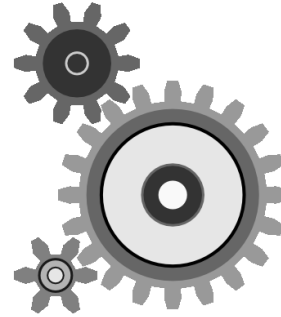
Sistemas de capas

Capa	Función
5	El operador
4	Programas de usuario
3	Administración de la entrada/salida
2	Comunicación operador-proceso
1	Administración de memoria y tambor
0	Asignación del procesador y multiprogramación

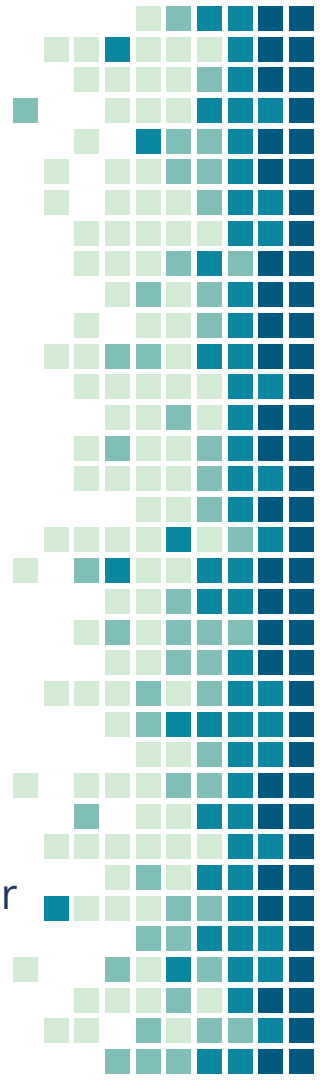
Generalización del monolítico, al estar organizado como una jerarquía de capas, cada una construida encima de otra.
THE, primer sistema de capas realizado en 1968.

Microkernels

Sólo un módulo se ejecuta en modo kernel, el microkernel
Proceso de usuario se puede configurar para evitar tanto poder.



- Coloca lo menos posible en modo kernel, dividiendo el sistema en módulos pequeños por lo que produce alta confiabilidad.

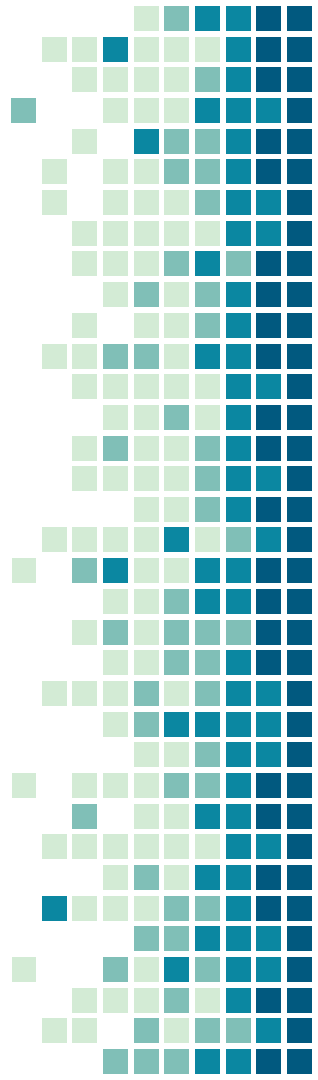


Microkernels

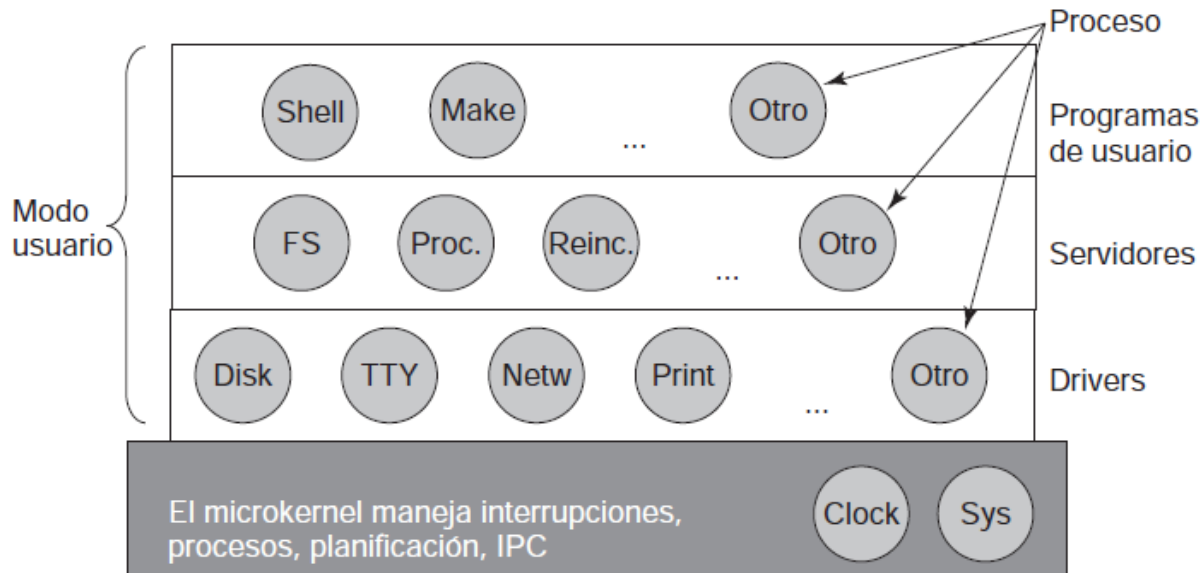
Programas adquieren servicios del sistemas mediante el envío de mensajes cortos.
Uno de las restricciones es que sólo los drivers utilizan el hardware



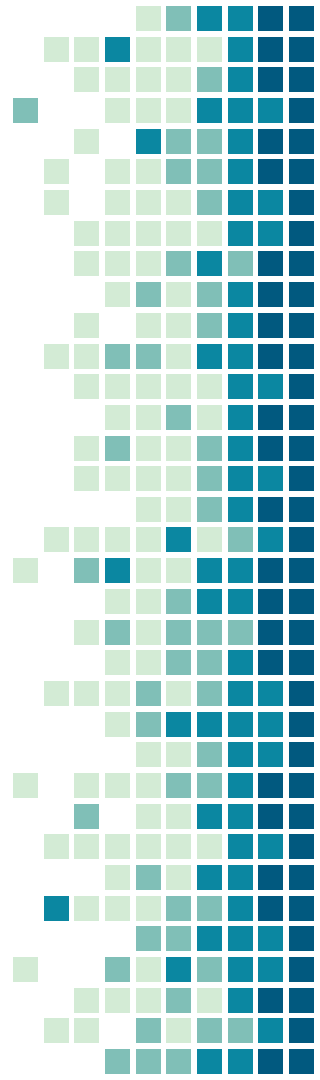
Coloca un mecanismo, pero no una directiva, ¿Cuál es la diferencia?



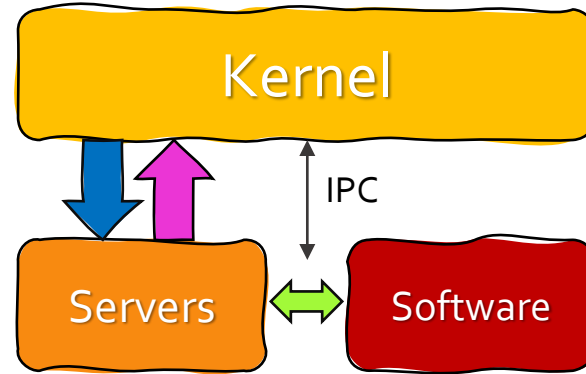
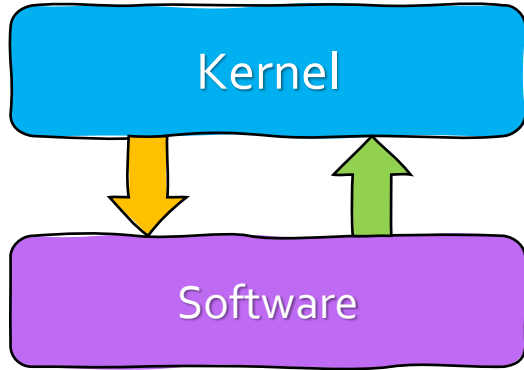
Estructura de MINIX



Torvalds VRS Tanenbaum



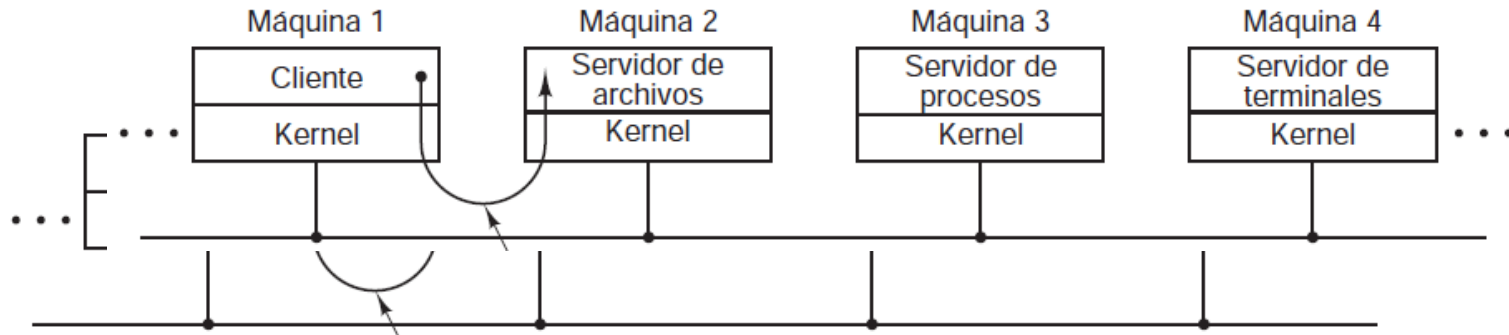
Torvalds VRS Tanenbaum



Modelo cliente-servidor

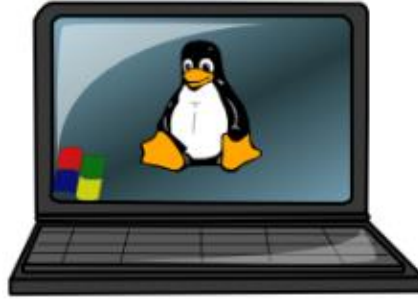
Es una ligera diferencia de microkernel, existen dos clases de procesos: los servidores, los cuales proporcionan un servicio, y los clientes, quienes los consumen.

La comunicación entre procesos se da por medio de mensajes.



Máquinas Virtuales

- 💡 Sistemas de tiempo compartido.
- 💡 Permiten la multiprogramación.
- 💡 Se puede analizar como copias exactas de hardware.
- 💡 Frecuentemente ejecutan SO diferentes a la máquina real.



Máquina virtual (Virtual VRS Transparentes)

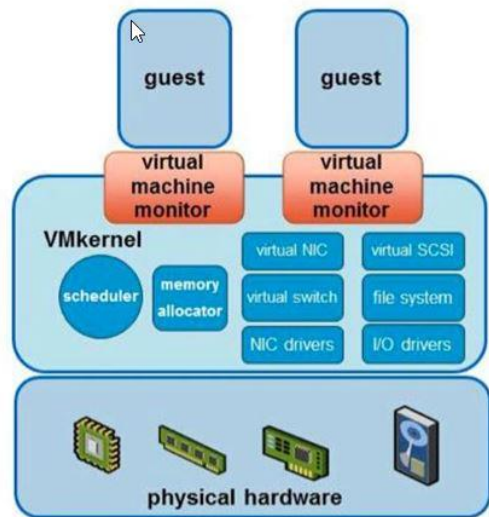
Transparente es lo que existe pero no se ve.

Virtual es lo que se ve, pero no existe.



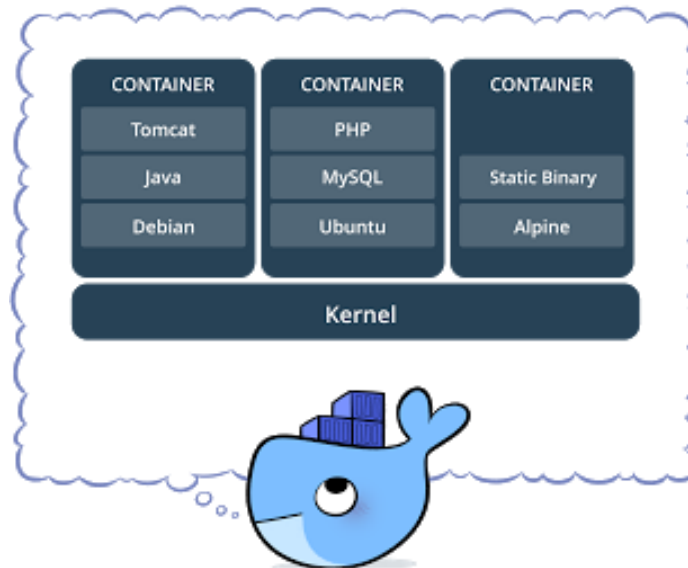
El concepto de VMM

- ✓ Es software que divide una sola máquina física en varias máquinas virtuales.

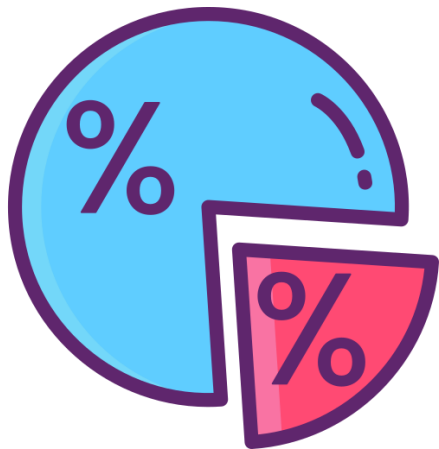


Contenedor

Agrupar y aislar entre sí aplicaciones o grupos de aplicaciones que se ejecutan sobre un mismo núcleo de sistema operativo



Exokernel



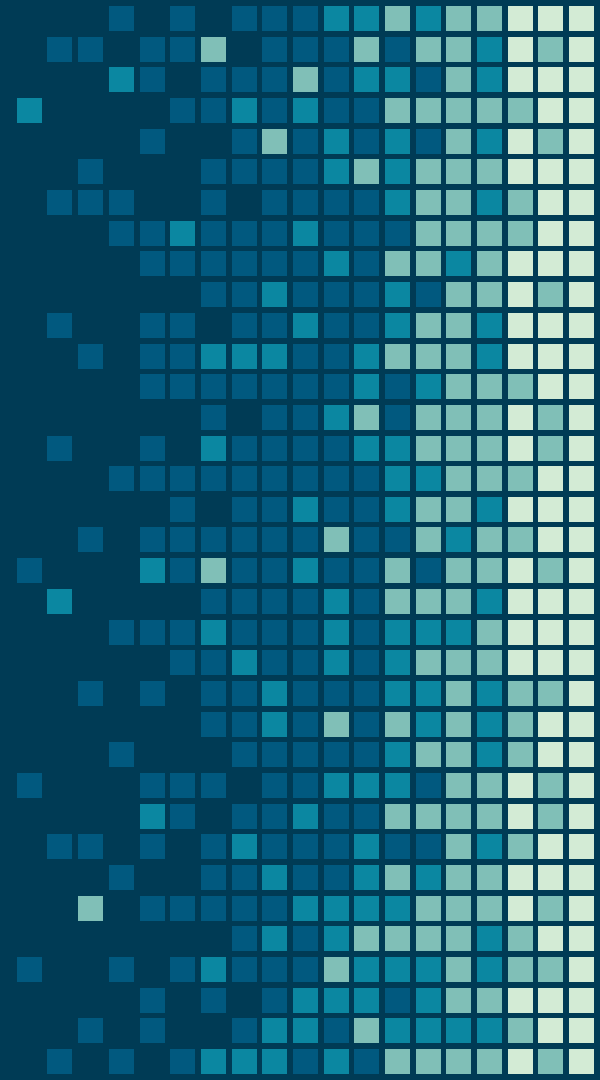
En vez de clonar una máquina, se particiona los recursos.

Así una máquina virtual podría obtener un conjunto de bloques de disco.

El principal objetivo es separar la protección de la administración, sin embargo, en máquinas virtuales asigna los recursos.

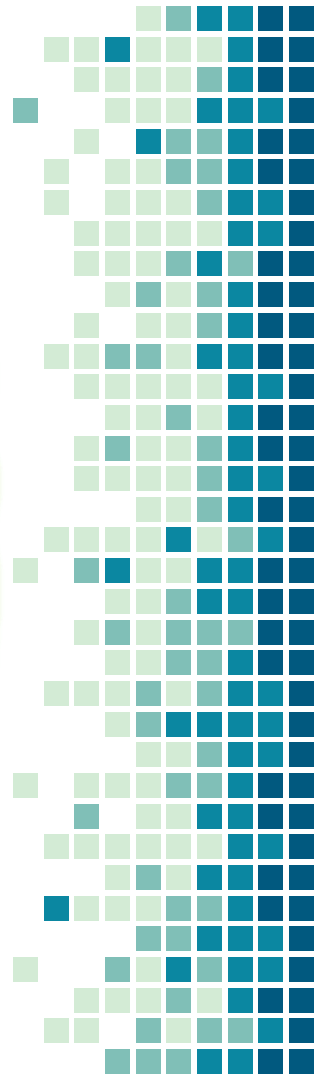
Cada máquina virtual utiliza su propio hardware. asignado

Llamadas al sistema e interrupciones



¿Cómo se protege el CPU?

Un programador hace un while infinito. A usted como diseñador Sistemas Operativos le asignan realizar un mecanismo para que dicho programa no se adueñe del CPU. Asuma una microarquitectura básica.

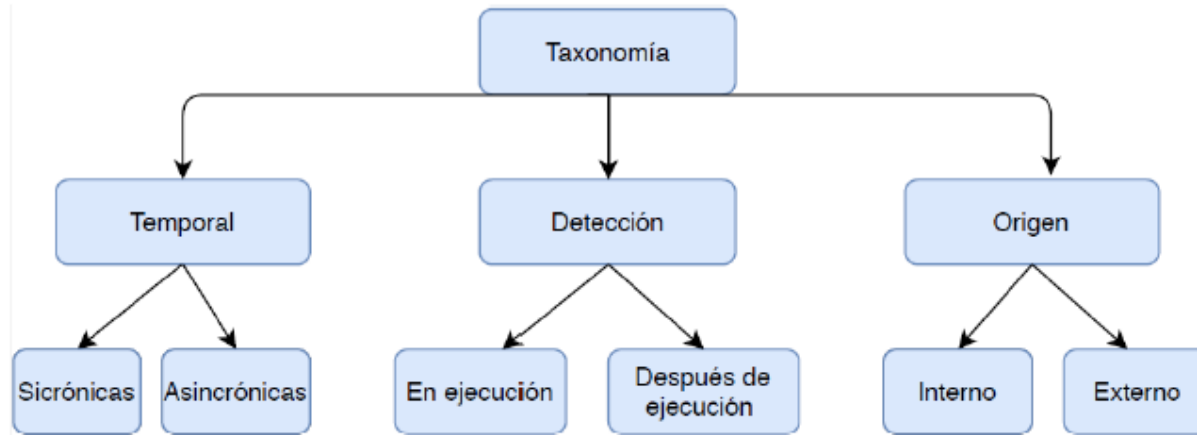


Clasificaciones de interrupciones

☞ Temporal.

☞ Detección.

☞ Origen.



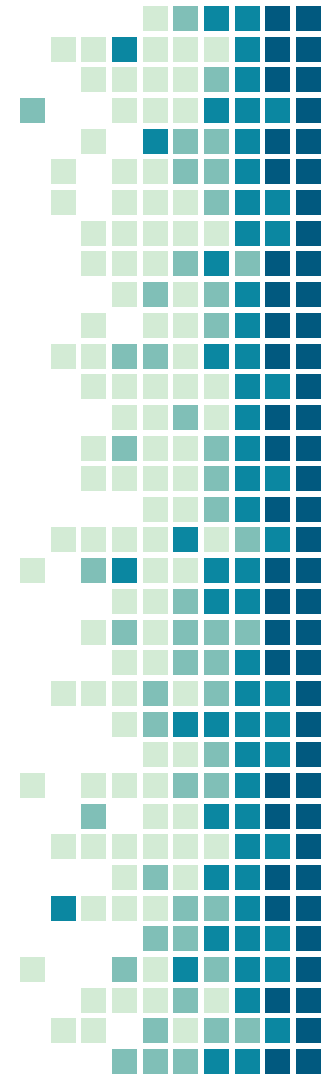
Llamadas al sistema

- - >Son operaciones extendidas que el sistema operativo ofrece para poder interactuar con los programas de usuario.
 - >Es la interfaz entre el sistema operativo y las aplicaciones.
 - >Si se está en modo de usuario y se necesita realizar alguna llamada al sistema, el programa debe ceder el control al SO, este la ejecuta y devuelve el control.

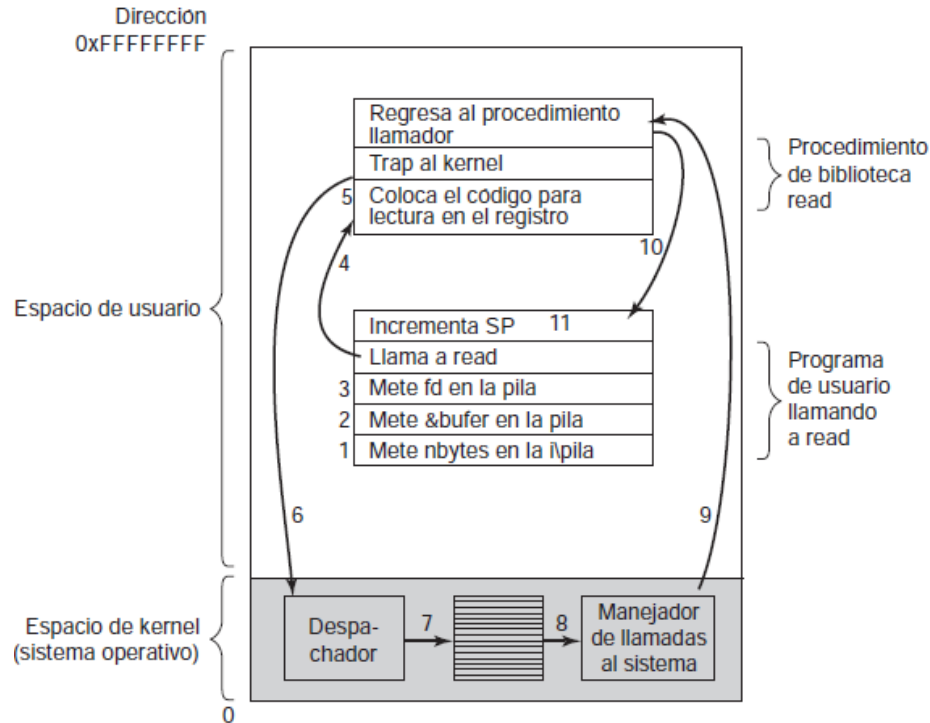


Llamada read(fd, bufer, nbytes)

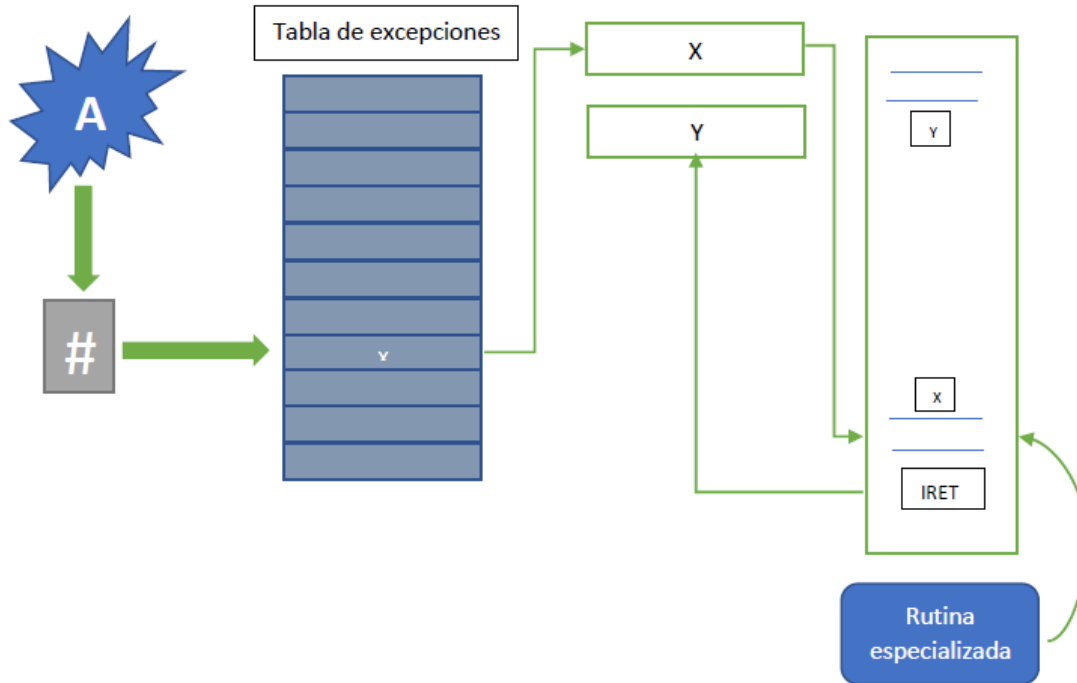
- ✓ ¿Qué ocurre si hay error al ejecutar la llamada?
- ✓ ¿Esta llamada se podría realizar en modo usuario?
- ✓ ¿Qué significan los parámetros?



Llamada al Sistema (Macro)

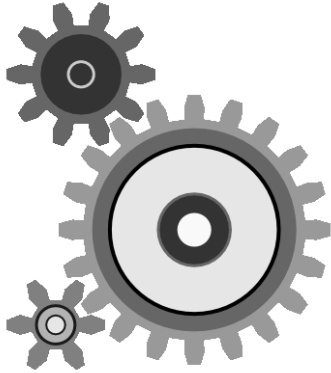


Pasos para la llamada al sistema (Micro)

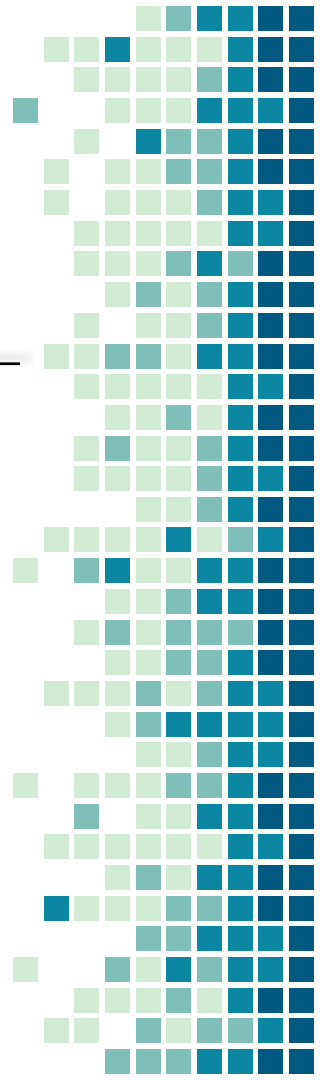


Actividad: Busque 10
systemcall (POSIX) y
comente la
funcionalidad y los
parámetros de cada
una.

Llamadas de administración de procesos



Llamada	Descripción
<code>Pid=fork()</code>	Crea un proceso hijo, idéntico al padre
<code>Pid=waitpid (pid, &statloc, opciones)</code>	Espera a que un hijo termine
<code>S=execve(nombre, argv, entorno)</code>	Remplaza la imagen del núcleo de un proceso
<code>Exit(estado)</code>	Termina la ejecución de un proceso y devuelve un resultado



Llamadas de administración de archivos

Llamada	Descripción
<code>Fd=open</code> (archivo,como...)	Abre un archivo para lectura, escritura o ambas
<code>s=close(fd)</code>	Cierra un archivo abierto
<code>n=read (fd, buffer, nbytes)</code>	Lee datos de un archivo y los coloca en un búfer
<code>n=write (fd, buffer, nbytes)</code>	Escribe datos de un búfer a un archivo
<code>Posición=lseek(fd, desplazamiento, de donde)</code>	Desplaza el apuntador de un archivo
<code>S=stat(nombre, &buf)</code>	Obtiene la información de estado de un archivo



Llamadas de administración de directorios y archivos

Llamada	Descripción
<code>s=mkdir(nombre, modo)</code>	Crea un nuevo directorio
<code>s=rmdir(nombre)</code>	Elimina un directorio vacío
<code>s=link(nombre1, nombre2)</code>	Crea un nuevo archivo llamado nombre2 que apunta nombre1
<code>s=unlink(nombre)</code>	Elimina una entrada de directorio
<code>S= mount(especial, nombre, bandera)</code>	Monta un sistema de archivos
<code>s=umount(especial)</code>	Desmonta un sistema de archivos




Equivalencias de llamadas

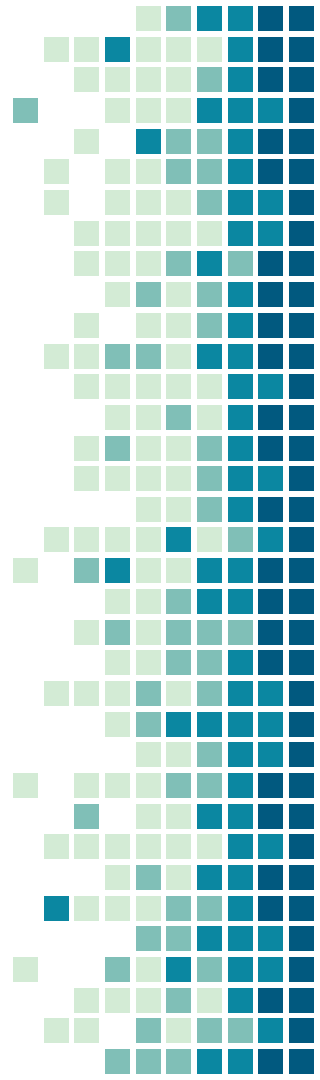
UNIX	Win32	Descripción
fork	CreateProcess	Crea un nuevo proceso
waitpid	WaitForSingleObject	Puede esperar a que un proceso termine
execve	(ninguno)	CreateProces = fork + execve
exit	ExitProcess	Termina la ejecución
open	CreateFile	Crea un archivo o abre uno existente
close	CloseHandle	Cierra un archivo
read	ReadFile	Lee datos de un archivo
write	WriteFile	Escribe datos en un archivo
lseek	SetFilePointer	Desplaza el apuntador del archivo
stat	GetFileAttributesEx	Obtiene varios atributos de un archivo
mkdir	CreateDirectory	Crea un nuevo directorio
rmdir	RemoveDirectory	Elimina un directorio vacío
link	(ninguno)	Win32 no soporta los enlaces
unlink	DeleteFile	Destruye un archivo existente
mount	(ninguno)	Win32 no soporta el montaje
umount	(ninguno)	Win32 no soporta el montaje
chdir	SetCurrentDirectory	Cambia el directorio de trabajo actual
chmod	(ninguno)	Win32 no soporta la seguridad (aunque NT sí)
kill	(ninguno)	Win32 no soporta las señales
time	GetLocalTime	Obtiene la hora actual

Funciones específicas de los SO

- 
- Realiza el inventario de los recursos del computador y el estado de los mismos.

- 
- Se define las políticas, es decir todo lo que permite realizar.

- 
- Separa el qué del cómo.



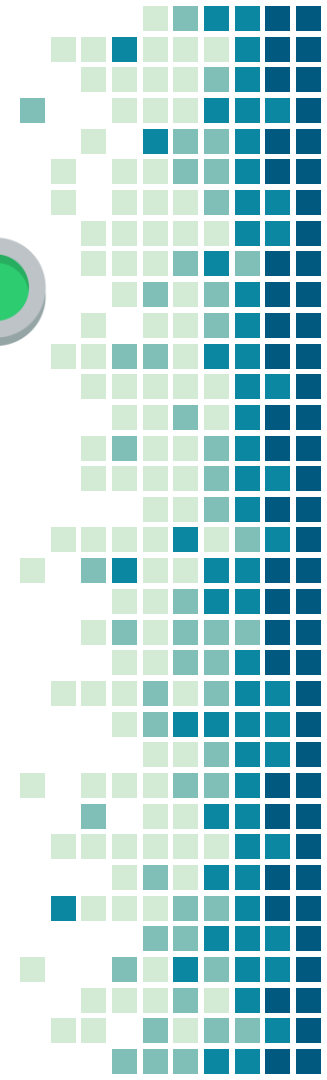
Proceso de arranque de la computadora



Proceso de arranque

—

- >El BIOS contiene software de bajo nivel de E/S.
- >Inicia reconociendo los componentes que está disponibles y que funcionen de manera correcta (RAM, ranuras, tarjetas).
- >El BIOS determina el dispositivo de arranque.
- >Se lee el primer sector del disco y se coloca en memoria para poder ejecutarse.
- >Este sector, contiene un programa que examina la tabla de particiones, para determinar cual está activa.



Proceso de arranque

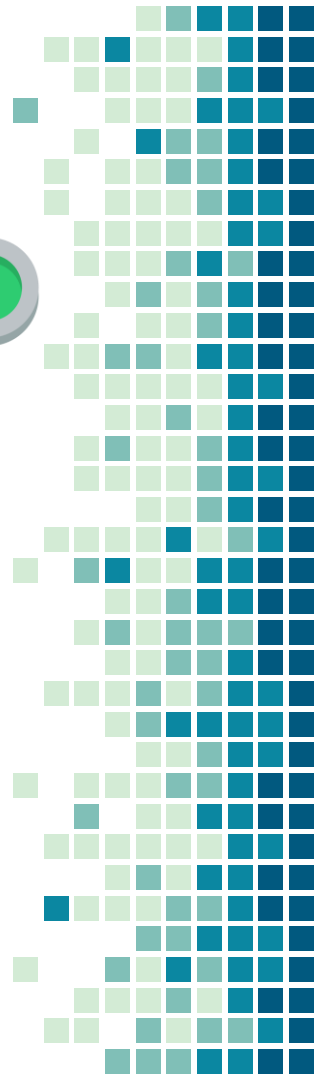
—

>Después se lee un cargador de arranque secundario de esta partición, el cual lee el sistema de dicha partición y lo inicia.

>Posteriormente el sistema consulta al BIOS para obtener los datos de configuración.

>Se hace la comprobación de los drivers de cada dispositivo.

>Se carga los drivers al kernel del SO.



Proceso de arranque

—

- >Inicializa las tablas.
- >Crea los procesos en segundo plano que se requieran.
- >Arranca programa de inicio de sesión o GUI.



Comparación de Tiempos

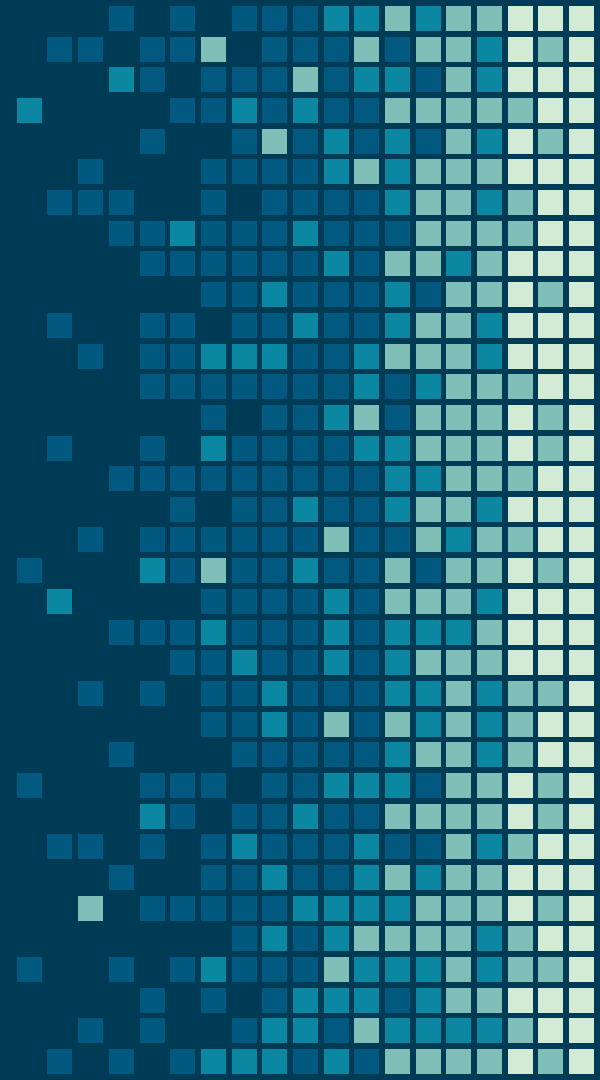
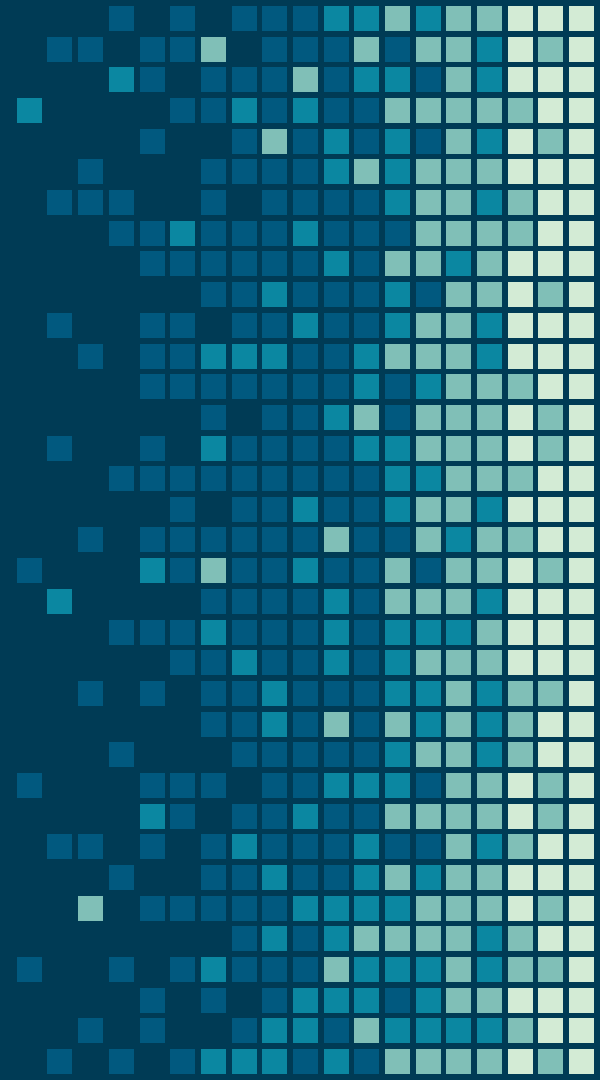


Tabla de comparación

- Procesador 2014 CPU 3.9 GHz

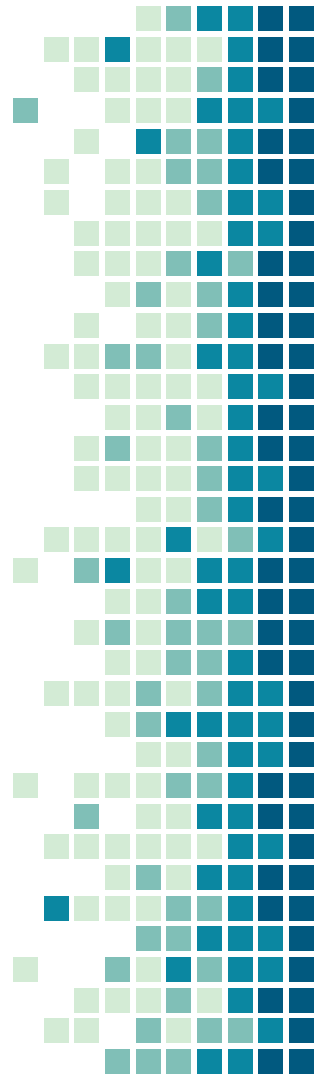
Actividad	Tiempo	Escala humana
Ciclo CPU	0.256 ns	1 segundo
Caché L1	1.026 ns	4 segundos
Caché L2	3.077 ns	12 segundos
Caché L3	6.154 ns	24 segundos
RAM	8.4 ns	32 segundos
Disco duro- mejor caso	2.9 ms	132 días
Disco duro- peor caso	12 ms	1.5 años
SSD	85 us	3 días y 20 horas
Cambio de contexto	10 us	10.80 horas
Quantum	100 ms	12.4 años

Deamons

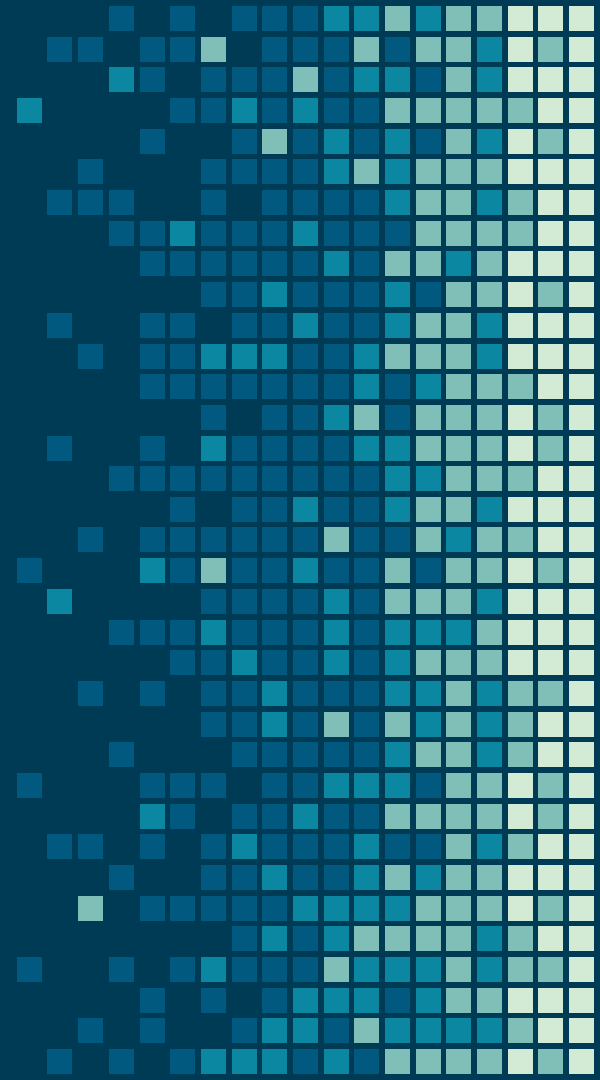


Demonios

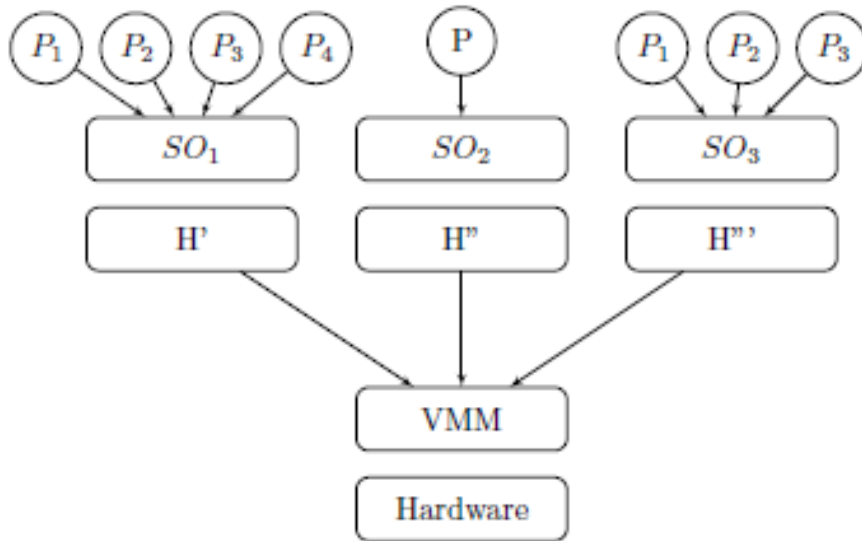
- Corresponden a los programas que se ejecutan en segundo plano.
- Generalmente no tienen interacción con el operador del sistema.
- Los sistemas operativos utilizan este tipo de proceso para proporcionar servicios.



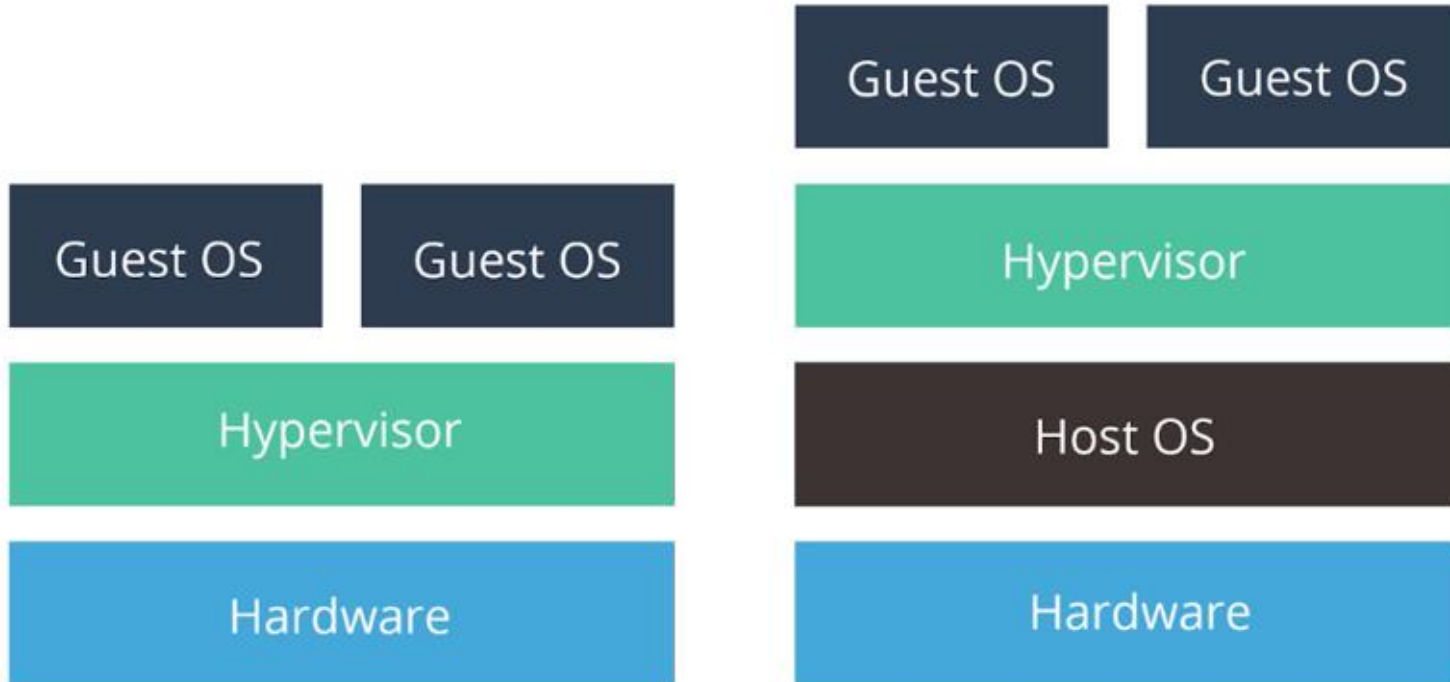
Interacción con VMM



Analizando diferentes casos

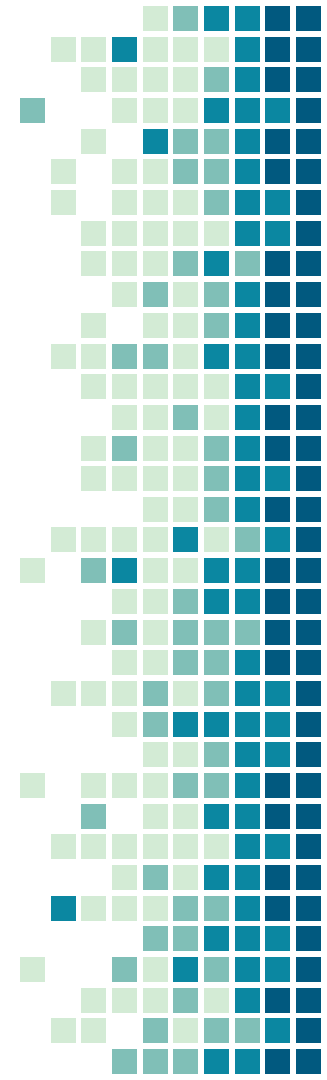


Tipos de VMM



Referencias

- ▶ Tanenbaum, A. S. (2022).
Sistemas operativos modernos. Pearson Educación
- ▶ Stallings, W. (2005).
Sistemas operativos. Martin Iturbide.
- ▶ Parshin, Y. G., & Pintiysky, V. V. (2013).
U.S. Patent No. 8,365,297. Washington, DC: U.S. Patent and Trademark Office.



¿Preguntas?

Realizado por: Jason Leitón Jiménez.

Tecnológico de Costa Rica

Ingeniería en Computadores

2023

TEC