Unidad 2: SISTEMAS OPERATIVOS.
Introducción.

SISTEMAS OPERATIVOS - INTRODUCCIÓN

¿QUÉ ES UN SISTEMA OPERATIVO?

Como sabemos, el hardware provee los recursos básicos de un sistema de computación. Mediante los programas de aplicación se definen las formas en que se usan esos recursos para resolver problemas para los usuarios. Hay diversos programas de aplicación. Un sistema operativo controla y coordina el uso del hardware entre los distintos programas de aplicación para los diversos usuarios.

El Sistema Operativo provee los medios para utilizar de forma correcta los recursos de un sistema de computación; no hace ninguna función útil por sí mismo. Simplemente provee un entorno en el cual otros programas pueden hacer trabajo útil.

Podemos ver al SO como un ***asignador de recursos***.

Un sistema tiene diversos recursos que pueden requerirse para resolver un problema: tiempo de CPU, espacio de memoria, espacio de almacenamiento de archivos, procesos, dispositivos de E/S, etc. El SO actúa como el gerente de esos recursos y se los otorga a programas específicos y usuarios a medida que son necesarios. Como puede haber diversos pedidos conflictivos para recursos, el SO tiene qué decidir que pedidos son atendidos y cuáles no. Su ***fuente principal de trabajo*** es un conjunto de ***tablas*** en las cuales ***se describen los recursos y a qué procesos están asignados los mismos***.

Otro punto de vista de los SO apunta a la necesidad de ***controla***r los distintos ***dispositivos de E/S y programas del usuario***.

Un SO es un programa de control. Un programa de control controla la ejecución de programas de usuario para prevenir errores y uso impropio de la computadora. Está relacionado especialmente con la operación y control de dispositivos de E/S.

DEFINICIÓN DE SISTEMA OPERATIVO

Un Sistema Operativo es un conjunto de módulos o funciones (software) que actúan como interfase entre el usuario de una computadora y el hardware de la misma. El propósito es proveer un en torno en el cual el usuario puede ejecutar programas.

OBJETIVOS DEL SISTEMA OPERATIVO

El ***objetivo principal*** de un SO es lograr que el Sistema de computación se use de manera cómoda, y el ***objetivo secundario*** es que el hardware del computador se emplee de manera eficiente. Todo esto se verá con mayor profundidad más adelante.

EVOLUCIÓN DE LOS SO

Para ver qué son los SO, estudiemos como fueron construidos en los últimos años. Viendo esta evolución, podemos identificar elementos en común, y cómo fueron mejorando.

Los SO y la arquitectura de los computadores tienen una gran influencia el uno sobre el otro. Para facilitar el uso del hardware se construyeron los SO. A medida que se diseñaron y usaron SO, se volvieron obvios

ciertos cambios en el diseño del hardware que simplificaron los sistemas operativos. En esta revisión histórica veremos como la introducción de nuevo hardware es la solución natural para muchos de los problemas de sistemas operativos.

1) LOS PRIMEROS SISTEMAS

Inicialmente sólo hubo hardware. Las primeras computadoras eran máquinas muy grandes que se programaban desde una consola. El programador podía escribir un programa, y luego operar el programa directamente desde la consola del operador.

Primero, el programa tenía que cargarse manualmente en la memoria, ya sea por medio de switches (llaves de conmutación), cinta de papel, o tarjetas perforadas. Luego, se apretaban los botones apropiados para cargar la dirección de comienzo y comenzaba la ejecución del programa. A medida que el programa corría, el programador/operador podía monitorear su ejecución por medio de luces en la consola. Si se descubrían errores, el programador podía parar el programa, examinar los contenidos de la memoria y registros, y corregir el programa directamente desde la consola. La salida se imprimía o perforaba en cintas o tarjetas para una impresión posterior. Un aspecto importante de este entorno era la naturaleza interactiva HANDS-ON.

El programador era el operador. Muchos sistemas usaban un esquema de reserva para otorgar tiempo de maquina: para usar la maquina se pedía un turno en una hoja de papel.

Esta aproximación tiene ciertos problemas: supongamos que uno reserva una hora, y necesita más que ese tiempo para hallar un error. Uno tiene que parar, recolectar lo que se pueda, y volver mas tarde para continuar (consideremos que no existían ensambladores, y mucho menos compiladores). Por otro lado, uno podía terminar antes de que se terminara su tiempo, quedando el resto del tiempo la máquina sin ser usada.

A medida que pasó el tiempo, se construyó software y hardware adicional. Lectoras de tarjeta, impresoras de línea y cintas magnéticas se convirtieron en lo más común. Se diseñaron assemblers, cargadores y linkers para facilitar la programación, así como también bibliotecas de funciones comunes. Estas podían copiarse en un nuevo programa sin tener que escribirse dos veces.

Las rutinas que hacen entrada y salida son especialmente importantes. Cada dispositivo de E/S nuevo tiene sus características propias, requiriendo una programación cuidadosa. Para cada dispositivo se escribe una subrutina especial, llamada el manejador del dispositivo (device driver). Este programa conoce como deben usarse los buffers, banderas, registros, bits de control y estado para un dispositivo en particular. Cada tipo distinto de dispositivo tiene su propio manejador. En vez de escribir el código necesario cada vez, el device driver se utilizaba directamente de la biblioteca. Posteriormente aparecen compiladores (Fortran, Cobol, etc.) que facilitan la tarea de programación, pero dificultan la tarea de operación de la computadora.

Para preparar un programa Fortran para ejecución había que ejecutar los siguientes pasos:

- Montar la cinta magnética o cargar las tarjetas perforadas que contenían el Compilador Fortran.
- Cargar el compilador Fortran.
- Leer el código fuente del programa de tarjetas perforadas y escribirlo en otra cinta magnética.
- Compilar el programa. El compilador Fortran producía una salida en lenguaje ensamblador.
- Montar la cinta que contiene el Assembler.
- Ensamblar el programa.
- Linkeditar la salida del programa para incluir rutinas de bibliotecas y subrutinas.
- Cargar el programa ejecutable y ejecutarlo, para su corrección.

Para ejecutar un trabajo hay una gran cantidad de tiempo de preparación (setup time), debido a la gran cantidad de pasos de que consta el trabajo. Si ocurre un error en algún paso, hay que comenzar nuevamente desde el comienzo, teniendo que cargar cintas o tarjetas.

2) MONITOR SIMPLE O SISTEMA BATCH SENCILLO

Vimos que el tiempo de preparación es un problema importante. Durante el tiempo de montado de cinta, o mientras el operador trabaja en la consola, la CPU esta desocupada. En esos tiempos, un computador era muy caro, con un tiempo de vida esperado corto. Por lo tanto, el tiempo de computación era muy valioso, y los propietarios querían utilizarlo lo más posible.

Hubo una solución doble:

- la primera fue contratar operadores profesionales, de tal forma que el programador no operara más la computadora, y eliminar el tiempo desperdiciado por las reservas. Los operadores no saben corregir los programas: lo único que hacen es obtener un vuelco de memoria y registros para el programador, y así poder seguir trabajando.
- La segunda solución trató de reducir el tiempo de preparación. Los trabajos con necesidades similares eran loteados (batched) juntos, como un solo grupo. Si el operador recibe varios trabajos Fortran y otros tantos Cobol, en vez de ejecutarlos en orden de llegada ejecuta los programas Fortran juntos, y los Cobol juntos, y se gana el tiempo de carga de los compiladores.

A pesar de mejorar la utilización, sigue habiendo problemas: si para un programa, el operador tiene que mirar la consola para ver la condición de parada, si fue un error, hacer un dump (vuelco de memoria), y luego cargar la lectora con el próximo trabajo. Durante este tiempo, la CPU se mantiene sin uso.

Para solucionar este problema se introduce el ***secuenciamiento automático de trabajos***, y con el, se crean los ***primeros sistemas operativos rudimentarios***. La idea es que un programa pequeño, llamado monitor residente, transfiera automáticamente el control de un trabajo al siguiente.

Inicialmente, el monitor tiene el control del computador. El mismo transfiere el control a un programa. Cuando éste termina, retorna el control al monitor, que dará control al próximo programa. El monitor tiene que saber qué programa ejecutar. Con este fin se introducen las ***tarjetas de control***, para dar información directamente al monitor. Estas son tarjetas especiales que se mezclan con las de datos o programa, y son directivas que indican qué programa se ejecutará y qué recursos necesitará.

Las ***tarjetas de control*** tienen una identificación especial para diferenciarlas de las de datos

Los errores son atrapados (interrupciones), y reciben un tratamiento adecuado (por ejemplo, un dump), y luego se sigue ejecutando la próxima tarjeta de control.

Para evitar que un programa, por error, lea tarjetas de control del siguiente trabajo, se utiliza un manejador de dispositivos (o sea que el encargado de leer es el Monitor), y esta instrucción (lectura) es una instrucción privilegiada.

Para hacer un pedido al SO, existen las llamadas al supervisor. Estas provocan una interrupción

3) BATCH SOFISTICADO (PERFORMANCE)

La aparición de sistemas batch mejora la utilización de los sistemas, y mejoran la performance y el rendimiento. A pesar que el secuenciamiento automático elimina la manipulación humana, que es muy lenta, la CPU sigue estando poco utilizada. El problema reside en los dispositivos de E/S, que al ser mecánicos son mucho más lentos que la CPU.

Por ejemplo, un assembler puede procesar 300 o más tarjetas por segundo, mientras que una lectora muy veloz puede leer solo 1200 tarjetas por minuto. Entonces, para ensamblar un programa de 1579 tarjetas, se tarda 4.8 segundos de ensamblado, mientras que 78.9 segundos se ocupan en leer tarjetas. El procesador espero 74.1 segundos, o sea, un desperdicio del 93.9%.

El mismo problema ocurre para las operaciones de salida. El problema es que mientras ocurre una operación de E/S, la CPU está desocupada esperando que termine la E/S, y mientras la CPU está ejecutando, los dispositivos de E/S están libres.

3.1) Operación offline

Una solución para el problema anterior fue reemplazar las lectoras de tarjetas e impresoras, con unidades de cinta magnética. En vez que la CPU leyera directamente de las tarjetas, las mismas primero se copiaban en una cinta magnética. Cuando ésta estaba suficientemente llena, se llevaba a la computadora. Cuando un programa quería leer una tarjeta, lee la cinta. Similarmente, la salida se hace en una cinta para imprimirse posteriormente. Las lectoras e impresoras operaban offline, no en la computadora principal.

La ventaja principal es que la CPU se libera de la espera de lectura de tarjetas, que son muy lentas, y la misma se hace por medio de cintas, que son mucho más rápidas. Además no son necesarios cambios en los programas de aplicación: los mismos llaman al driver de la lectora de tarjetas, que se reemplaza por un driver de lectura de cinta magnética. De esta forma, los programas de aplicación no deben ser cambiados: sólo el driver. Los programas usan dispositivos de E/S lógicos.

La ventaja real en la operación offline es la posibilidad de usar múltiples lectoras-escritoras magnéticas para una sola CPU. Si la CPU puede procesar entrada a dos veces la velocidad de la cinta, entonces dos lectoras trabajando simultáneamente pueden producir suficiente cinta para mantener a la CPU ocupada.

3.2) Buffering

Esta es otra solución a la lentitud de los dispositivos de E/S. La idea es muy simple: después que se leyó un dato, y la CPU está operando en el mismo, el dispositivo de entrada comienza a leer el próximo dato inmediatamente. La CPU y el dispositivo están ocupados simultáneamente. Un buffering similar se puede hacer para la salida de datos.

La unidad natural de datos es el registro. Puede ser un registro físico (una línea de impresión, un carácter del teclado, o un bloque de una cinta), o un registro lógico (una línea de entrada, una palabra, o un arreglo). Los registros lógicos están definidos por la aplicación; los físicos, por la naturaleza del dispositivo de E/S. Los registros son las unidades de datos usados para el buffering.

En la práctica, es difícil que el buffering pueda mantener ocupados simultáneamente a la CPU y a los dispositivos de E/S. Si la CPU está trabajando en un registro, mientras que un dispositivo está trabajando con otro, o la CPU o el dispositivo pueden terminar primero.

Si la CPU termina primero, tiene que esperar, porque no puede procesar otro registro hasta que el dispositivo no haya terminado con el anterior. Si el dispositivo termina primero, o espera o puede proceder a leer otro registro. Los buffers que contienen registros que ya han sido leídos y no procesados, generalmente se usan para poder mantener varios registros.

El problema principal del buffering es detectar que terminó una E/S tan pronto como sea posible, para poder lanzar la próxima. La solución para este problema son las interrupciones. Cuando un dispositivo de E/S termina con una operación, interrumpe a la CPU. Esta para de hacer lo que este haciendo, y se hace una transferencia de control a la rutina de atención de la interrupción. Esta rutina chequea si el buffer no está lleno (para un dispositivo de entrada) o vacío (para uno de salida), y lanza el próximo pedido de E/S. Entonces, la CPU resume el cálculo interrumpido. De esta forma, los dispositivos de E/S y la CPU operan a máxima velocidad.

El buffering afecta la performance suavizando las diferencias en las variaciones de tiempo que toma procesar un registro. Si, en promedio, las velocidades de CPU y de dispositivos de E/S son similares, el buffering permite que ambos procesen casi a velocidad máxima.

Sin embargo, si la CPU es mucho mas rápida que un dispositivo, el buffering no afecta demasiado la performance, porque la CPU tendrá que esperar que el dispositivo se libere aún cuando todos los buffers disponibles estén llenos.

Esta situación también ocurre con trabajos limitados por E/S (I/O-bound), donde la cantidad de E/S en relación con los cálculos es muy grande. Como la CPU es más rápida que los dispositivos, la velocidad de ejecución está limitada por la velocidad del dispositivo de E/S. Si, por otro lado, en el caso de trabajos limitados por CPU (CPU bound), donde la cantidad de cálculos es tan grande que los buffers de entrada están siempre llenos, y los de salida vacíos, la CPU no puede emparejarse con los dispositivos de E/S.

3.3) Spooling

Esta es la mejor solución para todos los problemas mencionados anteriormente.

En muchos sistemas se comenzó a reemplazar los sistemas offline y a utilizarse discos, al estar estos más disponibles. Los mismos mejoraron la operación offline, al ser más rápidos que las cintas. El problema con éstas es que no se puede escribir en un extremo de la misma mientras la CPU lee del otro. Los discos eliminan este problema: moviendo la cabeza de un área del disco a la otra, sin el problema del rebobinado, el disco puede cambiar del área de tarjetas que se están usando, para almacenar nuevas tarjetas en otra posición.

Basado en esta ventaja se crean los sistemas de SPOOL (Simultaneous Peripheral Operation On-Line). Se leen las tarjetas directamente desde la lectora en el disco. La ubicación de la imagen de las tarjetas en el disco se almacena en una tabla del sistema operativo. Cada trabajo se almacena en la tabla. Cuando se ejecuta un trabajo, sus pedidos para la lectora de tarjetas se satisfacen leyendo su imagen desde el disco. De la misma forma, cuando se pide la impresora, esa línea se copia en el disco. Cuando el trabajo termina, se imprime la salida completa.

El buffering superpone la E/S de un trabajo con sus propios cálculos. La ventaja del spooling es que superpone la E/S de un trabajo con los cálculos de otros trabajos. Esto tiene un efecto directo sobre la performance: la CPU y los dispositivos de E/S se mantienen ocupados con tasas muy altas, particularmente si hay una mezcla de trabajos CPU-bound e I/O-bound.

Además, el spooling provee una estructura de datos importante: una cola de trabajos. El spooling resulta en varios trabajos que han sido leídos y esperan en el disco, listos para ser ejecutados. Esta cola permite al sistema operativo elegir qué trabajo ejecutar luego, para aumentar la utilización de la CPU. Si los trabajos vienen directamente en tarjetas, o aún en una cinta, no es posible ejecutar trabajos en distinto orden al de llegada. En cambio, al tener los trabajos en un dispositivo de acceso directo surge la posibilidad de hacer una Planificación de Trabajos. Esta selección de programas lleva a la multiprogramación, o sea, tener más de un programa cargado en memoria.

---

4). MULTIPROGRAMACION

Es un intento de aumentar la utilización de la CPU, logrando que ésta siempre tenga algo que ejecutar. La idea es la siguiente: el SO levanta uno de los trabajos de la cola de trabajos y comienza a ejecutarlo. En el momento en que tenga que esperar por algo (montar una cinta, escribir algo en el teclado, terminar una operación de E/S), la CPU se entrega a otro trabajo, y así sucesivamente. En un sistema monoprogramado, la CPU hubiera estado desocupada, a pesar de que hay trabajo que hacer. Prácticamente por este tema es que surgen todos los administradores a estudiar, y lo que hace (de acuerdo a sus algoritmos), la mayor o menor eficiencia de un SO.

---

5). TIME SHARING

Los primeros sistemas batch consisten en el loteo de trabajos similares. Las tarjetas y cintas sólo permiten acceso secuencial a los programas y datos, y sólo se puede usar una aplicación a la vez.

Cuando aparecen los discos, se puede acceder simultáneamente a todas las aplicaciones. Ahora, los sistemas batch no están más definidos por el loteo de trabajos similares, sino por otras características. La característica principal de un sistema batch es la falta de interacción entre el usuario y el trabajo mientras está ejecutando. El trabajo se prepara y un tiempo más tarde, aparece la salida. Como los usuarios no pueden interactuar con sus trabajos mientras están ejecutando, deben colocar tarjetas de control para manejar todos los resultados posibles. En un trabajo multietapa, el resultado puede depender de los resultados anteriores. Puede ser difícil definir que hacer en todos los casos. Otra desventaja es que los programas deben corregirse en forma estática, por medio de dumps de memoria. Un programador no puede modificar un programa a medida que se ejecuta.

Un sistema Interactivo o Hands-on provee comunicación on-line entre el usuario y el sistema. El usuario da instrucciones al sistema operativo o a un programa directamente, y recibe una respuesta inmediatamente, por medio de una terminal. Cuando el sistema operativo termina la ejecución de un comando, pide la

próxima "tarjeta de control", no de una lectora de tarjetas sino del teclado. El usuario ejecuta un comando, espera la respuesta,y decide cuál será el próximo comando, basado en el resultado del anterior.

Los sistemas batch son apropiados para ejecutar trabajos grandes, que no tienen interacción. Los trabajos interactivos tienden a estar compuestos por muchas acciones cortas, donde los resultados de un comando puede ser impredecibles. Como el usuario se queda esperando el resultado, el tiempo de respuesta tiene que ser muy corto. Un sistema interactivo está caracterizado por el deseo de un tiempo de respuesta corto.

Como vimos, los primeros sistemas eran interactivos, pero se perdía mucho tiempo de CPU. Los sistemas de Tiempo Compartido son el resultado de tratar de obtener un sistema interactivo a un costo razonable. Estos sistemas usan multiprogramación y planificación de CPU, para que cada usuario tenga una parte pequeña del tiempo de la computadora. Cada usuario tiene un programa separado en memoria.

Un sistema de tiempo compartido permite a los usuarios compartir simultáneamente la computadora. Como cada acción en un sistema de tiempo compartido tiende a ser corta, solo hace falta un tiempo corto de CPU para cada usuario. Como el sistema cambia rápidamente de un usuario al otro, los usuarios tienen la impresión que cada uno tiene su computadora propia.

---

6) SISTEMAS DE TIEMPO REAL

Un sistema de tiempo real generalmente se usa como un dispositivo de control en una aplicación dedicada. Hay sensores que dan datos a la computadora, que los analiza y puede ajustar controles para modificar las entradas del sensor.

Se utilizan para sistemas médicos, controles industriales, controles de experimentos científicos, etc. La característica más importante de estos sistemas es que tienen restricciones de tiempo bien definidas, y el procesamiento tiene que hacerse dentro de ese tiempo.

---

7) MULTIPROCESAMIENTO

Son sistemas con más de una CPU, compartiendo memoria y periféricos. Se usan dos aproximaciones.

- La más común es asignar a cada procesador una tarea específica. Un procesador central controla el sistema, y los demás tienen tareas específicas, o le piden instrucciones al procesador principal. Este esquema define una relación maestro/esclavo. Ejemplos de estos sistemas son aquellos que usan un Procesador Frontal (Front-End Processor) para manejar lectoras e impresoras a alguna distancia del procesador central. Estos sistemas están compuestos generalmente de un computador grande, que es la computadora principal (Host o Mainframe), y un computador más pequeño que es responsable de la E/S de la terminal.
- El otro tipo común de multiprocesamiento es el uso de redes. En éstas, varias computadoras independientes pueden comunicarse, intercambiar archivos e información. Cada computador tiene su propio sistema operativo, y opera independientemente.

FUNCIONES (SERVICIOS) QUE BRINDAN LOS SISTEMAS OPERATIVOS

Un sistema operativo provee ciertos servicios a los programas y a los usuarios. Estos servicios difieren de un sistema a otro, pero hay cierta clase de servicios que pueden identificarse:

- Ejecución de programas: El SO debe ser capaz de cargarlos, darles el control y determinar su fin normal o anormal.
- Operaciones de E/S: El programa del usuario no puede ejecutar operaciones de E/S directamente, por lo tanto el sistema operativo tiene que proveer ciertos medios para hacerlo.
- Manipulación del Sistema de Archivos: Los programas quieren leer y escribir archivos. Esto se hace por medio del sistema de archivos.

- Detección de errores: Estos pueden ocurrir en la CPU y memoria, en dispositivos de E/S, o en el programa del usuario. Por cada tipo de error, el sistema operativo tiene que tomar un tipo de acción distinto.
- Administración de recursos: Cuando hay varios usuarios ejecutando al mismo tiempo, hay que darles recursos a cada uno de ellos. El SO maneja distintos tipos de recursos: CPU, memoria principal, archivos, dispositivos.
- Accounting: Se quiere contabilizar qué recursos son usados por cada usuario. Estos datos, usados con fines estadísticos para configurar el sistema, pueden mejorar los servicios de computación. También se utilizan estos datos para cobrar a cada usuario los recursos utilizados.
- Protección: Los dueños de la información la quieren controlar. Cuando se ejecutan distintos trabajos simultáneamente, uno no debe poder interferir con los otros. Las demandas conflictivas para determinados recursos, deben ser ad- ministradas razonablemente.

Los servicios de los sistemas operativos son provistos de distintas formas.

Dos métodos básicos son:

Llamadas al Sistema.

El nivel más fundamental de los servicios se maneja por medio del uso de llamadas al supervisor. Estas proveen la interfaz entre un programa en ejecución y el sistema operativo. Generalmente están disponibles como instrucciones del lenguaje ensamblador.

Podemos reconocer, básicamente, los siguientes tipos de llamadas al supervisor:

Control de Procesos

- Fin, Dump.
- Carga de otro programa (Load).
- Crear procesos, Terminar proceso.
- Esperar un tiempo.
- Esperar por un evento o una señal.

Manipulación de Archivos

- Crear, borrar archivos.
- Apertura y cierre de archivos.
- Lectura y escritura.

Manipulación de Dispositivos

- Pedir un dispositivo, liberar un dispositivo.
- Leer, escribir.

Mantenimiento de Información

- Fecha, Hora.
- Atributos de Procesos, Archivos, o Dispositivos

Hay distintas formas de implementar las llamadas al supervisor dependiendo de la computadora en uso. Puede hacerse como interrupciones o como llamadas a un servidor (server)

Es necesario identificar la llamada al sistema que se quiere hacer, y otro tipo de informaciones. Por ejemplo, para leer un registro, hace falta especificar el dispositivo o archivo a usar, y la dirección y longitud de memoria donde copiar el registro.

Programas del Sistema

Programa de ayuda a la operación de la computadora.

Ÿ **Programa cargador (loader):** Forma parte del núcleo (Kernel) del SO y su función es cargar el resto de las rutinas del SO cuando se lo requiera y también los programas del usuario. Asigna espacio en memoria central para programas; carga o introduce la instrucción en memoria junto con sus datos; encadena las referencias entre lotes de objetos; reasigna y ajusta todas las posiciones dependientes del direccionamiento.

Ÿ **Programa cargador inicial (IPL):** A través de una tecla de la consola o del teclado se manda un impulso que pone en marcha al I.P.L., quién carga el núcleo del SO en memoria central cada vez que el sistema se pone en marcha.

- **Cargadores absolutos o reubicables:** Es un software que coloca las instrucciones del programa que se pretende ejecutar con sus datos en las locaciones de Memoria Central. Un cargador absoluto lo hace en las locaciones indicadas por el programa a ejecutar en lenguaje de máquina; mientras que un cargador con reubicación puede cargar un programa en diversos lugares de memoria de acuerdo a la disponibilidad de espacio en el momento de su carga.

Ÿ **Intérprete de mandatos:** Su función principal es obtener el siguiente mandato especificado por el usuario y ejecutarlo. Hay dos maneras generales para poner en práctica estos mandatos:

- **Programa de control de trabajo (Job Control):** Puede o no estar en el núcleo y sus funciones son encadenar el flujo de trabajo; interpretar los comandos; suministrar al "loader" las referencias de ubicación del próximo programa a cargar. En general se busca reducir la intervención humana en un flujo de trabajos por lotes (Batch), a ser propuesto para ejecutar, mediante acciones automáticas al SO Estas acciones son incorporadas en el flujo de los trabajos y esto se hace mediante una serie de sentencias que se ocupan de:
  a) Comienzo y finalización de un trabajo (Job).
  b) Ordenes para cargar y ejecutar los programas.
  c) Ordenes de reserva de recursos o tiempos de uso de los mismos. Se ejecutan con un interprete asociado al Kernel o monitor.

- **Programa supervisor, núcleo o Kernel (ejecutivo, monitor o residente):** Establece las comunicaciones con el operador enviando o recibiendo los mensajes a través de la consola; organiza la operación de los periféricos y a través de los canales atiende las interrupciones, contiene las llamadas al sistema con las que administra todos los incidentes tales como errores de programas, fallas en los dispositivos, errores del sistema, etc.

Programas de ayuda a la programación

- **Compiladores (Compilers**): son un conjunto de programas que permiten la traducción de los lenguajes de alto nivel que que generalmente acompañan al equipo, al lenguaje de esa máquina o sea su juego de instrucciones. Hay tantos compiladores como lenguajes haya disponibles. Se construyen a medida para cada SO y por ello se los denominan programas utilitarios.

  Hay dos tipos de compiladores:

  - los rápidos y sucios, que producen un programa objeto en un código poco eficiente;
  - los optimizadores que generan un código de máquina de alta calidad y eficientizados, aunque son algo más lentos.

- **Interpretes**: Son traductores de instrucción de programa fuente a instrucción del lenguaje de máquina y no producen programas objetos evitando con ello el ensamblado o la compilación y su correspondiente ahorro de tiempo.

- **Editores de Enlaces (Linkeditors):** Son programas que permiten crear un programa ejecutable resolviendo las referencias de objetos del código objeto (salida del compilador) mediante direcciones definitivas a los objetos reubicables. También incorporan en el programa ejecutable las rutinas de bibliotecas (librery) necesarias para su ejecución.

- En general a estos programas (compiladores y editores de enlaces) se suelen englobar como programas ***traductores*** ya que se ocupan de traducir lenguajes o vinculaciones.

- **Debuggers**: son programas que permiten encontrar y depurar fácilmente errores en otros programas.

Programas de ayuda al manejo de funciones de Entrada/Salida o bibliotecas (Drivers).

Son programas de gestión de entradas y salidas, compuestos por:

1) Rutina de transferencia entre el canal o interfase de operaciones de E/S y la Memoria central o la manipulación de dispositivos.
2) Rutinas de verificación de los rótulos o directorios de los archivos.
3) Rutinas de desbloqueo de los registros físicos para suministrar los registros lógicos a las zonas de trabajo.
4) Rutinas de bloqueo.
5) Y otras rutinas más que dependen del proveedor del SO o del periférico.

Tipos de Sistemas Operativos

Existen distintas arquitecturas computacionales y por ende, distintos tipos de SO. Proponemos una clasificación de ellas:

Según la cantidad de tareas simultáneas que puede realizar:

Sistema Operativo Multitareas

Es el modo de funcionamiento disponible en algunos sistemas operativos, mediante el cual una computadora procesa varias tareas al mismo tiempo.

Sistema Operativo Monotarea

Los sistemas operativos monotareas son más primitivos y es todo lo contrario al visto anteriormente, es decir, solo pueden manejar un proceso en cada momento ya que solo puede ejecutar las tareas de una en una. Por ejemplo cuando la computadora esta imprimiendo un documento, no puede iniciar otro proceso ni responder a nuevas instrucciones hasta que se termine la impresión.

Según la cantidad de Usuarios que soporta:

Sistema Operativo Monousuario

Los sistemas monousuarios son aquellos que nada más pueden atender a un solo usuario, debido a las limitaciones creadas por el hardware, los programas o el tipo de aplicación que se este ejecutando.

Estos sistemas se basan en máquinas virtuales que admiten a un sólo usuario que utiliza todos los recursos sin compartirlos simultáneamente con otros. Es el caso de algunas computadoras personales o estaciones de trabajos específicas.

Sistema Operativo Multiusuario

Es todo lo contrario a monousuario. En esta categoría se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, que comparten mismos recursos. Este tipo de sistemas se emplean especialmente en redes.

Son ejemplos de estos SO:

a) Sistemas de Consultas de Información

Estos sistemas se caracterizan por las consultas triviales que se efectúan sobre archivos o bases de datos con el objeto de obtener una información dada en un tiempo relativamente corto. Es el caso de Bases de datos médicas, o bibliográficas, etc.

b) Sistemas de Gestión de Operaciones.

Se caracterizan por trabajar sobre una base de datos frecuentemente modificada, posiblemente varias veces por segundo, tal el caso de la gestión de operaciones bancarias o reservas de pasajes aéreos. Los principales problemas en éstos sistemas reside en mantener la información permanentemente actualizada y las operaciones simultáneas sobre los mismos datos. Estos problemas los debe resolver el SO.

Según la cantidad de Procesadores que intervienen en los procesos:

Sistema Operativo Multiprocesador

Dos o más procesadores centrales del mismo tipo pueden trabajan paralelamente; es decir que dos o más programas pueden ejecutarse en el mismo momento porque existen dos o más CPU que comparten una memoria central común.

Sistema Operativo Monoprocesador

Sólo permite la existencia de un procesador que es el encargado de realizar toda la tarea.

Según sus aplicaciones, los SO se pueden dividir en:

Sistemas de Propósito General.

Son los que proporcionan una amplia gama de servicios y deben adaptarse a cualquier ambiente, tipo de aplicaciones, modos de operación, dispositivos, etc.)

Los Sistemas Operativos de propósito general se emplean en computadores que soportan una amplia gama de aplicaciones. Estos Sistemas están diseñados para mantener un flujo constante de trabajo en forma de tareas a ser ejecutadas por la máquina. Debido al gran número, y diversidad de trabajos, el Sistema debe proveer soportes utilitarios y facilidades para soportar una gran cantidad de unidades periféricas. La disponibilidad y el control de éstos utilitarios constituyen, junto con la organización del flujo de trabajo, las principales funciones de un Sistema Operativo de propósito general.

Los Sistemas de propósito general pueden clasificarse en dos grandes grupos:

a) Sistemas Batch

Los Sistemas Batch se caracterizan por el hecho de que, una vez introducida una tarea en la máquina, el usuario no mantiene contacto con la misma hasta que concluye su ejecución. Hay dos modos de trabajar proponiendo las tareas al sistema. Una es a través del operador, quien recibe el trabajo y lo pone a ejecutar, cuando finaliza devuelve los resultados de la ejecución al usuario. La otra forma es un Sistema Batch denominado de "entrada remota de trabajos" (en inglés ***Remote Job Entry (RJE***)), que permite ordenar la ejecución de los trabajos a través de dispositivos de Entrada Salida (E/S) que pueden estar en lugares alejados (Remotos) de la máquina.

b) Sistemas de accesos mútiples (Multiaccess).

En estos sistemas de accesos múltiples, el usuario puede iniciar, vigilar, controlar, o suspender la ejecución de su programa desde cualquier terminal del sistema. El SO hace que los diferentes trabajos compartan los

recursos computacionales de forma que cada usuario tenga la sensación de disponer todo el Sistema en forma exclusiva para él solo.

Muchos SO combinan los modos de ejecución de Batch, para trabajos rutinarios que no requieran interactividad (como ser obtención de listados o control de stocks, etc), y de accesos múltiples, debido a las ventajas que presenta desde el punto de vista interactivo, como por ejemplo, depuración de errores o preparación de documentos; o nuevos desarrollos.

Los Sistemas pueden estar implementados sobre un solo procesador instalado en una sala o sobre varios procesadores distribuídos en lugares distantes entre sí, intercomunicados por medio de líneas de transmisión de datos o red. En estos últimos, el Sistema Operativo debe, además, coordinar las actividades de los diferentes computadores, asegurando que se lleve a cabo un adecuado flujo de información entre ellos.

Sistemas de Propósito Especial.

Estos sistemas son construidos a medida para arquitecturas especiales o aplicaciones con requerimientos especiales como control de procesos industriales.

Son algunos ejemplos de Sistemas Operativos de propósito especial:

a) SO en tiempo real.

Un sistema de tiempo real se usa generalmente como un dispositivo de control en una aplicación dedicada en que el procesamiento debe realizarse dentro de un tiempo dado. Las principales características que deben brindar son:

1. Garantizar la respuesta a eventos externos dentro de límites de tiempo preestablecidos.
2. Los parámetros más importantes son el tiempo de espera de la entrada, tiempo de procesamiento de la entrada, y un rápido almacenamiento de la misma.

Ejemplos de aplicaciones

Existen dos tipos de Sistemas de Tiempo Real:

Aquellos en que el tiempo de respuesta no es muy crítico: reservas de pasajes, aplicaciones comerciales online, control de tránsito vehicular, control de tráfico telefónico, etc.

Los que el tiempo de respuesta es muy crítico (Sistemas estimulados por eventos externos deben generar respuestas a éstos eventos): Control de procesos industriales, recolección de datos de experimentos, etc.

El sistema en tiempo real incluye el software (que debe organizar, administrar y operar la transferencia de los datos y proveer la adecuada sincronización de tiempos) y el hardware compuesto por:

sensores: elementos que detectan mediante una alteración de sus características un cambio en el ambiente en que miden.

transductores: Traducen un cambio físico en una corriente o tensión eléctrica.

conversores (adc/dca): Convierten las variaciones de corrientes o tensiones eléctricas en pulsos binarios (Analogic - Digital Converter / Digital - Analogic Converter).

interfases: Puertos, Impresores, Graficadores, Registradores, Visualizadores, Señalizadores, Alarmas, Consolas, Teclados, etc., etc.

procesadores: Son los responsables de las tareas de control, interrupciones, tiempos, almacenamiento de datos temporales y definitivos, generación de órdenes de control o mandos a distancia, etc.

Un caso particular de SO de Tiempo real son los Sistemas de ***Control de Procesos.***

Las características de los procesos, ya sean industriales, médicos, u otros; son que el computador recibe como entrada una información, la procesa y efectúa una acción como consecuencia del proceso. Esta acción se conoce como **realimentación** (en inglés feed-back) cuyo efecto es mantener la estabilidad del sistema. En algunos de estos procesos se requiere una respuesta de acuerdo a la evolución del fenómeno, entonces se dice que el sistema está trabajando en tiempo real. La principal función del SO en un control de proceso es la de proporcionar el máximo de fiabilidad con el mínimo de intervención humana. En caso de una falla del hardware el SO debe ser capaz de detener el proceso bajo control.

b) SO con tolerancias a fallas (fault-tolerance Operating System)

1. Usado en aplicaciones donde se debe proveer un servicio continuo o cuyo mantenimiento es dificultoso o muy costoso.
2. Se suele utilizar un conjunto de redundancias en recursos y chequeos internos.
3. EL SO detecta y corrige errores; y recupera el sistema habilitando reemplazos de los componentes en mal funcionamiento o vuelve atrás operaciones que motivaron perdidas de datos.

Ejemplos de aplicaciones

Sistemas de seguridad en el área nuclear, sistemas espaciales, cajeros automáticos, bases de datos on-line, etc.

c) SO virtuales

Especialmente diseñados para ejecutar varios SO (o distintas versiones de uno mismo) concurrentemente en una máquina creando la ilusión de varias máquinas idénticas.

Todos los SO trabajan en modo usuario respecto al SO virtual, pero están en modo privilegiado con respecto a los programas que corren bajo ese SO.

Ejemplos de aplicaciones

Verificación de un nuevo SO, migraciones de un Sistema Operativo a una nueva Versión (release), conversiones de aplicaciones a un nuevo ambiente.

Componentes mínimos de un Sistema Operativo.

Un SO proporciona el entorno dentro del cual se ejecutan los programas. Para construir este entorno, dividimos lógicamente al mismo en pequeños módulos y creamos una interfase bien definida para éstos programas. Internamente los SO varían en su estructura, organizándose de acuerdo con diferentes esquemas. Antes de diseñarlo es importante definir los objetivos del sistema. Para crearlo se divide en fragmentos más pequeños, si bien cada SO varía en estructura con respecto a otros, los modernos generalmente comparten el objetivo de dividirlo en los siguientes componentes o administradores:

- Administración de la Memoria Central.
- Administración de los procesos en ejecución, su coordinación, sincronización e intercomunicación.
- Administración de datos e información.
- Administración de programas.
- Administración de dispositivos de E/S.
- Administración del almacenamiento secundario (File Management).
- Administración de recursos (Pedidos, conflictos, devoluciones, etc.).
- Programas de servicio y utilitarios.
- Administración de redes (Networking), protección del Sistema, Intérprete de comandos, etc.