Sistemas Operativos

Unidad 1: Introducción a los Sistemas Operativos

En el principio...fue la línea de comandos ~ Neal Stephenson

~ Grady Booch

Introducción a los sistemas operativos

¿Qué es un sistema operativo?

Ante todo, un sistema operativo es un programa que permite controlar la ejecución de aplicaciones, que actúa como interfaz entre las aplicaciones y el hardware de la computadora.

Es complejo dar una definición precisa y parte de esta complejidad se debe a que un sistema operativo realiza dos funciones que en principio no se encuentran relacionadas entre sí. La primera de estas funciones es extender la funcionalidad de la máquina mientras que la segunda se refiere a la administración de los recursos de la máquina.

El sistema operativo como una máquina extendida

Desde el punto de vista de un programador/a, resultaría muy complejo tener que desarrollar software utilizando las instrucciones en lenguaje máquina de cada arquitectura de computadoras. Imaginemos que si por cada programa a desarrollar debemos implementar las rutinas para lectura o escritura de datos en disco o memoria, la programación se tornaría una actividad cuanto menos tediosa, sobre todo si tenemos en cuenta que estas instrucciones son dependientes de la arquitectura y por lo tanto cambian máquina a máquina.

Es por esto que una función del sistema operativo es presentar al usuario el equivalente de una **máquina extendida** o **máquina virtual** de forma tal que sea más fácil de programar que el hardware subyacente.

La forma en que el sistema operativo logra esto es proporcionando una variedad de servicios que los programas pueden obtener usando instrucciones especiales denominadas **llamadas al sistema**.

El sistema operativo como un administrador de recursos

El concepto anterior puede entenderse como una visión “top-down”. Una visión alternativa, “bottom-up” (de abajo hacia arriba), sostiene que el sistema operativo está para administrar todas las piezas de un sistema complejo.

Las computadoras modernas constan de procesadores, memorias, discos, interfaces de red, dispositivos de E / S y una amplia variedad de otros dispositivos. La función del sistema operativo es proporcionar una asignación ordenada y controlada de estos componentes entre los diversos programas que compiten por el uso de los mismos

La administración de recursos, implica que los mismos deben ser compartidos por los distintos programas o usuarios que los solicitan de dos maneras: tiempo y espacio.

Cuando un recurso es compartido en el tiempo, significa que los procesos (o usuarios) que lo solicitan, deben usar dicho recurso por turnos. Un ejemplo de esto sería que varios procesos requieran utilizar el CPU al mismo tiempo. Es responsabilidad del sistema operativo asignar los turnos para que cada proceso o usuario pueda utilizar el recurso.

Por otra parte, recursos como la memoria o el disco, pueden ser fraccionados y asignados de a partes entre los procesos o datos que lo requieran. Asignar los espacios (direcciones) de memoria o los bloques de disco es responsabilidad del sistema operativo.

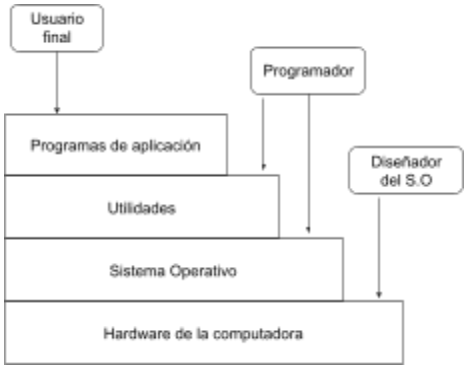
Objetivos y funciones

A grandes rasgos, un sistema operativo posee 3 objetivos que debe cumplir:

- Facilidad de uso
- Gestionar los recursos
- Capacidad para evolucionar

Facilidad de uso

El hardware y software utilizados para proporcionar aplicaciones a los usuarios se puede pensar como un sistema organizado jerárquicamente en capas, donde la capa inferior proporciona servicios a la capa superior.



Esta jerarquía de capas hace que un usuario final, que utiliza un programa de aplicación, no tenga que preocuparse por detalles del hardware. A su vez un programador de aplicaciones, para desarrollar una aplicación utiliza bibliotecas proporcionadas por el sistema operativo que le permiten controlar el hardware de la computadora. Imaginemos que sin estas bibliotecas cada

programador debería desarrollar todo un conjunto de instrucciones en código máquina que le permitan controlar la E/S (entrada/salida) de datos, manejo de archivos en disco, etc.

Finalmente el sistema operativo, resulta ser el software de base más importante ya que oculta los detalles del hardware a los programadores proporcionando una interfaz apropiada para su uso (bibliotecas) mientras que a las aplicaciones proporciona servicios y utilidades para realizar las tareas.

De manera resumida, un sistema operativo, proporciona servicios en las siguientes áreas:

- **Desarrollo y ejecución de programas**. Proporcionando bibliotecas y servicios al desarrollador. Realizando todos los pasos necesarios para ejecutar un programa.
- **Acceso a dispositivos de E/S**. Proporcionando un conjunto de instrucciones que pemita al programador y a las aplicaciones manipular de manera uniforme los dispositivos.
- **Acceso controlado a archivos**. Proporciona mecanismos de control y protección en el acceso.
- **Acceso al sistema**. Controla el acceso al sistema y a los recursos. Proporcionando protección a los recursos y datos.
- **Detección y respuesta a errores**. Ya sea por errores de hardware o software, debe proporcionar una respuesta que elimine la condición de error. La respuesta puede variar entre cerrar la aplicación que produjo el error hasta reintentar la operación.
- **Estadísticas**. Recolecta estadísticas de uso de los recursos y rendimiento. Estos datos son utilizados para mejorar el rendimiento del sistema.

Gestionar los recursos

La computadora es un conjunto de recursos (ej.: procesador, memoria, disco) que se utilizan para el ingreso, procesamiento y almacenamiento de datos así como también para llevar a cabo el control de estas operaciones. Ahora bien, el sistema operativo es el encargado de gestionar y controlar estos recursos.

Como ya mencionamos, el sistema operativo es un software que como tantos otros proporciona instrucciones al procesador de la máquina, pero lo que lo hace diferente es su objetivo. El sistema operativo dirige al procesador en el uso de los otros recursos de la computadora y administra su uso entre los distintos programas, es decir, el sistema operativo determina cuánto tiempo de procesador debe asignarse a la ejecución de un programa de aplicación/usuario.

Por otra parte, la memoria principal (RAM) es administrada en forma conjunta por el sistema operativo y el hardware de gestión de memoria del procesador.

Por último, es el sistema operativo el que debe decidir cuándo un programa puede utilizar un dispositivo de E/S (entrada / salida) a la vez que controla el acceso y uso de los archivos.

Capacidad para evolucionar

Un sistema operativo debe tener la capacidad de evolucionar en el tiempo debido a actualizaciones en el hardware y tipos de hardware, nuevos servicios o mejoras de los existentes y resolución de errores.

Un poco de historia y algo más

La primera computadora digital fue diseñada por el matemático inglés Charles Babbage (1792 - 1871). Esta máquina nunca llegó a construirse debido a que era 100% mecánica y en ese momento de la historia no contaba con la tecnología adecuada. Lo interesante de este episodio es que entre 1842 y 1843, *Ada Lovelace* comienza a trabajar con Babbage describiendo con un lenguaje técnico el funcionamiento de la máquina de Babbage. En su trabajo, incluyó varias notas entre las cuales figura una detallada y formal descripción de cómo obtener una secuencia de números de Bernoulli utilizando la máquina de Babbage. Gracias a esto, hoy es considerada la primera programadora de la historia.

El lenguaje de programación, Ada, fue nombrado así en honor a Ada Lovelace

Procesamiento en serie (primera generación)

Desde los años 40 y hasta mediados de la década del 50, el programador interactuaba directamente con el hardware de la computadora. No existía ningún sistema operativo. Los programas eran escritos en código máquina que se cargaba en la computadora a través de tarjetas perforadas en el dispositivo de entrada, un lector de tarjetas. Si el programa terminaba de forma normal, el resultado podía verse en la impresora, si se generaba un error, el programador podía examinar los registros del procesador y la memoria principal.

Este sistema tenía dos problemas. El primero era que la planificación se realizaba de forma manual. Esto implicaba reservar tiempo de procesamiento en una planilla con lo cual podía ocurrir que el programa terminara antes de lo estipulado, por lo que se desperdiciaba tiempo o bien que el programa tarde más de lo esperado y por lo tanto no se pudiera obtener los resultados.

El segundo problema era el tiempo de configuración. Cada vez que se tenía que ejecutar un programa, el programador debía cargar el compilador, el programa y los datos. Esto implicaba un tiempo considerable.

Se denominó *procesamiento en serie* debido a que los programas se ejecutaban uno atrás del otro.

Sistemas en lotes simple (segunda generación)

A mediados de los años 50, y con el surgimiento del transistor, las computadoras se volvieron más confiables y "aptas" para ser vendidas. Generalmente eran organismos gubernamentales y universidades quienes podían pagar las cifras multimillonarias.

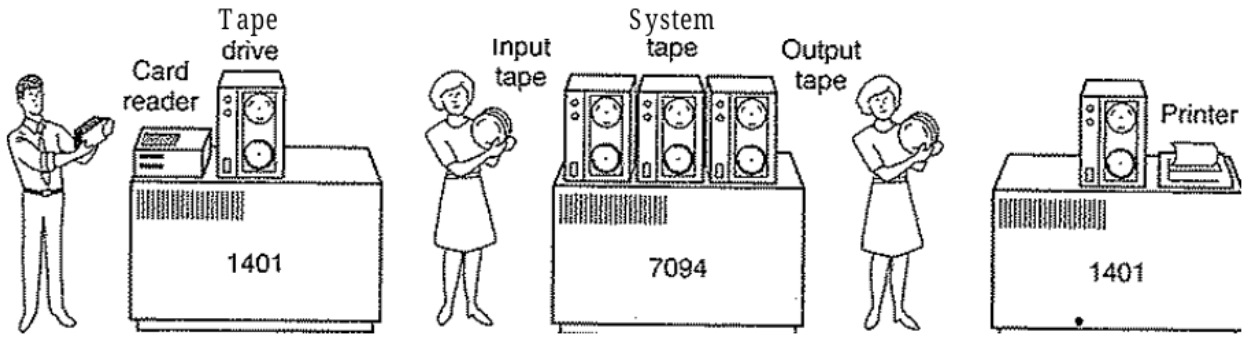
Estas máquinas, hoy conocidas como ***mainframes***, ocupaban grandes habitaciones que debían estar muy bien refrigeradas y debían ser operadas por personal capacitado.

Para salvar los problemas del procesamiento en serie y aprovechar mejor los tiempos, se desarrolló el concepto de sistema operativo en lotes. La idea central, es el uso de una pieza de software denominada **monitor** o **supervisor** que se encargaba de cargar y ejecutar los programas a medida que iban terminando.

De la misma manera que ocurría con los sistemas en serie, para ejecutar un programa (o trabajo), el programador debía enviar el mismo impreso en tarjetas perforadas al operador de la computadora, éste los organizaba y colocaba en el dispositivo de entrada, un lector de tarjetas.

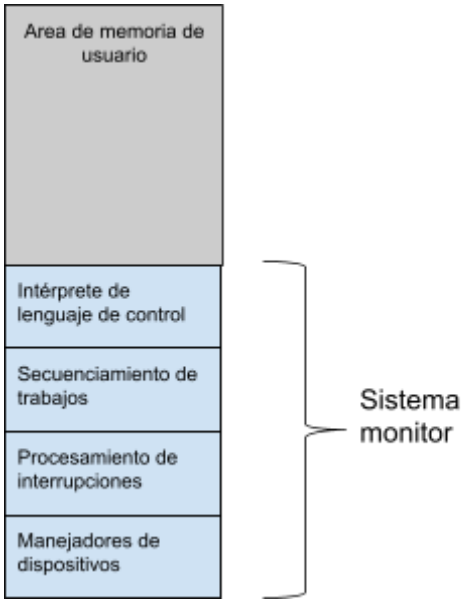
Para tratar de agilizar los tiempos, estos programas eran pasados de las tarjetas perforadas a una cinta magnética que luego era llevada por el operador a otra máquina. Esta segunda máquina es donde residía el ***sistema monitor***. Este sistema, podríamos decir el ancestro del sistema operativo moderno, era el encargado de cargar y ejecutar los programas de la cinta magnética. El resultado de cada programa era grabado en otra cinta magnética, que al finalizar todos los trabajos, el operador debía llevar a otra máquina que leía la cinta e imprimía los resultados para que los programadores pudieran analizarlos.

La ilustración siguiente resume los pasos mencionados:



Este nuevo esquema puede ser analizado desde 2 puntos de vista distintos, el del sistema monitor y el del procesador.

Monitor: es el encargado de controlar la secuencia de eventos. Para ello, gran parte del monitor debe estar siempre cargado en la memoria principal. A esta porción del monitor, se la llama ***residente***. El resto del monitor está formado por utilidades y funciones que se cargan a medida que los programas las requieren. El trabajo del monitor es leer cada uno de los trabajos de la cinta magnética y colocarlo en el área de usuario de la memoria principal. Una vez hecho esto, el monitor cede el control al programa. Una vez que el programa finalizó, el control es devuelto al monitor que cargará el siguiente trabajo.



Ahora bien, desde el punto de vista del procesador, el mismo sólo ejecuta instrucciones. Lo que sucede es que cuando el monitor carga un programa en la memoria de usuario, ejecuta una instrucción de salto que le indica al procesador que la próxima instrucción a ejecutar es la primer instrucción del programa cargado. De la misma forma, cuando el programa finaliza, el procesador encuentra una instrucción que le indica que la próxima instrucción a ejecutar es la siguiente del programa monitor.

Tener en cuenta, que el monitor no deja de ser un programa más, que delega responsabilidad en el procesador para cargar instrucciones de diferentes porciones de la memoria, que de forma alternativa le permiten ceder y recuperar el control.

Con el correr del tiempo, los fabricantes se dieron cuenta que había otras características del hardware que debían implementarse:

Protección de memoria: como mencionamos hay una porción de memoria destinada al sistema monitor y una a los programas de usuario. Este concepto apunta a que ningún programa de usuario modifique o acceda a la porción de memoria destinada al sistema monitor.

Instrucciones privilegiadas: hay instrucciones que pueden generar fallos y desestabilizar el sistema o que permiten acceder a recursos compartidos. Por ello estas instrucciones sólo pueden ser ejecutadas por el sistema monitor.

Interrupciones: esta es una característica muy avanzada, que se implementó mucho después del surgimiento del monitor. Otorga una mayor flexibilidad para ceder y retomar el control entre el sistema operativo y los programas de usuario.

Temporizador: se utiliza para evitar que un programa de usuario monopolice el sistema y evite que el resto de los programas se ejecuten. Cuando el temporizador expira se devuelve el control al sistema monitor.

Los conceptos de instrucciones privilegiadas y protección de memoria dieron origen a los modos de ejecución. Un programa de usuario ejecuta en **modo usuario**, mientras que el sistema operativo ejecuta en **modo núcleo (modo kernel)**.

Sistema en lotes multiprogramado (tercera generación)

A principio de los años 60, y con el surgimiento de los circuitos integrados, IBM desarrolló el equipo System/360 que básicamente integraba los 2 tipos de equipos que existían al momento, los orientados a cálculos matemáticos y los orientados a entrada y salida de datos, ordenamiento, etc. Los descendientes de estos equipos son ampliamente usados aún hoy en bancos y organizaciones con alto procesamiento de datos.

La innovación del System/360 era que tenía la capacidad de procesar varias tareas al mismo tiempo, lo que se dió a conocer como **multiprogramación** o **multitarea.**

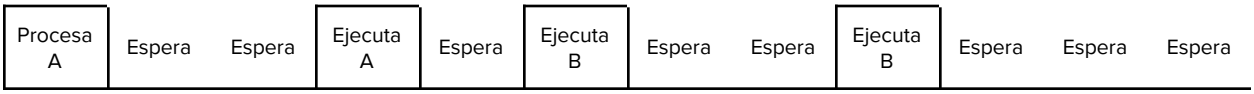
El principio detrás de este concepto es que el procesador se encuentra frecuentemente ocioso, por lo que se desaprovecha tiempo de procesamiento. Este tiempo ocioso, se debe principalmente a las operaciones de entrada / salida que generan los programas, lecturas y escrituras de datos desde y hacia discos (o cintas magnéticas), etc. Para tener una idea sobre este aspecto, consideremos el siguiente ejemplo:

Un programa realiza las siguientes operaciones: Leer un registro.....15 µs	Porcentaje de utilización de CPU: $\frac{1}{31} = 0,032 = 3,2\%$

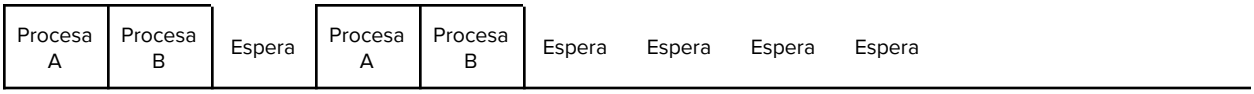
Ejecutar 100 operaciones.....1 µs Escribir un registro.....15 µs Total.....31 µs	Tiempo de CPU ocioso (aproximado): 96%

Bien, esta ineficiencia puede salvarse si consideramos que existe una cantidad de memoria principal tal que admita más de 1 programa cargado al mismo tiempo, además del sistema operativo (monitor residente). En este escenario, suponiendo que tenemos 2 programas cargados en la memoria, cuando el programa “A” realiza instrucciones de entrada/salida, el monitor puede asignar el control del procesador al programa “B” y viceversa.

Ejemplo:



a. sistema monoprogramado (lotes simples)



b. sistema multiprogramado

Al igual que los sistemas monoprogramados dependían de ciertas características del hardware para poder funcionar, los sistemas multiprogramados incorporan 2 características fundamentales para poder funcionar correctamente, que son:

- ❏ Interrupciones de E/S
- ❏ DMA: acceso directo a memoria

Las interrupciones de E/S, permiten que un controlador de un dispositivo de E/S interrumpa el accionar del procesador y que el control del mismo sea devuelto al programa que estaba esperando por esa operación de E/S.

DMA, permite a los controladores de dispositivos de E/S acceder directamente a la memoria principal sin tener que hacerlo a través del procesador.

Sistema de tiempo compartido

Con el advenimiento de los sistemas multiprogramados, el procesamiento se hizo cada vez más eficiente, pero en ciertos casos es deseable que el usuario pueda interactuar directamente con

la computadora, por ejemplo para realizar algún tipo de transacción. Es entonces cuando surge el concepto de sistemas de **tiempo compartido (time sharing).**

En los años 60, las computadoras personales no existían y para realizar trabajos interactivos los usuarios se conectaban desde una terminal o estación de trabajo a una computadora central que, a diferencia de las computadoras que conocemos hoy, ocupaba toda una habitación.

El concepto de *tiempo compartido*, como su nombre lo indica, comparte el uso de procesador entre los distintos usuarios conectados. Es un concepto similar a la multiprogramación pero que permite gestionar múltiples usuarios en vez de programas.

Estos sistemas entrelazan la ejecución de los programas de usuario utilizando pequeños intervalos de tiempo. Por lo tanto, si hay *n* usuarios activos solicitando un servicio a la vez, cada usuario sólo utilizará en promedio *1/n* de la capacidad de procesamiento de la computadora. Esta técnica debe su éxito a la lenta capacidad de reacción de los seres humanos en comparación con la velocidad en los tiempos de respuesta de las máquinas.

¿Cómo funciona un sistema de tiempo compartido?

Cuando el control se asigna a un usuario, el programa de usuario y los datos son cargados por el monitor en memoria principal. Un reloj del sistema genera una interrupción cada una determinada cantidad de segundos, en ese momento el sistema operativo retoma el control y asigna el procesador a otro usuario. Para preservar el estado del programa de usuario que es reemplazado, el programa y sus datos son escritos a disco antes de cargar el nuevo programa.

Para minimizar la escritura a disco, esta operación se realiza sólo en el caso de que el programa entrante requiera más memoria de la que el sistema tiene libre en ese momento.

Tanto la multiprogramación como los sistemas de tiempo compartido plantean nuevos desafíos. Al tener varios programas en memoria al mismo tiempo, se requieren técnicas para que un programa no lea/escriba la memoria de otro. Al tener múltiples usuarios el sistema de archivos debe ser protegido de forma que sólo los usuario autorizados tengan acceso a determinados archivos. Surgen los conflictos entre recursos como impresoras y dispositivos de almacenamiento.

Video recomendado:

- 1963 Timesharing: A Solution to Computer Bottlenecks: https://youtu.be/Q07PhW5sCEk

Conceptos Básicos

El procesador

El procesador, podríamos decir que es el “cerebro” de la computadora. Es quien controla el funcionamiento y realiza las funciones de procesamiento de datos.

Cada procesador posee su propio conjunto de instrucciones y es por esto que un Pentium no puede ejecutar un programa SPARC y viceversa.

El procesador tiene un conjunto de registros (una suerte de memoria interna) utilizados para guardar valores de variables y resultados intermedios de operaciones. Estos registros los podemos clasificar en:

Registros visibles para el usuario

Son registros que están disponibles para todos los programas de usuario y sistema. Son registros que normalmente se utilizan para almacenar datos, direcciones de memoria.

Registros de control y estado

Son registros que se utilizan para controlar el funcionamiento del procesador. El acceso a estos registros se realiza cuando se ejecuta en modo kernel. Algunos de los registros esenciales para la ejecución de programas son:

- PC (contador de programa): contiene la dirección de memoria de la próxima instrucción
- IR (registro de instrucción): contiene la instrucción ejecutada
- PSW (estado): contiene información del estado del programa

El shell

El sistema operativo es el que recibe y lleva a cabo las llamadas al sistema. Los editores, compiladores, ensambladores, enlazadores e intérpretes de comandos definitivamente no son parte del sistema operativo, aunque son importantes y útiles. El shell es un intérprete de comandos, es la interfaz principal entre un usuario sentado en su terminal y el sistema operativo, a menos que el usuario utilice una interfaz gráfica (GUI). Existen muchos shells, incluidos sh, csh, ksh y bash. Todos ellos admiten la funcionalidad que se describe a continuación, que se deriva del shell original (sh).

Cuando un usuario inicia sesión, se inicia un shell. El shell presenta una terminal como entrada y salida estándar. Comienza escribiendo el indicador, el carácter con el signo pesos, que le dice al usuario que el shell está esperando para aceptar un comando. Si el usuario ahora escribe el

comando “*date*”, por ejemplo, el shell crea un proceso hijo y ejecuta el programa fecha como hijo. Mientras se ejecuta el proceso hijo, el shell espera a que termine. Cuando el proceso hijo termina, el shell vuelve a escribir el indicador e intenta leer la siguiente línea de entrada.

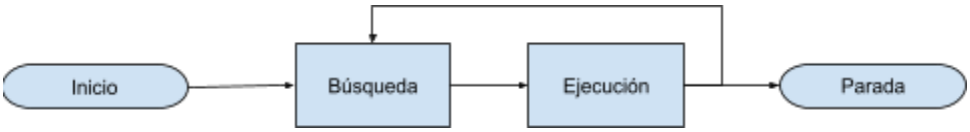
La mayoría de los sistemas operativos modernos, de propósito general, usan una GUI. De hecho, la GUI es solo un programa que se ejecuta en la parte superior del sistema operativo, como un shell.

Ciclo de ejecución de instrucciones

Al principio de cada ciclo, el procesador lee una instrucción de la memoria. El PC almacena la dirección de la siguiente instrucción a leer. El PC siempre se incrementa en 1 para leer la próxima instrucción, salvo que se indique lo contrario.

La instrucción leída se carga en el IR. Esta instrucción indica al procesador la acción a realizar. Éstas acciones pueden ser de tipo:

- **Procesador - memoria**: involucra transferencia de datos desde y hacia la memoria
- **Procesador - E/S**: involucra transferencia de datos desde y hacia un módulo de E/S
- **Procesamiento de datos**: operaciones aritméticas o lógicas
- **Control**: permiten alterar la secuencia de ejecución.



System Calls

Hemos visto que los sistemas operativos tienen dos funciones principales: proporcionar abstracciones a los programas de usuario y administrar los recursos de la computadora. En su mayoría, la primer función trata de proporcionar interfaces que permitan la interacción entre los programas de usuario y el sistema operativo; por ejemplo, crear, escribir, leer y eliminar archivos.

La segunda función, la gestión de recursos es en gran medida transparente para los usuarios y se realiza de forma automática.

La interfaz entre los programas de usuario y el sistema operativo está implementada principalmente como llamadas al sistema o **system calls.** Cada sistema operativo dispone de un conjunto de system calls que varían de uno a otro (aunque los conceptos subyacentes tienden a ser similares).

Es importante tener en cuenta que cualquier computadora con un único CPU puede ejecutar solo una instrucción a la vez. Si un proceso está ejecutando un programa de usuario en modo de usuario y necesita un servicio del sistema, como leer datos de un archivo, tiene que ejecutar una instrucción para transferir el control al sistema operativo. A continuación, el sistema operativo lleva a cabo la llamada al sistema y devuelve el control a la instrucción que sigue a la llamada al sistema.

Interrupciones

En el sentido más amplio, una interrupción es un mecanismo mediante el cual se puede interrumpir la ejecución normal del procesador. Cómo vimos al describir la multiprogramación, las interrupciones permiten realizar un uso más eficiente del procesador.

Hay 4 tipos de interrupciones:

- **De programa**: se producen como resultado de una ejecución de una instrucción. Por ejemplo, desbordamiento aritmético, división por cero, referenciar memoria fuera del espacio del programa, intento de apertura/lectura de un archivo, etc.
- **De E/S**: generada por un controlador de E/S para indicar la finalización de una operación.
- **Por temporizador**: permite al sistema operativo realizar ciertas funciones de forma regular.
- **Por fallo de hardware**

Ciclo de instrucción con interrupciones

Debido a la implementación de interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras que una operación de E/S se lleva a cabo.

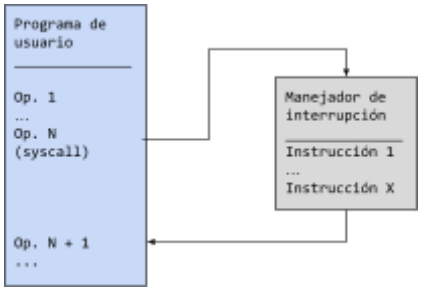
Para explicar este concepto, imagine un programa que debe, luego de procesar algunos datos, imprimirlos a través de una impresora o guardarlos en el disco rígido. Para esto el programa realizará una llamada al sistema para imprimir los datos.

En ese momento el sistema operativo toma el control y ejecuta una rutina de preparación del dispositivo de E/S donde se almacenan los datos que el programa quiere escribir en el buffer del dispositivo. Una vez terminada esta rutina, es decir, que el dispositivo tiene los datos, se comienza con la operación de escritura, momento en el cual el sistema operativo devuelve el control al programa de usuario que realizó la llamada al sistema para que continúe su ejecución.

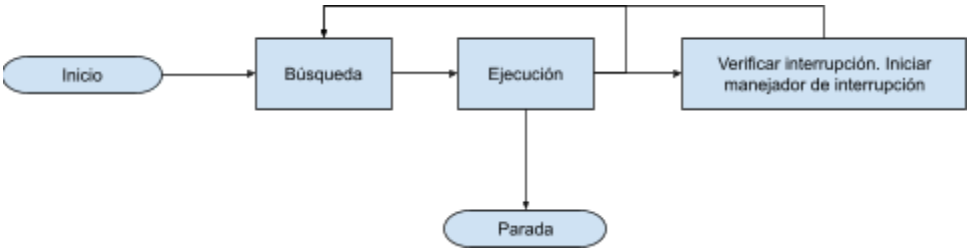
Cuando la operación de E/S termina, es decir que el dispositivo terminó de escribir los datos, el controlador del mismo manda una señal de *petición de interrupción* al procesador para indicar que está listo para realizar otra operación. El procesador termina de ejecutar la instrucción en

curso e interrumpe nuevamente el programa de usuario en ejecución para procesar la interrupción. Tener en cuenta que antes de comenzar a procesar la interrupción, el procesador debe salvar el estado del proceso actual para luego poder reanudar la ejecución.

Gracias a que las rutinas para manejo de interrupciones son parte del sistema operativo, los programas de usuario no deben preocuparse por implementar nada de esto. La ejecución de un programa de usuario se vería típicamente como:



Si al ciclo de instrucción explicado en el punto anterior le sumamos las interrupciones, se ve modificado de la siguiente manera:



Técnicas de comunicación de E/S[2]

Existen 3 técnicas principales para el manejo de la E/S de datos:

1. E/S programada
2. E/S por interrupciones
3. DMA o acceso directo a memoria

E/S programada

Con esta técnica, el procesador es responsable de extraer los datos de la memoria principal en una operación de salida y de almacenarlos en ella en una operación de entrada. El mayor problema o desventaja es que el proceso permanece en un bucle de comprobación constante hasta que haya finalizado la operación (**busy waiting**). Podríamos decir que en cierta forma el

[2] Entrada / Salida

procesador se acopla al dispositivo de entrada-salida y se lo mantiene ocupado innecesariamente.

E/S por interrupciones

Al procesador ser interrumpido por el módulo de E/S, representa una mejora respecto de la E/S programada ya que el procesador no está todo el tiempo consultando si la operación finalizó.

Desde el punto de vista de un módulo de E/S, se recibe la orden de lectura (o escritura) del procesador. Se comienza a leer los datos de un periférico asociado. Una vez que los datos están en el registro de datos del módulo, genera una interrupción al procesador. A partir de ese momento el módulo queda a la espera de que el procesador solicite los datos. El procesador pide los datos al módulo de E/S quien los coloca en el bus de datos.

Desde el punto de vista del procesador, el procesador genera la orden de lectura de datos, salva el contexto de ejecución del programa actual (PC, PSW, etc) y pasa a realizar otra acción. Al finalizar cada instrucción comprueba si hay alguna interrupción, en caso de que la haya, salva el contexto del programa actual y comienza a ejecutar el programa de manejo de interrupciones. El procesador lee la palabra del módulo de E/S y la almacena en memoria, restaura el contexto del programa de usuario que había realizado la petición de datos y reanuda la ejecución.

Si bien esta técnica es más eficiente que la E/S programada, todavía consume tiempo valioso del procesador ya que para que el programa pueda obtener los datos sí o sí interviene el procesador.

DMA

DMA o acceso directo a memoria (por sus siglas en inglés) es una técnica conveniente cuando se requiere transferir grandes volúmenes de datos. Esta técnica funciona de la siguiente manera. Cuando el procesador requiere leer o escribir datos, genera una orden para el módulo de DMA enviando la siguiente información:

1. Tipo de operación (escritura o lectura)
2. Dispositivo de E/S involucrado
3. Posición inicial en donde leer o escribir los datos
4. Cantidad de palabras a leer o escribir.

A continuación el procesador continúa con la ejecución de otro programa. El módulo de DMA, se encargará de transferir los datos desde o hacia la memoria sin utilizar el procesador, por lo que el procesador sólo se involucra al principio y al final de toda la operación.

Los 5 pilares de un sistema operativo

Los 5 conceptos en los que se basa el desarrollo de un sistema operativo son:

1. Procesos
2. Gestión de almacenamiento
3. Protección y seguridad de la información
4. Planificación y gestión de los recursos
5. Estructura del sistema

Procesos

Definición	**¿Qué es un proceso?** Es una unidad de actividad caracterizada por un solo hilo secuencial de ejecución, un estado actual y un conjunto de recursos del sistema asociados.

En otras palabras, un proceso es un programa en ejecución, que tiene asociado un espacio de direcciones de memoria (*address space*) que puede leer y escribir. Este espacio de direcciones contiene las instrucciones, los datos y la pila de ejecución (stack) del proceso. También existen algunos registros asociados como el PC (program counter), SP (stack pointer) y varios más, necesarios para posibilitar la ejecución del proceso.

Cada proceso está formado por 3 componentes:

- Instrucciones
- Datos
- Contexto de ejecución.

El contexto de ejecución o estado del proceso, es el conjunto de datos que utiliza el sistema operativo (junto al procesador) para gestionar y ejecutar el proceso. Son los registros del procesador como PC, IR, PSW. Esta información está separada del proceso ya que contiene información que el proceso no debe poder manipular, como por ejemplo el modo de ejecución, que es un bit que se encuentra dentro del PSW.

Por otra parte, el procesador mantiene, además de los registros mencionados, otros registros que hacen al contexto del programa como ser el registro que guarda la posición inicial de memoria de un proceso (base) y el registro que guarda la posición final de memoria del proceso (límite).

El PC (program counter) y todas las referencias a datos de un proceso se interpretan de forma relativa al registro base; y es gracias a estos 2 registros (base y límite) que el procesador puede realizar las tareas de verificación respecto de que un proceso no lea memoria por fuera de su espacio (*address space*).

Para tener un acercamiento a cómo funcionan los procesos, el mejor ejemplo es pensar en un sistema multiprogramado. Periódicamente, el sistema operativo decide detener la ejecución de un proceso y comenzar a ejecutar otro, por ejemplo, porque el primero ha tenido más tiempo de CPU de lo que le correspondía en el último segundo. Cuando un proceso se suspende temporalmente de esta manera, debe luego reiniciarse exactamente en el mismo estado que tenía cuando se detuvo. Esto significa que toda la información sobre el proceso debe guardarse explícitamente en algún lugar durante la suspensión. Por ejemplo, el proceso puede tener varios archivos abiertos para leer a la vez. Asociado con cada uno de estos archivos hay un puntero que da la posición actual (es decir, el número del byte o registro que se leerá a continuación). Cuando un proceso se suspende temporalmente, todos estos punteros deben guardarse para que una llamada de lectura ejecutada después de reiniciar el proceso lea los datos adecuados. En muchos sistemas operativos, toda la información sobre cada proceso, que no sea el contenido de su propio espacio de direcciones, se almacena en una tabla del sistema operativo llamada tabla de procesos (**process table**), que es una matriz de estructuras, una para cada proceso.

Gestión de almacenamiento

Un sistema operativo posee 5 responsabilidades esenciales respecto del almacenamiento:

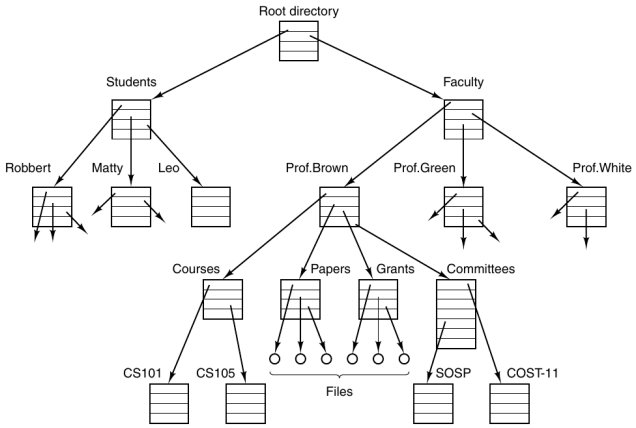
- ❏ Aislamiento de procesos: evitar que los procesos interfieran en el espacio de memoria de otro proceso
- ❏ Asignación y gestión automática: la asignación de memoria debe ser transparente al programador
- ❏ Programación modular: el sistema operativo debe dar soporte a la programación modular.
- ❏ Protección y control de acceso: en ciertos escenarios es deseable que un proceso pueda direccionar la memoria de otro. El sistema operativo debe permitir que varios usuario accedan de distintas formas a porciones de la memoria
- ❏ Almacenamiento a largo plazo: este aspecto lo lleva a cabo a través del uso de archivos y tiene que ver con almacenar información incluso cuando el equipo se haya apagado.

Archivos

El almacenamiento a largo plazo, el sistema operativo lo lleva a cabo a través de la implementación de archivos (ficheros).

Definición	**¿Qué es un archivo?** Es un concepto lógico, conveniente para el programador y es una unidad útil para el control de acceso y protección para el sistema operativo

La mayoría (o todos) los sistemas operativos modernos implementan además, el concepto de directorio como una forma de agrupar archivos. Las entradas de cada directorio pueden ser archivos u otros directorios posibilitando de esta manera, el armado de una estructura jerárquica de archivos y directorios.



La jerarquía mencionada permite establecer permisos de lectura/escritura tanto sobre los archivos como los directorios de forma tal que se pueda permitir o restringir el acceso y las acciones que se realizan sobre ellos.

Un aspecto a destacar, es que cada proceso tiene un directorio de trabajo actual (working directory), a partir del cual se buscan todas las referencias a archivos cuya ruta no comienzan con una barra. Por ejemplo, si /Faculty/Prof.Brown fuera el directorio de trabajo, entonces el uso del nombre “Courses/CSlOl” haría referencia al mismo archivo que si indicáramos el nombre completo de la ruta (PATH absoluto), por ejemplo “/Faculty/Prof.Brown/Courses/CSlOl”.

Cuando una ruta comienza con “/” significa que se debe buscar desde el directorio raíz.

Los procesos pueden cambiar su directorio de trabajo utilizando una llamada al sistema y especificando el nuevo directorio de trabajo. Antes de poder leer o escribir un archivo, el mismo debe abrirse, momento en el que se verifican los permisos. Si se permite el acceso, el sistema devuelve un entero llamado descriptor de archivo para usar en operaciones posteriores. Si el acceso está prohibido, se devuelve un código de error.

Memoria Virtual

Otro aspecto interesante y útil sobre la gestión de almacenamiento que realiza un sistema operativo es el de la ***memoria virtual.*** La memoria virtual, es un concepto que permite a los programas direccionar la memoria desde un punto de vista lógico, es decir, sin importar la cantidad de memoria principal disponible.

La memoria virtual fue concebida como un método para poder mantener múltiples trabajos en memoria principal al mismo tiempo (*multiprogramación*), para acortar los tiempos de espera entre la ejecución de un proceso y otro.

Con el surgimiento de la multiprogramación, los diseñadores de sistemas operativos se vieron forzados a pensar un esquema de asignación de memoria distinto y más eficiente que el usado hasta ese momento con los sistemas por lotes. Hasta ahora sabemos que cuando el sistema operativo necesita cargar un programa y no tiene memoria disponible, lo que hace es sacar un proceso de la memoria y guardarlo en disco. Bueno, este método no es realmente eficiente, por lo que se pensó un esquema donde la memoria esté fraccionada en bloques de tamaño fijo llamados "***página***". De esta forma se daba origen a los sistemas de paginación donde un programa que es cargado en memoria puede ocupar varias páginas y además, cuando se requiere espacio no es necesario reemplazar todo un programa sino que se puede reemplazar sólo algunas páginas del mismo. Este esquema también modificó la forma de direccionar las palabras ya que a partir de ahora se direccionan a través de un número de página más un desplazamiento dentro la página.

¿Qué pasa cuando un programa necesita una instrucción o un dato que se encuentra en una página que fue llevada a disco?

Esto es lo que se conoce como ***page fault*** o fallo de página.

Por cada instrucción o dato requerido por el programa en ejecución, el sistema operativo lo busca primeramente dentro de las páginas del programa cargadas en memoria principal. Si no lo encuentra (recordemos que una dirección de memoria se compone de número de página + desplazamiento dentro de la misma) se dispara un mecanismo para recuperar una o varias páginas de la memoria secundaria que contengan la dirección buscada. Esta página deberá reemplazar a otra página.

Protección y seguridad de la información

El trabajo de un sistema operativo de cara a la seguridad y protección, puede agruparse en:

- Disponibilidad: relacionado con la protección del sistema frente a interrupciones

- Confidencialidad: proteger el acceso de los datos de aquellos que no tienen autorización de acceso.
- Integridad: proteger la modificación de los datos de aquellos que no tienen autorización.
- Autenticidad: verificar la identidad de los usuarios

Planificación y gestión de los recursos

El sistema operativo es responsable por la correcta administración de los recursos (memoria, dispositivos de E/S, procesador) para su uso por los distintos procesos activos. Un sistema operativo debe garantizar:

- Equitatividad: el sistema operativo debe garantizar que los procesos tengan un acceso equitativo a los recursos.
- Respuesta diferencial: el sistema operativo debe poder discriminar los procesos y otorgar distintas prioridades a cada uno.
- Eficiencia: el sistema operativo debe maximizar la productividad, minimizar los tiempos de respuesta.

Para poder cumplir con estos requisitos, el sistema operativo debe poder planificar de forma correcta los procesos que desean ejecutarse. Para esto cuenta con colas de prioridad, que son listas de procesos que esperan usar el procesador. Estas colas pueden ser a largo plazo, donde se almacenan los procesos que ingresan y que aún no están listos para ejecutar, así como también aquellos que quedan esperando por algún dispositivo de E/S. La otra lista es a corto plazo donde se alojan los procesos que están listos para ejecutar. Es responsabilidad del sistema operativo seleccionar uno de los procesos, de la lista a corto plazo, para ser ejecutado.

Estructura del sistema

El tamaño de un sistema operativo y las dificultades que afronta llevaron a tener en cuenta las siguientes cuestiones para su desarrollo:

- Modularidad para organizar el desarrollo, facilidad de mantenimiento, corrección de errores.
- Interfaces bien definidas y sencillas: facilita la programación. Permite modificar un módulo con mínimo impacto en los otros módulos
- Estructura jerárquica (niveles): permite una mejor abstracción de la información, separar sus funciones.

Esta última característica conduce a que cada nivel realice un subconjunto de funciones del sistema operativo y cada nivel delega en el nivel inmediatamente inferior la ejecución de

funciones más primitivas. Dicho en otras palabras, cada nivel proporciona servicios a la capa inmediatamente superior.

La forma en que cada sistema operativo implementa estos conceptos varía enormemente. Una propuesta teórica de dichos niveles es la siguiente:

Nivel	Nombre	Objetos	Parte de
13	Intérprete de comandos	programas de usuario, entornos de programación	Sistema operativo
12	Procesos de usuario		
11	Directorios	Directorios	
10	Dispositivos	Periféricos	
9	Sistema de archivos	Archivos	
8	Comunicaciones	Tuberías	
7	Memoria virtual	Segmentos y páginas	
6	Almacenamiento secundario (discos)	Bloques de datos	
5	Procesos primitivos	Semáforos, lista de procesos	
4	Interrupciones	Programa de gestión de interrupciones	Hardware
3	Procedimientos	Pila de llamadas, registros de activación	
2	Conjunto de instrucciones	Pila de evaluación, intérprete de microprogramas	
1	Circuitos electrónicos	Registros del procesador, buses, etc.	

Cabe aclarar que del nivel 1 al 4 constituyen el hardware del procesador y no forman parte del sistema operativo. Exceptuando el nivel 4 donde se empiezan a mezclar elementos del sistema operativo como el manejador de interrupciones.

Diseño de un sistema operativo

A continuación se describen cinco diseños diferentes que se han probado en la construcción de sistemas operativos. Los cinco diseños son sistemas monolíticos, sistemas en capas, máquinas virtuales, exokernels y sistemas cliente-servidor.

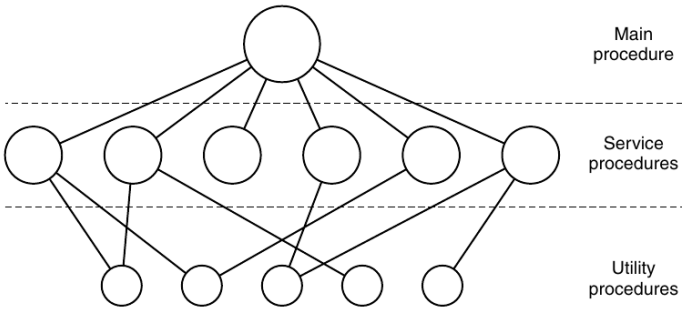
Monolíticos

El sistema operativo está escrito como una colección de procedimientos, cada uno de los cuales puede llamar a cualquier otro procedimiento cuando lo necesite.

Cuando se utiliza esta técnica, cada procedimiento en el sistema tiene una interfaz bien definida en términos de parámetros y resultados, y cada uno es libre de llamar a cualquier otro.

Para construir el sistema operativo cuando se usa este enfoque, primero se compilan todos los procedimientos individuales, o archivos que contienen los procedimientos, y luego se vinculan todos juntos en un solo archivo usando el enlazador del sistema.

En términos de ocultamiento de información, esencialmente no hay ninguno: todos los procedimientos son visibles para todos los demás procedimientos (a diferencia de una estructura que contiene módulos o paquetes, en los que gran parte de la información está oculta dentro de los módulos, y solo los puntos de entrada designados oficialmente pueden llamarse desde fuera del módulo). Sin embargo, incluso en sistemas monolíticos, es posible tener al menos una pequeña estructura. En este modelo, para cada llamada al sistema hay un procedimiento de servicio que se encarga de ello. Esta división de los procedimientos en tres capas se muestra en la siguiente figura:



En capas

Es una generalización del esquema monolítico, donde una capa se construye sobre la anterior. El primer sistema construido con este esquema fue llamado THE (Technische Hogeschool Eindhoven). Fue construído por E. W. Dijkstra (1968) y sus alumnos. El sistema tenía 6 capas

Capa	Función
5	El shell del usuario
4	Gestiona los programas de usuario. Proporciona bibliotecas para manejo de memoria, I/O, etc.
3	Gestión de la Entrada / Salida
2	Maneja la comunicación entre procesos
1	Gestión de memoria
0	Gestión de procesos y multiprogramación

Otro ejemplo que implementó este diseño fue el sistema MULTICS, que se organiza en forma de anillos concéntricos, donde los más internos tenían más privilegios que los más externos. Cuando un proceso de un anillo externo necesitaba invocar un proceso de un anillo interno, debía realizar el equivalente a una llamada al sistema.

Microkernels

Con el enfoque en capas, los diseñadores pueden elegir dónde trazar el límite entre el espacio del usuario y el del kernel. Tradicionalmente, todas las capas iban en el kernel, pero eso no es necesario. De hecho, resulta conveniente que la capa del kernel posea menos responsabilidades porque un error en el kernel puede derribar el sistema instantáneamente. Por el contrario, los procesos de usuario se pueden configurar para que tengan menos poder de acción, de modo que un error no sea fatal.

La idea básica detrás del diseño del microkernel es lograr una alta confiabilidad dividiendo el sistema operativo en módulos pequeños y bien definidos, solo uno de los cuales, el microkernel, se ejecuta en modo kernel y el resto se ejecuta como procesos de usuario. La justificación detrás de este enfoque es que al ejecutar cada controlador de dispositivo y sistema de archivos como un proceso de usuario, un error en uno de estos sólo bloquea ese componente, pero no todo el sistema.

Por el contrario, en un sistema monolítico con todos los controladores en el kernel, un controlador de audio defectuoso puede hacer referencia fácilmente a una dirección de memoria no válida y hacer que el sistema se detenga instantáneamente.

Este diseño de sistema operativo, es particularmente común en sistemas de tiempo real, industrial, aviónica y aplicaciones militares que son de misión crítica y tienen requisitos de

confiabilidad muy altos. Algunos de los microkernels más conocidos son Integrity, K42, L4, PikeOS, QNX, Symbian y MINIX 3.

Cliente-Servidor

Una variación de la idea del microkernel es distinguir dos clases de procesos, los de tipo servidor, que proporciona algún servicio, y los de tipo cliente, que utilizan estos servicios. Este modelo se conoce como modelo cliente-servidor. A menudo, la capa más baja es un microkernel, pero no es necesario.

La comunicación entre clientes y servidores se realiza a menudo mediante el paso de mensajes. Para obtener un servicio, un proceso cliente construye un mensaje diciendo lo que quiere y lo envía al servicio apropiado. El servicio luego hace el trabajo y envía la respuesta.

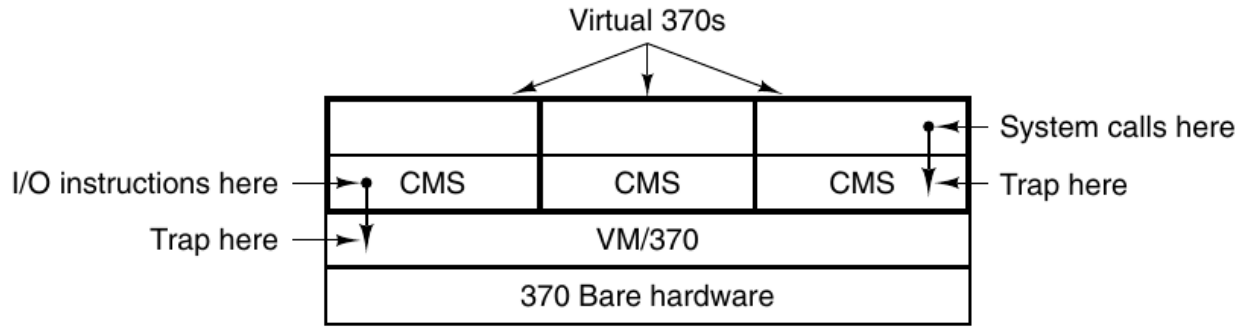
Máquina Virtual

El primer sistema en implementar este tipo de diseño surgió como respuesta a implementar un sistema de tiempo compartido. Originalmente llamado CP / CMS y luego rebautizado como VM / 370, este sistema se basó en la siguiente observación: un sistema de tiempo compartido proporciona:

1. multiprogramación
2. una máquina extendida con una interfaz conveniente para el uso del hardware.

La esencia del VM / 370 era separar completamente estas dos funciones.

El corazón del sistema, conocido como monitor de la máquina virtual, se ejecuta en el hardware básico y realiza la multiprogramación, proporcionando no una, sino varias máquinas virtuales a la siguiente capa, como se muestra en la siguiente figura. Sin embargo, a diferencia de todos los demás sistemas operativos, estas máquinas virtuales no son máquinas extendidas, con archivos y otras características interesantes. En cambio, son copias exactas del hardware básico, incluido el modo kernel / usuario, E / S, interrupciones y todo lo demás que tiene la máquina real.



Debido a que cada máquina virtual es idéntica al hardware real, cada una puede ejecutar cualquier sistema operativo que se ejecute directamente en el hardware básico. Diferentes máquinas virtuales pueden ejecutar, y con frecuencia lo hacen, diferentes sistemas operativos.

Cuando un programa ejecuta una llamada al sistema, la llamada queda atrapada en el sistema operativo en su propia máquina virtual, no en VM / 370, tal como lo haría si se estuviera ejecutando en una máquina real en lugar de una virtual. A continuación, se emiten las instrucciones de E / S de hardware normales para leer su disco virtual o lo que sea necesario para realizar la llamada. Estas instrucciones de E / S son capturadas por VM / 370, que luego las ejecuta como parte de su simulación del hardware real.

Al hacer una separación completa de las funciones de multiprogramación y proporcionar una máquina extendida, cada una de las piezas puede ser mucho más simple, más flexible y más fácil de mantener.

Exokernel

En lugar de clonar la máquina real, como se hace con las máquinas virtuales, otra estrategia es particionarla, en otras palabras, darle a cada usuario un subconjunto de los recursos.

En la capa inferior, que se ejecuta en modo kernel, hay un programa llamado exokernel, su trabajo es asignar recursos a las máquinas virtuales y luego verificar los intentos de usarlos para asegurarse de que ninguna máquina esté tratando de usar los recursos de otra persona. Cada máquina virtual a nivel de usuario puede ejecutar su propio sistema operativo, como en VM / 370 y Pentium virtual 8086s, excepto que cada una está restringida a usar solo los recursos que ha solicitado y asignado.

La ventaja del esquema de exokernel es que ahorra una capa de mapeo. En los otros diseños, cada máquina virtual piensa que tiene su propio disco, con bloques que van desde 0 hasta un máximo, por lo que el monitor de la máquina virtual debe mantener tablas para reasignar direcciones de disco (y todos los demás recursos). Con el exokernel, esta reasignación no es necesaria. El exokernel sólo necesita realizar un seguimiento de qué máquina virtual se ha asignado a qué recurso. Este método todavía tiene la ventaja de separar la multiprogramación (en el exokernel) del código del sistema operativo del usuario (en el espacio de usuario), pero con menos sobrecarga, ya que todo lo que tiene que hacer el exokernel es mantener las máquinas virtuales fuera del alcance de los demás..

Bibliografía utilizada

● William Stallings. Sistemas operativos. Pearson Education. S.A., Madrid, 2005. ISBN-84-205-4462-0.

● AndrewS.Tanenbaum. Modern Operating System. Pearson Education Inc., 2009. ISBN-Q-lB-filBMST-L

● Multilevel Feedback queue. Wikipedia, La enciclopedia libre, 2019 [consulta: 21 de marzo del 2019]. Disponible en https://en.wikipedia.org/wiki/Multilevel_feedback_queue