

Un paseo por la historia

Lidón García, Luis Peralta y Samuel Fernández

Resumen

El documento presenta la evolución de los sistemas operativos a lo largo de la historia, introduciendo los conceptos fundamentales relativos a éstos, como podrían ser la gestión de procesos o su arquitectura.

Así mismo, se ha tratado de dar todas las referencias posibles sobre S.O.'s desconocidos para la mayoría y que han jugado un papel más o menos determinante en la historia. Se han incluido algunos que no han pasado del borrador pero que citamos como simple curiosidad. De cualquier manera, la lista proporcionada es muy incompleta dado el gran número de Sistemas Operativos que existen o que han existido.

1. Introducción

Antaño, con las primeras máquinas, era algo muy complicado ser programador... y no sólo porque los lenguajes de programación no habían evolucionado, sino porque se debía manejar el ordenador desde la consola y la consola en aquellos tiempos significaba un puñado de interruptores. Afortunadamente, esto ha ido cambiando y se lo debemos, en parte, a que han nacido y evolucionado los sistemas operativos. Como también lo han hecho las máquinas, los lenguajes de programación e incluso las ideas.

2. Necesidad de Sistema Operativo

- 2.1. Vista histórica [Tanenbaum92]

En un principio solo existía el *hardware* del computador. Los primeros computadores eran (físicamente) grandes máquinas que se operaban desde una consola. El programador escribía un programa y luego lo controlaba directamente. En primer lugar, el programa se cargaba manualmente en la memoria, desde los interruptores del tablero frontal, desde una cinta de papel o desde tarjetas perforadas. Luego se pulsaban los botones adecuados para establecer la dirección de inicio y comenzar la ejecución del programa. Mientras este se ejecutaba, el programador-operador lo podía supervisar observando las luces en la consola. Si se descubrían errores, el programador podía detener el programa, examinar el contenido de la memoria y los registros y depurar el programa directamente desde la consola. La salida del programa se imprimía, o se perforaba en cintas de papel o tarjetas para su impresión posterior. Un

aspecto importante de ese entorno era su naturaleza interactiva directa. El programador también era el operador del sistema de computación.

La mayoría de los sistemas utilizaban un esquema de reservas o de registro para asignar tiempo de máquina. Si alguien quería usar el computador, acudía a la hoja de registro, buscaba el siguiente tiempo libre de la máquina que fuera conveniente y se anotaba para usarla. Sin embargo, con este procedimiento se presentaban ciertos problemas. Si un programador se había registrado para usar una hora de tiempo del computador dedicada a ejecutar el programa que estaba desarrollando, pero se topaba con algún error difícil y no podía terminar en esa hora. Si alguien más había reservado el siguiente bloque de tiempo, el programador debía detenerse, rescatar lo que pudiera y volver mas tarde para continuar. Por otra parte, si el programa se ejecutaba sin problemas, podría terminar en 35 minutos; pero como pensó que necesitaría la máquina durante más tiempo, se registró para usarla un hora, y permanecería inactiva durante 25 minutos.

Conforme transcurrió el tiempo, se desarrollaron *software* y *hardware* adicionales; empezaron a popularizarse los lectores de tarjetas, impresoras de líneas y cintas magnéticas. Se diseñaron ensambladores, cargadores y ligadores para facilitar las tareas de programación, y se crearon bibliotecas de funciones comunes, de manera que éstas podían copiarse a un nuevo programa sin tener que escribirlas de nuevo.

Se podía necesitar un considerable tiempo de operación para realizar un trabajo. Cada trabajo consistía en varios pasos distintos: cargar la cinta del compilador de FORTRAN, ejecutar el compilador, descargar la cinta del compilador, cargar la cinta del ensamblador, ejecutar el ensamblador, descargar la cinta del ensamblador, cargar el programa objeto y ejecutarlo. Si ocurría un error en cualquiera de los pasos, probablemente habría que comenzar desde el principio. Cada paso del trabajo podía ocasionar la necesidad de cargar y descargar cintas magnéticas, cintas de papel o tarjetas perforadoras, por lo que al avanzar el tiempo aparecieron los primeros sistemas operativos.

Los computadores, sobretudo los centrales de gran tamaño, han sido siempre máquinas muy costosas. Por ello, los dueños han deseado que estas máquinas efectuen la mayor cantidad posible de cálculos. Hoy en día, esta situación también se aplica a los microprocesadores de menor coste, de los cuales, a pesar de no ser tan costosos, siempre se espera que logren el mayor número posible de cálculos. El cambio a los sistemas por lotes con secuenciación automática de trabajos se efectuó para mejorar el rendimiento. El problema es que las personas somos demasiado lentas. Por lo tanto, es deseable sustituir la intervención humana por el *software* del *sistema operativo*. Aún con esta secuenciación automática de trabajos, la *UCP* todavía tiene periodos de inactividad. El problema es la

velocidad de los dispositivos mecánicos de E/S, los cuales son intrínsecamente más lentos que los dispositivos electrónicos. Una *UCP* lenta trabaja en el orden de los microsegundos, pues ejecuta millones de instrucciones por segundo. La solución que se encontró a esto fue el uso de *buffers* o *tampones* que se anticipaban al programa leyendo los datos y dejándolos en memoria antes de que se produjese la orden de leerlos, con lo que se acelera la ejecución total del programa.

- 2.2. Definición [Peterson91]

Definamos ahora el concepto de Sistema Operativo: Es un programa que actúa como intermediario entre el usuario y el *hardware* de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas. El objetivo principal de un sistema operativo 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. Una definición más común es que el sistema operativo es el programa que se ejecuta todo el tiempo en el computador, siendo programas de aplicación todos los demás.

- *Objetivo principal:* los sistemas operativos existen porque se supone que es mas fácil trabajar con uno de ellos que sin él. Esta situación es particularmente clara cuando se observan los sistemas operativos para los pequeños computadores personales.
- *Objetivo secundario:* es la utilización eficiente del sistema de computación. Este propósito tiene una importancia especial en los grandes sistemas multiusuario compartidos. En el pasado, las consideraciones de eficiencia a menudo eran más importantes que la comodidad, por lo que gran parte de la teoría de sistemas operativos se concentra en el uso óptimo de los recursos de computación.

Los sistemas operativos y la arquitectura del computador tienen una gran influencia mutua. Los sistemas operativos se desarrollan para facilitar el uso del *hardware*. Al ir diseñando y utilizando los sistemas operativos, se hizo patente que podrían simplificarse con cambios en el diseño del *hardware*.

3. Tipos de gestión de procesos de un Sistema Operativo [Tanenbaum91]

En un principio, los computadores se utilizaban desde la consola central. El *software* mejoró la comodidad de programar, pero necesitaba un tiempo considerable de preparación. Para reducir este tiempo, se contrataron operadores y los trabajos semejantes se agruparon en lotes. El computador ya no tenía que esperar la intervención humana, aun así, la utilización de la *UCP* era muy lenta. Con el fin de mejorar el rendimiento global del sistema se introdujo el concepto de multiprogramación, gracias al cual se almacenan en la

memoria varios trabajos al mismo tiempo, lo que aumenta el rendimiento de la *UCP* y reduce el tiempo de ejecución de los trabajos.

- 3.1. Sistemas por lotes.

Cuando se desarrollaron por primera vez, estaban caracterizados por la "agrupación en bloques" de trabajos similares. Los modernos sistemas utilizan otras características. El rasgo característico de un sistema por lotes es la ausencia de interacción entre el usuario y el trabajo mientras éste se ejecuta. El trabajo se prepara y se envía. Tiempo después aparece la salida.

- 3.2. Multiprogramación.

Un solo usuario no puede, en general, mantener todo el tiempo ocupado a la *UCP* o a los dispositivos de E/S. La multiprogramación aumenta la utilización de la *UCP* organizando los trabajos de manera que ésta siempre tenga algo que ejecutar. El *S.O.* escoge uno de los trabajos del depósito y comienza a ejecutarlo. En algún momento el trabajo tendrá que esperar, ya que el sistema ha pasado el control a otro programa y así sucesivamente. Mientras haya otro trabajo por ejecutar, la *UCP* nunca estará inactiva.

Dentro de los sistemas multiprogramados tenemos tres tipos:

- 3.2.1. Tiempo compartido.

Utiliza la planificación de la *UCP* y la multiprogramación para proporcionar a cada usuario, que tiene su propio programa en memoria, una pequeña porción de un computador de tiempo compartido. La E/S interactiva es demasiado lenta para un computador por lo que, para que la *UCP* no permanezca inactiva, el *S.O.* la cambiará al programa de otro usuario. Ésto ocurre tan rápidamente que cada usuario tiene la impresión de que cuenta con su propio computador, cuando en realidad todos lo comparten.

- 3.2.2 Tiempo real.

Suele usarse como dispositivo de control en una aplicación dedicada. Tiene restricciones temporales bien definidas, por lo que el procesamiento debe llevarse a cabo dentro de los límites definidos o el sistema fallará. Puede parecer extraña la utilidad de este tipo de gestión, así que pondremos un ejemplo: una nave espacial se dispone a acoplarse a la estación espacial MIR, nos interesa conocer las coordenadas de la MIR en todo momento para compararlas con las nuestras y así actuar en consecuencia. De

nada nos sirve que se resuelvan los cálculos una vez nos hemos estrellado porque un astronauta estaba jugando con el mismo ordenador al *tetris* y este consumía toda la potencia de cálculo del ordenador.

- 3.2.3 Combinados

Es una mezcla de los dos anteriores. Aunque se ha intentado combinar la funcionalidad del tiempo compartido y el tiempo real en un solo *S.O.*, los resultados han sido pésimos debido a los obvios conflictos entre los requisitos de ambos tipos.

- 3.3. Sistemas distribuidos. [Goscinski91]

Es un sistema débilmente acoplado, es decir, los procesadores no comparten ni memoria ni reloj, cada uno cuenta con su propia memoria local y se comunican a través de distintas líneas de comunicación. Los procesadores pueden variar de tamaño y función. Las principales ventajas son:

- Compartición de recursos.
- Aceleración de los cálculos.
- Fiabilidad.
- Comunicación.

4. Arquitectura de un *S.O.* [Milenkovic94]

Para comenzar este apartado podemos plantear la siguiente pregunta: "¿Qué pasa si el propio sistema operativo necesita hacer algo que es capaz de hacer pero no se encuentra en el código que está ejecutando en ese momento?" Dicho de otra manera, vamos a ver dónde se encuentran las diferentes funciones de un *S.O.* y qué tiene que hacer para usarlas. Hasta ahora hemos visto parte del aspecto "exterior" del *S.O.*, que es la organización que da a los programas a la hora de ejecutarlos. Pasemos a examinar las diferentes posibilidades de implementación de un *S.O.* "por dentro", es decir, no cómo organiza el sistema operativo al resto de programas, sino cómo se organiza respecto a sí mismo.

- 4.1. Kernel monolítico: La estructura de esta arquitectura es simplemente no tener ninguna. A nivel de núcleo no se produce ninguna abstracción, es decir, si un procedimiento necesita a otro es libre de hacerlo en cualquier momento. Fue el primer enfoque en la historia, el resto son evoluciones.
- 4.2. Microkernel o micronúcleo: En este caso, el *S.O.* se ocupa solo de unas pocas funciones, reduciendo el núcleo a su mínima expresión. El resto de funciones pasan a estar en el espacio de usuario.
- 4.3. Maquinas virtuales: El primer sistema con esta arquitectura nació con la idea de separar completamente las dos funciones características de un *S.O.* de tiempo compartido: multiprogramación y un interfaz más apropiado que el del puro *HW*. El centro del sistema, también conocido como monitor de la máquina

virtual, se ejecuta directamente sobre el propio *HW*, encargándose de la multiprogramación. De esta forma, ofrece al nivel superior varias máquinas virtuales, que son copias exactas del *hardware*, por lo que se puede dar el caso de ejecutar varios *S.O.* sobre cada una de ellas (de hecho, el caso mas usual).

- 4.4. Modelo cliente-servidor: Esta es la tendencia en cuanto a arquitectura de los *S.O.* hoy en día. Consiste en reducir al mínimo el kernel, al igual que en el caso de los *microkernels*, pero en este caso la única función del kernel es de servir de *punto* entre procesos: cuando una función necesita de otra es el *kernel* el que se encarga de mantener la comunicación entre ellas, pero nada más.

5. Diferentes funciones de un *S.O.* [Stallings97]

El sistema operativo crea un entorno para la ejecución de cada proceso, además de ofrecer ciertos servicios a los programas y a sus usuarios. Vamos a ver algunos de ellos.

Los servicios específicos que ofrece cada *S.O.* suelen ser diferentes, dependiendo del sector al que están destinados, aunque algunas clases de ellos son comunes: las operaciones de *E/S*, la manipulación del sistema de archivos, la detección de errores, etc. Estas funciones tienen el propósito, en su mayoría, de favorecer la tarea del programador, haciendo más fácil su trabajo. Lo que realmente nos está dando el *S.O.* es una capa de abstracción del *hardware* particular sobre el que corre. Por si no ha quedado claro el porqué de estas facilidades, pongamos un ejemplo: un programador de bases de datos que trabaje con dos máquinas distintas en *HW* se encontrará con el problema de que tiene que saber manipular los archivos tanto en una como en otra plataforma. Si el *S.O.* cumple bien el objetivo de abstracción, un mismo programa puede funcionar sobre dos máquinas de arquitectura diferente con el mismo sistema operativo, sin que haga falta modificar el código del programa.

Además de estas funciones, el *S.O.* tiene otro conjunto destinado al funcionamiento eficiente de la máquina: asignación de recursos, gestión de procesos, ...

Prácticamente, estos dos grandes grupos de funciones determinan todo lo que hace un *S.O.*: por un lado, las que facilitan la vida al programador y, por otro, las que hacen que éstas últimas se ejecuten en un medio con consistencia. De nada serviría que el sistema permitiese abrir 5 archivos simultáneamente si es incapaz de dejar algo de memoria para la ejecución del programa que los ha abierto.

Veamos ahora algunos de estos tipos de llamadas al sistema, o servicios que ofrece el sistema:

- Para la gestión de procesos: Dentro de este tipo nos encontramos todas aquellas llamadas necesarias para la ejecución de programas, así como la eficiente distribución de recursos entre éstos.

- Para el acceso a dispositivos de *E/S*: Como hemos dicho antes, el *S.O.* debe abstraer el funcionamiento del *hardware* al programador. Para ello, el sistema nos provee de ciertas funciones genéricas para su acceso.
- Para la detección de errores y respuesta: A pesar de ser máquinas, en los sistemas se pueden producir multitud de errores, tanto de *SW* como de *HW*, algunos de ellos pueden ser:
 - Acceso a zonas prohibidas de memoria (*SW*)
 - Imposibilidad de asignar los recursos solicitados por una aplicación (*SW*)
 - Fallo de algún dispositivo de almacenamiento (*HW*)

En todos los casos el sistema ha de ser capaz de responder con éxito a estos sucesos, o en su defecto, evitar la pérdida de datos. Alguno de estos problemas, como la imposibilidad de asignar recursos solicitados por una aplicación han causado muchos quebraderos de cabeza a los programadores y se han llegado a escribir capítulos enteros sobre las posibles soluciones.

- Para la contabilidad del sistema: Si bien este tipo de funciones no las encontramos en todos los *S.O.*, son bastante comunes en aquellos que tienen la particularidad de ser multiusuario.

6. Otros aspectos.

Hasta ahora nos hemos centrado en describir qué es lo que nos ofrecía un *S.O.* objetivamente. Es hora de ver cuál es el resultado de emplear un tipo determinado de estructura o de gestión de procesos.

Por norma, nos encontraremos generalmente con el dilema de que más comodidad significa automáticamente menos eficiencia y viceversa. Ocurre lo mismo con los lenguajes de programación: el ensamblador es tremendamente eficiente pero, sin embargo, es mucho más difícil de usar.

Otro aspecto importante a considerar es la facilidad que tiene el sistema operativo para adaptarse a los nuevos cambios, o sea, evolucionar. Un *kernel* monolítico será mucho más difícil de mantener que un *microkernel* donde podríamos hacer los cambios simplemente en la capa de usuario.

Un paseo por la historia.

A continuación se presenta una tabla cronológica que nos va a ayudar a relacionar diferentes *S.O.*'s según sus características más notables.

Tabla de características

S.O.	Año	Autor	Gestión de procesos	Arquitectura	Multiusuario
Atlas	50-60	University of Manchester	Lotes	monolítico	No
The		Universidad de Eindhoven	Lotes	modular	no
RC4000		Brinch Hansen de Regenecentralen	S.O. Completo	modular	no
Solo		Brinch Hansen de Regenecentralen	multiprogramado	modular	no
CTSS		MIT	multiprogramado-tº compartido	monolítico	si
Multics		MIT	multiprogramado-tº compartido	modular	si
Unix	1969	Ritchie Thompson /	multiprogramado-tº compartido	monolítico	si
Sprite	1984		multiprogramado	modular	si
Merlin	1984		Lotes	monolítico	no
Windows NT	1985	Microsoft	multiprogramado	modular	si
Mach	1986	Darpa	multiprogramado	monolítico	si
Amoeba	1994 (En desarrollo)		distribuido	microkernel	si
Windows 95/98	1995/98	Microsoft	multiprogramado	monolítico	no
Coyote	1996	Trinity College Dublín	distribuido	modular	si
Exokernel	En desarrollo		micro-kernel	monolítico	si

Referencias

- [Goscinski91] Goscinski, Andrej (1991). *Distributed Operating Systems: The logical Design*. Prentice-Hall. 2 edición.
- [Milenkovic94] Milenkovic, Milan (1994). *Sistemas Operativos: conceptos y diseño*. McGraw-Hill. 2ª edición.
- [Peterson91] James L. Peterson, Abraham Silberschatz (1991). *Sistemas Operativos, conceptos fundamentales*. Editorial Reverté.
- [Stallings97] William Stallings (1997). *Sistemas Operativos*. Prentice-Hall. 2ª edición.

- [Tanenbaum91] Andrew Tanenbaum (1991). *Sistemas Operativos: Diseño e implementación*. Prentice Hall. 1ª edición
- [Tanenbaum92] Andrew Tanenbaum (1992). *Modern Operating Systems*. Prentice-Hall. 1ª edición.

Conclusiones

Como hemos visto, existen multitud de Sistemas Operativos, muchos de los cuales el público desconoce, pero que nosotros, como informáticos tenemos la 'obligación' de saber de ellos o incluso probarlos. Muchos de ellos son incluso de libre distribución.

De nada nos servirá que las tiendas ofrezcan ahora preinstalar Windows o Linux, habiendo una baraja entera de S.O.'s entre los que elegir. Estas iniciativas están lejos de complacer a todos aquellos que tratan de llegar más lejos, es decir, de abrir los ojos.

Sistemas Operativos



Augusto Marconcini
Computer Story Foundation

Concepto básico

Es un soporte lógico que controla el funcionamiento del equipo físico. Cualquier sistema de computación puede dividirse en cuatro partes, el Hardware, el sistema operativo, los programas de aplicación y los usuarios.

Un **Sistema operativo** puede verse como un asignador de recursos (Hardware y Software), el S.O (Sistema Operativo) opera de administrador de estos recursos y los asigna a usuarios o programas en la medida que lo requieran. Un S.O es un programa de control y como tal controla la ejecución de los programas de los usuarios para prevenir el uso inadecuado del computador.

Las primeras computadoras eran grandes donde el único lenguaje que era posible usar era el lenguaje de la maquina.

Los programas se cargaban de forma manual mediante tarjetas perforadas.

Tiempo libre era un término que definía al tiempo en que un usuario debía anotarse para usar el computador. Se utilizó un programa llamado **monitor residente** que se podía considerar como el Primer S.O que facilitaba la tarea de cargar la consistencia o reconocimiento de las rutinas que debían ejecutarse en la consola.

Términos pasados de Moda

OFF-Line:

Se llama al tiempo que le quedaba libre al procesador para realizar tareas, mientras la maquina estaba por otro lado realizando tareas, como por ejemplo copiado de cintas (Si Cintas, que viejo!)

Buffering:

Es la técnica en la cual se emplea una memoria la cual es cargada para luego volcar esos datos en algún dispositivo.

Spooling:

Esta técnica permite que la salida de un programa se carga en un buffer (memoria) y posteriormente sea llevado a un soporte magnético u otro dispositivo.

Multiprogramación:

Es un modo de trabajo en el cual se pueden realizar varias tareas simultáneamente con el propósito de aprovechar al máximo los recursos de la computadora. Surge de la necesidad de ocupar el procesador cuando está ocioso.

Tiempo de Sharing:

Es una extensión de la multiprogramación en la cual el usuario interactúa con el proceso. El usuario podía ir dando los datos a medida que el programa lo requería.

Tiempo real:

Es una modalidad en la cual se necesita un tiempo limitado de respuesta con el ordenador y el operador.

Sistemas Distribuidos:

Es una tendencia de los sistemas de computación en distribuir cálculos en varios procesos

Monousuario:

A medida que el Hardware reduce sus costos es mas fiable poseer una computadora, nacen las PCs (Computadoras Personales)

Sistemas Operativos de Red y Distribuidos

Introducción

Sin el software una computadora es en esencia una masa metálica sin utilidad. Con el software, una computadora puede almacenar, procesar y recuperar información, encontrar errores de ortografía e intervenir en muchas otras valiosas actividades para ganar el sustento. El software para computadoras puede clasificarse en general, en 2 clases: los programas de sistema, que controlan la operación de la computadora en sí y los programas de aplicación, los cuales resuelven problemas para sus usuarios. El programa fundamental de todos los programas de sistema, es el Sistema Operativo, que controla todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación.

Un Sistema Operativo es un programa que actúa como intermediario entre el usuario y el hardware de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas. El objetivo principal de un Sistema Operativo es, entonces, 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.

Con el paso del tiempo, los Sistemas Operativos fueron clasificándose de diferentes maneras, dependiendo del uso o de la aplicación que se les daba. A continuación se mostrarán diversos tipos de Sistemas Operativos que existen en la actualidad, con algunas de sus características:

Sistemas Operativos por lotes.

Se reúnen todos los trabajos comunes para realizarlos al mismo tiempo, evitando la espera de dos o más trabajos como sucede en el procesamiento en serie. Estos sistemas son de los más tradicionales y antiguos, y fueron introducidos alrededor de 1956 para aumentar la capacidad de procesamiento de los programas.

Sistemas Operativos de tiempo real.

Los Sistemas Operativos de tiempo real son aquellos en los cuales no tiene importancia el usuario, sino los procesos. Por lo general, están subutilizados sus recursos con la finalidad de prestar atención a los procesos en el momento que lo requieran. Se utilizan en entornos donde son procesados un gran número de sucesos o eventos.

Sistemas Operativos de multiprogramación (o Sistemas Operativos de multitarea).

Se distinguen por sus habilidades para poder soportar la ejecución de dos o más trabajos activos (que se están ejecutando) al mismo tiempo. Esto trae como resultado que la Unidad Central de Procesamiento (CPU) siempre tenga alguna tarea que ejecutar, aprovechando al máximo su utilización.

Sistemas Operativos de tiempo compartido.

Permiten la simulación de que el sistema y sus recursos son todos para cada usuario. El usuario hace una petición a la computadora, esta la procesa tan pronto como le es posible, y la respuesta aparecerá en la terminal del usuario.

Sistemas Operativos paralelos.

En estos tipos de Sistemas Operativos se pretende que cuando existan dos o más procesos que compitan por algún recurso se puedan realizar o ejecutar al mismo tiempo.

Sistemas Operativos distribuidos.

Permiten distribuir trabajos, tareas o procesos, entre un conjunto de procesadores. Puede ser que este conjunto de procesadores esté en un equipo o en diferentes, en este caso es transparente para el usuario.

Los sistemas distribuidos deben de ser muy confiables, ya que si un componente del sistema se descompone otro componente debe de ser capaz de reemplazarlo.

Entre los diferentes Sistemas Operativos distribuidos que existen tenemos los siguientes: Sprite, Solaris-MC, Mach, Chorus, Spring, Amoeba, Taos, etc.

Características de los Sistemas Operativos distribuidos:

- Colección de sistemas autónomos capaces de comunicación y cooperación mediante interconexiones hardware y software
- Objetivo clave es la transparencia.
- Generalmente proporcionan medios para la compartición global de recursos.
- Servicios añadidos: denominación global, sistemas de archivos distribuidos, facilidades para distribución de cálculos (a través de comunicación de procesos internodos, llamadas a procedimientos remotos, etc.).

Sistemas Operativos de red.

Son aquellos sistemas que mantienen a dos o más computadoras unidas a través de algún medio de comunicación (físico o no), con el objetivo primordial de poder compartir los diferentes recursos y la información del sistema.

Principios De Los Sistemas Operativos

El diseño e investigación de herramientas para los sistemas operativos centralizados convencionales, los cuales corren en sistemas de uno o varios procesadores, esta muy bien entendido. Sin embargo la proliferación de estaciones de trabajo personales y redes de área local ha llevado al desarrollo de nuevos conceptos del sistema operativo, a saber sobre, sistemas operativos en red y sistemas operativos distribuidos. Es por eso que nos enfocaremos más en estos dos tipos de sistemas.

Antes de empezar no hay que confundir un **Sistema Operativo de Red** con un **Sistema Operativo Distribuido**. En un **Sistema Operativo de Red** las computadoras están interconectadas por medios de comunicación: software y hardware. En este tipo de red los usuarios saben donde están ejecutando su trabajo y guardando su información. En cambio en los **Sistemas Operativos Distribuidos**

existe un software que distribuye las tareas de los usuarios sobre una red de computadoras y para los usuarios es transparente donde realizan sus tareas y guardan su información.

Existen dos esquemas básicos de éstos sistemas. Un **sistema fuertemente acoplado** es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores. En un **sistema débilmente acoplado** los procesadores no comparten ni memoria ni reloj, ya que cada uno cuenta con su memoria local.

A medida que se van desarrollando los sistemas operativos, van adquiriendo nuevos conceptos, como son:

Interoperabilidad, que es la habilidad del sistema de facilitar intercambio de información entre los componentes heterogéneos en el sistema.

Transparencia, este concepto es muy parecido al de máquina virtual en los sistemas operativos tradicionales, la transparencia en los sistemas operativos distribuidos, ya que esta es la propiedad que permite a los usuarios ver al conjunto de máquinas en las que esta trabajando como un sola máquina.

Autonomía, esto es la independencia de los sistemas operativos con respecto al hardware, lo que permite que el sistema trabaje con unidades autónomas.

Sistemas Operativos Centralizados

Esta clase de sistemas es fuertemente acoplado ya que sus recursos son compartidos internamente. Un sistema operativo es una pieza grande de software formado por miles de millones de líneas de código. Cuando se implementa esta clase de sistemas es necesario estructurar el software en módulos manejables.

Podemos considerar un módulo como una colección de instrucciones que realiza un servicio del sistema.

Servicios de los sistemas operativos

Tipos de servicios

- **Ejecución de programas.** El sistema tiene que ser capaz de cargar un programa en memoria y ejecutarlo.
- **Operaciones de entrada/salida.** Como un programa no puede acceder directamente a un dispositivo de E/S el sistema operativo debe facilitarle algunos medios para realizarlo.
- **Manipulación del sistema de archivos.** El sistema operativo debe facilitar las herramientas necesarias para que los programas puedan leer, escribir y eliminar archivos.
- **Detección de errores.** El sistema operativo necesita constantemente detectar posibles errores. Los errores pueden producirse en la CPU y en el hardware de la memoria, en los dispositivos de E/S o bien en el programa de usuario. Para cada tipo de error, el sistema operativo debe

adoptar la iniciativa apropiada que garantice una computación correcta y consistente.

La estructura del **Kernel** en estos sistemas generalmente es monolítica, es decir, está dividido 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. Esto hace que el Kernel sea más manejable.

El modelo de Cliente-Servidor es una forma de describir la iteración entre procesos, a través del paso de mensajes, en donde el proceso cliente espera algún servicio (por ejemplo la lectura de datos de algún archivo) entonces envía un mensaje al servidor y después espera el mensaje de respuesta, en la forma más simple de este modelo, el sistema tiene solo dos primitivas **Enviar** y **Recibir**. La primera primitiva indica el destino y un mensaje, la segunda indica de quien es el mensaje y provee de un buffer donde será almacenado el mensaje.

Un sistema operativo estándar puede ser implementado en un núcleo mínimo, esto es un **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. En los sistemas operativos 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.

Los usos más comunes de los micronúcleos es en los sistemas operativos que intentan ser distribuidos, y en aquellos que sirven como base para instalar sobre ellos otros sistemas operativos. Por ejemplo, el sistema operativo AMOEBA intenta ser distribuido y el sistema operativo MACH sirve como base para instalar sobre él DOS, UNIX, etc.

Funciones de Manejo

Uno de los módulos más importantes de un sistema operativo es la de administrar los procesos y tareas del sistema de cómputo. Para realizar esto el sistema operativo ocupa la multiprogramación, este es un método para incrementar el empleo de la CPU disponiendo en todo momento de algo que la CPU pueda ejecutar. El trabajo que se realiza es el siguiente, cuando un proceso deja libre la

CPU para realizar una E/S, el sistema operativo cambia a otro trabajo y lo ejecuta. Cuando este último deba esperar por una E/S, la CPU pasará a otro trabajo y así sucesivamente.

Un sistema operativo multiprogramado es bastante sofisticado. Tener varios trabajos para ejecutar significa tenerlos simultáneamente en memoria. Tener varios programas en memoria requiere una gestión de memoria, además el sistema operativo deberá seleccionar para elegir uno de ellos. Esto se denomina planificación de CPU.

La planificación del procesador se refiere a la manera o técnicas que se usan para decidir cuánto tiempo de ejecución y cuando se le asignan a cada proceso del sistema. Obviamente, si el sistema es monousuario y monotarea no hay mucho que decidir, pero en el resto de los sistemas esto es crucial para el buen funcionamiento del sistema.

Lo más importante en esto es el uso de semáforos, en 1965, E.W. Dijkstra sugirió el uso de una variable entera para contar el número de despertares almacenados para su uso posterior.

En su propuesta se presentó un nuevo tipo de variable, llamada Semáforo. Un semáforo puede tener el valor 0, lo que indica que no existen despertares almacenados; o bien algún valor positivo si están pendientes uno o más despertares.

Dijkstra, propuso dos operaciones, DOWN y UP (generalizaciones de SLEEP y WAKEUP, respectivamente). La operación Down verifica si el valor de un semáforo es mayor que 0. En este caso, decrementa el valor (es decir, utiliza un despertar almacenado) y continúa. Si el valor es cero, el proceso se va a dormir. La verificación y modificación del valor, así como la posibilidad de irse a dormir se realiza en conjunto, como una sola e indivisible acción atómica. Se garantiza que al iniciar una operación con un semáforo, ningún otro proceso puede tener acceso a semáforo hasta que la operación termine o se bloquee. Esta atomicidad es absolutamente esencial para resolver los problemas de sincronización y evitar condiciones de competencia.

La operación UP incrementa el valor del semáforo correspondiente. Si uno o más procesos dormían en ese semáforo y no podían completar una operación Down anterior, el sistema elige alguno de ellos (por ejemplo, en forma aleatoria) y se le permite terminar Down. Así, después de un UP en un semáforo con procesos durmiendo, el semáforo seguirá con valor cero, pero habrá un menor número de procesos durmiendo. La operación de incremento del semáforo y despertar de un proceso también es indivisible. Ningún proceso llega a bloquear mediante un UP.

Un semáforo es una variable protegida, cuyo valor sólo puede ser leído y alterado mediante las operaciones P y V, y una operación de asignación de valores iniciales y (Inicia semáforo).

Operación P (Sobre el semáforo S)

si $S > 0$

entonces $S := S - 1$;

sino (esperar S)

Operación V (Sobre el semáforo S)

si (Uno o más procesos esperan S)

entonces (dejar que prosiga uno de esos procesos)

sino $S := S + 1$

Otra forma de controlar la comunicación entre procesos es la **sincronización**. En muchos casos, los procesos se reúnen para realizar tareas en conjunto, a este tipo de relación se le llama procesos cooperativos. Para lograr la comunicación, los procesos deben sincronizarse, de no ser así pueden ocurrir problemas no deseados. La sincronización es la transmisión y recepción de señales que tiene por objeto llevar a cabo el trabajo de un grupo de procesos cooperativos. Es la coordinación y cooperación de un conjunto de procesos para asegurar la comparación de recursos de cómputo. La sincronización entre procesos es necesaria para prevenir y/o corregir errores de sincronización debidos al acceso concurrente a recursos compartidos, tales como estructuras de datos o dispositivos de E/S, de procesos contendientes. La sincronización entre procesos también permite intercambiar señales de tiempo (ARRANQUE/PARADA) entre procesos cooperantes para garantizar las relaciones específicas de precedencia impuestas por el problema que se resuelve. Sin una sincronización adecuada entre procesos, la actualización de variables compartidas puede inducir a errores de tiempo relacionados con la concurrencia que son con frecuencia difíciles de depurar. Una de las causas principales de este problema es que procesos concurrentes puedan observar valores temporalmente inconsistentes de una variable compartida mientras se actualizan. una aproximación para resolver este problema es realizar actualizaciones de variables compartidas de manera mutuamente exclusiva. Se pueden mejorar permitiendo que a lo más un proceso entre a la vez en la sección crítica de código en la que se actualiza una variable compartida o estructura de datos en particular.

La sincronización en sistemas de un único ordenador no requiere ningún aspecto especial en el diseño del sistema operativo, ya que un reloj único proporciona de forma regular y precisa el tiempo en cada momento; sin embargo los sistemas distribuidos tienen un reloj por cada máquina por lo que es fundamental una coordinación entre todos los relojes para mostrar una hora única. Debemos sincronizar los relojes; ésta no es trivial por que se realiza a través de mensajes por la red, cuyo envío puede ser variable y depender de muchos factores, como la distancia, velocidad de transmisión o la propia saturación de la red.

Esto también para evitar las **regiones críticas**, cuando hablamos de región crítica nos referimos a un recurso que quiere ser utilizado por uno o varios procesos. Siempre es un recurso que está siendo utilizado por un proceso y existe otro proceso lo desea por lo que debe esperar que el primer recurso lo libere y de esta manera pueda utilizarlo.

Si un proceso se encuentra en su sección crítica, otros procesos pueden seguir ejecutándose fuera de sus secciones críticas. Cuando un proceso abandona su región, otro proceso que esperaba entrar en su propia sección podrá hacerlo. El problema de la programación concurrente está en que se cumpla la exclusión mutua.

Cuando un proceso se encuentra en una región crítica se está hablando de un estado especial que se concede a un proceso. El proceso tiene acceso exclusivo a los datos compartidos y los demás procesos que requieren acceso a esos datos y en ese momento deben esperar; por esto las secciones críticas deben ejecutarse tan rápido como sea posible. Un proceso no se debe bloquear dentro de su propia sección crítica y estas deben codificarse con mucho cuidado

Necesitamos 4 condiciones para poder obtener una buena solución:

1. Dos procesos no deben encontrarse al mismo tiempo dentro de sus secciones críticas .
2. No se deben hacer hipótesis sobre la velocidad o el número de UCP.
3. Ninguno de los procesos que estén en ejecución fuera de su sección crítica puede bloquear a otros procesos.
4. Ningún proceso debe esperar eternamente para entrar a su sección crítica.

Manejo de los dispositivos de E/S

El código destinado a manejar la entrada y salida de los diferentes periféricos en un sistema operativo es de una extensión considerable y sumamente complejo. Resuelve la necesidades de sincronizar, atrapar interrupciones y ofrecer llamadas al sistema para los programadores.

Los dispositivos de entrada salida se dividen, en general, en dos tipos: dispositivos orientados a bloques y dispositivos orientados a caracteres. Los dispositivos orientados a bloques tienen la propiedad de que se pueden direccionar, esto es, el programador puede escribir o leer cualquier bloque del dispositivo realizando primero una operación de posicionamiento sobre el dispositivo. Los dispositivos más comunes orientados a bloques son los discos duros, la memoria, discos compactos y, posiblemente, unidades de cinta. Por otro lado, los dispositivos orientados a caracteres son aquellos que trabajan con secuencias de bytes sin importar su longitud ni ninguna agrupación en especial. No son dispositivos direccionables. Ejemplos de estos dispositivos son el teclado, la pantalla o display y las impresoras.

En el manejo de los dispositivos de E/S es necesario, introducir dos nuevos términos:

Buffering (uso de memoria intermedia)

El buffering trata de mantener ocupados tanto la CPU como los dispositivos de E/S. La idea es sencilla, los datos se leen y se almacenan en un buffer, una vez que los datos se han leído y la CPU va a iniciar inmediatamente la operación con ellos, el dispositivo de entrada es instruido para iniciar inmediatamente la siguiente lectura. La CPU y el dispositivo de entrada permanecen ocupados. Cuando la CPU esté libre para el siguiente grupo de datos, el dispositivo de entrada habrá terminado de leerlos. La CPU podrá empezar el proceso de los últimos datos leídos, mientras el dispositivo de entrada iniciará la lectura de los datos siguientes.

Para la salida, el proceso es análogo. En este caso los datos de salida se descargan en otro buffer hasta que el dispositivo de salida pueda procesarlos.

Este sistema soluciona en forma parcial el problema de mantener ocupados todo el tiempo la CPU y los dispositivos de E/S. Ya que todo depende del tamaño del buffer y de la velocidad de procesamiento tanto de la CPU como de los dispositivos de E/S.

El manejo de buffer es complicado. Uno de los principales problemas reside en determinar tan pronto como sea posible que un dispositivo de E/S a finalizado una operación. Este problema se resuelve mediante las *interrupciones*. Tan pronto como un dispositivo de E/S acaba con una operación interrumpe a la CPU, en ese momento la CPU detiene lo que está haciendo e inmediatamente transfiere el control a una posición determinada. Normalmente las instrucciones que existen en esta posición corresponden a una rutina de servicio de interrupciones. La rutina de servicio de interrupción comprueba si el buffer no está lleno o no está vacío y entonces inicia la siguiente petición de E/S. La CPU puede continuar entonces el proceso interrumpido.

Cada diseño de computadora tiene su propio mecanismo de interrupción, pero hay varias funciones comunes que todos contemplan.

El buffering puede ser de gran ayuda pero pocas veces es suficiente.

Spooling

El problema con los sistemas de cintas es que una lectora de tarjetas no podía escribir sobre un extremo mientras la CPU leía el otro. Los sistemas de disco eliminaron esa dificultad, moviendo la cabeza de un área del disco a otra.

En un sistema de discos, las tarjetas se leen directamente desde la lectora sobre el disco. La posición de las imágenes de las tarjetas se registran en una tabla mantenida por el sistema operativo. En la tabla se anota cada trabajo una vez leído. Cuando se ejecuta un trabajo sus peticiones de entrada desde la tarjeta se satisfacen leyendo el disco. Cuando el trabajo solicita la salida, ésta se copia en el buffer del sistema y se escribe en el disco. Cuando la tarea se ha completado se escribe en la salida realmente.

Esta forma de procesamiento se denomina *spooling*, utiliza el disco como un buffer muy grande para leer tan por delante como sea posible de los dispositivos de entrada y para almacenar los ficheros hasta que los dispositivos de salida sean capaces de aceptarlos.

La ventaja sobre el buffering es que el spooling solapa la E/S de un trabajo con la computación de otro. Es una característica utilizada en la mayoría de los sistemas operativos.

Afecta directamente a las prestaciones. Por el costo de algo de espacio en disco y algunas tablas, la CPU puede simultanear la computación de un trabajo con la E/S de otros. De esta manera, puede mantener tanto a la CPU como a los dispositivos de E/S trabajando con un rendimiento mucho mayor.

Además mantiene una estructura de datos llama job spooling, que hace que los trabajos ya leídos permanezcan en el disco y el sistema operativo puede seleccionar cual ejecutar, por lo tanto se hace posible la planificación de trabajos.

Manejo de Archivos

Un archivo es un conjunto de información, que se encuentra almacenada o guardada en la memoria principal del computador, en el disco duro, en el disquete flexible o en los discos compactos (Cd-Rom).

Antes de que un archivo pueda leerse o escribirse en él, debe abrirse, momento en el cual se verifican los permisos. Estos archivos se abren especificando en el computador la ruta de acceso al archivo desde el directorio raíz, que es la unidad principal del disco del computador, este puede ser un disco duro o disco flexible. Entonces el sistema operativo visualiza el entorno al abrir un archivo.

Uno de los problemas mas frecuentes en el manejo de archivos son los **deadlock**, un deadlock es una situación no deseada de espera indefinida y se da cuando en un grupo de procesos, dos o más procesos de ese grupo esperan por llevar a cabo una tarea que será ejecutada por otro proceso del mismo grupo, entonces se produce el bloqueo. Los bloqueos se dan tanto en los sistemas operativos tradicionales como en los distribuidos, solo que en estos últimos es más difícil de prevenirlos, evitarlos e incluso detectarlos, y si se los logra detectar es muy complicado solucionarlos ya que la información se encuentra dispersa por todo el sistema.

Una vez que un deadlock se detecta, es obvio que el sistema está en problemas y lo único que resta por hacer es una de dos cosas: tener algún mecanismo de suspensión o reanudación que permita copiar todo el contexto de un proceso incluyendo valores de memoria y aspecto de los periféricos que esté usando para reanudarlo otro día, o simplemente eliminar un proceso o arrebatarle el recurso, causando para ese proceso la pérdida de datos y tiempo.

Seguridad en los Sistemas Operativos

En los sistemas operativos se requiere tener una buena seguridad informática, tanto del hardware, programas y datos, previamente haciendo un balance de los requerimientos y mecanismos necesarios. Con el fin de asegurar la integridad de la información contenida.

Dependiendo de los mecanismos utilizados y de su grado de efectividad, se puede hablar de sistemas seguros e inseguros. En primer lugar, deben imponerse ciertas características en el entorno donde se encuentra la instalación de los equipos, con el fin de impedir el acceso a personas no autorizadas, mantener un buen estado y uso del material y equipos, así como eliminar los riesgos de causas de fuerza mayor, que puedan destruir la instalación y la información contenida.

En la actualidad son muchas las violaciones que se producen en los sistemas informáticos, en general por acceso de personas no autorizadas que obtienen información confidencial pudiendo incluso manipularla. En ocasiones este tipo de incidencias resulta grave por la naturaleza de los datos; por ejemplo si se trata de datos bancarios, datos oficiales que puedan afectar a la seguridad de los estados, etc.

El software mal intencionado que se produce por diversas causas, es decir pequeños programas que poseen gran facilidad para reproducirse y ejecutarse, cuyos efectos son destructivos nos estamos refiriendo a los virus informáticos.

Para esto, se analizan cuestiones de seguridad desde dos perspectivas diferentes la seguridad externa y la seguridad interna.

Todos los mecanismos dirigidos a asegurar el sistema informático sin que el propio sistema intervenga en el mismo se engloban en lo que podemos denominar **seguridad externa**.

La seguridad externa puede dividirse en dos grandes grupos :

Seguridad física. Engloba aquellos mecanismos que impiden a los agentes físicos la destrucción de la información existente en el sistema; entre ellos podemos citar el fuego, el humo, inundaciones descargas eléctricas, campos magnéticos, acceso físico de personas con no muy buena intención, entre otros.

Seguridad de administración. Engloba los mecanismos más usuales para impedir el acceso lógico de personas físicas al sistema.

Todos los mecanismos dirigidos a asegurar el sistema informático, siendo el propio sistema el que controla dichos mecanismos, se engloban en lo que podemos denominar **seguridad interna**.

Sistemas Operativos

UNIX: El fenómeno

Los orígenes de UNIX se remontan al año 1962 en el que el CTSS y el MIT se encuentran investigando en áreas de tiempo compartido y protección. En 1965, Bell Labs (la división de investigación de AT&T), General Electric y el MIT se encuentran trabajando en un macroproyecto llamado MULTICS, previsto para desarrollar una gran potencia de cálculo y almacenamiento de muchos usuarios. De este proyecto, se obtuvieron interesantes resultados (capacidad de multiproceso, árbol de ficheros, shell); pero, como todo proyecto gigante, su complejidad desbordó al equipo que lo emprendió (seguramente no habían estudiado cibernética o teoría de complejidad) así que en 1969 fue abandonado. El caso, es que una de las mejores "cosas" que salieron de allí fue un tal Ken Thompson, un tanto "mosqueado", eso sí, pero con ideas propias que le llevaron a desarrollar ese mismo año un sistema de ficheros propio. A Thompson, en realidad, lo que le interesaba era derrotar al imperio Klingon jugando al Star Trek, así que se montó una simulación de la galaxia que quitaba el aliento en un sistema GECOS. Y si no lo quitaba, al menos eso le pareció a una tal Dennis Ritchie, que pasaba por allí y también veía Star Trek. El caso es que Thompson encontró un PDP-7 (otro ordenador más potente) y construyó para él su sistema de ficheros para poder jugar mejor con Ritchie sin que nadie les viera.

Bueno, esta es la leyenda que dice que los orígenes de UNIX vienen "de Vulcano". Puede que no fuera así, pero lo cierto es que muchos grandes avances han surgido del desarrollo que grandes hombres han hecho para su disfrute en ratos de "ocio" y este fue uno de ellos. Sea como fuere, en el año '71, Ritchie y Kernighan crean y utilizan el lenguaje C en un PDP-11 (algo así como un AT), lenguaje nacido para la programación de sistemas. Así, dos años después en 1973, Ritchie y Thompson re-escriben su engendro en lenguaje C, pero esta vez desarrollan un sistema multiusuario. UNIX había nacido. El sistema, nacido del trabajo y la ilusión de sólo dos hombres, demostró ser algo tan bueno que ese mismo año Bell Labs contaba con 25 instalaciones funcionando con UNIX.

En 1974 aparece un artículo en las comunicaciones del ACM (Association for Computer Machinery) y se distribuye a las universidades. En 1977 ya son 500 los centros y 125 las universidades que utilizan el sistema. Su expansión es fulgurante ya que se distribuye sin licencias y con fuentes. Entre 1977 y 1982 se combina con un sistema comercial y nace UNIX System III. Ya en 1984 existen 100.000 sistemas UNIX en todo el mundo.

Paralelamente en Berkeley... Entre las universidades a la que llegó UNIX, se encontraba la University of California Berkeley. Allí se modificó el sistema incorporando una variante notable: la utilización de memoria virtual paginada. Así en 1978 surge UNIX 3BSD (Berkeley Software Distribution). En 1980 DARPA (Defense Advanced Research Projects Office), verdadero motor de investigación en USA, (sí, sí, la de la famosa DARPA InterNet, que pasó a ARPANet o ARPA Internet y de ahí a sólo internet), subvenciona el desarrollo de 4BSD. Poco después surge 4.1BSD incorporando nuevas utilidades como el clásico editor vi y la shell csh. En 1982 SUN desarrolla para sus arquitecturas el sistema SunOS basado en la versión BSD. Un año después surge la versión 4.2BSD que incorpora DEC en VAX y adopta SUN Microsystems. 1984 marca un nuevo hito en la historia de UNIX ya que SUN desarrolla los conceptos de RPC (Remote Procedure Call) y NFS (Network File System) y los incorpora a SunOS.

Panorama actual: familias de UNIX. Cada fabricante ha ido desarrollando sus estándares: AT&T SVID, el interfaz SYSTEM V; HP, UP-UX (tipo SYSTEM V); DEC, ULTRIX (tipo 4.2BSD); Microsoft, XENIX; e IBM, AIX. Como se puede ver, en líneas generales, existen dos tipos de UNIX: tipo BSD y tipo SYSTEM V. También existen implementaciones de tipo académico, como MINIX, desarrollada por Tanenbaum con afán didáctico en 1983; o XINU, desarrollada por Comer en 1984. La razón de los distintos nombres que recibe el sistema es que UNIX es una marca registrada de AT&T. Así, cada implementación recibe el suyo propio.

En la actualidad Berkeley acaba de anunciar la salida de 4.4BSD y su retirada del mundo UNIX, por lo que el futuro es de SYSTEM V. SUN, por ejemplo, ha pasado ya a este sistema en la última versión de su S.O. Solaris. Sin embargo, en los últimos años se ha buscado la convergencia de los sistemas. Así, desde el punto de vista de programación, BSD ha ganado la batalla ya que su interfaz de sockets a pasado a ser el medio clásico de comunicación entre procesos. En su momento, los usaremos.

Se pueden encontrar muchos otros UNIX: SCO, NetBSD, FreeBSD, LINUX, GNU (...) y cada vez más para el mundo PC. En este momento nos detendremos para dirigir una breve mirada a LINUX.

Cronología

1969 Desarrollo original por Thompson y Ritchie (Laboratorios Bell AT&T) sobre un PDP-7.

1970 Versión de 2 usuarios sobre DEC PDP-11.

1971 Versión multiusuario sobre PDP-11/40,45,70.

1973 Re-escritura del S.O. en C (Kernighan, Ritchie) ya que originalmente estaba programado en ensamblador. De esta forma se podía transportar a otras máquinas.

1974 Empieza la explotación comercial (25.000 dólares) de las fuentes.

1975 Versión 6 de UNIX y cesión a Universidades para su enseñanza.

1983 Aparece el UNIX System V (ATT) con soporte para mensajes y memoria compartida. También aparece una versión para PC: XENIX (Microsoft).

1989 UNIX System V, R4 con soporte para RFS, Streams, Redes y Tiempo Real.

En 1993 Novell compra UNIX a la compañía AT&T

En 1994 Novell le da el nombre "UNIX" a X/OPEN

En 1995 Santa Cruz Operations le compran UnixWare a Novell. Santa Cruz Operations y Hewlett-Packard anuncia que desarrollarán conjuntamente una versión de Unix de 64-bit.

En 1996 International Data Corporation prevee que en 1997 habrá mas de 3 millones de computadoras con Unix a nivel mundial.

Introducción

El padre de Linux es Linus Torvalds, un programador finlandés de 21 años que inicialmente no tenía más pretensión que 'divertirse' creando un sistema operativo para su uso personal. Torvalds colocó Linux en Internet para que cualquiera lo bajara gratis, en 1991, y desde entonces participan en su desarrollo cientos de voluntarios. Hoy Linux se difunde más rápido que cualquier otro sistema operativo,

es venerado por una comunidad de diez millones de usuarios y comienza a verse como una alternativa real a Windows. Esta es su historia.

1991

- En abril, Linus Torvalds comenzó a crear un programa que varios meses después se convertiría en Linux, un sistema operativo Unix para PC (Unix es un sistema operativo usado en estaciones de trabajo y otros computadores de alto rendimiento; hay muchas versiones de Unix). Linux nació como un pasatiempo de Torvalds, que en esa época tenía 21 años y estudiaba segundo año de ciencias de los computadores en la Universidad de Helsinki (Finlandia); su intención inicial no era crear un sistema operativo, sino experimentar creando software para manejar ciertas funciones básicas del PC que había comprado cuatro meses antes (un sistema operativo es el programa que controla el funcionamiento de un computador). La inquietud surgió porque el sistema operativo que usaba en su PC, llamado Minix, era una versión de Unix limitada, y él necesitaba algo mejor; Torvalds no usaba DOS o Windows porque le parecían –aún hoy– sistemas pobres y poco confiables (Minix es un sistema operativo Unix experimental, creado por un profesor holandés para enseñar a los estudiantes el funcionamiento de los sistemas operativos).

- A mediados del año, el programa estaba avanzado y Torvalds comenzó a pensar seriamente en que podría crear el kernel de un nuevo sistema operativo, similar a Minix pero mejor (el kernel es el corazón de un sistema operativo). Torvalds no tenía grandes pretensiones; él dice que no pensó en crear Linux para que fuera un sistema operativo profesional, sino que lo diseñó para su uso personal. Sin embargo, poco a poco su pasatiempo se fue convirtiendo en un proyecto más serio.

- El 5 de octubre, Torvalds publicó en un grupo de noticias sobre Minix, en Internet, un mensaje en el que anunció la creación de Linux, un sistema operativo para PC basados en procesadores Intel 386. El mensaje decía que el sistema (Linux versión 0.02) estaba todavía en desarrollo, pero ya funcionaba, y lo ofrecía gratis a cualquiera que deseara bajarlo. También invitó a los programadores interesados en sistemas operativos a usarlo, y enviarle correcciones y mejoras para incluirlas en la próxima versión. Ese fue un suceso clave en la historia de Linux; gracias a Internet, Linux pronto se convertiría en un fenómeno mundial.

1992

- En enero, Linux tenía cerca de 100 usuarios, y varios de ellos ya participaban en el desarrollo de Linux con mejoras y correcciones que enviaban a Torvalds por Internet. Él lanzó ese mes la versión 0.12 de Linux; esa fue la primera versión que incluyó partes desarrolladas por otros programadores y la primera que realmente se desempeñaba mejor que Minix en ciertos aspectos.

-Microsoft lanzó Windows 3.1.

- El número de usuarios de Linux comenzó a crecer rápidamente, y no era extraño que tuviera gran acogida. Al igual que Torvalds, muchos estudiantes de sistemas y gomosos de la computación amaban los sistemas operativos Unix por su estabilidad y potencia, pero estos eran inalcanzables porque una versión comercial de Unix costaba en esa época 4.000 o 5.000 dólares, y casi todas funcionaban exclusivamente en estaciones de trabajo de 10.000 o más dólares (no en PC). Linux, en cambio, era un sistema Unix gratuito, y funcionaba en PC basados en procesadores Intel (386, 486, etc.).

- A medida que creció el número de usuarios, también aumentaron los programadores voluntarios que se involucraron en el desarrollo de Linux. Torvalds distribuyó Linux bajo un tipo de licencia llamada GPL, que permite a cualquier persona bajar, usar, modificar e incluso vender Linux, sin pagar un peso; la única condición es que los cambios o mejoras que una persona o compañía realicen en Linux también deben ser públicos. Es generó un fenómeno de colaboración mundial sin precedentes. Programadores de todo el planeta enviaron a Torvalds mejoras para el kernel, reportaron errores y comenzaron a crear controladores de dispositivos para Linux. Se calcula que al final de 1992 Linux tenía aproximadamente 1.000 usuarios.

1993

- Se estima que este año Linux completó 20.000 usuarios en el mundo, y más de 100 programadores contribuyeron en su desarrollo. Para poder manejar esas colaboraciones, Torvalds delegó las labores de revisión del código de programación de Linux a cinco personas, que se convirtieron en sus 'oficiales' principales. A diferencia de los programas comerciales, que se actualizan cada dos o tres años, en el mundo Linux aparecen actualizaciones y mejoras menores cada pocas semanas; eso ha permitido que Linux evolucione rápidamente.
- Microsoft lanzó Windows NT, una versión de Windows para servidores y estaciones de trabajo (es, por ello, rival de los sistemas operativos Unix).

1994

- En marzo se lanzó la primera versión 'completa' del sistema operativo de Torvalds: Linux 1.0. Esta versión ofreció soporte a redes de computadores, e incluyó docenas de utilidades, programas de desarrollo de aplicaciones y otras herramientas.
- Se fundó Red Hat Software, una empresa que hoy es el principal distribuidor de Linux. Aunque Linux se puede bajar gratis de Internet, hay docenas de empresas – como Red Hat Software y Caldera– que elaboran sus propias versiones, y las venden en CD-ROM, junto con manuales, soporte técnico y programas adicionales (esto es lo que se conoce en el mundo de Linux como una distribución). Estas distribuciones cuestan entre 10 y 70 dólares, dependiendo de la empresa.
- Este año Linux completó aproximadamente 100.000 usuarios.

1995

- En agosto, Microsoft lanzó Windows 95.
- A finales de este año Linux tenía aproximadamente 500.000 usuarios.

1996

- El 9 de junio se lanzó la versión 2.0 de Linux. Una de las principales novedades fue el soporte a multiprocesamiento simétrico (el sistema aprovechaba el poder de computadores con más de un procesador). Además, Linux 2.0 no solo trabajaba en PC con procesadores Intel x86 (como el 386, 486 y Pentium), sino también en estaciones de trabajo con procesadores Alpha.
- Se calcula que este año Linux completó 1,5 millones de usuarios.

1997

- Linus Torvalds se fue a vivir a Santa Clara (California, Estados Unidos), debido a que fue contratado por una compañía llamada Transmeta (es una empresa de chips, que no está relacionada con Linux). Sin embargo, Torvalds continuó encabezando el equipo de gente que se encarga del desarrollo del kernel de Linux.

- La firma de investigaciones Datapro dijo que Linux era el segundo sistema operativo más popular en los servidores web de Internet, después de Solaris (un sistema Unix de Sun Microsystems).

- Se estima que Linux completó 3,5 millones de usuarios. Este año se lanzó la versión 2.1.

1998

- Varios de los principales fabricantes de programas para el mercado corporativo, como Oracle, Informix, Computer Associates (CA) y Netscape, anunciaron que lanzarán versiones para Linux de sus productos. El respaldo de estas empresas ha sido clave para la consolidación de Linux en las empresas.

- En junio, Microsoft lanzó Windows 98.

- En septiembre, Intel Corporation y Netscape anunciaron una inversión de capital en la empresa Red Hat Software. Este hecho fue muy importante para aumentar la credibilidad de Linux, debido a que Intel y Netscape son dos de los líderes de la industria de los computadores.

- En diciembre, Corel Corporation lanzó una versión para Linux de su procesador de palabra WordPerfect 8.

El programa se colocó en Internet para que los usuarios lo pudieran probar gratis durante 90 días, y en los primeros seis meses lo bajaron un millón de personas.

- A finales de 1998, Linux dominaba cerca del 17 por ciento del mercado de sistemas operativos para redes, según la empresa de investigación de mercados International Data Corporation (IDC).

- Se calcula que Linux completó 7,5 millones de usuarios. Y el número de programadores que participan en el desarrollo y pruebas del programa creció a 10.000.

1999

- En enero se lanzó la versión 2.2 de Linux, que ofreció un mejor rendimiento y soporte para procesadores Sparc, Motorola 68000, PowerPC y MIPS. Esta versión, al igual que la 2.0, soporta computadores con 8 procesadores, pero el multiprocesamiento es mucho más eficiente en la versión 2.2.

- Corel Corporation anunció que antes de terminar este año lanzará Corel Linux, una distribución de Linux dirigida a usuarios de PC. Aunque hay muchas empresas que ofrecen versiones comerciales de Linux, esta tiene gran relevancia por estar dirigida a usuarios comunes y por ser producida por uno de los más grandes fabricantes de software del mundo. Corel también dijo que en el año 2000 lanzará una versión para Linux del programa gráfico más importante del mundo Windows, CorelDraw, lo mismo que una versión de su paquete de programas Corel WordPerfect Suite.

- Linus Torvalds dijo que a finales de 1999 se lanzará la versión 2.4 del kernel de Linux. La versión 2.4 (la actual es la 2.2.11) mejorará el soporte a multiprocesamiento simétrico, y soportará tecnologías como Universal Serial Bus (USB) y PCMCIA (actualmente tienen soporte, pero por medio de programas adicionales).

- Actualmente, Linux tiene gran acogida y un futuro prometedor. Se calcula que cuenta con más de diez millones de usuarios. Además, trabaja en los principales tipos de procesadores: Intel x86, Motorola 680x0, MIPS, PowerPC (los que usan los Macintosh), Alpha y Sparc (estos dos últimos son procesadores de 64 bits, más potentes que los chips Intel x86). Incluso, hay una versión que funciona en el computador de mano PalmPilot. De otro lado, varios de los principales fabricantes de computadores, como Dell Computer, Compaq y Hewlett-Packard, venden equipos que traen Linux preinstalado.

Sistemas Operativos

OS/2: El divorcio entre IBM y Microsoft

OS/2 fue desarrollado originalmente entre IBM y Microsoft como un sucesor multiproceso del DOS para CPUs 286 y mejor, pero la versión 1.x nunca fue aceptada excepto para algunas aplicaciones específicas. Con la versión 2.0, Microsoft dejó la sociedad OS/2, e IBM promovió el OS/2 como un sistema operativo de 32-bit que requería un CPU 386 o mejor. Esta configuración básica no ha cambiado en el OS/2 2.1 o 3.0. La versión que siguió de OS/2 "Merlin" (probablemente llamada 4.0 cuando fuese liberada), no se probará con CPUs 386, y a estas alturas está desconocido si trabajará en un sistema 386 en lo absoluto.

El OS/2 Warp 3.0 es multiproceso, 32-bit, mono-usuario para 386SX y CPU mejores con 4MB o más de RAM. Esta simula al DOS en varias maneras (tal como los órdenes de línea de comandos, y la presencia de un archivo CONFIG.SYS), pero se parece al Mac en otras maneras (ej., la representación de iconos representando archivos) y tiene una similitud con otros operativos en otras maneras (ej., menús que aparecen al pulsar un botón en el tablero (desktop) mismo, estas se usan en X Windows bajo UNIX). Warp incluye una Interface Gráfica para el Usuario (GUI) conocido como el Manejador de Presentación (PM), y un Manejador del escritorio del cual se ejecutan los programas y manipulan los archivos conocido como el WorkPlace Shell (WPS). El PM se asemeja al Windows pero no es igual. El WPS es similar a Windows 95 y el Buscador de Mac, pero generalmente es más flexible y más orientado a objetos que ambos. Una versión de Windows del WPS esta disponible.

OS/2 Warp 3.0 entra a varias versiones con varios niveles de gestión de redes. La primera versión tiene gestión de redes solo por medios Telefónicos(Dial-Up). Otras dos versiones, "Warp Connect" y "Warp Serve", incluye capacidades cliente /servidor, respectivamente. La próxima versión de OS/2 cuyo nombre código es "Merlin", incluirá manejo de red cliente en el paquete básico. Al usarse con Windows para trabajo en grupos 3.1, (WFW), la capacidad de red esta inhabilitada bajo OS/2, pero se puede utilizar esta en WFW, si se ejecuta desde DOS nativo.

La versión mínima de OS/2 también viene en dos versiones cada una, una que incluye una versión recopilada de Windows (También llamada Win-OS/2), requiere que el usuario tenga Windows 3.1 para ejecutar programas Windows desde OS/2 (También referenciada como la versión para Windows, este termino no es oficial). La versión win-OS/2 algunas veces ejecuta programas Windows con leve mejora de velocidad que en la versión para Windows, y es mas fácil de instalar si el usuario no tiene instalado Windows 3.1; pero la versión para Windows es mas barata.

Warp puede ejecutar en modo texto a OS/2 GUI, y programas DOS. Windows es un programas DOS que Warp puede ejecutar, y es de esta forma que OS/2 provee soporte para Windows; ejecutando Windows sobre modo DOS. Este método de soporte Windows no cambió en Merlin. Nótese que Windows 95 no proveerá soporte de programas Windows para OS/2.

Mucho usuarios nuevos de OS/2, podrán conseguir la versión original "Para Windows", de Warp. La versión "Para Windows", es un poco más barata y usa menos espacio en disco que la versión completa. Alguno que se actualice de la versión 2.1 OS/2 completa debe comprar la versión completa de actualización Warp, que incluye la detección de la versión vieja 2.1 y no se instalará si no la detecta. Alguno que este armando una computadora nueva y que aún no tenga Windows u OS/2 2.X, pero que desee ejecutar programas Windows, debe comprar

la versión completa del OS/2 que no es actualización la cual es más cara, pero es más conveniente que comprar la versión "Para Windows", y una copia separada de Windows.

Empresa que lo creó OS/2

Fue originalmente diseñado por Microsoft con la ayuda de IBM. Desde este punto de vista estas compañías OS/2 iba a reemplazar a MS-DOS. Esto nunca sucedió OS/2 se entregó tarde en incompleto. Aunque tenía unas ventajas obvias sobre MS-DOS, como el uso real de memoria, la ejecución en modo protegido y el soporte de multiprogramación en forma elegante, los usuarios no se interesaron en él.

Versiones de OS/2

OS/2 1.0 (1987) fue originalmente diseñado por Microsoft con la ayuda de IBM. Cuando el procesador 286 era el último y mas grande chip. Desde entonces Microsoft se percató que Dos se estaba quedando atrás.

Ellos introdujeron OS/2 1.0 en 1987, el cual corre en modo texto programas que eran extremadamente poderosos.

Desde entonces sobrepasan más de los límites de DOS, usando aun comandos estilo DOS en la línea de prompt. Con esta ventaja usted no tiene que conocer UNIX para obtener aplicaciones multitareas.

OS/2 1.1(1988) - 1.3 (1991) incluyó el administrador de presentación, el cual se veía como Windows 3.X. (Windows 3.X tardaría para ser mas avanzado que el Presentation Manager). El apoyo a DOS fue adicionado con algunas otras características como alta o excelente ejecución del sistema de archivos, el cual en verdad permitía nombre de archivos largos, y esto es típicamente mucho más rápido que (V) FAT y permitía volúmenes de tamaño mas largo son aquellos torpes o en cómodos programas administradores de discos. Justo alrededor de esto Microsoft abandonó el soporte a OS/2 y seguro con el desarrollo de Windows, esperando obtener o abarcar el mercado con su versión de bajo consumo de memoria OS/2 Presentation Manager la cual podrá correr en los sistemas de DOS existentes, Windows y entonces regresar y desarrollar el proyecto de OS/2 dentro de Windows NT, la cual obtendría la otra gran parte del mercado. Mientras el plan de alcanzar la parte más baja del mercado tuvo éxito. IBM ya ha obtenido la parte más alta del mercado con el OS/2.

Actualmente la batalla continua. Mientras IBM lucha para mantener a OS/2 mejor sistema en la parte alta del mercado y trazó su plan para competir en la parte más baja del mercado en los próximos 2 años.



Linux, otra opción en sistemas operativos

Raul Lusky

Linux es una versión de Unix libremente distribuible e independiente, para plataformas con maquinas x86, Motorola 68k, Digital Alpha, Sparc, Mips y Motorola Power PC. En la actualidad, este sistema operativo es utilizado por miles de usuarios para desarrollo de software, redes y para plataformas de usuarios finales.

Linux, entre los miles de sistemas operativos alternos que existen, se ha convertido en una opción interesante, independientemente de que estas vengan de UNIX o de las más conocidas donde se encuentra Windows y NT.

Es una implantación de la especificación POSIX con la cual cumplen todas las verdaderas versiones de UNIS.

El núcleo de Linux no usa código de AT&T o de cualquier otra fuente propietaria, la mayoría de los programas disponibles para Linux es desarrollado por el proyecto GNU de la Free Software Foundation.

Este soporta un amplio espectro de aplicaciones o paquetes de programación tales como X Window, Emacs, redes de datos bajo protocolos TCP/IP (incluyendo SLIP, PPP, ISDN). Linux está disponible en Internet en cientos de servidores ftp y en distribuidores en discos CD-ROM de revendedores que lo ofrecen empacado con manuales e información que es realmente la del costo, pues el programa es gratuito.

Algunos de estos son: [Caldera](#), [Debian](#), [Slackware](#), [Red Hat](#), etc. Uno de los servidores más populares que ofrecen Linux está ubicado en <ftp://sunsite.unc.edu/pub/Linux/distributions>, con una gran cantidad de mirrors alrededor del mundo.

El núcleo del Linux está legalmente protegido por la licencia publica GNU (GPL). Linux incluye compiladores, ensambladores, debuggers, editores de texto, paquetes de email, lectores de noticias, navegadores, servidores y programas para la creación y edición gráfica. Linux, maneja los archivos de forma jerárquica, de la misma forma que DOS, con la diferencia que el DOS está diseñado para procesadores x86 que no soportan verdaderas capacidades de múltiples tareas.

Historia

Linux fue creado originalmente por Linus Benedict Torvalds en la Universidad de Helsinki en Finlandia.

Este ha sido desarrollado con la ayuda de muchos programadores a través de Internet. Linus originalmente inició el hacking del núcleo como su proyecto favorito, inspirado por su interés en MINIX, un pequeño sistema Unix. El se propuso a crear lo que en sus propias palabras sería un mejor Minix que el Minix.

El 5 de octubre de 1991, Linus anunció su primera versión "oficial" de linux, versión 0.02. Desde entonces, muchos programadoras han respondido a su llamado, y han ayudado a construir Linux como el sistema operativo completamente funcional que es hoy. La ultima versión estable es la versión 2.2, que soporta muchos más periféricos, desde procesadores hasta joysticks, sintonizadores de televisión, CD ROMs no ATAPI y reconoce buena cantidad de tarjetas de sonido. Incluye también soporte para tipos de archivos para Macintosh HFS, Unix UFS y en modo de lectura, HPFS de OS/2 y NTFS, de NT.

Ventajas

1. Precio.
2. [Estabilidad](#), no se traba a cada rato.
3. [Seguridad](#), es mucho mas seguro que otros servidores.
4. [Compatibilidad](#), reconoce la mayoría de los otros sistemas operativos en una red.
5. [Velocidad](#), es mucho mas veloz para realizar las tareas.
6. Posee el apoyo de miles de programadores a nivel mundial.
7. El paquete incluye el código fuente, lo que permite modificarlo de acuerdo a las necesidades del usuario.
8. Ideal para la programación, ya que se puede programar en Linux para distintas plataformas, como para Windows.
9. Un sistema de crecimiento rápido.
10. Se puede usar en casi cualquier computadora, desde una 386.
11. Multitareas [REAL](#).
12. Puede manejar múltiples procesadores. Incluso hasta 16 procesadores.
13. [Libre de virus](#), aun no se conoce ningún virus para Linux.
14. Maneja discos duros de hasta 16 TeraBytes.
15. Se consiguen parches con facilidad, además de ser gratuitos.
16. Se posee el apoyo de millones de usuarios a nivel mundial.
17. Los fabricantes de Hardware le están dando su apoyo, como [IBM](#) y [COMPAQ](#).
18. Vendedores y desarrolladores implementan un sistema de certificación para Linux.
19. La corporación DATA Internacional predice que el crecimiento de este programa será del orden de un 25 por ciento anual en el nuevo milenio.

Desventajas

1. Linux no cuenta con una empresa que lo respalde, por lo que no existe un verdadero soporte como el de otros sistemas operativos.
2. Linux corre el riesgo de llegar a fragmentarse como fue el caso de UNIX.
3. Algunas empresas pueden llegar a ayudar a Linux con la intención de mejorar sus relaciones públicas, aunque en el fondo no tengan ninguna intención de utilizarlo fielmente.