Antes de la clase:

¿El mismo sistema operativo de un servidor de Azure son los mismos de una máquina virtual normal?

R/ No

Los 4 retos de las máquinas virtuales:

1. Estén aisladas
2. Cómo virtualizar la memoria
3. CPU sencillo con calendarización
4. Overhead las abstracciones lo provocan

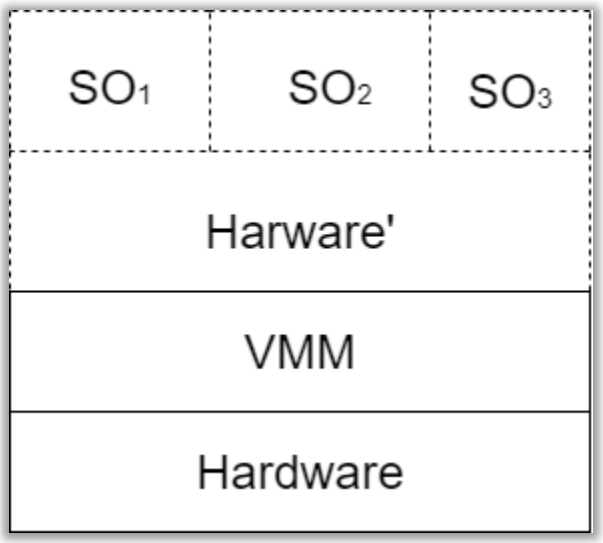
En un solo hardware debe correr todo.

¿Cómo fueron pensadas y creadas las máquinas virtuales?

Re/ Las creo IBM

Virtual Machine Monitor (VMM): crea capa de abstracción portea el sistema operativo en un hardware virtual.

➔ Hypervisor está dentro del VMM



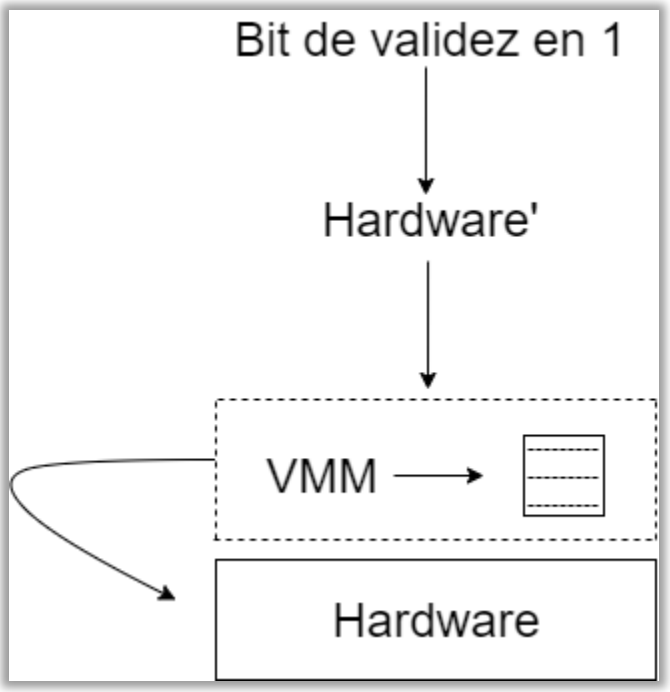
Esto logra “crear otro hardware” por lo que cada sistema operativo cree que ocurre en un hardware real.

Tengo n sistemas operativos n maneras de portearlos.

solo cambia el kernel, en Azure, por lo que agunta muchas máquinas virtuales.

¿Qué pasa si hay un programa (P) P1, que corre en el sistema operativo y divide entre 0?

R/ Se ejecuta la interrupción

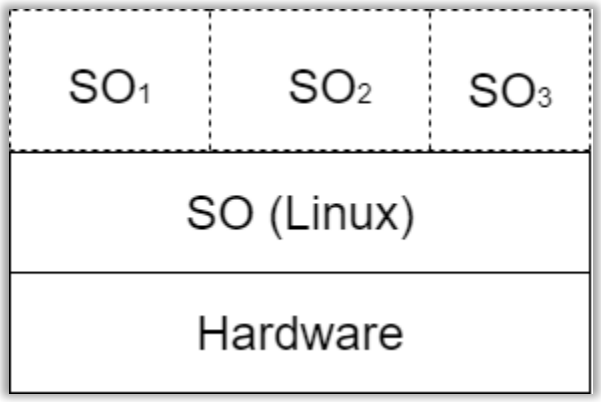


Llega al hardware, cómo se ejecuta por cambio de contexto, settea el bit de validez en uno, además necesita que ambos bits de validez estén en uno, sistema operativo que llamó la excepción y el hardware real.

¿Qué pasa si el sistema operativo está en 0 y quiero escribir en memoria física?

R/ Lo rechaza porque el bit de validez del sistema operativo que ejecuta la excepción es diferente al bit de validez del hardware real, levanta una excepción y la devuelve al sistema operativo que generó la excepción.

Otro enfoque



Manera más lenta necesita calendarización mucho overhead ya que lo ve como un proceso común y corriente.

Diferencias entre una máquina local y una virtual

¿Porque la máquina virtual es más rápida que la local?

R/ Porque lo que se ejecuta en la máquina virtual es solamente el sistema operativo y el programa en cambio en la máquina local se pueden tener muchos procesos ejecutándose a la vez lo que retrasa la ejecución del programa.

Noticias

Examen semana 9 o 10, después de Semana Santa.

La otra semana traer computadora, decirle el viernes al profesor qué día.

Adelanto: Implementar excepciones en C.

Comienza la lección del día

Procesos e hilos en un sistema operativo

Tema central del curso, fundamental.

“Los problemas que estaban en los 60 son los mismos de ahora solo que han cambiado las dimensiones”

¿Cómo se hace para ejecutar varias aplicaciones al mismo tiempo?

R/ Con multiplexaje generalmente de tiempo saltar en el tiempo de uso del procesador, no es al mismo tiempo.

¿Cómo se hace para comunicar dos aplicaciones?

R/ memoria compartida e intercambio de mensajes, la diferencia entre éstos es un buffer que se encarga de los mensajes, el buffer está en memoria en memoria compartida.

¿Cómo se asignan los recursos computacionales?

Re/ El sistema operativo se encarga asignarlos se asignan de manera peculiar tabla de procesos PID.

¿Qué pasa cuando un web server tiene muchos clientes a la vez y dan click al mismo tiempo?

R/ El sistema operativo tiene que identificar y calendarizar todas, se atiende una y se envía en procesos.

Procesos en un sistema operativo

Uno de los conceptos más importantes en sistemas operativos e informática.

Abstracción de un programa en ejecución, hacer la ejecución de un programa de manera sencilla.

Para que haya un proceso se debe de ejecutar un programa.

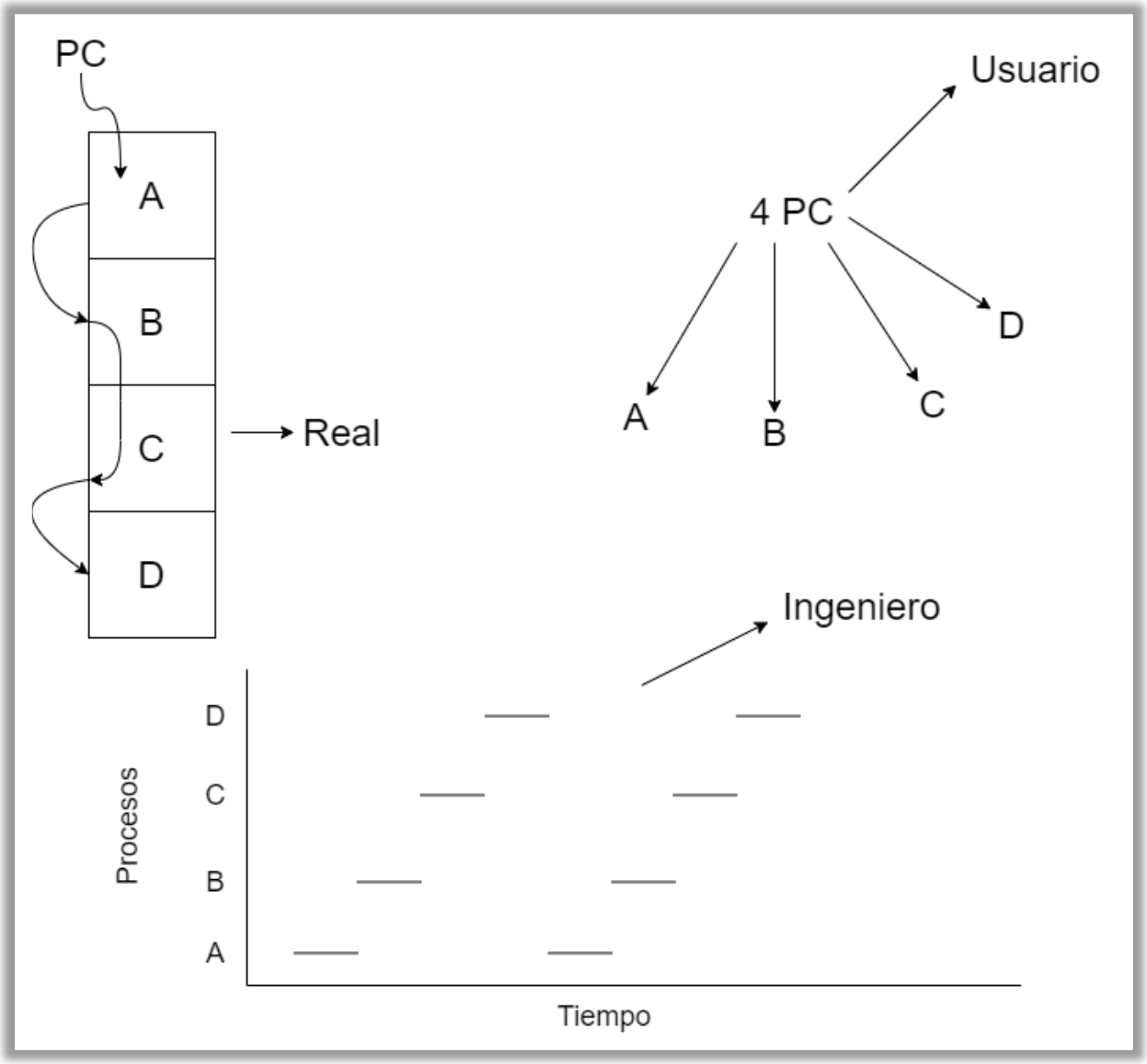
Proporciona la capacidad de realizar concurrencia en un solo procesador.

Todo sobre ejecutable se organiza en varios procesos ejecutables, cierto para programas complejos, por ejemplo, Word.

Cada proceso posee un CPU virtual.

Es una instancia de un programa incluyendo todo lo de este, registros y variables.

Una de las garantías de la multiprogramación todo proceso corre como un proceso aparte es decir cree que tiene un CPU y memoria solo para él.



Se da una conmutación de procesos: es quitar y poner otros. se llama cambio de contexto en la literatura.

¿Cuánto tarda el procesador en conmutar un proceso?

R/ Depende del programa, es una trampa, se necesitan más datos.

¿Qué se podría decir de la rapidez de los procesos en un procesador específico?

R/ Es imposible reproducir la ejecución de un programa variables caché no se puede afirmar nada excepto que no duran lo mismo.

Proceso vs Programa

Se utiliza un ejemplo sencillo

- ➔ Existe un científico con mente culinaria
- ➔ Posee la receta y todo el equipo
- ➔ Posee ingredientes

quién es quién

- ➔ La receta es el programa porque es una serie de pasos.
- ➔ El científico es el procesador, maneja los recursos.
- ➔ Los ingredientes son los datos de entrada.
- ➔ La cocina del hardware porque es donde se va a hacer.
- ➔ El proceso es hacerlo posible.

Ocurre un evento, el hijo del científico entra gritando.

¿Qué debe hacer científico?

R/ cambio de contexto, es decir dejar todo como está para atender el evento.

El evento es una interrupción

Diferencia puntual entre proceso y programa

Los programas son instrucciones y el proceso es la ejecución con los elementos necesarios.

¿Cuándo se crean los procesos en el sistema operativo?

R/

- ➔ Durante el boot.
- ➔ La ejecución de un proceso de una llamada al sistema para la creación de otro proceso.
- ➔ Petición del usuario.
- ➔ Inicio de un trabajo por lotes.

Creación de procesos

Generalmente, necesita que un proceso existente ejecute una llamada al sistema.

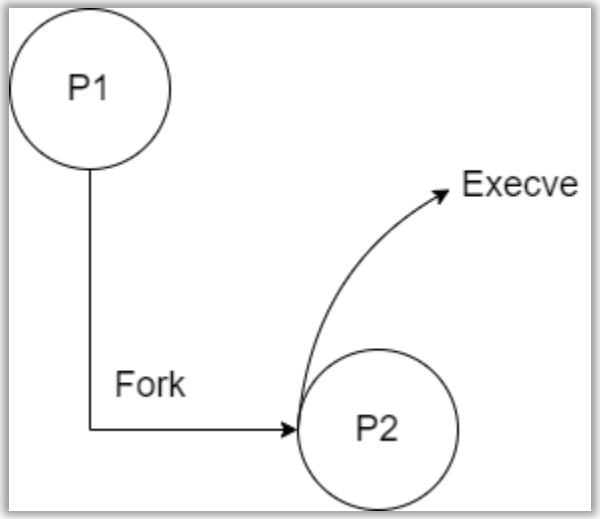
La llamada indica al sistema operativo que cree un proceso y le indica a cuál programa asociarlo.

Fork: crear proceso en Unix.

Crea un clon exacto del programa que realizó la llamada.

Los dos procesos tienen la misma imagen de memoria.

El proceso creado ejecuta otra llamada, execve, qué cambia la imagen de memoria y ejecuta el programa correspondiente.



CreateProcess: crear proceso en Windows

Crea un clon exacto del proceso que realizó la llamada y carga de una vez el programa correcto.

Posee más parámetros que fork, cómo 10.

¿Cuál es mejor?

R/ En Unix, al copiar todo, es más eficiente. Ya que tiene acceso a los mismos archivos que el proceso que lo llamo. Es mejor Unix.

CreateProcess vs Fork

Ambos cuentan con espacios de direccionamiento distintos.

En Windows desde el inicio en Unix está el paso intermedio.

No se comparte memoria excepto del inicio en Unix.

Demonios

Son procesos que permanecen en 2º plano, realizando tareas para administrar ciertas actividades el usuario no se tiene que dar cuenta.

Generalmente, el sistema operativo posee docenas de demonios realizando tareas que son necesarias.

Terminación de procesos

- ➔ Salida normal
- ➔ Salida por error de usuario
- ➔ Error fatal
- ➔ Eliminado por otro proceso

Salida normal

Proceso termina porque terminó su trabajo.

Llamada exit en Unix, ExitProcees En Windows.

Salida por error de usuario

Proceso termina porque se descubre un error.

Ejemplos: Mal comando, Escribe mal un nombre.

Error fatal

Error en el programa

Ejemplos: División entre 0, Instrucción ilegal.

Es una interrupción sincrónica.

Eliminado por otro proceso

Es el caso cuando un proceso podría ejecutar una llamada al sistema para eliminar otro proceso.

kill en unix, TerminateProcess Windows.

Jerarquías en los procesos

¿Por qué dicen que Windows no tiene jerarquía de procesos?

R/ Porque se puede referenciar procesos donde sea con una llamada.