# Unidad I Conceptos fundamentales

# TABLA DE CONTENIDOS

# Introducción a los sistemas operativos:

El objetivo de este módulo, es brindar al alumno los conceptos básicos y las diferentes maneras de resolver los múltiples problemas que se presentan a la hora de utilizar un sistema informático y que son resueltos y atendidos sin ser percibidos por el usuario gracias a la intervención del Sistema Operativo.

Como todos sabrán, un "sistema informático" posee componentes de Hardware, Firmware y Software, que deben ser coordinados, administrados y controlados para que funcionen adecuadamente y puedan cumplir con los propósitos de los usuarios. Es por este motivo, que se hace necesario contar con un programa que asuma la difícil tarea de administrar los recursos del computador (CPU, Memoria, Pantalla, Teclado, Mouse, Impresoras, Scanners, lectoras de discos flexibles, formas de guardar y recuperar la información, etc) y ese programa se denomina ***"Sistema Operativo".***

Con la intención de aclarar con puntualidad el rol que tiene un Sistema Operativo, vale realizar la comparación con el rol que cumple el gobierno de un País, Provincia o una Localidad. El gobierno como tal, es el administrador de los recursos y bienes limitados en cantidad que le son propios de ese ámbito; por lo que una función es destinar los ingresos provenientes de los impuestos a la educación, la salud y las obras públicas, de manera equitativa y eficiente hacia todos los sectores de la población.

Con los ***"Sistemas operativos",*** la situación es muy parecida, debido a que es él, el que tiene que administrar la CPU, la Memoria y los dispositivos de Entrada/Salida como la impresora por ejemplo, para que un programa pueda ser ejecutado, y luego que éste concluye con su tarea se pueda asignar a otro programa o usuario, y luego a otro. En todos los casos la complejidad propia del Hardware es manejada exclusivamente por el Sistema Operativo, brindando a los usuarios y programas un entorno mucho más "amigable" para ellos.

Por todo lo expuesto, estamos en condiciones de definir: **¿Qué es un Sistema Operativo desde el punto de vista de la Informática** ?

***"Es el programa del sistema que controla todos los recursos del Hardware del computador que son finitos en cantidad, proporcionando una 'interfaz' o 'máquina virtual' para que puedan escribirse y ejecutarse los programas de aplicación de los usuarios; es decir brindar soporte al Software de Aplicación"***

De la definición se deduce que para lograr una correcta interpretación y compresión de los conceptos que desarrollaremos, es de fundamental importancia manejar con claridad los conceptos de Hardware, desarrollados en los módulos de **"Arquitectura ".**

# *Evolución de los Sistemas Operativos*

Los Sistemas Operativos y la arquitectura de computadores han evolucionado de manera interrelacionada: para facilitar el uso de los computadores, se desarrollaron los sistemas operativos. Al construir y usar los sistemas operativos, se hace obvio que ciertos cambios en la arquitectura o tecnología aplicada los simplificaría. Por eso, es conveniente echar una mirada a la historia, porque cada generación sirvió de base a la siguiente, a punto tal que hay conceptos y políticas de administración de recursos computacionales que son utilizados por los sistemas operativos modernos y, por lo tanto, vigentes en la generación de computadoras actuales.

## *La primera generación (1945-1955): tubos de vacío[1]*

Los antecedentes del primer computador digital se remontan casi un siglo y medio atrás, y su diseñador fue el matemático Inglés Charles Babbage. Era un computador totalmente mecánico, que Babbage nunca pudo terminar de construir, principalmente debido a que la tecnología de la época no era capaz de producir las piezas con la precisión requerida.

Después de eso, poco se hizo hasta la Segunda Guerra Mundial: alrededor de 1940 se construyeron las primeras máquinas calculadoras usando TUBOS DE VACÍO. Estas máquinas de varias toneladas de peso eran diseñadas, construidas, programadas y operadas por el mismo grupo de personas. No había ni Lenguajes de Programación[2], ni Compiladores[3]; mucho menos Sistema Operativo. Cada programa

[1] Dispositivos electromecánicos que procesaban la Información.

[2] Un lenguaje de programación es un “Programa con el que se pueden hacer programas”. Esto es un software que contiene un repertorio de instrucciones (comandos que se pueden leer en nuestro idioma) que se le puede pedir a la máquina que las ejecute, por ejemplo: escribir en la pantalla , leer un archivo, imprimir, realizar cálculos, etc., y como resultado del trabajo se obtiene un archivo ejecutable, un programa.

[3] Algunos lenguajes de programación implementan en forma separada un software que se encarga de traducir las instrucciones inteligibles por nosotros a sus equivalentes en el lenguaje de la máquina. Este tipo de software se denomina “Compilador”.

se escribía en lenguaje de máquina, usando tableros con enchufes, cables e interruptores y tenía que manejar todo el sistema, desde sus funciones propias hasta las funciones del Sistema Operativo (lo que era factible gracias a que el programador era el mismo habría diseñado y construido la máquina).

En este caso, por ejemplo, un Sistema de Gestión del Censo de un país, además de tener que controlar todos sus propios problemas de cálculo y estadística, tenía que preocuparse por controlar los problemas inherentes a la máquina, cubriendo así las dos funciones del Sistema Operativo que no existían, por lo que cambiar de programa como hoy día alguien cambia de una aplicación a otra, significaba diseñar y programar la máquina de nuevo.

La operación con estas máquinas se efectuaba desde una consola consistente en unos indicadores luminosos, unos conmutadores, algún tipo de dispositivo de entrada de datos y una impresora. Si se detenía el programa por algún error, se indicaba esta condición mediante los indicadores luminosos. El operador podía examinar la memoria principal para determinar la causa del error. Si el programa continuaba hasta su culminación la salida aparecía en la impresora.

Estos primeros sistemas presentaban dos problemas principales:

- Planificación: La mayoría de las instalaciones empleaban un formulario de reserva de tiempo de máquina. Normalmente, un usuario podía reservar bloques de tiempo en múltiplos de media hora o aproximadamente. Un usuario podía tener reservada una hora y terminar a los 45 minutos, generando esto un desperdicio del computador. Por el contrario, un usuario podía tener dificultades, no terminar en el tiempo asignado y verse forzado a parar sin haber solucionado el problema.
- Tiempo de preparación: un programa sencillo se componía de varios pasos para su preparación, hasta que se lo podía ejecutar. Si se producía un error el infortunado operador tenía que volver al inicio del proceso de preparación perdiendo de esta manera mucho tiempo, hasta ejecutar nuevamente su programa.

Todas estas dificultades técnicas y operativas, hacían que los equipos no fueran muy populares y que su costo sea muy elevado

### *La segunda generación (1955-1965): transistores y procesamiento por lotes*

La introducción de los TRANSISTORES[4] permitió aumentar sustancialmente la confiabilidad de los computadores, lo que a su vez hizo factible construir máquinas comerciales. Por primera vez hubo una separación entre diseñadores, constructores, y programadores.

La aparición de los primeros compiladores (uno de los primeros que apareció fue el que trabajaba con un lenguaje llamado FORTRAN) facilitó la programación, a costa de hacer mucho más compleja la operación de los computadores. Por ejemplo, para probar un programa escrito en FORTRAN, el programador debía esperar su turno:

1. Cargaba el compilador de FORTRAN, típicamente desde una cinta magnética[5].
2. Ponía el montón de tarjetas perforadas correspondientes al programa escrito en FORTRAN y ejecutaba el compilador para que transformara las instrucciones. El compilador, en lugar de generar directamente lenguaje de máquina, generaba códigos intermedios denominados assembler, muy cercanos al lenguaje de la máquina, pero no legibles por ella.
3. Consecuentemente, cargaba el ensamblador para traducirlo a lenguaje de máquina. Este paso requería poner otra cinta con el ensamblador, en lugar del compilador FORTRAN. Luego ejecutar el ensamblador, para generar el programa ejecutable.
4. Por último ejecutaba el programa para probar si todos los cambios que había hecho funcionan.

[4] Dispositivos más rápidos, pequeños y confiables que los tubos de vacío. 200 transistores podían acomodarse en la misma cantidad de espacio que un tubo de vacío.

[5] Medio de almacenamiento de información similar a un cassette de música. Son de gran capacidad y se leen secuencialmente.

Si había errores en cualquiera de estos pasos, el programador debía corregir el programa y comenzar todo de nuevo. Obviamente, mientras el programador ponía cintas y tarjetas, y mientras se devanaba los sesos para descubrir por qué el programa no funcionaba, la CPU, de millones de dólares de costo, se mantenía completamente desocupada.

Más que rápido, se idearon mecanismos para mantener a la CPU siempre ocupada. El primero fue separar el rol de programador del rol de operador. (Figura 1) Un operador profesional demoraba menos en montar y desmontar cintas, y podía acumular lotes de trabajos con requerimientos similares: por ejemplo, si se acumulaba la compilación de varios programas FORTRAN, entonces el compilador de FORTRAN se carga una sola vez para todos los trabajos.

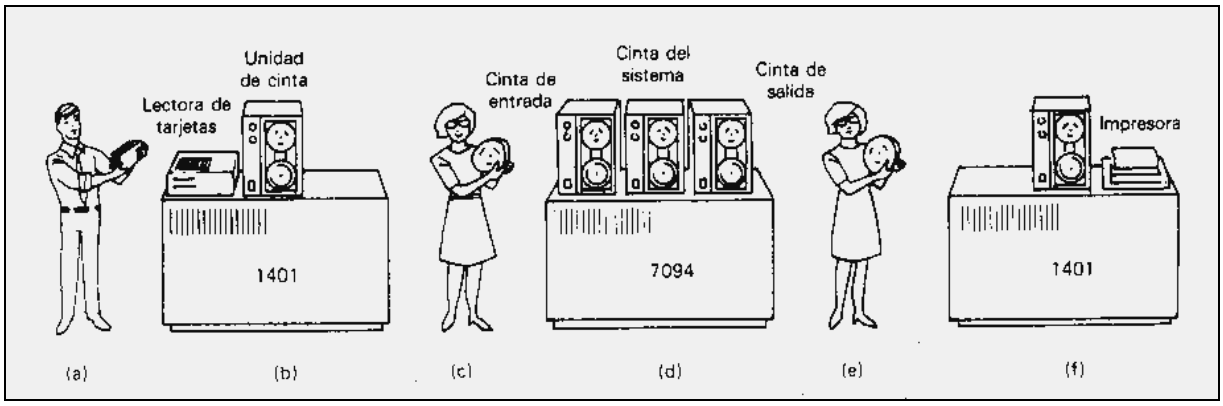



**Figura 1 Sistema por lotes**

(a) Los programadores traen las tarjetas.

(b) Un equipo especializado lee las tarjetas y las pasa a la Cinta

(c) Otro operador, lleva la cinta a un equipo más grande que la procesará

(d) El equipo realiza las operaciones de cálculo contenidas en la cinta.

(e) El operador lleva el resultado del proceso a un equipo especial para imprimir la salida

(f) El equipo transfiere los resultados contenidos en la cinta al papel para ser retirados por el programador

Aún así, en la transición de un trabajo a otro la CPU se mantenía desocupada. Aparecieron entonces los MONITORES RESIDENTES, que fueron los precursores de los sistemas operativos. Un monitor residente es un pequeño programa que está siempre en la memoria del computador, desde que éste se enciende. Cuando un programa termina, se devuelve el control al monitor residente, quien inmediatamente selecciona otro programa para ejecutar. Con esta técnica los programadores, en vez de informar al operador qué programa cargar, debían informárselo al monitor (mediante tarjetas de control especiales) (Figura 2).

Esto se conoce como PROCESAMIENTO POR LOTES: el programador deja su pila de tarjetas, y después vuelve a retirar la salida que se emite por la impresora (y que podría ser nada más que la notificación de que el programa tenía un error).


**Figura 2 Monitor Residente**

Con el uso de esta clase de Sistema operativo, los usuarios ya no tenían acceso directo a la máquina. En su lugar, el usuario debía entregar los trabajos en tarjetas o en cita al operador del computador, quien agrupaba secuencialmente los trabajos por lotes y ubicaba los lotes enteros en un dispositivo de entrada para su empleo por parte del monitor. Cada programa se construía de modo tal que volviera al monitor al

terminar su procesamiento y en ese momento el monitor comenzaba a cargar automáticamente el siguiente programa.

Para entender cómo funciona este esquema, se va a ver desde dos puntos de vista: el del monitor y el del procesador.

### ***El monitor visto desde le Monitor***

Desde el punto de vista del monitor, él es quien controla la secuencia de sucesos. Para que esto sea posible, gran parte del monitor debe estar siempre en memoria principal y disponible para su ejecución. Esta parte del monitor se conoce como monitor residente. El resto del monitor consta de utilidades y funciones comunes que se cargan al comienzo de cualquier trabajo que las necesite. El monitor lee los trabajos uno a uno del dispositivo de entrada (normalmente, un lector de tarjetas o una unidad de cinta magnética). A medida que lo lee, el trabajo actual se ubica en la zona del programa de usuario y el control pasa al trabajo. Cuando el trabajo termina, se devuelve el control al monitor, quien lee inmediatamente un nuevo trabajo. Los resultados de cada trabajo se imprimen y entregan al usuario.

### ***El monitor visto desde el Procesador***

Considérese ahora esta secuencia desde el punto de vista del procesador. En un cierto momento, el procesador o CPU estará ejecutando instrucciones de la zona de memoria principal que contiene al monitor. Estas instrucciones hacen que el trabajo siguiente sea leído en otra zona de la memoria principal. Una vez que el trabajo se ha leído, el procesador encuentra en el monitor una instrucción de desvío que ordena al procesador continuar la ejecución en el inicio del programa de usuario. El procesador ejecuta entonces las instrucciones del programa de usuario hasta que encuentre una condición de finalización o de error. Cualquiera de estos dos sucesos provocan que el procesador vaya por la instrucción siguiente del programa monitor. De este modo la frase "el control se le pasa al trabajo" quiere decir simplemente que el procesador pasa a leer y ejecutar instrucciones del programa de usuario, mientras que la frase "el control vuelve al monitor" quiere decir que el procesador pasa ahora a leer y ejecutar las instrucciones del programa monitor.

Debe quedar claro que es el monitor el que gestiona el problema de la planificación. Se pone en cola un lote de trabajo y éstos son ejecutados tan rápido como es posible, sin que haya tiempo alguno de desocupación.

### *La tercera generación (1965-1980): circuitos integrados y multiprogramación*

El procesamiento por lotes evita que la CPU tenga que esperar tareas ejecutadas por lentos seres humanos. Pero ahora el cuello de botella se trasladó a los dispositivos mecánicos (impresoras, lectoras de tarjetas y de cinta), por naturaleza más lentos que las CPUs electrónicas. Para resolver esto, aparece, dentro de la tercera generación de computadores, la **MULTIPROGRAMACIÓN**: varios trabajos se mantienen permanentemente en memoria (Figura 3); cuando uno de ellos tiene que esperar que una operación (como grabar un registro en cinta) se complete, la CPU continúa con la ejecución de otro trabajo. Si se mantiene un número suficiente de trabajos en la memoria, entonces la CPU puede estar siempre ocupada. (Figura 3)

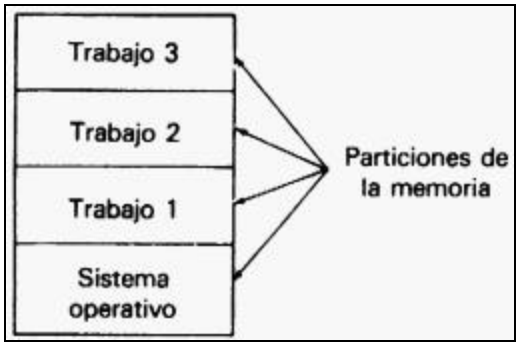



**Figura 3 Multiprogramación con tres trabajos en memoria junto al Sistema Operativo.**

Veamos un ejemplo, la Tabla 1 detalla un cálculo representativo. Los números corresponden a un programa que procesa un archivo de registros (por ahora consideremos un registro como un renglón de una lista, mas adelante se verá

exhaustivamente las características de los registros) y ejecuta, en promedio, 100 instrucciones de máquina o 100 ciclos de CPU por cada registro.

En este ejemplo, el computador gasta más del 96% del tiempo esperando a que los dispositivos de E/S terminen de transferir sus datos. El procesador consume parte del tiempo ejecutando instrucciones de cálculos, hasta que encuentra una instrucción de Entrada Salida (E/S) por ejemplo leer en el disco, esperar algo desde el teclado, imprimir etc.. Entonces debe esperar a que concluya la instrucción de E/S antes de continuar ejecutando las siguientes instrucciones, ya que los dispositivos de entrada salida se manejan independientemente, dejando libre al procesador mientras que trabajan con estos dispositivos de E/S.

| | |
|---|---|
| Leer un registro | 0,0015 segundos |
| Ejecutar 100 instrucciones | 0,0001 segundos |
| Escribir un registro | 0,0015 segundos |
| TOTAL | 0,0031 segundos |
| Porcentaje de Utilización de la CPU | 0,0001/ 0,0031 = 0,032  3,2% |

**Tabla 1 Requerimientos de programa**

Esta ineficiencia no es necesaria. Se sabe que hay memoria suficiente para almacenar el Sistema Operativo (el monitor residente) y un programa de usuario. Supóngase que hay espacio suficiente para el Sistema Operativo y dos programas usuarios. Ahora, cuando un trabajo necesite esperar una E/S, el procesador o CPU puede cambiar al otro trabajo, que probablemente no estará esperando a la E/S. Además, se podría ampliar la memoria para almacenar tres, cuatro o más programas y conmutar o cambiar entre todos ellos. Este proceso es lo que conocemos como MULTIPROGRAMACIÓN. Este es el punto central de los Sistemas Operativos modernos.

Para ilustrar aún más este beneficio proporcionado por la multiprogramación, consideremos el siguiente ejemplo. Sea un computador con 256K de memoria disponible (no utilizadas por el Sistema Operativo), un disco, una terminal y una impresora. Tres programas, TRABAJO1, TRABAJO2 y TRABAJO3, son enviados para su ejecución al mismo tiempo, con los atributos o características que se enumeran en la Tabla 2. Por lo que se ve, el TRABAJO1 ocupará mucha CPU para

realizar sus cálculos, mientras que el TRABAJO2 tendrá tiempos donde hará esperar a la CPU hasta que se ingrese información desde el teclado para seguir trabajando, y el TRABAJO3 además del teclado la hará esperar mientras utiliza el disco.

| | Trabajo 1 | **TRABAJO 2** | **TRABAJO 3** |
|---|---|---|---|
| Tipo de Trabajo | Cálculo intensivo | E/S Intensiva | E/S Intensiva |
| Duración | 5 min. | 15 min. | 10 min. |
| Memoria exigida | 50 K | 100 K | 80 K |
| ¿Necesita Disco? | No | No | Sí |
| ¿Necesita Terminal? | No | Sí | No |
| ¿Necesita Impresora? | No | No | Sí |

**Tabla 2 Atributos de ejemplo de la ejecución de un programa**

En un sistema sencillo por **lotes**, estos trabajos llegan todos juntos y serían ejecutados en secuencia. Así pues, el TRABAJO1 termina en 5 minutos. El TRABAJO2 debe esperar a que transcurran esos 5 minutos y terminar 15 minutos después. El TRABAJO3 comienza después de los 20 minutos para terminar 30 minutos después del momento en que fue lanzado. La utilización promedio de los recursos, la productividad y los tiempos de respuesta se ilustran en la columna de monoprogramación de la

Tabla **3**. Es evidente que hay poco aprovechamiento de todos los recursos cuando se promedian los tiempos de uso en el período exigido de 30 minutos.

Supóngase ahora que los trabajos se ejecutan concurrentemente en un Sistema Operativo con monoprogramación. Como hay poca contención de recursos entre los trabajos, cada uno de los tres puede ejecutarse en un tiempo cercano al mínimo mientras coexiste con los otros en el computador (suponiendo que a TRABAJO2 y TRABAJO3 se les adjudica tiempo suficiente de procesador para mantener activas sus operaciones de E/S). El TRABAJO1 requerirá 5 minutos para terminar, pero al finalizar este tiempo, el TRABAJO2 estará terminado en una tercera parte y el TRABAJO3 estará a la mitad. Los tres trabajos habrán terminado dentro

de 15 minutos. La mejora es evidente cuando se examina la columna de multiprogramación[6] de la tabla 2.

| | MONOPROGRAMACIÓN | MULTIPROGRAMACIÓN |
|---|---|---|
| Uso del procesador | 17% | 33% |
| Uso de la memoria | 30% | 67% |
| Uso del disco | 33% | 67% |
| Uso de la impresora | 33% | 67% |
| Tiempo transcurrido | 30 min | 15 min |
| Tasa de productividad | 6 trabajos/hora | 12 trabajos/hora |
| Tiempo medio de respuesta | 18 min | 10 min |

**Tabla 3 Efectos de la multiprogramación sobre la utilización de recursos.**

Al igual que un sistema sencillo por lotes, un sistema por lotes con multiprogramación tiene que depender de ciertas características del hardware del computador. El procesador tiene que poder enviar una orden de E/S para un trabajo y continuar con la ejecución de otro, mientras la E/S es efectuada por otro dispositivo conocido como controlador del dispositivo. Cuando termina la operación de E/S, el procesador es interrumpido y el control pasa al Sistema operativo. El Sistema Operativo le pasa entonces el control a otro trabajo.

Los Sistemas Operativos con multiprogramación son bastante más sofisticados en comparación con los sistemas de monoprogramación o de un solo programa. Para tener varios trabajos listos para ejecutar, éstos deben mantenerse en la memoria principal, lo que requiere cierto tipo de gestión de memoria. Además, si hay varios trabajos listos para ejecutarse, el procesador debe decidir cuál de ellos va a ejecutar, lo que requiere un algoritmo de planificación.

Pero el sistema sigue siendo esencialmente un sistema de procesamiento por lotes; los programadores no interactúan en línea con el computador, los tiempos de

[6] Multiprogramación es la capacidad de un sistema de atender a más de un programa simultáneamente. Si bien no los atiende al mismo tiempo, alterna entre ellos atendiendo una fracción de tiempo a cada uno, de manera que en conjunto, los usuarios no perciben el engaño y piensan que el sistema está atendiendo únicamente a uno mismo.

respuesta desde que se deja un trabajo para ejecución hasta conocer el resultado siguen siendo grandes. De ahí nace el concepto de **TIEMPO COMPARTIDO** que es una variante de la multiprogramación en la cual una CPU atiende simultáneamente los requerimientos de varios usuarios conectados en línea a través de terminales. Ya que los usuarios humanos demoran bastante entre la emisión de un comando y otro, una sola CPU es capaz de atender, literalmente, a cientos de ellos simultáneamente (bueno, en realidad, uno después de otro, pero los usuarios tienen la ilusión de la simultaneidad). Por otra parte, cuando no hay ningún comando que ejecutar proveniente de un usuario interactivo, la CPU puede cambiar a algún trabajo por lote.

Veamos como es esto: La técnica de tiempo compartido, refleja el hecho de que el tiempo del procesador es compartido entre los diversos usuarios. Básicamente consiste en tener a varios usuarios utilizando simultáneamente el sistema mediante terminales, mientras que el Sistema Operativo intercala la ejecución de cada programa de usuario en ráfagas cortas de cómputo o cuantos (quantum). De esta manera, si hay 10 usuarios que solicitan servicio a la vez, cada usuario sólo dispondrá, en promedio, de 1/10 de la atención efectiva del computador, sin contar con la sobrecarga del Sistema Operativo. Sin embargo, dado el tiempo de reacción relativamente lento que tiene el ser humano, el tiempo de respuesta en un sistema correctamente diseñado debería ser comparable al que se obtendrá si cada uno de los 10 usuarios tuviera su propia máquina.

Tanto la multiprogramación por lotes como el tiempo compartido utilizan multiprogramación. Las diferencias básicas se enumeran en la Tabla 4.

| | **MULTIPROGRAMAICÓN POR LOTES** | **TIEMPO COMPARTIDO** |
|---|---|---|
| Objetivo Principal | Maximizar la utilización del procesador | Minimizar el tiempo de respuesta |
| Origen de las instrucciones al Sistema Operativo | Instrucciones de un lenguaje de control de trabajos incluidas junto con el trabajo | Ordenes dadas en el Terminal por el usuario |

Tabla 4 Multiprogramación por Lotes frente a Tiempo compartido

Uno de los primeros sistemas de tiempo compartido que se desarrollaron fue el Sistema Compatible de Tiempo Compartido (CTSS, Compatible Time-Sharing System), desarrollado en el MIT

Comparado con sistemas posteriores, el CTSS era bastante primitivo y su funcionamiento básico es fácil de explicar. El sistema se ejecutaba en una máquina con una memoria de 32K, con un monitor residente que consumía 5K del total. Cuando había que asignar el control a un usuario interactivo, el programa del usuario y los datos eran cargados en las restantes 27K de la memoria principal. Un reloj del sistema generaba interrupciones a razón de aproximadamente una cada 0,2 segundos. En cada interrupción de reloj, el Sistema Operativo se adueñaba del control y le podía asignar el procesador a otro usuario. De esta manera, a intervalos regulares, el usuario en curso era expulsado y se cargaba otro usuario en su lugar. Para conservar el estado del usuario anterior, para su reanudación posterior, los programas del usuario anterior y sus datos eran escritos en el disco antes de leer los programas del nuevo usuario y sus datos. En consecuencia, el espacio de memoria del usuario anterior debía ser restaurado cuando le llegara de nuevo su turno.

Para minimizar el tráfico en el disco, la memoria del usuario se escribía a disco sólo cuando el nuevo programa a cargar podía sobreescribirla.

El tiempo compartido y la multiprogramación plantean una multitud de problemas nuevos para el Sistema Operativo. Si hay varios trabajos en memoria, entonces deben protegerse de injerencias unos de otros, como, por ejemplo, que uno modifique los datos de otro. Con varios usuarios interactivos, el sistema de archivos debe protegerse de forma que sólo los usuarios autorizados puedan tener acceso a un archivo en particular. La contención de recursos tales como la impresora y los dispositivos de almacenamiento masivo debe estar controlada.

Hasta aquí estuvimos enfocando dos tipos de programas esencialmente distintos en su forma de relacionarse con el usuario. Por un lado tenemos los Trabajos por Lotes en los que se prepara el programa y mientras éste se ejecuta, no se solicita nada al usuario, el programa tiene incorporado en sí mismo toda la información que requiere para lograr su objetivo; simplemente se lo ejecuta y se espera el resultado final. Mientras ello ocurre, no existe ninguna pregunta por

pantalla o cosa parecida; tampoco tenemos la posibilidad de seleccionar ninguna variante para la ejecución del programa.

Por otro lado están los sistemas interactivos, que contrariamente a los anteriores, están pensados para relacionarse asiduamente con el usuario, realizando cuanta pregunta y solicitud sea necesaria para poder trabajar y permitiendo que el usuario establezca durante la ejecución del programa todas las selecciones que éste le permita. Estos requieren que el usuario esté constantemente controlando qué es lo que el programa necesita o que es lo que está preguntado. Hoy día la mayoría de los programas son de estas características

### *La cuarta generación (1980-): computadores personales*

Con el advenimiento de la INTEGRACIÓN A GRAN ESCALA, que permitió concentrar en un solo chip miles, y luego millones de transistores, nació la era de la computación personal. Los conceptos utilizados para fabricar los Sistemas Operativos de la tercera generación de computadores siguen siendo, en general, adecuados para la cuarta generación. Algunas diferencias destacables:

- Dado los decrecientes costos de las CPUs, ya no es nada grave que un procesador esté desocupado.
- La creciente capacidad de las CPUs permite el desarrollo de interfaces gráficas; buena parte del código de los Sistemas Operativos de hoy es para manejar la interfaz con el usuario.

Las redes de computadores y sistemas distribuidos abren nuevas posibilidades e imponen nuevas obligaciones a los Sistemas Operativos.

## *Estructuras de Sistemas Operativos*

A continuación se presentan algunos diseños de Sistemas Operativos que se han implementado en la práctica.

Las distintas estructuras presentan enfoques posibles para establecer el funcionamiento del Sistema operativo. Para ello en todos los casos, el esquema general es el mismo: los programas que utiliza el usuario están en un lado y los

recursos del sistema (todo lo que la máquina dispone para trabajar) están del otro. Los programas utilizan estos recursos y es el Sistema Operativo es el que va a trabajar en el medio (de intermediario) para coordinar las tareas tal y como se vio en la descripción de los objetivos de todo Sistema Operativo.

Es así que cada estructura de las que vamos a analizar tratará de resolver este trabajo de una manera distinta teniendo por ello cada uno sus pro y contras.

### *Sistemas Monolíticos*

Esta estructura es muy utilizada en los Sistemas Operativos y consiste en una estructura muy rudimentaria. Básicamente todas las funcionalidades del Sistema Operativo son proporcionadas a través de un conjunto de procedimientos, los que pueden ser llamados por otros cada vez que se requiera. A este conjunto de procedimientos se le llama, como ya se han mencionado anteriormente, ***Llamadas al Sistema.***

Entonces, por ejemplo, cuando un programa de usuario requiera de algún recurso del sistema como por ejemplo leer del disco, realizará una “Llamada al sistema”; le solicitará al Sistema Operativo que le resuelva ese problema. Para ello, el Sistema operativo tiene una lista de Llamadas declaradas, que cada programa de usuario puede utilizar. Por cada llamada, el Sistema Operativo tiene lo que se conoce como Procedimiento asociado, que no es otra cosa que los pasos que hay que seguir para efectuar la llamada y solucionar el problema presentado. Entonces una vez que el Programa de usuario efectuó la llamada se dice que el sistema pasa a Modo Núcleo, donde el que toma el control es el Sistema Operativo y lo primero que realiza es verificar que sea una llamada válida, de no serlo genera un error y el programa se planta. Si la llamada realizada era válida; el Sistema Operativo busca en una tabla donde tiene todas las llamadas almacenadas y localiza cuál es el procedimiento asociado a la misma. Una vez localizado el procedimiento lo ejecuta de manera de conseguir la solución del problema planteado. Una vez finalizado el procedimiento se devuelve el resultado al programa de usuario y éste toma nuevamente el control cambiando el modo de trabajar a Modo Usuario. Esta manera de trabajar está ejemplificada en la Figura 4

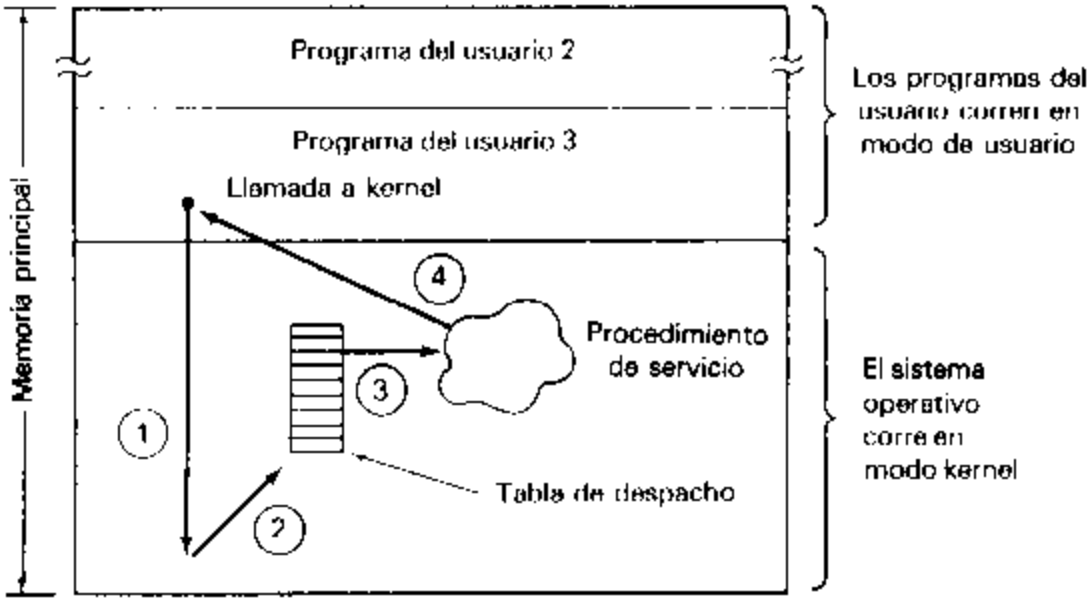



Figura 4 Sistema de Estructura Monolítica

Puede ser también que los procedimientos asociados tengan que requerir colaboración entre ellos, por ejemplo dada una llamada **x** puede ser que el procedimiento asociado a ella, cuando se está ejecutando, convoque a otro procedimiento para completar su tarea, por ello se dice que en realidad no existe una estructura formal, y se suele conocer a esta Estructura como “El gran embrollo” ya que el sistema tiene todos los procedimientos que resuelven los problemas que se pueden presentar pero éstos están a disposición para convocarlos cuando sea necesario y sin necesidad de seguir algún determinado orden.

Lo que sí esta bien establecido son los dos modos de trabajar: por un lado está el modo usuario, donde el que controla la situación es el programa del usuario y como tal no tiene acceso a todos los recursos; está limitado a aquella porción que le fue asignada cuando se cargó. Cuando necesita algún recurso adicional utiliza llamadas al sistema es entonces cuando se cambia al Modo Núcleo y toma control el Sistema Operativo, quien si tiene acceso a todos los recursos tratará de solucionar el problema planteado. Una vez que lo logra, o se genera un error el control vuelve al programa de usuario cambiando nuevamente a Modo Usuario y así sucesivamente.

En este tipo de estructura se pueden distinguir 3 niveles. En la Figura 5 se distinguen estos niveles.

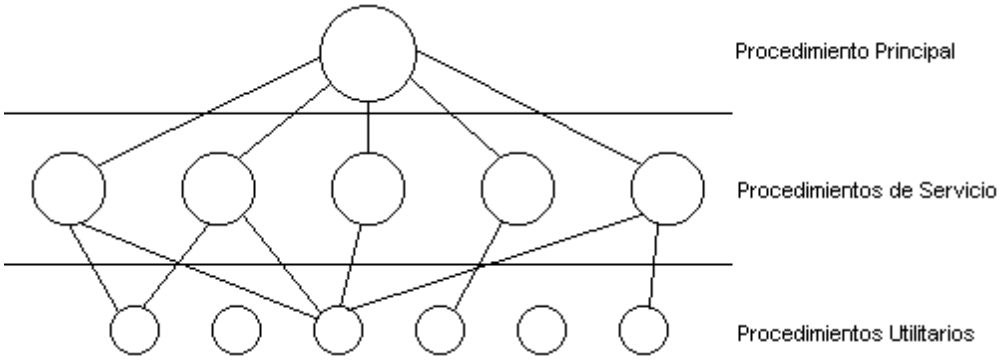



**Figura 5 Relación entre los procedimientos en un Sistema de estructura Monolítica**

- Nivel Superior: Programas de aplicación que llaman los procedimientos de servicio
- Nivel Intermedio: Conjunto de procedimientos de servicio que realizan los llamados a sistema
- Procedimientos utilitarios: Ayudan a los procediminetos de servicio

### *Sistemas con Capas*

El enfoque por capas consiste en dividir el Sistema Operativo en varias capas, cada una de las capas se construye sobre una capa inferior. La capa 0 (la más baja) es el hardware y la capa N (la más alta) es la interfaz con el usuario. (Figura 6)

De esta manera el Sistema operativo tendrá una x cantidad de “Capas” o “Niveles” estos estarán organizados de manera tal que el nivel de abajo es el que se encargará de realizar el manejo directo con el Hardware. Esto es manipular los dispositivos electrónicos para realizar las tareas solicitadas por los usuarios, y lo hará como un especialista. Y la capa de arriba estará en contacto con el usuario, se encargará de realizar todo lo necesario como para que el usuario pueda utilizar el Sistema (comandos, botones, ventanas etc.) Estas son las dos puntas de nuestra estructura; por un lado el Hardware y por otro lado, el Usurario.

En medio de estas capas periféricas van a existir tantas capas como sea necesario para ir solucionando todos los demás problemas, como por ejemplo, sobre la capa 0 existirá un nivel que maneje la memoria. Esta distribuirá espacio para los procesos contenidos en la memoria central. Si no alcanza la memoria para todos los procesos se encargará de pasar los procesos que no se están utilizando al disco para liberar la memoria principal; controlará que cada proceso acceda al área que se le asignó y no sobreescriba los demás procesos cargados, etc.. Sobre esta capa, los procesos no tienen que preocuparse si estaban en la memoria o en el disco; o que no les alcance la memoria para trabajar, o bien, de cualquier otro problema de memoria que pueda presentarse. El software de la capa 1 se hará cargo de solucionarlo.

La capa 2 manejará la comunicación entre cada proceso y la consola del operador (la máquina donde está trabajando). Si el proceso tenía que avisarle algo al operador este nivel del Sistema Operativo se encargaba de realizarlo. Sobre esta capa, cada proceso tenía en forma efectiva su consola de operador simplemente lo solicita y la capa 2 se encarga de ver cómo realizarlo.

La capa 3 se hará cargo de manejar los dispositivos de E/S y de separar la información en flujo que entra y sale de ellos. Sobre el estrato 3, cada proceso podrá trabajar solicitando simplemente escribir en disco o leer del teclado, sin tener que preocuparse por saber cómo se realiza esta tarea porque para ello está la capa 3.

En el nivel 4 se encuentran los programas de los usuarios. Estos no tienen que preocuparse por el manejo de los procesos, ni de la memoria, ni de la consola o de la E/S ya que todos estos problemas están solucionados en las capas inferiores. Sobre esta capa estará, entonces, el operador del sistema, el usuario, en el nivel 5 haciendo uso de todos los servicios proporcionados por el Sistema Operativo.

Visto de esta manera, cada capa se preocupa de realizar bien su tarea, puede pedir los servicios de las capas inferiores y atender las solicitudes de las capas superiores.

| | |
|---|---|
| **5** | Operador del Sistema |
| **4** | Programas del Usuario |
| **3** | Administración de Entrada / Salida |
| **2** | Comunicación entre el Operador y el Proceso |
| **1** | Administración de la memoria y el Disco |
| **0** | Distribución del Procesador y multiprogramación |

Figura 6 Sistema Operativo THE, diseñado con estructura en Capas

La característica más importante de este tipo de estructura es la modalidad. Las capas se seleccionan de manera que cada una utilice funciones y servicios sólo de las capas inferiores. Este enfoque permite simplificar la verificación y la depuración del sistema. Así, por ejemplo si durante la depuración de un nivel en particular se encuentra un error, se sabe que el error se produjo en este nivel, pues los niveles inferiores a esta capa ya se depuraron. La característica de esta modalidad proporciona una ventaja en el diseño e implementación de un sistema.

En cuanto a la implementación de un sistema por capas, cada capa utiliza solamente las operaciones que proporciona la capa inferior. Una capa no necesita saber cómo se implementan las capas inferiores sino, sólo las operaciones que ésta le ofrece.

El primer sistema creado mediante este enfoque se llamó THE (Technische Hogeschool Eindhoven), el cual se definió en seis capas. También se implementaron algunas versiones de Unix mediante este enfoque. También la primera versión de OS/2 (cuya idea era mejorar características de MS-DOS) se utilizó este esquema para el Sistema Operativo.

El enfoque por capas también tiene sus desventajas, la más importante se centra en la dificultad de definir los niveles, debido a que cada capa utiliza los servicios de la capa inferior el diseño debe realizarse con mucho cuidado. Existen operaciones que debe realizar el Sistema Operativo, pero según algunos, sería mejor que se ubicaran en una capa superior, y según otros, sería bueno que

estuvieran en una capa inferior. Estas dificultades han producido un retroceso en la aplicación de este enfoque en el diseño e implementación de sistemas operativos.

### *Máquinas virtuales*

Los primeros Sistemas Operativos estaban orientados al procesamiento por lotes, sin embargo, pronto las necesidades de los usuarios fueron exigiendo la utilización de tiempo compartido. Esta característica proporciona la multiprogramación y la máquina extendida con una buena interfaz.

Este tipo de sistema consiste en que se ejecuta el núcleo del sistema, que se le llama monitor de la máquina virtual, sobre el hardware y realiza la multiprogramación, proporcionando al nivel superior varias máquinas virtuales. Estas máquinas virtuales son simulaciones del hardware, con su modo núcleo/usuario, Entrada/Salida, y todos los demás aspectos físicos del hardware.

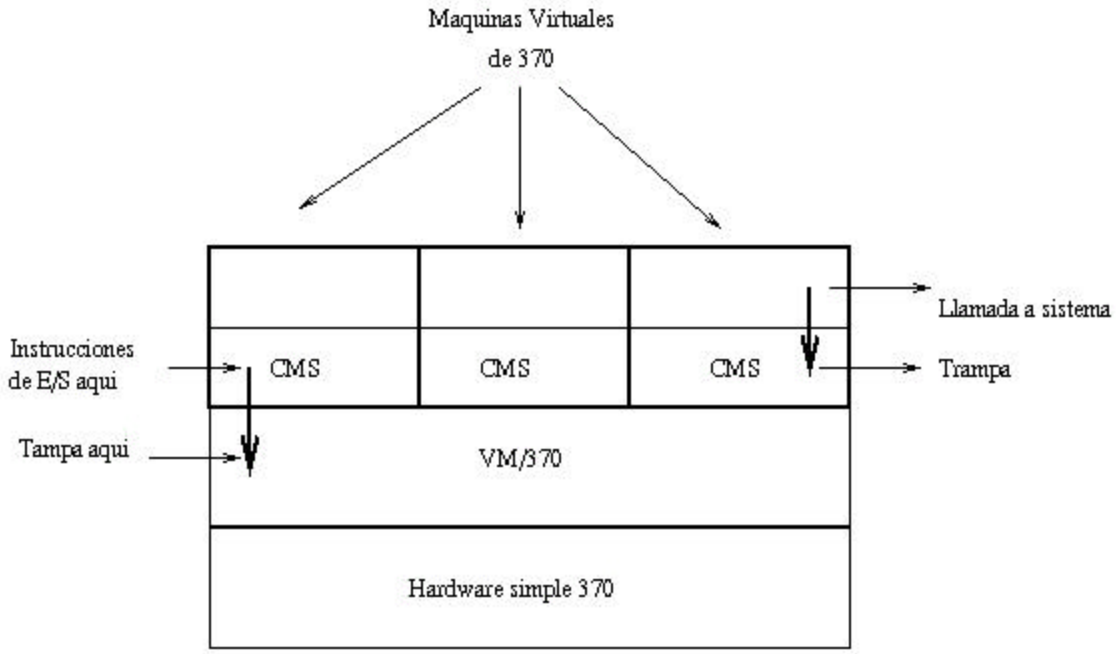
Dado que las máquinas virtuales son copias del hardware, sobre ellas es posible montar cualquier Sistema Operativo que se ejecute sobre el hardware. A diferencia de los sistemas de tiempo compartido, en un sistema de tipo Máquina Virtual, lo que el usuario ve en su terminal, no es un reflejo del equipo central, sino que tiene una máquina propia. Cada usuario ve como una máquina propia, sobre la cual puede cargar cualquier otro sistema a su gusto y trabajar tranquilamente. Estas "máquinas personalizadas" en realidad son siempre las mismas: el equipo central. Lo que pasa es que el Sistema Operativo de tipo Máquina Virtual realiza la simulación y permite que la visión de cada usuario sea así. De ahí que se llame "Máquina Virtual" porque en realidad, la máquina que ven los usuarios en sus terminales –que parece una para ellos solos, sobre la cual cargan cualquier otro Sistema Operativo– no existe; es una simulación, pero funciona.

Las llamadas a sistema que se ejecuten sobre cada uno de los Sistemas Operativos son atendidas por los correspondientes sistemas montados sobre la máquina virtual correspondiente. Las instrucciones que deba ejecutar el Sistema Operativo particular serán atrapadas (en una "Trampa") por el monitor de la máquina virtual y ejecutadas realmente en el hardware.

Uno de los sistemas creados bajo este enfoque es el VM/370, construido para máquinas IBMs. En los tiempos que fue creado se solía utilizar, sobre las máquinas virtuales, un Sistema Operativo para procesos por lotes y un Sistema Operativo de tiempo compartido llamado CMS (Conversational Monitor System) .

Este enfoque de estructura resulta ser bastante complejo y no maximiza el rendimiento, pues hay una simulación de hardware que razonablemente incurre en muchos costos de cálculo y tiempo de procesamiento.

A continuación se presenta una figura que esquematiza estos conceptos.




### *Sistemas tipo Cliente Servidor*

La tendencia en el desarrollo de los Sistemas Operativos apunta a mover la mayor cantidad posible del código del Sistema Operativo a las capas superiores, de manera de minimizar el tamaño del núcleo del sistema. A este tipo de diseño se le conoce en la literatura como sistemas cliente/servidor, aunque también se le encuentra con la denominación de microkernel. El diseño consiste en implementar la mayor cantidad de funciones en modo usuario, como procesos clientes y procesos servidores. Entonces, todos los recursos de la máquina se implementan como

servidores, por ejemplo: Servidor de Disco, Servidor de Impresión, Servidor de Comunicaciones, etc. y los procesos que se ejecuten serán Clientes.

Los procesos clientes solicitan los servicios a los procesos servidores. Bajo este enfoque el núcleo del sistema se encarga sólo de controlar la comunicación entre los clientes y los servidores.

Dado que los clientes y los servidores se ejecutan en modo usuario, si uno de ellos llega a fallar no resulta perjudicado el funcionamiento de toda la máquina, simplemente se inhabilita el servidor o cliente afectado y el resto del sistema sigue funcionando.

En general, este enfoque apunta a diseñar los mecanismos en el núcleo del Sistema Operativo y las políticas en modo usuario, tratando de minimizar el tamaño del núcleo. Cuando se habla de mecanismos se está refiriendo a cómo realizar las acciones, en cambio cuando se habla de políticas se refiere a qué hacer. La separación de estos dos conceptos es muy importante para conseguir flexibilidad, pues es probable que las políticas cambien con mayor frecuencia, y se supone que los cambios en las políticas sólo implicarían la redefinición de ciertos parámetros del sistema. Sin embargo, los cambios en los mecanismos se supone no ocurren con mucha frecuencia, por lo cual sería deseable que fuesen generales.

La Figura 7 muestra una esquematización de este sistema.

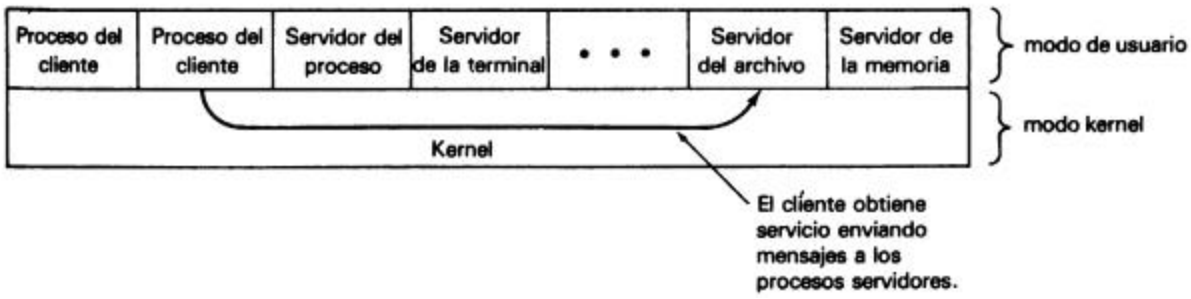



Figura 7 Sistema con estructura Cliente / Servidor

# *Ejemplos de Sistemas Operativos*

En la actualidad existe una gran cantidad de Sistemas Operativos. Analizaremos un par de ellos para ilustrar varios de los conceptos y como estos se pueden materializar en un sistema real.

Como ejemplo, tomaremos, principalmente dos sistemas, uno es Unix y el otro es MS-DOS. Ambos sistemas se encuentran en operación en nuestro departamento y además son sistemas muy utilizados, al menos Unix, MS-DOS se supone está perdiendo adeptos últimamente con las bondades de Windows en sus distintas versiones.

## *Sistema Operativo Unix*

Unix es un Sistema Operativo construido por Ken Thompson, aproximadamente en 1970, en los Laboratorios Bell, de AT&T, para ser utilizado en una máquina PDP-7. Posteriormente, Thomson se unió a Dennis Ritchie para producir el Unix, que usamos hoy día, escrito en lengua C

El diseño de Unix está basado en muchas características de MULTICS, como la organización básica del sistema de archivos, la idea del intérprete de comandos como proceso de modo usuario, etc.

A medida que el grupo investigador fue mejorando el sistema fueron apareciendo versiones. A partir de la Sexta versión Berkeley, otras instituciones se interesaron en tomar como base esos desarrollos y continuar por su cuenta.

A partir del momento en que la Universidad de Berkeley comenzó a trabajar con Unix, se distinguen dos grandes productores de Unix., por un lado, AT&T y por otro, Berkeley.

### ***Características de Unix***

Unix es un Sistema Operativo multiusuario, interactivo de tiempo compartido. La mayoría de los algoritmos que rigen el sistema se eligieron por su sencillez. La idea inicial fue que el núcleo y las bibliotecas proporcionaran un pequeño conjunto

de recursos con el poder suficiente para construir un sistema más complejo. Desde este punto de vista el sistema fue diseñado elegantemente.

La interfaz con el usuario es sencilla, a través del intérprete de comandos (Shell), el cual hace un uso intensivo de llamados a sistema, llama a los utilitarios, permite comunicar programas, permite manipular procesos que se ejecuten en forma directa (foreground) con participación del usuario y a la vista en la pantalla o bien indirecta, sin participación del usuario y sin que éste lo vea en la pantalla (background) y también permite que sea programada.

El sistema de archivos es tipo árbol con múltiples niveles que permite a los usuarios crear sus propios directorios. Además los usuarios pueden proteger los archivos y directorios en forma flexible.

Unix permite la ejecución de múltiples procesos, con la posibilidad de que los procesos creen otros procesos, modifiquen su acción o los maten; siendo el motor de estas acciones la adecuada y sencilla planificación de la CPU que implementa (ya lo veremos más adelante).

### *Sistema Operativo MS-DOS*

Ahora veremos a MS-DOS, a diferencia de Unix que se puede ejecutar en una gran cantidad de arquitecturas de computadores, MS-DOS sólo se puede ejecutar en la familia Intel 80x86. Sin embargo, es un sistema muy utilizado, pero se ve venir una disminución en su utilización con la aparición de Windows95.

Alrededor de 1975, MS-DOS surgió cuando Bill Gates escribió un pequeño sistema para la máquina Altair que tenía un procesador que era un Intel 8080. Posteriormente muchas empresas empezaron a fabricar computadores con el procesador 8080, de esta manera se fue popularizando este chip. El Sistema Operativo que se construyó para manejar estos computadores se llamó CP/M, producido por Digital Research.

Posteriormente, IBM pidió a la empresa Microsoft que le construyera un Sistema Operativo para sus computadores personales. Aquí surge la idea de confeccionar uno similar al CP/M, y en este sentido apareció MS-DOS. Enseguida

surge una seguidilla de versiones de MS-DOS, desde la versión 1 a la actual que es la versión 6.22.

MS-DOS es un sistema monousuario, sólo permite operar en el sistema a un usuario; y monotarea, pues sólo permite un sólo programa en ejecución.

Con respecto al intérprete de comandos, podemos decir en primer lugar que es bastante limitado, pues no es tan amigable y no es posible su programación en buena forma. La única posibilidad de programación que ofrece es a través de los archivos .bat. Además no permite la ejecución de procesos en background.

El sistema de archivos también es tipo árbol, pero no es posible proteger los archivos y directorios. Aunque existen algunos mecanismos rudimentarios, como ocultar los archivos, dejarlos de lectura solamente, etc.; estos mecanismos no son nada de seguros porque si entra otro usuario y desea desproteger los archivos lo hace sin problemas. Otro aspecto relativo al sistema de archivos es que DOS no hace diferencias de las letras minúsculas de las mayúsculas, pero sí está limitado el largo que pueden tomar éstos.

Otro aspecto interesante de mencionar respecto a MS-DOS, es que es un sistema desprotegido de los usuarios. Por ejemplo, si un usuario realiza algún programa que maneje el reloj, por ejemplo, y no lo hace bien el sistema puede verse perjudicado de ahí que tenga que sufrir los ataques de "Virus" que no son otra cosa que programas hechos por usuarios del sistema muy experimentados, que lo burlan y realizan estragos en él. En Unix este flagelo no existe ya que la estructura de Unix es muy cerrada.

# Evaluación Sistemas Operativos módulo I

Alumno: .....

**Desarrollar**

1. ¿Cuál es el objetivo de un Sistema Operativo?
2. Dar una analogía del papel de un sistema operativo dentro de un sistema informático, distinta a la del gobierno de un país, planteado en el módulo.
3. Exponer las características distintivas de cada generación de computadoras.
4. ¿En qué consiste la multiprogramación?.
5. ¿Qué es es tiempo compartido?
6. Explicar el funcionamiento de cada una de las cuatro estructuras de los sistemas operativos estudiadas.
7. Graficar cada estructura de sistemas operativos. (no se permite realizar fotocopias del material ni escanearlo).
8. Marcar 3 características distintivas entre los sistemas operativos Unix y Dos