INTRODUCCION

A finales de los 40's el uso de computadoras estaba restringido a aquellas empresas o instituciones que podían pagar su alto precio, y no existían los sistemas operativos. En su lugar, el programador debía tener un conocimiento y contacto profundo con el hardware, y en el infortunado caso de que su programa fallara, debía examinar los valores de los registros y páneles de luces indicadoras del estado de la computadora para determinar la causa del fallo y poder corregir su programa, además de enfrentarse nuevamente a los procedimientos de apartar tiempo del sistema y poner a punto los compiladores, ligadores, etc; para volver a correr su programa, es decir, enfrentaba el problema del procesamiento serial (serial processing) [Stal92].

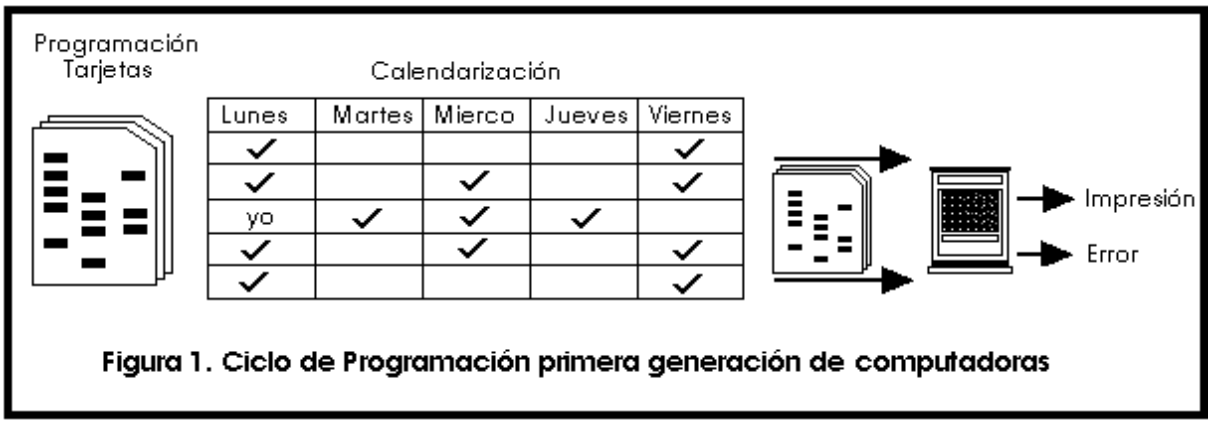


Figura 1. Ciclo de Programación primera generación de computadoras

La importancia de los sistemas operativos nace históricamente desde los 50's, cuando se hizo evidente que el operar una computadora por medio de tableros enchufables en la primera generación y luego por medio del trabajo en lote en la segunda generación se podía mejorar notoriamente, pues el operador realizaba siempre una secuencia de pasos repetitivos, lo cual es una de las características contempladas en la definición de lo que es un programa. Es decir, se comenzó a ver que las tareas mismas del operador podían plasmarse en un programa, el cual a través del tiempo y por su enorme complejidad se le llamó "Sistema Operativo". Así, tenemos entre los primeros sistemas operativos al Fortran Monitor System (FMS) e IBSYS [Tan92].

Posteriormente, en la tercera generación de computadoras nace uno de los primeros sistemas operativos con la filosofía de administrar una familia de computadoras: el OS/360 de IBM. Fue este un proyecto tan novedoso y ambicioso que enfrentó por primera vez una serie de problemas conflictivos debido a que anteriormente las computadoras eran creadas para dos propósitos en general: el comercial y el científico. Así, al tratar de crear un solo sistema operativo para computadoras que podían dedicarse a un propósito, al otro o ambos, puso en evidencia la problemática del trabajo en equipos de análisis, diseño e implantación de sistemas grandes. El resultado fue un sistema del cual uno de sus mismos diseñadores patentizó su opinión en la portada de un libro: una horda de bestias prehistóricas atascadas en un foso de brea.

Surge también en la tercera generación de computadoras el concepto de la multiprogramación, porque debido al alto costo de las computadoras era necesario idear un esquema de trabajo que mantuviese a la unidad central de procesamiento más tiempo ocupada, así como el encolado (spooling) de trabajos para su lectura hacia los lugares libres de memoria o la escritura de resultados. Sin embargo, se puede afirmar que los sistemas durante la tercera generación siguieron siendo básicamente sistemas de lote.

En la cuarta generación la electrónica avanza hacia la integración a gran escala, pudiendo crear circuitos con miles de transistores en un centímetro cuadrado de silicón y ya es posible hablar de las computadoras personales y las estaciones de trabajo. Surgen los conceptos de interfaces amigables intentando así atraer al

público en general al uso de las computadoras como herramientas cotidianas. Se hacen populares el MS−DOS y UNIX en estas máquinas. También es común encontrar clones de computadoras personales y una multitud de empresas pequeñas ensamblándolas por todo el mundo.

Para mediados de los 80's, comienza el auge de las redes de computadoras y la necesidad de sistemas operativos en red y sistemas operativos distribuidos. La red mundial Internet se va haciendo accesible a toda clase de instituciones y se comienzan a dar muchas soluciones (y problemas) al querer hacer convivir recursos residentes en computadoras con sistemas operativos diferentes. Para los 90's el paradigma de la programación orientada a objetos cobra auge, así como el manejo de objetos desde los sistemas operativos. Las aplicaciones intentan crearse para ser ejecutadas en una plataforma específica y poder ver sus resultados en la pantalla o monitor de otra diferente (por ejemplo, ejecutar una simulación en una máquina con UNIX y ver los resultados en otra con DOS). Los niveles de interacción se van haciendo cada vez más profundos.

TIPOS DE SISTEMAS OPERATIVOS

En esta sección se describirán las características que clasifican a los sistemas operativos, básicamente se cubrirán tres clasificaciones: sistemas operativos por su estructura (visión interna), sistemas operativos por los servicios que ofrecen y, finalmente, sistemas operativos por la forma en que ofrecen sus servicios (visión externa).

2.1 Sistemas Operativos por su Estructura

Según [Alcal92], se deben observar dos tipos de requisitos cuando se construye un sistema operativo, los cuales son:

Requisitos de usuario: Sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso al que se le quiere destinar.

Requisitos del software: Donde se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad.

A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

2.1.1 Estructura monolítica.

Es la estructura de los primeros sistemas operativos constituídos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra (Ver Fig. 2). Las características fundamentales de este tipo de estructura son:

Construcción del programa final a base de módulos compilados separadamente que se unen a través del ligador.

Buena definición de parámetros de enlace entre las distintas rutinas existentes, que puede provocar mucho acoplamiento.

Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.

Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo

mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

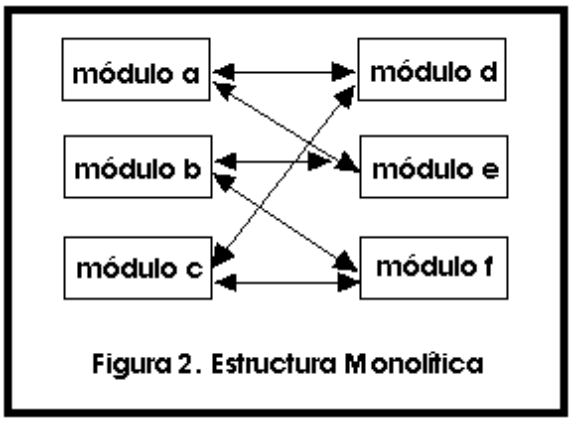


Figura 2. Estructura Monolítica

2.1.2 Estructura jerárquica.

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía subpartes y esto organizado en forma de niveles.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos.

Se constituyó una estructura jerárquica o de niveles en los sistemas operativos, el primero de los cuales fue denominado THE (Technische Hogeschool, Eindhoven), de Dijkstra, que se utilizó con fines didácticos (Ver Fig. 3). Se puede pensar también en estos sistemas como si fueran `multicapa'. Multics y Unix caen en esa categoría. [Feld93].

Figura 3. Sistema jerárquico THE

En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos o "rings" (Ver Fig. 4).

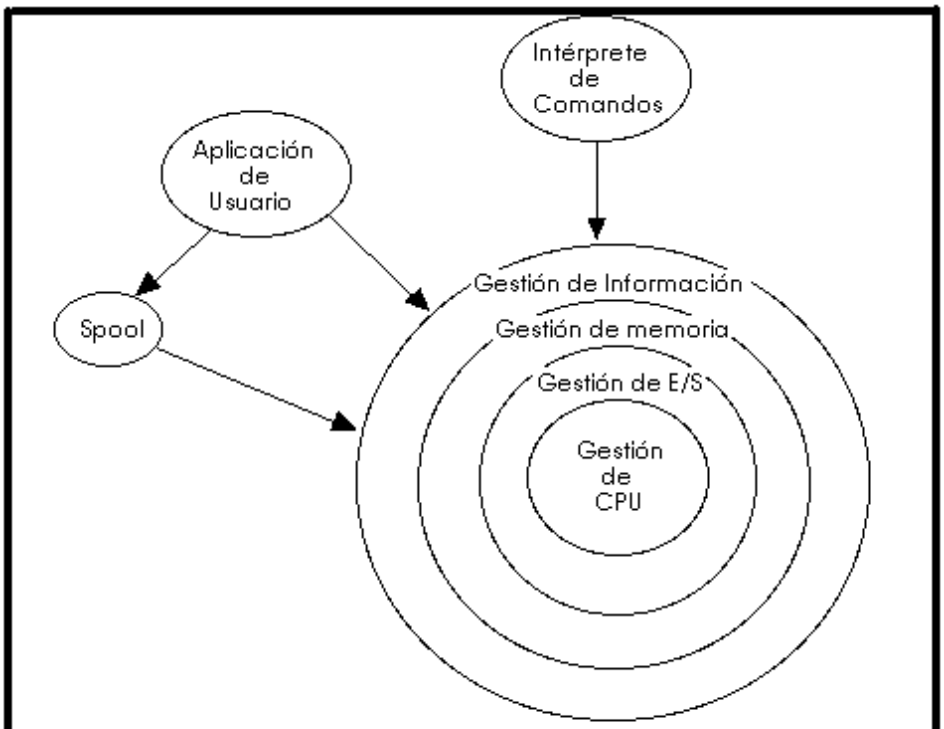


Figura 4. Organización jerárquica (anillos)

En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

2.1.3 Máquina Virtual.

Se trata de un tipo de sistemas operativos que presentan una interface a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente. Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de sistemas: la multiprogramación y la máquina extendida. El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

El núcleo de estos sistemas operativos se denomina monitor virtual y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario (Ver Fig. 5).

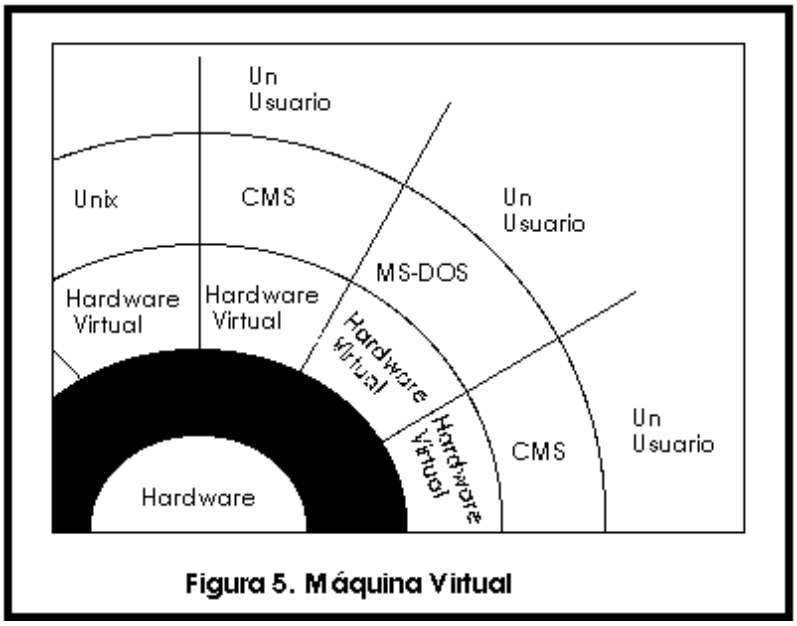


Figura 5. Máquina Virtual

2.1.4 Cliente−servidor (Microkernel)

El tipo más reciente de sistemas operativos es el denominado Cliente−servidor, que puede ser ejecutado en la mayoría de las computadoras, ya sean grandes o pequeñas.

Este sistema sirve para toda clase de aplicaciones por tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro." [Alcal92]. Este paradigma ofrece gran flexibilidad en cuanto a los servicios posibles en el sistema final, ya que el núcleo provee solamente funciones muy básicas de memoria, entrada/salida, archivos y procesos, dejando a los servidores proveer la mayoría que el usuario final o programador puede usar. Estos servidores deben tener mecanismos de seguridad y protección que, a su vez, serán filtrados por el núcleo que controla el hardware. Actualmente se está trabajando en una versión de UNIX que contempla en su diseño este paradigma.

2.2 Sistemas Operativos por Servicios

Esta clasificación es la más comúnmente usada y conocida desde el punto de vista del usuario final. Esta clasificación se comprende fácilmente con el cuadro sinóptico que a continuación se muestra en la Fig. 6.

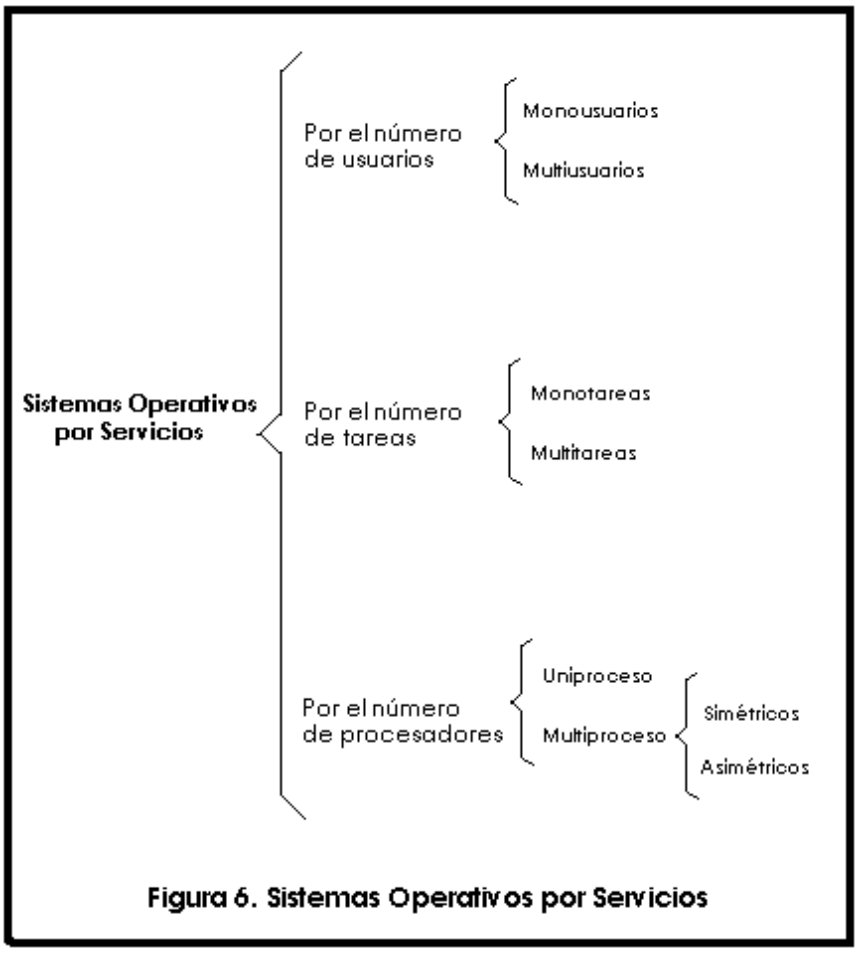


Figura 6. Sistemas Operativos por Servicios

2.2.1 Monousuarios

Los sistemas operativos monousuarios son aquéllos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Las computadoras personales típicamente se han clasificado en este renglón.

2.2.2 Multiusuarios

Los sistemas operativos multiusuarios son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

2.2.3 Monotareas

Los sistemas monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo solo una tarea a la vez.

2.2.4 Multitareas

Un sistema operativo multitarea es aquél que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background. Es

común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

2.2.5 Uniproceso

Un sistema operativo uniproceso es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil. El ejemplo más típico de este tipo de sistemas es el DOS y MacOS.

2.2.6 Multiproceso

Un sistema operativo multiproceso se refiere al número de procesadores del sistema, que es más de uno y éste es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas: simétrica o asimétricamente. Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos. Cuando se trabaja de manera simétrica, los procesos o partes de ellos (threads) son enviados indistintamente a cualesquira de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema.

Se dice que un thread es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto. Us aspecto importante a considerar en estos sistemas es la forma de crear aplicaciones para aprovechar los varios procesadores. Existen aplicaciones que fueron hechas para correr en sistemas monoproceso que no toman ninguna ventaja a menos que el sistema operativo o el compilador detecte secciones de código paralelizable, los cuales son ejecutados al mismo tiempo en procesadores diferentes. Por otro lado, el programador puede modificar sus algoritmos y aprovechar por sí mismo esta facilidad, pero esta última opción las más de las veces es costosa en horas hombre y muy tediosa, obligando al programador a ocupar tanto o más tiempo a la paralelización que a elaborar el algoritmo inicial.

2.3. Sistemas Operativos por la Forma de Ofrecer sus Servicios

Esta clasificación también se refiere a una visión externa, que en este caso se refiere a la del usuario, el cómo accesa los servicios. Bajo esta clasificación se pueden detectar dos tipos principales: sistemas operativos de red y sistemas operativos distribuídos.

2.3.1 Sistemas Operativos de Red

Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras computadoras por medio de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades. El punto crucial de estos sistemas es que el usuario debe saber la sintaxis de un cinjunto de comandos o llamadas al sistema para ejecutar estas operaciones, además de la ubicación de los recursos que desee accesar. Por ejemplo, si un usuario en la computadora hidalgo necesita el archivo matriz.pas que se localiza en el directorio /software/codigo en la computadora morelos bajo el sistema operativo UNIX, dicho usuario podría copiarlo a través de la red con los comandos siguientes: hidalgo% hidalgo% rcp morelos:/software/codigo/matriz.pas . hidalgo% En este caso, el comando rcp que significa "remote copy" trae el archivo indicado de la computadora morelos y lo coloca en el directorio donde se ejecutó el mencionado comando. Lo importante es hacer ver que el usuario puede accesar y compartir muchos recursos.

2.3.2 Sistemas Operativos Distribuídos

Los sistemas operativos distribuídos abarcan los servicios de los de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso) en una sola máquina virtual que el usuario accesa en forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y simplementa los usa como si todos ellos fuesen locales a su lugar de trabajo habitual. Todo lo anterior es el marco teórico de lo que se desearía tener como sistema operativo distribuído, pero en la realidad no se ha conseguido crear uno del todo, por la complejidad que suponen: distribuír los procesos en las varias unidades de procesamiento, reintegrar sub−resultados, resolver problemas de concurrencia y paralelismo, recuperarse de fallas de algunos recursos distribuídos y consolidar la protección y seguridad entre los diferentes componentes del sistema y los usuarios. [Tan92]. Los avances tecnológicos en las redes de área local y la creación de microprocesadores de 32 y 64 bits lograron que computadoras mas o menos baratas tuvieran el suficiente poder en forma autónoma para desafiar en cierto grado a los mainframes, y a la vez se dio la posibilidad de intercomunicarlas, sugiriendo la oportunidad de partir procesos muy pesados en cálculo en unidades más pequeñas y distribuirlas en los varios microprocesadores para luego reunir los sub−resultados, creando así una máquina virtual en la red que exceda en poder a un mainframe. El sistema integrador de los microprocesadores que hacer ver a las varias memorias, procesadores, y todos los demás recursos como una sola entidad en forma transparente se le llama sistema operativo distribuído. Las razones para crear o adoptar sistemas distribuídos se dan por dos razones principales: por necesidad (debido a que los problemas a resolver son inherentemente distribuídos) o porque se desea tener más confiabilidad y disponibilidad de recursos. En el primer caso tenemos, por ejemplo, el control de los cajeros automáticos en diferentes estados de la república. Ahí no es posible ni eficiente mantener un control centralizado, es más, no existe capacidad de cómputo y de entrada/salida para dar servicio a los millones de operaciones por minuto. En el segundo caso, supóngase que se tienen en una gran empresa varios grupos de trabajo, cada uno necesita almacenar grandes cantidades de información en disco duro con una alta confiabilidad y disponibilidad. La solución puede ser que para cada grupo de trabajo se asigne una partición de disco duro en servidores diferentes, de manera que si uno de los servidores falla, no se deje dar el servicio a todos, sino sólo a unos cuantos y, más aún, se podría tener un sistema con discos en espejo (mirror) a través de la red,de manera que si un servidor se cae, el servidor en espejo continúa trabajando y el usuario ni cuenta se da de estas fallas, es decir, obtiene acceso a recursos en forma transparente.

2.3.2.1 Ventajas de los Sistemas Distribuídos

En general, los sistemas distribuídos (no solamente los sistemas operativos) exhiben algunas ventajas sobre los sistemas centralizados que se describen enseguida.

- Economía: El cociente precio/desempeño de la suma del poder de los procesadores separados contra el poder de uno solo centralizado es mejor cuando están distribuídos.
- Velocidad: Relacionado con el punto anterior, la velocidad sumada es muy superior.
- Confiabilidad: Si una sola máquina falla, el sistema total sigue funcionando.
- Crecimiento: El poder total del sistema puede irse incrementando al añadir pequeños sistemas, lo cual es mucho más difícil en un sistema centralizado y caro.
- Distribución: Algunas aplicaciones requieren de por sí una distribución física.

Por otro lado, los sistemas distribuídos también exhiben algunas ventajas sobre sistemas aislados. Estas ventajas son:

- Compartir datos: Un sistema distribuído permite compartir datos más fácilmente que los sistemas aislados, que tendrian que duplicarlos en cada nodo para lograrlo.
- Compartir dispositivos: Un sistema distribuído permite accesar dispositivos desde cualquier nodo en forma transparente, lo cual es imposible con los sistemas aislados. El sistema distribuído logra un efecto sinergético.
- Comunicaciones: La comunicación persona a persona es factible en los sistemas distribuídos, en los

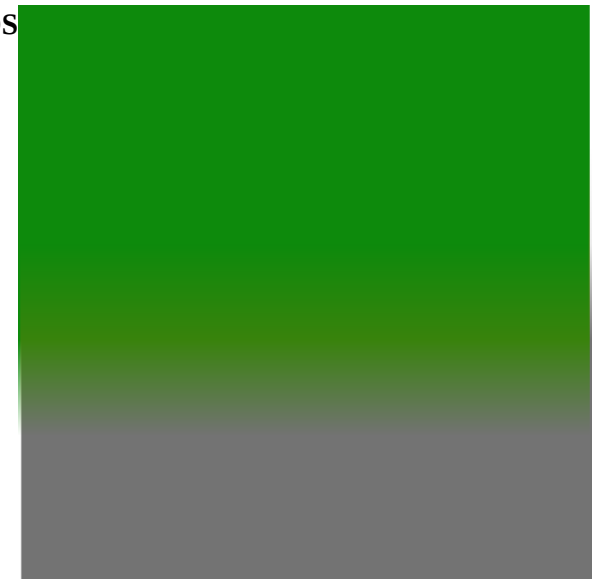
sistemas aislados no. _ Flexibilidad: La distribución de las cargas de trabajo es factible en el sistema distribuídos, se puede incrementar el poder de cómputo.

2.3.2.2 Desventajas de los Sistemas Distribuídos

Así como los sistemas distribuídos exhiben grandes ventajas, también se pueden identificar algunas desventajas, algunas de ellas tan serias que han frenado la producción comercial de sistemas operativos en la actualidad. El problema más importante en la creación de sistemas distribuídos es el software: los problemas de compartición de datos y recursos es tan complejo que los mecanismos de solución generan mucha sobrecarga al sistema haciéndolo ineficiente. El checar, por ejemplo, quiénes tienen acceso a algunos recursos y quiénes no, el aplicar los mecanismos de protección y registro de permisos consume demasiados recursos. En general, las soluciones presentes para estos problemas están aún en pañales.

Otros problemas de los sistemas operativos distribuídos surgen debido a la concurrencia y al paralelismo. Tradicionalmente las aplicaiones son creadas para computadoras que ejecutan secuencialmente, de manera que el identificar secciones de código `paralelizable' es un trabajo ardúo, pero necesario para dividir un proceso grande en sub−procesos y enviarlos a diferentes unidades de procesamiento para lograr la distribución. Con la concurrencia se deben implantar mecanismos para evitar las condiciones de competencia, las postergaciones indefinidas, el ocupar un recurso y estar esperando otro, las condiciones de espera circulares y , finalmente, los "abrazos mortales" (deadlocks). Estos problemas de por sí se presentan en los sistemas operativos multiusuarios o multitareas, y su tratamiento en los sistemas distribuídos es aún más complejo, y por lo tanto, necesitará de algoritmos más complejos con la inherente sobrecarga esperada.

NUCLEOS DE SISTEMAS OPERATIVOS



Para comprender mejor qué diferencias existen entre ambas categorías, se necesita revisar algunos conceptos.

Trabajos, Procesos y Thread

EStos tres conceptos van definiendo el grado de granularidad en que el sistema operativo trata a las masas de operaciones que se tienen que realizar. Un trabajo se conceptualiza como un conjunto de uno o más procesos. Por ejemplo, si se tiene que hacer el trabajo de correr el inventario, tal vez se subdivida ese trabajo en varios procesos: obtener la lista de artículos, número en existencia, artículos vendidos, artículos extraviados, etc. Un proceso se define como la imagen de un programa en ejecución, es decir, en memoria y usando el CPU. A este nivel de granularidad, un proceso tiene un espacio de direcciones de memoria, una pila, sus registros y su

'program counter'. Un thread es un trozo o sección de un proceso que tiene sus propios registros, pila y 'program counter' y puede compartir la memoria con todos aquellos threads que forman parte del mismo proceso.

Objetos

Un objeto es una entidad que contiene dos partes principales: una colección de atributos y un conjunto de métodos (también llamados servicios). Generalmente los atributos del objeto no pueden ser cambiados por el usuario, sino solamente a través de los métodos. Los métodos sí son accesibles al usuario y de hecho es lo único que él observa: los métodos conforman lo que se llama la 'interfaz' del objeto. Por ejemplo, para el objeto 'archivo' los métodos son abrir, cerrar, escribir, borrar, etc. El cómo se abre, se cierra, se borra, etc; está escondido para el usuario, es decir, los atributos y el código están 'encapsulados'. La única forma de activar un método es a través del envío de mensajes entre los objetos, o hacia un objeto.

Cliente − Servidor

Un cliente es un proceso que necesita de algún valor o de alguna operación externa para poder trabajar. A la entidad que prove ese valor o realiza esa operación se le llama servidor. Por ejemplo, un servidor de archivos debe correr en el núcleo (kernel) o por medio de un proceso 'guardián' al servidor de archivos que escucha peticiones de apertura, lectura, escritura, etc; sobre los archivos. Un cliente es otro proceso guardián que escucha esas peticiones en las máquinas clientes y se comunica con el proceso servidor a través de la red, dando la apariencia de que se tienen los archivos en forma local en la máquina cliente.

Núcleo Monolítico

Los núcleos monolíticos generalmente están divididos en dos partes estructuradas: el núcleo dependiente del hardware y el núcleo independiente del hardware. El núcleo dependiente se encarga de manejar las interrupciones del hardware, hacer el manejo de bajo nivel de memoria y discos y trabajar con los manejadores de dispositivos de bajo nivel, principalmente. El núcleo independiente del hardware se encarga de ofrecer las llamadas al sistema, manejar los sistemas de archivos y la planificación de procesos. Para el usuario esta división generalmente pasa desapercibida. Para un mismo sistema operativo corriendo en diferentes plataformas, el núcleo independiente es exactamente el mismo, mientras que el dependiente debe re−escribirse.

Microkernel

Un núcleo con 'arquitectura' micronúcleo es aquél que contiene únicamente el manejo de procesos y threads, el de manejo bajo de memoria, da soporte a las comunicaciones y maneja las interrupciones y operaciones de bajo nivel de entrada−salida. [Tan92]. En los sistemas oprativos que cuentan con este tipo de núcleo se usan procesos 'servidores' que se encargan de ofrecer el resto de servicios (por ejemplo el de sistema de archivos) y que utilizan al núcleo a través del soporte de comunicaciones.

Este diseño permite que los servidores no estén atados a un fabricante en especial, incluso el usuario puede escoger o programar sus propios servidores. La mayoría de los sistemas operativos que usan este esquema manejan los recursos de la computadora como si fueran objetos: los servidores ofrecen una serie de 'llamadas' o 'métodos' utilizables con un comportamiento coherente y estructurado. Otra de las características importantes de los micronúcleos es el manejo de threads. Cuando un proceso está formado de un solo thread, éste es un proceso normal como en cualquier sistema operativo.