SOFTWARE DE BASE

A. TRADUCTORES

I. Traductores de bajo nivel

1.Ensambladores

- ***Pasadas del texto fuente***

Tratamiento de operandos

Las instrucciones de maquina operan sobre los datos, las categorías mas importantes de datos son:

Numéricos

- Enteros o de punto fijo.
- Punto flotante.
- BCD (Decimal codificado ó binario).- Los datos BCD comprimidos presentan dos dígitos por byte y los no comprimidos presentan un dígito por byte.

Caracteres

Unidades de 8 bits que normalmente contienen representaciones ASCII de símbolo.

Datos lógicos

- Byte con o sin signo (8 bits).- El bit 7 es el bit de signo de un byte sin signo.
- Palabra con o sin signo (word-16 bits).- El bit 15 es el bit de signo de una word sin signo
- Doble palabra con o sin signo (dword-32 bits).- El bit 31 es el bit de signo de una dword sin signo
- Cuádruple palabra con o sin signo (qword-64 bits).- El bit 63 es el bit de signo de una qword sin signo

Direcciones

Son una forma de dato. En muchos casos, algún calculo se debe ejecutar sobre la referencia de un operando en una instrucción para determinar la dirección de la memoria principal o virtual. En este contexto, las direcciones se pueden considerar como enteros sin signo.

- Desplazamiento.- Son cantidades de 16 o 32 bits que contienen la distancia de la dirección base o la dirección referenciada.
- Punteros.- Consistentes en selectores de segmento de 16 bits y un desplazamiento de 16 o 32 bits.

Estilo de programación.- Un programa en ensamblador es una serie de instrucciones ejecutables que le dice al ensamblador que sentencia tiene que ejecutar. Cada sentencia esta compuesta por cuatro campos:

a) Campo nombre.- Se emplea con frecuencia como punto de entrada o regreso. Algunas veces es denominado campo del rotulo, asigna un nombre simbólico a la dirección del comienzo de memoria real de una instrucción de ensamblador. Debe comenzar con un carácter alfabético de 31 caracteres máximo.

b) Campo operación.- Contiene un mnemónico de 2 a 6 caracteres normalmente es una abreviatura en ingles, puede representar una instrucción maquina, una macroinstruccion, o una pseudo-operación.

c) Campo operando.- Contiene la posición o posiciones donde están los datos que van a ser manipulados por la instrucción de la operación, si la instrucción mide 1 o 2 operandos, están separados de la instrucción por lo menos de un espacio en blanco, si hay dos operandos estos están separados por una coma.

d) Campo comentario.- Se utiliza para documentar internamente el código, este debe comenzar con un punto y coma (;).

Formato general de una sentencia

[nombre] mnemónico [operando] [; comentario]

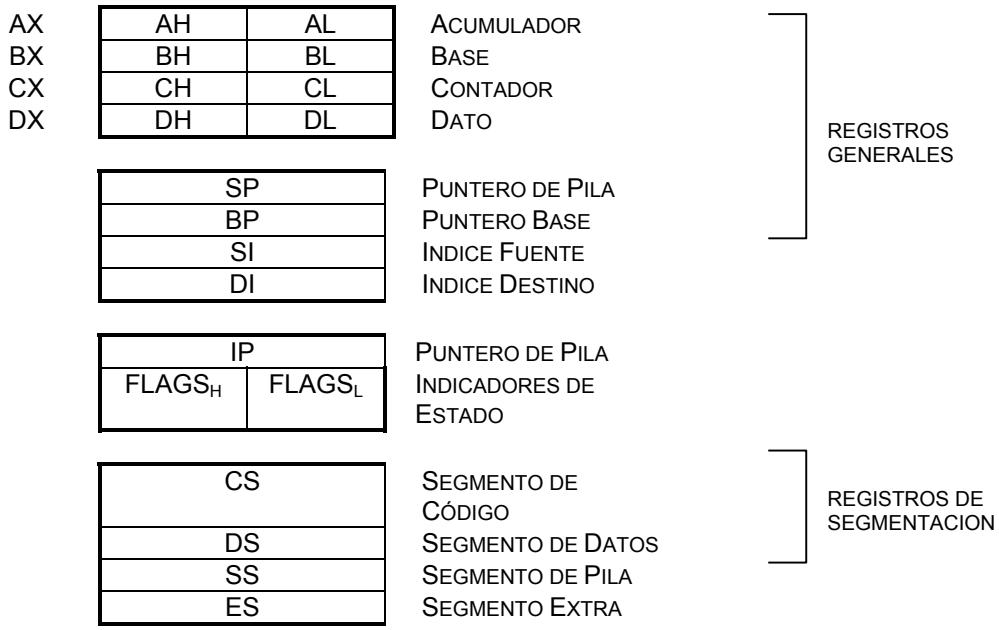
destino, fuente

MOV AX, BX

Operando destino.- La operación puede producir un resultado.

Operando fuente.- La operación puede involucrar uno o mas operandos fuentes, esto es, operandos que son entradas para la operación.

REGISTROS 8086



Técnicas de direccionamiento

El campo o campos de dirección en un formato de instrucción típico son algo limitados. Para hacer referencia a un rango mas grande de localidades en memoria principal o, para sistemas en memoria virtual. Para lograr este objetivo, se han empleado una variedad de técnicas de direccionamiento. Todos involucran un trueque entre el rango de direcciones y/o la flexibilidad de direccionamiento por una parte, y el numero de referencias de memoria y/o la complejidad de calculo de la dirección por otro.

- Direccionamiento Inmediato. El microprocesador decodifica el modo de direccionamiento que está siendo referenciado. Se transfiere un byte, word o dword de datos inmediatos hacia el registro o localidad en la memoria del destino. Ejemplo:

  ```
  MOV  AL, 22H
  ```

  La instrucción copia el 22H de tamaño byte al registro AL.
- Direccionamiento de registros. Transfiere un byte, word o dword desde el registro fuente o localidad en la memoria, hasta el registro o localidad destino en la memoria. Ejemplos:

  ```
  MOV  AX,  BX
  MOV  CX,  DX
  ```

  La instrucción copia el contenido de tamaño de palabra en el registro DX y lo pasa al registro CX. NOTA: de los 8 modos de direccionamiento, el inmediato y el de registros tienen los ciclos de ejecución mas pequeños, como el dato operando puede ser incluido en la misma instrucción y ya este almacenado internamente se evita el tiempo de acceso a dispositivos externos o memoria externa.
- Direccionamiento directo. Mientras el direccionamiento inmediato solamente almacena datos en registros o locaciones de memoria, el direccionamiento directo mueve datos de locaciones de memoria a registro ó desde registro a locaciones de memoria. Ejemplo:

  ```
  LOC_1 DB      00
  LOC_2 DW      0000
  MOV   AL, LOC_1
  MOV   BX, LOC_2
  MOV   LOC_1, AH
  ```
- Direccionamiento indirecto por registros. En lugar de diferenciar con un rotulo de dirección del operando fuente, el valor del operando es señalado por una dirección de desplazamiento almacenada en alguno de los siguientes registros: SI (índice fuente), DI (índice destino), BX (registro base) y BP (puntero base). El designador del operando fuente es reconocido por corchetes ([ ]). Ejemplo:

  ```
  MOV  AX, [BX]
  ```

La instrucción copia los datos en una dirección del segmento de datos con un desplazamiento dado por BX y lo pasa al registro AX.

- Direccionamiento relativo a la base. La dirección efectiva de un operando se obtiene de la suma del desplazamiento y contenido de un registro base (BX o BP) relativo al segmento relacionado. Ejemplo:

  MOV AL, [BX] + 4 ó MOV AL, [BX + 4] ó MOV AL, 4[BX]

  La instrucción copia los datos de una dirección en el segmento de datos, formada por el contenido de BX mas 4, y lo pone en el registro AL.
- Direccionamiento indexado directo. La dirección del desplazamiento del operando se calcula sumando el desplazamiento a un registro índice (SI o DI) en el desplazamiento seleccionado. Frecuentemente se utiliza para acceder a los elementos de un array estático. Ejemplo:

  MOV AL, ARRAY [SI]

  La instrucción copia la dirección formada de sumar ARRAY y SI entonces la coloca en el registro AL.
- Direccionamiento base indexado. El operando se localiza en el segmento seleccionado en un desplazamiento determinado por la suma de los contenido del registro base, registro índice y opcionalmente el desplazamiento. Si no se incluye desplazamiento entonces, el registro base indexado se utiliza con mas frecuencia para acceder a los elementos de un array dinámico. Ejemplo:

  MOV AX, ARRAY [BX+ DI]

  La instrucción copia la dirección formada de sumar ARRAY, BX y DI entonces la coloca en el registro AX.
- Extensiones del 80386. Los modos de direccionamiento de 32 bits se extienden para permitir que cualquier registro sea utilizado como registro base o registro índice. Los modos de 32 bits requieren que los registros índices y bases si se usan tengan valores validos de 32 bits.

Ensambladores residentes y cruzados

Los ensambladores son traductores que transforman programas fuentes escritos en lenguaje ensamblador, en programas objetos equivalentes escritos en lenguaje máquina. La traducción se realiza de tal forma que cada instrucción escrita en ensamblador se transforma en una única instrucción en lenguaje maquina. El lenguaje ensamblador es una simplificación simbólica del lenguaje maquina y el programa ensamblador es su traductor.

- Ensamblador cruzado (cross assembler). Es un traductor de lenguaje ensamblador a lenguaje maquina que traduce en una computadora y ejecuta en otra distinta. La ventaja que ofrece este tipo de traductor es utilizar una computadora de características potentes para desarrollar programas que van dirigidos a otra cuya potencia y facilidades para el programador están ciertamente limitados.
- Macroensamblador (macroassembler). Es un ensamblador que posee la característica de permitir el uso de lo que se denomina microinstrucción. Una micro instrucción no es mas que un grupo de instrucciones que de forma global reciben nombre simbólico al que se puede hacer referencia en un programa tantas veces como se desee. El macroensamblador coloca en la traducción el mencionado grupo de instrucciones en cada una de las referencias (expansión de macros).
- Microensamblador. Es un traductor utilizado en la microprogramacion que algunas computadoras tienen. Estos microprogramas permiten cierta flexibilidad al repertorio de instrucciones maquina de la computadora.
- Ensamblador de una pasada o increméntales. Traducen en una sola pasada construyendo la tabla de símbolos a medida que van apareciendo las variables y etiquetas. Tienen el pequeño inconveniente de no permitir lo que se denomina referencias adelantadas, es decir, referencias a líneas de programa posteriores no traducidos.
- Ensamblador de dos pasadas. Realizan la traducción en dos pasadas, en la primera leen el programa fuente construyendo la tabla de símbolos y asignando las correspondientes direcciones, en la segunda vuelven a leer el programa traduciéndolo a lenguaje maquina. Si están permitidos las referencias adelantadas, siendo este tipo de ensamblador los mas utilizados actualmente.

2. Macroprocesadores

- ***Bibliotecas de macros***

En lenguajes ensambladores una rutina prescrita que es llamada en varios lugares del programa es llamada macro. En el momento de ensamblar las llamadas a la macro son sustituidas ya sea por la subrutina completa o por una serie de instrucciones que derivan a la subrutina. El equivalente en lenguaje de alto nivel es la función. Una vez que en una macro se establece una operación o código solo necesita ser llamado el nombre de la macro desde el nombre del programa para utilizar el código. Cada vez que se llama una macro en un programa el MASM copia y pega el código real de la macro en el programa en lugar del nombre de la macro, entonces se dice que se ejecutan en línea, ya que el flujo del programa es ininterrumpido. Las macros pueden ser creadas en el programa real o llamadas desde

una biblioteca establecida de macros. Una biblioteca de macros es un archivo de macros que puede ser llamado desde el programa actual en tiempo de ensamblamiento.

II. Traductores de alto nivel

Lenguajes de alto nivel
Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de textos. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.
Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.
Intérpretes y compiladores
La traducción de una serie de instrucciones en lenguaje ensamblador (el código fuente) a un código máquina (o código objeto) no es un proceso muy complicado y se realiza normalmente por un programa especial llamado compilador. La traducción de un código fuente de alto nivel a un código máquina también se realiza con un compilador, en este caso más complejo, o mediante un intérprete. Un compilador crea una lista de instrucciones de código máquina, el código objeto, basándose en un código fuente. El código objeto resultante es un programa rápido y listo para funcionar, pero que puede hacer que falle el ordenador si no está bien diseñado. Los intérpretes, por otro lado, son más lentos que los compiladores ya que no producen un código objeto, sino que recorren el código fuente una línea cada vez. Cada línea se traduce a código máquina y se ejecuta. Cuando la línea se lee por segunda vez, como en el caso de los programas en que se reutilizan partes del código, debe compilarse de nuevo. Aunque este proceso es más lento, es menos susceptible de provocar fallos en la computadora.

1. Intérpretes

- ***Lenguajes para aplicaciones específicas susceptibles o idóneos para interpretación***

Interpretación directa o mediante pseudo código
Traductores que realizan lo que podemos llamar traducción directa, es decir, traducir y ejecutar simultáneamente. La ventaja frente a un compilador es la ocupación en memoria es menor ya que suelen ser programas mas reducidos en tamaño que un compilador. La desventaja es el tiempo de ejecución que tarda un programa sujeto a este tipo de traducción. En la actualidad existen interpretes que incorporan una pequeña fase de análisis antes de la interpretación para evitar los problemas originados por la suspensión del programa en ejecución al aparecer el primero de los errores, con lo cual todo proceso transcurrido se pierde. Una buena aplicación de los interpretes es la de depurar programas que una vez en funcionamiento correcto se compilan ejecutándose el producto de dicha compilación en el uso normal del programa.

2. Compiladores

- ***Tipos de gramáticas formales***

Una gramática se define como un conjunto finito de reglas sintácticas. La gramática esta formada de 4 elementos los cuales se identifican como (V, T, P, Z) donde:

V es el alfabeto.
T es el conjunto de símbolos terminales, de donde T esta en V.
P es el conjunto de reglas sintácticas.
Z es un símbolo de V - T, al cual se le llama el símbolo inicial.

El lenguaje de una gramática, es el conjunto de cadenas de símbolos terminales, los cuales pueden ser generados desde Z. Para resolver el problema del análisis de enunciados en un lenguaje de programación se desarrollo la teoría de los lenguajes formales, que enumera diferentes clases de lenguajes con ciertas propiedades, y que pueden ser definidos en términos de sus gramáticas, las cuales generan solamente lenguajes de esa clase, Se definen 4 tipos básicos de gramáticas, los cuales son:

- De tipo 0 ò sin restricciones
- De tipo 1 ó dependientes del contexto

- De tipo 2 ó libres del contexto
- De tipo 3 o regulares

El tipo de gramática que usa un compilador es de tipo 2 ó libre de contexto. Este tipo de gramáticas tienen 4 componentes:

1.- Un conjunto de componentes léxicos, denominados símbolos terminales.
2.- Un conjunto de no terminales
3.- Un conjunto de producciones, en el cada producción consta de un no terminal, llamado lado izquierdo de la producción, una flecha y una secuencia de componentes léxicos y no terminales, o ambos llamado lado derecho de la producción.
4.- La denominación de uno de los no terminales como símbolo inicial.

Ejemplo: prop ⟶ **if** (expr) prop **else** prop

Dicha regla se denomina producción. En una producción, los elementos léxicos, como la palabra clave **if** y los paréntesis, se llaman componentes léxicos. Las variables expr y prop representan secuencias de componentes léxicos y se llaman no terminales. Se supone que los dígitos, los signos y las palabras clave son terminales.

Compilador.- Es un programa que lee un programa escrito en un lenguaje, el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto. El compilador informa a su usuario de la presencia de errores en el programa fuente. En la compilación hay dos partes: análisis y síntesis. La parte del análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia del programa fuente. La parte de la síntesis construye el programa objeto deseado a partir de la representación intermedia. En la compilación, el análisis consta de 3 fases: Las funciones de estos módulos son las siguientes:

Analizador lexicográfico: Las principales funciones que realiza son:

- Identificar los símbolos.
- Eliminar los blancos, caracteres de fin de línea, etc...
- Eliminar los comentarios que acompañan al fuente.
- Crear unos símbolos intermedios llamados tokens.
- Avisar de los errores que detecte.

Ejemplo: A partir de la sentencia en PASCAL siguiente

posición := inicial + velocidad * 2

genera un código simplificado para el análisis sintáctico posterior, por ejemplo:

<id1> <:=> <id2> <+> <id3> <*> <ent>

Nota: Cada elemento encerrado entre <> representa un único token. Las abreviaturas id y ent significan identificador y entero, respectivamente.

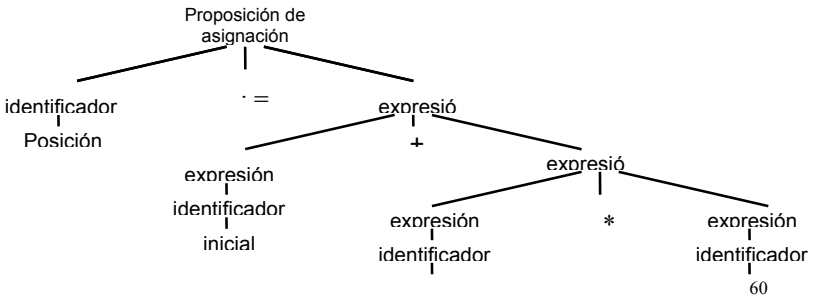
- ***Notación formal de sintaxis***

Analizador sintáctico: Comprueba que las sentencias que componen el texto fuente son correctas en el lenguaje, creando una representación interna que corresponde a la sentencia analizada. De esta manera se garantiza que sólo serán procesadas las sentencias que pertenezcan al lenguaje fuente. Durante el análisis sintáctico, así como en las demás etapas, se van mostrando los errores que se encuentran. Ejemplo: El esquema de la sentencia anterior corresponde al de una sentencia de asignación del lenguaje Pascal. Estas sentencias son de la forma: <id> <:=> <expresión> y la parte que se denomina <expresión> es de la forma:

- Cualquier identificador <id> es una <expresión>
- Cualquier numero es una <expresión>
- Si una expresion$_1$ y expresion$_2$ son expresiones, entonces también lo son

expresion$_1$ + expresion$_2$ expresion$_1$ * expresion$_2$ (expresion$_1$)

La estructura de la sentencia queda, por tanto, de manifiesto mediante el siguiente esquema:



60

Análisis semántico: Se ocupa de analizar si la sentencia tiene algún significado. Se pueden encontrar sentencias que son sintácticamente correctas pero que no se pueden ejecutar porque carecen de sentido. En general, el análisis semántico se hace a la par que el análisis sintáctico introduciendo en éste unas rutinas semánticas.
Ejemplo: En la sentencia que se ha analizado, existe una variable entera. Sin embargo, las operaciones se realizan entre identificadores reales, por lo que hay dos alternativas: o emitir un mensaje de error "Discordancia de tipos", o realizar una conversión automática al tipo superior, mediante una función auxiliar entareal.

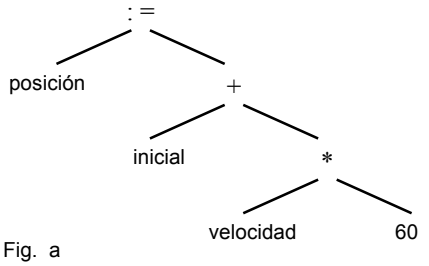


Fig. a

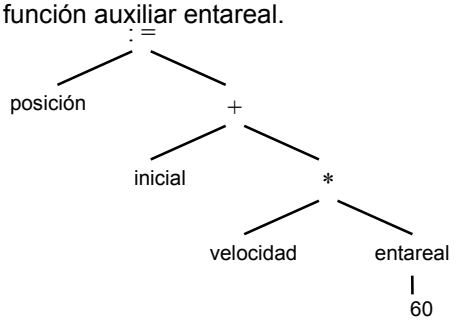


Fig. b

a) La verificación de tipos revela que * se aplica a un real, velocidad, y a un entero, 60.

b) El tratamiento general es convertir el entero a real. Esto se ha logrado creando un nodo extra para el operador entareal que de manera explícita convierte un entero a real.

- ***Análisis lexicográfico***

Generadores de analizadores léxicos
Estas herramientas generan automáticamente analizadores léxicos, por lo general a partir de una especificación basada en expresiones regulares. La organización básica del analizador léxico resultante es en realidad un autómata finito. La forma mas sencilla para crear una analizador léxico consiste en la construcción de un diagrama que ilustre la estructura de los componentes léxicos del lenguaje fuente, y después hacer la traducción “a mano” del diagrama a un programa para encontrar los componentes léxicos. En la construcción de los analizadores léxicos el problema de fondo es la especificación y diseño de programas que ejecuten las acciones activadas por patrones dentro de las cadenas. Como la programación dirigida por patrones es de mucha utilidad, se introduce un lenguaje de patrón-acción, llamado LEX, para especificar los analizadores léxicos. En este lenguaje, los patrones se especifican por medio de expresiones regulares, y un compilador de LEX puede generar un reconocedor de las expresiones regulares mediante un autómata finito eficiente.

- ***Generación de código, códigos intermedios***

Generador de código: A partir de los análisis anteriores y de las tablas que estos análisis van creando durante su ejecución produce un código o lenguaje objeto que es directamente ejecutable por la máquina. Es la fase final del compilador. Las instrucciones del código intermedio se traducen una a una en código máquina reubicable.
Nota: Cada instrucción de código intermedio puede dar lugar a más de una de código máquina.
Ejemplo: Utilizando los registros 1 y 2, la traduccion del codigo anterior podria convertirse en:

```
MOVF   id,     R2
MULF   #60.0, R2
MOVF   id2,    R1
ADDF   R2,     R1
MOVF   R1,     id1
```

El primero y segundo operandos de cada instrucción especifican una fuente y un destino, respectivamente. La F de cada instrucción indica que las instrucciones trabajan con números de punto flotante. Este código traslada el contenido de la dirección id3 al registro 2, después lo multiplica por la constante real 60.0. El signo # significa que 60.0 se trata como una constante. La tercera instrucción pasa id2 al registro 1. La cuarta instrucción le suma el valor previamente calculado en el registro 2. Por ultimo el valor del registro 1 se pasa a la dirección de id1.
Generador de código intermedio: El código intermedio es un código abstracto independiente de la máquina para la que se generará el código objeto. El código intermedio ha de cumplir dos requisitos importantes: ser fácil de producir a partir del análisis sintáctico, y ser fácil de traducir al lenguaje objeto. Esta fase puede no existir si se genera directamente código máquina, pero suele ser conveniente emplearla.
Ejemplo: Consideremos, por ejemplo, un código intermedio de tercetos, llamado así porque en cada una de sus instrucciones aparecen como máximo tres operandos. La sentencia traducida a este código intermedio quedaría :
temp1 := entareal (60)

temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3

- ***Optimización de código***

Optimizador de código: A partir de todo lo anterior crea un nuevo código más compacto y eficiente, eliminando por ejemplo sentencias que no se ejecutan nunca, simplificando expresiones aritméticas, etc... La profundidad con que se realiza esta optimización varía mucho de unos compiladores a otros. En el peor de los casos esta fase se suprime.
Ejemplo: Siguiendo con el ejemplo anterior, es posible evitar la función entareal mediante el cambio de 60 por 60.0, obviando además una de las operaciones anteriores. El código optimizado queda como sigue :
temp1 := id3 * 60.0
id1 := id2 + temp1
La tabla de símbolos: Es el medio de almacenamiento de toda la información referente a las variables y objetos en general del programa que se está compilando.
Ejemplo: Hemos visto que en ciertos momentos del proceso de compilación debemos hacer uso de cierta información referente a los identificadores o los números que aparecen en nuestra sentencia, como son su tipo, su posición de almacenamiento en memoria, etc... Esta información es la que se almacena en la tabla de símbolos.
Rutinas de errores: Están incluidas en cada uno de los procesos de compilación (análisis lexicográfico, sintáctico y semántico), y se encargan de informar de los errores que encuentran en texto fuente.
Ejemplo: El analizador semántico podría emitir un error (o al menos un aviso) cuando detectase una diferencia en los tipos de una operación.
Generadores de compiladores
Poco después de escribir el primer compilador, aparecieron sistemas para ayudar en el proceso de escritura de compiladores. Se les conoce como generadores de compiladores, compiladores de compiladores o sistemas generadores de traductores. En gran parte, se orientan en torno a un modelo particular de lenguaje, y son mas adecuados para generar compiladores de lenguajes similares al modelo. Se han creado algunas herramientas generales para el diseño automático de componentes específicos de un compilador. Estas herramientas utilizan lenguajes especializados para especificar e implementar el componente, y pueden utilizar algoritmos bastantes complejos. Las herramientas mas efectivas son las que ocultan los detalles del algoritmo de generación y producen componentes que se pueden integrar con facilidad al resto del compilador. Herramientas útiles para la construcción de un compilador:

- Generadores de analizadores sintácticos.- Estos generadores producen analizadores sintácticos, normalmente a partir de una entrada fundamentada en una gramática independiente del contexto. Estos utilizan poderosos algoritmos de análisis sintáctico, y son demasiado complejos para realizarlos manualmente.
- Generadores de analizadores léxicos.- Estas herramientas generan automáticamente analizadores léxicos, por lo general a partir de una especificación basada en expresiones regulares. La organización básica del analizador léxico resultante es en realidad un autómata finito.
- Dispositivos de traducción dirigida por sintaxis.- Estos producen grupos de rutinas que recorren el árbol de análisis sintáctico, generando código intermedio.
- Generadores automáticos de código.- Tales herramientas toman un conjunto de reglas que definen la traducción de cada operación del lenguaje intermedio al lenguaje de maquina para la maquina objeto. Las reglas deben incluir suficiente detalle para poder manejar los distintos métodos de acceso posibles a los datos. La técnica fundamental es la de “concordancia de plantillas”. Las proposiciones de código intermedio se reemplazan por “plantillas” que representan secuencias de instrucciones de maquina, de modo que las suposiciones sobre el almacenamiento de las variables concuerden de plantilla a plantilla.
- Dispositivos para análisis de flujo de datos.- Mucha de la información necesaria para hacer una buena Optimización de código implica hacer un “análisis de flujo de datos”, que consiste en la recolección de información sobre la forma en que se transmiten los valores de una parte de un programa a cada una de las otras partes.

Tópicos de compilación
Tratamiento de recursividad
Un procedimiento es recursivo si puede comenzar una nueva activación antes de que haya terminado una activación anterior del mismo procedimiento. En términos mas formales, se dice que un lenguaje cuyas frases pueden ser generadas por un procedimiento es recursivamente enumerable, mientras que se le llama recursivo si existe un algoritmo para reconocerlo. Que un lenguaje sea recursivamente enumerable significa que se puede producir mediante una gramática; que sea recursivo significa que existe para el un algoritmo de reconocimiento: una maquina de turing que llega a un estado final (un si o un no) cuando analiza las frases que lo componen.

Tratamiento de lenguajes orientados a objetos
Los programas orientados a objetos se enfatizan en los datos, al contrario que la programación estructurada que se enfatiza en los algoritmos.
Clase.- Es una plantilla o modelo que define los datos y las funciones que actúan sobre esos datos (llamados métodos).
Objeto.- Elemento dato que es miembro de una clase.
Herencia.- Las clases se pueden subdividir en subclases, dicho de otro modo, una clase se puede deducir o derivar de otra clase. La clase original se denomina clase base y merced a la propiedad de herencia, una clase derivada o descendiente puede heredar las estructuras de datos y los métodos de su clase base. Además, la nueva clase puede añadir nuevos elementos datos y métodos a los que hereda de su clase base.
Instanciamiento.- Las clases son tipos de datos definidos por el usuario que definen un tipo de objeto. Un objeto es un modelo o instancia de este tipo o clase. De modo que un objeto es a una clase como una variable a un tipo. Por eso, a veces, el termino objeto se utiliza indistintamente como instancia o modelo de una clase, y también como variable.

3. Ambientes integrados

- ***Depuración interactiva***

Permiten la ejecución paso a paso ó por tramos del programa manteniendo el entorno que se va produciendo (valores de variables). El programador en cada parada de la ejecución de su programa puede comprobar e incluso modificar valores de las variables, con lo cual resulta muy interesante este tipo de ejecución para detección de errores y comprobación del buen funcionamiento del programa. Los puntos de parada en la ejecución son determinados por el propio programador mediante sentencias destinadas a tal fin.
Tipos de depuradores:
DUMP (Volcado de memoria).- Exhibición del contenido de la memoria. Cuando un programa tiene una terminación anormal, puede hacerse un volcado de la memoria para examinar el estado del programa en el momento del estallido, el programador mira dentro de los buffer para ver con que elementos de datos estaba trabajando cuando fallo. Los contadores conmutadores, y señales del programa también pueden ser inspeccionados.
TRACE (Trazo).- Un trazo es un ayudante de la depuración consistiendo de una presentación que describe la crónica de acciones y resultados de los pasos individuales en un programa; la sección es algunas veces usada por un programa de control que produce esta clase de manifestación. El proceso de depuración precipita incontables problemas, cuyas correcciones requieren detalladamente registros de la ejecución del programa, un trazo es diseñado para proveer este tipo de información, tomando el programa del usuario y localizándolo bajo el control de una rutina especial, la cual monitorea el proceso del programa.

B. SISTEMAS OPERATIVOS

I. Conceptos y estructuras básicas

1. Historia y evolución

- ***Necesidad del sistema operativo, administración de recursos y desempeño del sistema.***

Cuando aparecieron los primeros ordenadores, la programación de estos era hecha íntegramente en código máquina, lo cual resultaba una tarea extremadamente pesada: cada vez que se escribía un nuevo programa, además de escribir el algoritmo adecuado era preciso añadir todo el código necesario para que el ordenador pudiese leer datos desde una cinta perforada, imprimir en un teletipo, etc. Dado que, en general, todas estas rutinas eran exactamente iguales para todos los programas que se hacían, pronto los programadores de aquellas máquinas aprendieron a organizarlas en bibliotecas de rutinas. Cada vez que había que escribir un nuevo programa, solo tenían que ir a su libreta y copiar las rutinas de Entrada/Salida que necesitaban, lo cual les simplificaba un poco el trabajo. Otro sistema era el que la propia empresa que fabricaba el ordenador incluyese un paquete de fichas perforadas con dichas rutinas, de modo que el programador solo tenía que coger las que le interesasen y añadirlas estratégicamente en su paquete. El siguiente paso fue generalizar este conjunto de rutinas. La idea era incluir juntas todas las rutinas necesarias para acceder al hardware, y hacerlas accesibles a cualquier programador en forma de llamadas a

subrutina. De este modo, cada vez que se hacía un programa no era preciso incluir en él todas esas rutinas. Había nacido el Sistema Operativo. Los primeros 'Sistemas Operativos' (si es que podían recibir ese nombre) no eran más que un conjunto de subrutinas que ayudaban al programador, ofreciéndole servicios básicos como lectura de caracteres desde un teletipo, escritura en tambor, disco, etc. Sin embargo, pronto se vio la necesidad de un nuevo enfoque. Los ordenadores eran caros, y su mantenimiento también, por lo que solo algunas grandes empresas y universidades podían disponer de ellos. Para amortizarlos, se alquilaba tiempo de proceso, de modo que se podía ir allí con un programa, ejecutarlo, y pagar por el tiempo que le llevase hacerlo al ordenador. El primer problema que había era que se tardaba mucho en cargar cada programa. Cuando el ordenador terminaba de ejecutar uno, el operador tenía que insertar el siguiente, lo cual era una tarea bastante pesada y larga. Durante el tiempo que se hacía esto, el ordenador estaba totalmente inactivo, se estaba desperdiciando tiempo. Fue entonces cuando surgieron los sistemas de proceso por lotes (batch). En los sistemas de proceso por lotes, los programas se almacenan en una cinta, todos seguidos. Cada vez que el ordenador terminaba de ejecutar un programa, leía el siguiente de dicha cinta. Si llegaba alguien con un programa, este se añadía a continuación del último que hubiese, mientras el ordenador iba ejecutando el que ya tenía en memoria. El coste de mantenimiento del nuevo sistema era ligeramente superior, pues había que añadir al sistema el lector de cintas en donde se escribían los programas; pero como podía ejecutar muchos más programas en el mismo tiempo (porque no se perdía parte de este en cargar los programas), el precio de alquiler bajó drásticamente, permitiendo que más gente tuviese acceso a los ordenadores.

- ***Generaciones de las computadoras***

El desarrollo de los sistemas operativos esta muy relacionado con el desarrollo de las arquitecturas de los sistemas computacionales, por esta razón clasificaremos el desarrollo de los sistemas operativos en generaciones que han sido marcadas en base a las tecnologías de construcción de dichas arquitecturas.

Primera Generación (1945-1955)

Este periodo se caracteriza porque existía un sólo grupo de personas que se dedicaba a diseñar, construir, programar, operar y mantener las máquinas. La tecnología que se utilizaba era la de Tubos al vacío, los que ocupaban gran espacio, lentos y disipaban mucho calor. La programación en esta época era exclusivamente en lenguaje de máquina o bien en acciones directas en el hardware, modificando circuitos (Hardware). En esta etapa no existe el concepto de sistema operativo, todo lo debe realizar el programador, no hay un ente software intermedio entre la máquina y el programador. Por otro lado, los recursos del sistema eran asignados a una sola tarea, no era posible realizar tareas simultáneamente con distintos recursos. En el último tiempo de este período, aparecieron las tarjetas perforadas, las que en algún sentido permitieron agilizar el proceso de programación, pues con ellas era posible escribir y leer los programas en las tarjetas. Sin embargo, la programación igual se realizaba en lenguaje de máquina.

Segunda Generación (1955-1965)

Esta generación se caracteriza por la utilización de la tecnología que incluye transistores. Estos proporcionan mejores características que los tubos al vacío : son más pequeños, disipan menos calor y son más rápidos. En esta época, ya no hay un solo grupo que se dedica a diseñar, construir, operar y mantener el sistema computacional, sino que la manutención la realizan otras personas. La programación, en este período se ve apoyada por la creación de lenguajes ensambladores y lenguajes de alto nivel; como Fortran y Cobol. Para optimizar el tiempo de utilización del sistema computacional, se adoptó el Procesamiento por Lotes. Este procedimiento se puede resumir en los siguientes pasos:

- Varios trabajos son leídos desde tarjetas perforadas y almacenados en cinta
- El computador ejecuta programas que lee de una cinta y escribe los resultados en otra cinta.
- Los resultados escritos en la cinta son desplegados a través de la impresora.

Para hacer la transferencia de los datos entre el computador y la cinta se construyó un programa, éste es el que se conoce como el primer sistema operativo. Como consecuencia de la utilización de tecnología de mejores características, el procesamiento por lotes, la utilización de un operador de sistema y un sistema operativo primitivo se logró minimizar el tiempo ocioso de los computadores.

Tercera Generación (1965-1980)

La tecnología utilizada en este período corresponde a la de circuitos integrados, lo cual permitió construir máquinas más baratas y con mejor rendimiento. La empresa IBM fue la primera en utilizar esta tecnología. Una de las principales características de los sistemas operativos de hoy, que aparecen en esta generación es la Multiprogramación. Esta técnica consiste en particionar la memoria de manera de contener simultáneamente muchos trabajos, lo que permite independencia en las acciones de la máquina. Por ejemplo, si un trabajo está esperando por

Entrada/Salida, otro trabajo puede estar utilizando la CPU. Con la utilización de la multiprogramación, se mejora el tiempo de utilización de la CPU, pero se crean problemas de protección y seguridad, debido al particionamiento de la memoria. Otra característica importante de los sistemas operativos de esta generación es la aparición de la técnica de Spooling (Operación simultánea y en línea de periféricos). Esta consiste en la capacidad de leer los trabajos de cinta al disco, haciendo posible que a medida que hubiera espacio en memoria se podía pasar al siguiente trabajo. De esta manera las operaciones de periféricos se realizan en paralelo con la utilización de la CPU. Durante esta generación aparecen los primeros sistemas de Tiempo compartido, esto consiste en que se le asigna a cada trabajo un tiempo fijo de utilización de CPU, este período de tiempo (denominado time slot) es pequeño, pues el uso de tiempo compartido tiene como objetivo manejar a distintos usuarios mediante terminales que tengan la ilusión de estar ocupando la máquina como si fueran únicos. El primer sistema operativo importante de tiempo compartido es el (CTSS) y fue desarrollado en MIT (Massachusetts Institute of Technology). Otros de los sistemas que surgió en este periodo es MULTICS, el que sirvió de base para el desarrollo del sistema UNIX.

Cuarta Generación (1980-1990)

Esta generación se caracteriza por la utilización de los microprocesadores para la construcción de computadores de menor tamaño, haciendo posible que una sola persona tuviera su propio computador personal. Los computadores personales con mayores capacidades y que se empezaron a utilizar en la empresa y en las universidades recibieron el nombre de estaciones de trabajo. Durante esta generación aparecen los sistemas operativos que proporcionan una interfase más amigable al usuario, esto se traduce en mejores herramientas de interacción con el computador, tales como capacidades gráficas, iconos, sistemas de menús, etc. Los sistemas operativos que cobran gran popularidad en este período son MS-DOS de Microsoft y UNIX. Este último tuvo sus primeros desarrollos en Berkeley y posteriormente en AT&T.

Quinta Generación (1990-)

Desde aproximadamente mediados de los '80 surgió el crecimiento de las redes de computadores con sistemas operativos de red y sistemas operativos distribuidos. En los sistemas operativos de red cada computador tiene su propia copia de sistema operativo, los usuarios saben que existen varios computadores, pueden conectarse explícitamente a diferentes máquinas remotas para transferir archivos, hacer búsquedas, etc. En los sistemas operativos distribuidos existe una integración real de los recursos, la red es transparente a los usuarios, es decir, que éstos pueden no darse cuenta de la existencia de varias máquinas conectadas. Sobre este tipo de sistema operativo, una tarea puede ser ejecutada en varios nodos a la vez, pues existen facilidades de migración de procesos. Además este sistema computacional puede crecer fácilmente y la tolerancia a fallas se realiza en mejor forma (la tolerancia a fallas está relacionada con la confiabilidad, por ejemplo, si un computador falla puede seguir operando en su reemplazo).

2. Esquema básico

- ***Estructura interna: monolítico y modular***

El sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales puede llamar a los demás cada vez que así lo requiera. Cuando se usa esta técnica, cada procedimiento del sistema tiene una interfaz bien definida en términos de parámetros y resultados, y cada uno de ellos es libre de llamar a cualquier otro, si éste último proporciona un cálculo útil para el primero. Para construir el programa objeto real del sistema operativo siguiendo este punto de vista, se compilan de forma individual los procedimientos, o los ficheros que contienen los procedimientos, y después se enlazan en un sólo fichero objeto con el enlazador. En términos de ocultación de la información, ésta es prácticamente nula: cada procedimiento es visible a los demás (en contraste con una estructura con módulos o paquetes, en la que la mayoría de la información es local a un módulo, y donde sólo los datos señalados de forma expresa pueden ser llamados desde el exterior del módulo). Los servicios (mediante llamadas al sistema) que proporciona el sistema operativo se solicitan colocando los parámetros en lugares bien definidos, como los registros o la pila, para después ejecutar una instrucción especial de trampa, a veces referida como llamada al núcleo o llamada al supervisor. Esta instrucción cambia la máquina del modo usuario al modo núcleo (también conocido como modo supervisor), y transfiere el control al sistema operativo. Esta organización sugiere una estructura básica del sistema operativo:

- Un programa principal que llama al procedimiento del servicio solicitado.
- Un conjunto de procedimientos de servicio que lleva a cabo las llamadas al sistema.
- Un conjunto de procedimientos de utilidades que ayudan a los procedimientos de servicio.

En este modelo, para cada llamada al sistema existe un procedimiento de servicio que se encarga de ella. Los procedimientos de utilidad hacen cosas necesarias para varios procedimientos de servicio, como por ejemplo, buscar los datos del programa del usuario.

Modelo cliente-servidor

Una tendencia de los sistema operativos modernos es la de trasladar el código a capas superiores, y eliminar la mayor parte posible del sistema operativo para mantener un núcleo mínimo. El punto de vista usual es el implantar la mayoría de las funciones del sistema operativo como procesos de usuario. Para solicitar un servicio, como la lectura de un bloque de cierto fichero, un proceso de usuario (denominado en este caso proceso cliente) envía la solicitud a un proceso servidor, que realiza el trabajo y devuelve la respuesta. Lo único que hace el núcleo es controlar la comunicación entre los clientes y los servidores. Al separar el sistema operativo en partes, cada una de ellas controla una faceta del sistema, como el servicio a ficheros, servicio a procesos, servicio a terminales o servicio a la memoria; cada parte es pequeña y controlable. Además, puesto que todos los servidores se ejecutan como procesos en modo usuario, y no en modo núcleo, no tienen acceso directo al hardware. En consecuencia, si hay un error en el servidor de ficheros éste puede fallar, pero esto no afectará en general a toda la máquina. Otra de las ventajas del modelo cliente-servidor es su capacidad de adaptación para su uso en sistemas distribuidos. Si un cliente se comunica con un servidor mediante mensajes, el cliente no necesita saber si el mensaje se gestiona de forma local, en su máquina, o si se envía por medio de una red a un servidor en una máquina remota. En lo que respecta al cliente, lo mismo ocurre en ambos casos: se envió una solicitud y se recibió una respuesta. La idea anterior de un núcleo que sólo controla el transporte de mensajes de clientes a servidores, y viceversa, no es totalmente real. Algunas funciones del sistema operativo (como la introducción de órdenes en los registros físicos de los controladores de E/S) son difíciles, si no es que imposible de realizar, a partir de programas de usuario. Existen dos formas de afrontar este problema. Una es hacer que algunos procesos de servidores críticos (por ejemplo, los gestores de los dispositivos de E/S) se ejecuten en realidad en modo núcleo, con acceso total al hardware, pero de forma que se comuniquen con los demás procesos mediante el mecanismo normal de mensajes. La otra forma es construir una cantidad mínima de mecanismos dentro del núcleo, pero manteniendo las decisiones de política relativos a los usuarios dentro del espacio de los usuarios. Por ejemplo, el núcleo podría reconocer que cierto mensaje enviado a una dirección especial indica que se tome el contenido de ese mensaje y se cargue en los registros del controlador de algún disco, para iniciar la lectura del disco. En este ejemplo, el núcleo ni siquiera inspeccionaría los bytes del mensaje para ver si son válidos o tienen algún sentido; sólo los copiaría ilegalmente en los registros del controlador del disco. Es evidente que debe utilizarse cierto esquema para limitar tales mensajes sólo a los procesos autorizados. La separación entre mecanismos y política es un concepto importante, aparece una y otra vez en diversos contextos de los sistemas operativos.

- ***Tipos de sistemas: monousuario, multiusuario, distribuido, de red, de tiempo real, de propósito especial***

Los sistemas operativos son clasificados tomando en cuenta los siguientes elementos: planificación del uso del procesador, gestión de la memoria, gestión de los dispositivos de E/S, y gestión de los archivos.

- Sistemas operativos de lotes
- Sistemas operativos de multiprogramación
- Sistemas de tiempo compartido
- Sistemas de tiempo real
- Sistemas operativos distribuidos
- Sistemas de red
- Sistemas de propósito especial

A) Sistemas operativos de lotes:

El procesamiento por lotes precisa que las funciones a realizar sean agrupada en forma de trabajos (``jobs''). Normalmente, no hay interacción entre el usuario y los programas (i.e. procesamiento de nóminas). planificación: lineal (solo un proceso a la vez). Se pueden aplicar algunas políticas para mejorar el uso de los recursos. Por ejemplo, procesar el trabajo mas pequeño primero. memoria: normalmente se divide en dos partes, una conteniendo los programas del SO, y la segunda conteniendo programas transitorios. E/S: no hay gestión de dispositivos debido a la planificación lineal. archivos: no hay protección (solo un usuario a la vez puede accesar los archivos).

B) Sistemas operativos de multiprogramación:

La característica principal de estos sistemas es que multiplexan los recursos entre un grupo de programas activos. Definamos algunos términos:

Multitarea: capacidad de soportar la ejecución de 2 o mas procesos activos. Manteniendo el código y los datos varios procesos en memoria principal, el procesador y los dispositivos son multiplexados entre ellos. Para esto, es necesario un sistema de protección de la memoria para evitar que algunos procesos corrompan el funcionamiento de otros procesos. multiprogramación: este término designa a un sistema operativos que además de soportar multitarea, proporciona formas sofisticadas de protección de memoria y fuerza el control de la concurrencia cuando los procesos acceden a dispositivos de E/S. Estos sistemas normalmente permiten el acceso múltiple de usuarios.
Multiacceso: se permite el acceso simultáneo de varios usuarios desde terminales remotas. El Multiacceso no implica multiprogramación. Pensemos en los sistemas bancarios o de reservas aéreas. multiprocesamiento: estos sistemas gestionan la operación de sistemas con varios procesadores. Como consecuencia, estos sistemas son multitareas. La multitarea puede estar o no implementada en procesadores individuales. Ejemplos de estos sistemas son Solarios, CRAY-UNIX, .

C) Sistemas de tiempo compartido:

Estos sistemas son normalmente multiprogramados y multiusuario. **Planificación**: el tiempo del procesador es repartido entre los procesos aplicando una política de prioridades (``time-slice''). Normalmente, son implementados utilizando colas de espera para evitar monopolios. memoria: es necesario asegurar el aislamiento y la protección entre las áreas de memoria asignadas a los procesos. E/S: sofisticada, debe asegurar la integridad del sistema. archivos: debido a que múltiples usuarios o procesos pueden accesar los mismos archivos, es necesario un sistema capaz de proporcionar protección y control de acceso.

D) Sistemas de tiempo real:

Estos sistemas se caracterizan por la prioridad que se da a los eventos externos. Normalmente, son aplicados para controlar dispositivos como robots, reactores nucleares, telefonía, etc. planificación: dado la importancia de los eventos externos, estos están ligado a ciertos procesos por medio de interrupciones de hardware (que son tratadas independientemente). Se aplica una planificación expropiativa basada en prioridades (diversos procesos pueden ser ejecutados en diferentes frecuencias). memoria: dada la importancia de la velocidad para procesar eventos externos, estos están normalmente cargados permanentemente en memoria principal. Por esto, la gestión de la memoria es sencilla. Dado que frecuentemente los procesos requieren comunicarse, es necesario tener un soporte para la separación y compartición de áreas de memoria. E/S: el tratamiento de algunos dispositivos críticos en tiempo es la característica principal de estos sistemas. Se requiere pues un sistema sofisticado de tratamiento de interrupciones que permita ligar procesos del usuario a vectores de interrupción. archivos: algunos sistemas de tiempo real no necesitan memoria secundaria (la computadora de abordo de los coches por ejemplo). En muchas ocasiones, el sistema de archivos es conectado por medio de una red.

E) Sistemas operativos distribuidos:

Estos son sistemas autónomos capaces de comunicarse y cooperar entre si para resolver tareas globales. Necesariamente, el uso de redes es indispensable para intercambiar datos. Además de los servicios típicos de un sistema operativos, un sistema distribuido debe gestionar la distribución de tareas entre los diferentes nodos conectados. También, debe proporcionar los mecanismos necesarios para compartir globalmente los recursos del sistema.
Planificación: para compartir los procesadores y poder sincronizar los procesos, se deben soportar las llamadas remotas (a otros nodos). Un ejemplo claro de esto es RPC (Remote Procedure Call).
Archivos: un ejemplo típico de estos sistemas es NFS (Network File System) que permite accesar datos en otros nodos.

F) Sistemas de red:

Sus características principales son:
- software débilmente acoplado en hardware débilmente acoplado es la combinación más común de encontrar.
- un ejemplo típico, es una red de estaciones de trabajo de ingeniería o workstation, conectadas mediante una LAN. En este modelo, cada usuario tiene una estación de trabajo para su uso exclusivo. Puede o no tener un disco duro. En definitiva, tiene su propio sistema operativo. Lo normal es que todos los comandos se ejecuten en forma local, justo en la estación de trabajo.
- Sin embargo, a veces es posible que un usuario se conecte de manera remota con otra estación de trabajo mediante un comando como:

rlogin máquina o telnet máquina

- El efecto de estos comandos es convertir la propia estación de trabajo del usuario en un terminal remoto enlazada con la máquina remota. Los comandos escritos en el teclado se envían a la máquina remota y la salida de la máquina remota se exhibe en la pantalla.

- Para alternar con otra máquina remota, primero es necesario desconectarse de la primera, utilizar después el comando rlogin para conectarse a la otra. En cualquier instante, sólo se puede utilizar una máquina y la selección de ésta se realiza de forma manual.
- Las redes de las estaciones de trabajo también tienen un comando de copiado remoto para copiar archivos de una máquina a otra. Por ejemplo, un comando como :

rcp máquina1:archivo1 máquina2:archivo2

copiaría el archivo archivo1 de máquina1 a máquina2 con el nombre de archivo2. De nuevo, el movimiento de los archivos es explícito y se requiere que el usuario esté completamente consciente de la posición de todos los archivos y el sitio donde se ejecutan todos los comandos.
- Un método mas moderno y transparente consiste en proporcionar un sistema de archivos global compartido, accesible desde todas las estaciones de trabajo.
- Una o varias máquinas, llamadas servidores de archivos, soportan al sistema de archivos. Los servidores aceptan solicitudes de los programas de usuarios, los cuales se ejecutan en las otras máquinas (no servidoras), llamadas clientes, para la lectura y escritura de archivos.
- Cada una de las solicitudes que llegue se examina, se ejecuta y la respuesta se envía de regreso.
- Los servidores de archivos tienen, por lo general, un sistema jerárquico de archivos, cada uno de los cuales tiene un directorio raíz, con subdirectorios y archivos.
- Las estaciones de trabajo pueden importar o montar estos sistemas de archivos, lo que aumenta sus sistemas locales de archivos con aquellos localizados en los servidores.
- Puesto que cada una de las estaciones de trabajo operan en forma relativamente independiente del resto, no existe garantía alguna de que todas presenten la misma jerarquía de directorios a sus programas.
- El sistema operativo a utilizar en este tipo de ambiente debe controlar las estaciones de trabajo en lo individual, a los servidores de archivo y también debe encargarse de la comunicación entre ellos.
- Es posible que todas las máquinas ejecuten el mismo sistema operativo, pero esto no es necesario. Si los clientes y los servidores ejecutan diversos sistemas, entonces, como mínimo, deben coincidir en el formato y significado de todos los mensajes que podrían intercambiar.
- En una situación como ésta, en la que cada máquina tiene un alto grado de autonomía y existen pocos requisitos a lo largo de todo el sistema, las personas se refieren a ella como un sistema operativo de red.

G) Sistemas de propósito especial:

Algunos de los Sistemas de Propósito Especial son los siguientes:

- S.O. de tiempo real.
- S.O. con tolerancia a fallas.
- S.O. virtuales.

S.O. de tiempo real

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

1. Garantizar la respuesta a eventos externos dentro de limites de tiempo reestablecidos.
2. Los parámetros más importantes son el tiempo de procesamiento de la entrada y un rápido almacenamiento de la misma.
3. Existen dos tipos de Sistemas de tiempo real: En los que el tiempo de respuesta no es muy critico, y en los que lo es.
4. El sistema de tiempo real incluye software y hardware compuesto por:

Sensores: elementos que detectan mediante la alteración de sus características un cambio en el ambiente que miden.
Traductores: traducen un cambio físico en una corriente o tensión eléctrica.
Conversores: convierten las variaciones de corriente o tensión en pulsos binarios.
Interfaces: puertos, registradores, alarmas, consolas, etc.
Procesadores: son los responsables de las tareas de control, interrupciones, tiempos de almacenamiento, etc.
Un caso particular de estos sistemas es el del Sistema de Control de Procesos.

S.O. con tolerancia a fallas

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

- ***Lenguajes de control (shell), interfaces gráficas y utilitarios de un sistema operativo***
- ***Llamadas al sistema***

3. Arquitectura de un sistema operativo

- ***Manejo del procesador: políticas y técnicas para la gestión (scheduling)***

Los sistemas operativos multiprogramados necesitan del concepto de proceso. El sistema operativo debe entremezclar la ejecución de un número de procesos para maximizar la utilización de los recursos del ordenador. Al mismo tiempo, los sistemas de tiempo compartido deben proporcionar un tiempo de respuesta razonable. El sistema operativo debe asignar recursos a los procesos de acuerdo a una política específica (ciertas funciones o aplicaciones son de mayor prioridad), mientras impide los interbloqueos. Por último, el sistema operativo debe ofrecer un soporte para llevar a cabo la comunicación entre procesos.

Qué es un proceso

Un **programa** es una secuencia de instrucciones escrita en un lenguaje dado. Un **proceso** es una instancia de ejecución de un programa, caracterizado por su contador de programa, su palabra de estado, sus registros del procesador, su segmento de texto, pila y datos, etc. Un programa es un concepto estático, mientras que un proceso es un concepto dinámico. Es posible que un programa sea ejecutado por varios usuarios en un sistema multiusuario, por cada una de estas ejecuciones existirá un proceso, con su contador de programa, registros, etc. El sistema operativo necesita el concepto de proceso para poder gestionar el procesador mediante la técnica de multiprogramación o de tiempo compartido, de hecho, el proceso es la unidad planificable, o de asignación de la CPU.

Estados de un proceso y Transiciones de estado de los procesos

Durante su vida, un proceso puede pasar por una serie de estados discretos, algunos de ellos son:

En ejecución: El proceso ocupa la CPU actualmente, es decir, se está ejecutando.

Listo o preparado: El proceso dispone de todos los recursos para su ejecución, sólo le falta la CPU.

Bloqueado: Al proceso le falta algún recurso para poder seguir ejecutándose, además de la CPU. Por recurso se pueden entender un dispositivo, un dato, etc. El proceso necesita que ocurra algún evento que le permita poder proseguir su ejecución. Por sencillez, se considera un sistema con una sola CPU, aunque no es difícil la extensión a múltiples procesadores. Solamente puede haber un proceso en ejecución a la vez, pero pueden existir varios listos y varios pueden estar bloqueados. Así pues, se forman una lista de procesos listos y otra de procesos bloqueados. La lista de procesos listos se ordena por prioridad, de manera que el siguiente proceso que reciba la CPU será el primero de la lista. La lista de procesos bloqueados normalmente no está ordenada; los procesos no se desbloquean (es decir, no pasan a ser procesos listos) en orden de prioridad, sino que lo hacen en el orden de ocurrencia de los eventos que están esperando. Como se verá más adelante, hay situaciones en las cuales varios procesos pueden bloquearse esperando la ocurrencia del mismo evento; en tales casos es común asignar prioridades a los procesos que esperan.

Transiciones de estado de los procesos

De ejecución a Bloqueado: al iniciar una operación de E/S, al realizar una operación WAIT sobre un semáforo a cero (en el tema de procesos concurrentes se estudiarán los semáforos).

De ejecución a Listo: por ejemplo, en un sistema de tiempo compartido, cuando el proceso que ocupa la CPU lleva demasiado tiempo ejecutándose continuamente (agota su cuanto) el sistema operativo decide que otro proceso ocupe la CPU, pasando el proceso que ocupaba la CPU a estado listo.

De Listo a en ejecución: cuando lo requiere el planificador de la CPU (veremos el planificador de la CPU en el tema de planificación de procesos).

De Bloqueado a Listo: se dispone del recurso por el que se había bloqueado el proceso. Por ejemplo, termina la operación de E/S, o se produce una operación SIGNAL sobre el semáforo en que se bloqueó el proceso, no habiendo otros procesos bloqueados en el semáforo.

Niveles de Planificación

La planificación de la CPU, en el sentido de conmutarla entre los distintos procesos, es una de las funciones del sistema operativo. Este despacho es llevado a cabo por un pequeño programa llamado planificador a corto plazo o dispatcher (despachador). La misión del dispatcher consiste en asignar la CPU a uno de los procesos ejecutables del

sistema, para ello sigue un determinado algoritmo. Los acontecimientos que pueden provocar la llamada al dispatcher dependen del sistema (son un subconjunto de las interrupciones), pero son alguno de estos:

- El proceso en ejecución acaba su ejecución o no puede seguir ejecutándose (por una E/S, operación WAIT, etc).
- Un elemento del sistema operativo ordena el bloqueo del proceso en ejecución.
- El proceso en ejecución agota su cuantum o cuanto de estancia en la CPU.
- Un proceso pasa a estado listo.

Hay que destacar el hecho de que cuanto menos se llame al dispatcher menos tiempo ocupa la CPU un programa del sistema operativo, y, por tanto, se dedica más tiempo a los procesos del usuario (un cambio de proceso lleva bastante tiempo). Así, si sólo se activa el dispatcher como consecuencia de los 2 primeros acontecimientos se estará haciendo un buen uso del procesador. Este criterio es acertado en sistemas por lotes en los que los programas no son interactivos. Sin embargo, en un sistema de tiempo compartido no es adecuado, pues un proceso que se dedicara a realizar cálculos, y no realizara E/S, monopolizaría el uso de la CPU. En estos sistemas hay que tener en cuenta el conjunto de todos los procesos, activándose el dispatcher con la circunstancia tercera y, posiblemente, la cuarta. Los sistema operativos en que las dos siguientes circunstancias no provocan la activación del dispatcher muestran preferencia por el proceso en ejecución, si no ocurre esto se tiene más en cuenta el conjunto de todos los procesos.

Se puede definir el scheduling -algunas veces traducido como -planificación- como el conjunto de políticas y mecanismos construidos dentro del sistema operativo que gobiernan la forma de conseguir que los procesos a ejecutar lleguen a ejecutarse.

El **scheduling** está asociado a las cuestiones de:

- Cuándo introducir un nuevo proceso en el Sistema.
- Determinar el orden de ejecución de los procesos del sistema.
- El scheduling está muy relacionado con la gestión de los recursos.
  - Planificador de la CPU o a corto plazo.
  - Planificador a medio plazo.
  - Planificador a largo plazo.

Planificación a largo plazo

Este planificador está presente en algunos sistemas que admiten además de procesos interactivos trabajos por lotes. Usualmente , se les asigna una prioridad baja a los trabajos por lotes, utilizándose estos para mantener ocupados a los recursos del sistema durante períodos de baja actividad de los procesos interactivos. Normalmente, los trabajos por lotes realizan tareas rutinarias como el cálculo de nóminas; en este tipo de tareas el programador puede estimar su gasto en recursos, indicándoselo al sistema. Esto facilita el funcionamiento del planificador a largo plazo. El objetivo primordial del planificador a largo plazo es el de dar al planificador de la CPU una mezcla equilibrada de trabajos, tales como los limitados por la CPU (utilizan mucho la CPU) o la E/S. Así, por ejemplo, cuando la utilización de la CPU es baja, el planificador puede admitir más trabajos para aumentar el número de procesos listos y, con ello, la probabilidad de tener algún trabajo útil en espera de que se le asigne la CPU. A la inversa, cuando la utilización de la CPU llega a ser alta, y el tiempo de respuesta comienza a reflejarlo, el planificador a largo plazo puede optar por reducir la frecuencia de admisión de trabajos. Normalmente, se invoca al planificador a largo plazo siempre que un proceso termina. La frecuencia de invocación depende, pues, de la carga del sistema, pero generalmente es mucho menor que la de los otros dos planificadores. Esta baja frecuencia de uso hace que este planificador pueda permitirse utilizar algoritmos complejos, basados en las estimaciones de los nuevos trabajos.

Planificación a Medio Plazo

En los sistemas de multiprogramación y tiempo compartido varios procesos residen en la memoria principal. El tamaño limitado de ésta hace que el número de procesos que residen en ella sea finito. Puede ocurrir que todos los procesos en memoria estén bloqueados, desperdiciándose así la CPU. En algunos sistemas se intercambian procesos enteros (swap) entre memoria principal y memoria secundaria (normalmente discos), con esto se aumenta el número de procesos, y, por tanto, la probabilidad de una mayor utilización de la CPU. El planificador a medio plazo es el encargado de regir las transiciones de procesos entre memoria principal y secundaria, actúa intentando maximizar la utilización de los recursos. Por ejemplo, transfiriendo siempre a memoria secundaria procesos bloqueados, o transfiriendo a memoria principal procesos bloqueados únicamente por no tener memoria.

- ***Manejo de memoria secundaria: políticas y técnicas para la gestión***

La organización y gestión de la memoria

Para que un proceso pueda ejecutarse debe estar ubicado en la memoria principal del ordenador. Una parte del sistema operativo se va a encargar de gestionar la memoria principal, de forma que los procesos puedan residir en la memoria sin conflictos. La gestión de la memoria implica varias tareas, una de ellas es llevar un registro de qué zonas están libres (es decir, no están siendo utilizadas por ningún proceso), y qué zonas están ocupadas por qué procesos. Otra tarea importante surge en sistemas en los que no todos los procesos, o no todo el código y datos de un proceso, se ubican en la memoria principal. En estos sistemas, a menudo se debe pasar parte, o la totalidad del código y datos de un proceso, de memoria a disco, o viceversa; siendo el sistema operativo responsable de esta tarea. De esta forma se libera al usuario de realizar estas transferencias de información, de las cuales no es consciente. Otros dos temas importantes en la gestión de la memoria son el de la carga de los programas de disco a memoria y el de la protección. Desde el momento en que varios procesos deben compartir la memoria del ordenador surge el problema de la protección. En general, se pretende que un proceso no pueda modificar las direcciones de memoria en las que no reside. Esto es así ya que en las direcciones de memoria donde no está ubicado el proceso pueden residir otros procesos, o código o estructuras de datos del S.O. Si un proceso puede modificar indiscriminadamente la memoria, podría, por ejemplo, cambiar el valor de una dirección de memoria donde residiera una variable de otro proceso, con la consecuente ejecución incorrecta del proceso propietario de la variable. Algunos sistemas ni siquiera permiten que un proceso pueda leer las direcciones de memoria en las que no reside, con esto se consigue privacidad sobre el código y datos de los procesos. Existen varias formas de gestionar la memoria. Por lo común, la forma de gestión dependerá de la máquina virtual que se quiera proporcionar y del hardware subyacente. Con independencia de la forma de gestión es necesario decidir qué estrategias se deben utilizar para obtener un rendimiento óptimo. Las estrategias de administración de la memoria determinan el comportamiento de una organización de memoria determinada cuando se siguen diferentes políticas: ¿ Cuándo se coge un nuevo programa para colocarlo en la memoria ? ¿ Se coge el programa cuando el sistema lo necesita, o se intenta anticiparse a las peticiones del sistema ? ¿ En qué lugar de la memoria principal se coloca el siguiente programa por ejecutar ? ¿ Se colocan los programas lo más cerca posible unos de otros en los espacios disponibles de la memoria principal para reducir al mínimo el desperdicio de espacio, o se colocan lo más rápido posible para reducir el tiempo empleado en tomar la decisión ?

Jerarquía de la memoria

Los programas y datos necesitan estar en la memoria principal para ser ejecutados, o para poder ser referenciados. Los programas o datos que no se necesitan de inmediato pueden guardarse en la memoria secundaria hasta que se necesiten, y en ese momento se transfieren a la memoria principal para ser ejecutados o referenciados. Los soportes de memoria secundaria, como cintas o discos, son en general menos caros que la memoria principal, y su capacidad es mucho mayor. Normalmente, es mucho más rápido el acceso a la memoria principal que a la secundaria. En los sistemas con varios niveles de memoria hay muchas transferencias constantes de programas y datos entre los distintos niveles. Estas transferencias consumen recursos del sistema, como tiempo de la CPU, que de otro modo podrían utilizarse provechosamente. En los años sesenta se hizo evidente que la jerarquía de la memoria podía extenderse un nivel más, con una clara mejora del rendimiento. Este nivel adicional, la memoria caché, es una memoria de alta velocidad, mucho más rápida que la memoria principal. La memoria caché es extremadamente cara, si se compara con la principal, por lo que sólo se utilizan memorias caché relativamente pequeñas. La memoria caché introduce un nivel adicional de transferencia de información en el sistema. Los programas en memoria principal se pasan a la memoria caché antes de ejecutarse. En la memoria caché se pueden ejecutar mucho más rápido que en la principal. La esperanza de los diseñadores es que el trabajo extra requerido por la transferencia de programas sea mucho menor que el incremento del rendimiento obtenido por la ejecución más rápida en la caché.

Gestión de la memoria en los sistemas monoprogramados

En los sistemas de monoprogramación sólo existe un proceso de usuario, que disfruta de todos los recursos del ordenador. Esto va a simplificar notablemente la gestión de la memoria, ya que ésta sólo debe ser compartida por los programas del sistema operativo, y por el único proceso de usuario existente. Dependiendo de detalles de diseño, el sistema operativo ocupará la parte baja de la memoria RAM, o la parte alta de la memoria ROM. El PC de IBM ubica parte del sistema operativo en RAM, y los gestores de dispositivos en ROM; a esta última parte se le llama BIOS (Basic Input/Output System, sistema básico de entrada/salida). Si el usuario conoce la ubicación en la memoria del sistema operativo, entonces puede escribir programas en términos de direcciones absolutas de memoria. Una dirección absoluta de memoria es una dirección física (es decir, real) de la memoria. En contraposición se tienen las direcciones relativas. Un programa está escrito en término de direcciones relativas cuando se escribe suponiendo que empieza a cargarse en la dirección cero de la memoria. Por lo general, los usuarios escriben programas en

lenguajes de alto nivel, por lo que son los traductores los encargados de generar las direcciones que ocupan las variables, procedimientos, etc. en la memoria. Los compiladores no generan direcciones absolutas de memoria, pues no saben dónde se almacenarán los procesos. Por lo común, los sistemas operativos monousuario de monoprogramación (muy comunes en las microcomputadoras) no tienen protección de la memoria. Por lo tanto, el único proceso de usuario que existe en la memoria, puede modificar posiciones de memoria pertenecientes al sistema operativo, esto provocaría errores al ejecutarse la zona modificada. La protección se puede realizar mediante un registro de límite integrado en la CPU. El registro de límite contendrá la dirección de inicio de carga del S.O. El hardware, en tiempo de ejecución, verifica que las direcciones generadas por el proceso de usuario no son superiores al valor del registro de límite. En caso de ser superior, el proceso de usuario intenta acceder al S.O., esto provoca una interrupción hardware que gestiona el S.O., normalmente eliminando al proceso.

- ***Manejo de dispositivos de E/S***

Su control adecuado es indispensable para el funcionamiento de la interfaz y del sistema de archivos y para la comunicación de la máquina con el exterior, leyendo y escribiendo datos a través de diversos dispositivos. Los componentes de E/S en un S.O está constituida básicamente por los manejadores de dispositivos de los cuales existe uno para cada tipo.

Dispositivos

A cada dispositivo físico reconocido le asigna un nombre interno. Un dispositivo físico puede tener varios nombres y algunos nombres pueden ser asignados dinámicamente a diferentes dispositivos. Al conjunto de nombres internos de los dispositivos se les denomina dispositivos lógicos. El manejo de dispositivos lógicos permite cambiar el destino de un archivo, sin modificar el programa que lo genera, asignándole el nombre a otro archivo. En un sistema pueden existir dispositivos especiales, tal como el dispositivo del sistema (aquel donde reside el sistema operativo). Por otra parte, los dispositivos pueden agruparse en tres tipos, de acuerdo a su uso:

a) asignados o dedicados
b) compartidos y
c) virtuales

a) Dispositivos asignados. Estos son aquellos que únicamente pueden atender a una tarea, por un tiempo relativamente largo. Por ello se asigna a la tarea que los requiere y son de su uso exclusivo hasta que los libere o concluya su ejecución.

b) Dispositivos compartidos. Existen dispositivos que por su naturaleza permiten que las tareas los empleen por tiempos muy cortos, de tal forma que el efecto real es de que varias tareas los comparten (como el caso de los discos magnéticos y los discos ópticos).

c) Dispositivos virtuales. Emulador. Es la simulación de un dispositivo asignado en otro compartido.

Controladores

La mayoría de los dispositivos se conectan a la computadora a través de un controlador. Existen controladores que soportan un solo dispositivo. Cada controlador recibe ordenes del procesador, generadas por el SO u otro programa; los datos los extrae directamente de memoria. El controlador realiza sus instrucciones haciendo trabajar a los dispositivos que tiene asociados, y supervisando su operación. Los resultados se envía de nuevo al procesador generalmente a través de interrupciones. Desde el punto de vista del S.O. un controlador puede verse como un conjunto de registros de uso especial, que realizan una o más de las siguientes funciones: transmisión de órdenes, envío y recibo de datos y control del estado de los dispositivos conectados. Cuando se desea realizar una operación de entrada o salida, es necesario averiguar el estado del dispositivo, enviar las órdenes necesarias y enviar o recibir los datos.

Relación con el resto del sistema operativo

Los manejadores correspondientes a dispositivos, son los programas que realizan el proceso de transferencia de datos y órdenes con el controlador. Su acción es iniciada desde algún otro programa, especialmente del subsistema de archivos. En la entrada-salida, como se maneja usualmente desde un lenguaje de alto nivel, las ordenes siempre pasan por el sistema de archivos, donde se determina el dispositivos y el archivo a que se refiere la orden.

Estructura de datos

En el manejo de los dispositivos existen una cuantas estructuras básicas: los buffers que recibe del sistema de archivos y que regresa al mismo; los registros del controlador y los dispositivos, que debe consultar y modificar, y las que le son propias: descriptores de dispositivos y cola de solicitudes pendientes. Los registros de controladores y dispositivos son partes de estos que sirven para comunicarse con el resto de la computadora. Su consulta y actualización se realiza por medio de subrutinas primitivas de la parte central del SO. Estos registros contienen los

siguientes elementos: orden específica recibida, resultado de una operación, parámetros recibidos, estado del dispositivo y códigos de error. Los descriptores son tablas que contienen información acercar de los dispositivos. Los elementos mínimos de un descriptor de dispositivos son los siguientes:

- nombre del dispositivo (lógico)
- controlador del que depende
- estado actual (listo, en falla)
- apuntadores a lista de solicitudes pendientes

La cola de dispositivos pendientes se refiere a aquellas solicitudes originadas en las tareas, a través del sistema de archivos o de otras partes del sistema operativo, y que deben ser satisfechas por algún dispositivo.

Manejadores

Un manejador es una rutina que se encarga de programar a un controlador, y de esta manera realizar la entrada salida de datos. Un manejador básicamente recibe ordenes generales de entrada salida, como "envía" el contenido del buffer a la impresora, y luego las convierte a secuencias de instrucciones que envía al controlador correspondiente, esperando sus respuesta. Los manejadores pueden agruparse en dos categorías:

- Aquellas que manejan registros más o menos grandes, que son enviados o recibidos en memoria, (ej. disco o cinta magnética).
- Aquellos que manejan cadenas de caracteres enviados (o recibidos) uno a uno (ej. las terminales).

Otra diferencia se da entre los dispositivos asignados y compartidos. En el primer caso la activación del manejador queda bajo la responsabilidad de la tarea solicitante, mientras que en el segundo el manejador permanece activo mientras haya solicitudes en la cola asignada a él, venga de donde sea. Un manejador puede verse como un paquete que contiene varias rutinas. La principales son:

- atención a la interrupción del dispositivo
- inicio de atención a solicitud
- tiempo límite del dispositivo
- cancelación de una solicitud

- ***Sistema de archivos: archivos y directorios. Estructura, organización y tipos.***

Se dice que le sistema de archivos es el esqueleto sobre le que descansa gran parte del sistema operativo y de las aplicaciones. Los archivos comienzan a surgir cuando el sistema crece y se requiere de mayor capacidad y rapidez; en ese momento se necesitan dispositivos de acceso directo como son los discos en todas sus variedades. Un archivo desde el punto de vista del S.O es un conjunto de datos que deseamos manejar juntos. Físicamente es una colección de bloques de disco o cinta, que incluyen a los datos mismo y además otros datos para dar unidad a los bloques que lo forman. Estos últimos se denominarán Apuntadores Bloques de Datos. Los apuntadores a bloques de datos se organizan en varias formas:

- como un dato al final de cada bloque, enlazándolos en una lista ligada.
- como un bloque de apuntadores a los bloques.
- como una lista ligada externa que apunta a los bloques de datos.

Componentes

Un sistema de archivos, de cualquier tipo, tiene dos componentes básicas: una que permite el acceso a los archivos y otra que controla el uso de éstos, cada una conformada por un conjunto de estructuras y una colección de operaciones.

Llamada a las funciones de acceso y control

El control de archivos incluye dos aspectos: el control físico del espacio disponible y el control lógico de ellos. El primer incluye los Apuntadores a Bloques de Datos, el Bit Map y la tabla de archivos dañados. El segundo incluye los directorios y tablas de archivos activos. El propósito del control lógico es el manejo de archivos en forma unificada y homogénea, sin atender a los aspectos físicos. Para algunas funciones, especialmente las que tienen que ver con la componente de control, se requiere de parámetros como estos:

a) identificación del archivo (dispositivo, directorio, nombre, tipo, versión)
b) modo de empleo (por registro, por bloque, por carácter).
c) forma de acceso (secuencial o directo).
d) uso propuesto (lectura, escritura, adición de registros).
e) organización (montón, secuencial, secuencial indizado, etc).

Directorios

Cuando un dispositivo es compartido por varios usuarios, es conveniente organizar sus archivos en grupos. También cuando un usuario tiene archivos de varios tipos, aplicaciones o proyectos, resulta conveniente distinguirlos. Así surgen los **directorios** como una organización de tipo jerárquico impuesta sobre los archivos. La organización de los archivos puede hacerse de diversas formas y empleando diferentes atributos para clasificarlos. Los más frecuente son la clave o cuenta del usuario y el tipo de archivo. Sin embargo también existen sistemas con directorios arbitrarios. En general se podría decir que un directorio es un archivo que contiene información acerca de un grupo de archivos que se desea manejar como una unidad. La estructura básica de un directorio es con una raíz, una serie de directorios a uno o más niveles y, al final, los descriptores de archivos, que vienen a ser las hojas. La representación interna de un directorio se realiza generalmente como árbol binario donde los nodos hermanos forma una lista que se revisa secuencialmente y solo se cambia a otro archivo cuando se pasa a un nodo descendiente. Debido a este tipo de estructura, deben evitarse dos extremos al estructurar los archivos en directorios: si hay pocos niveles, la búsqueda secuencial será larga; s hay demasiados niveles, los cambios de nodo serán muchos haciendo lento el proceso de localización de archivos.

Módulo de Acceso

El módulo de acceso es el que se encarga de las operaciones básicas sobre un archivo y es el enlace con el manejador del dispositivo donde éste se localiza. Las operaciones básicas que realiza son cuatro:

a) Abrir . Esta operación consiste de los siguientes pasos: localizar el archivo en el directorio y de ahí en el índice; con ello, se tiene la dirección inicial del archivo.

b) Cerrar. Pasos inversos a la operación de abrir.

c) Leer. Esta operación transfiere datos a la rutina llamadora, de acuerdo a la solicitud. Generalmente los datos se transmiten del buffer a la tarea, registro por registro, pero pueden ser también por variable o por carácter.

d) Escribir. Todo lo que el usuario escribe se va acumulando en el buffer y cuando este se llena, se le envía al archivo a través del manejador del dispositivo correspondiente, después de verificar que exista espacio disponible.

Modulo de control

En este módulo importan más sus estructuras que sus procesos.

Para los *discos magnéticos:*

La principales estructuras son:

A) Control físico del espacio
   a) bit map
   b) archivo de bloques dañados

B) Control lógico
   c) directorios
   d) encabezados
   e) tabla de archivos activos
   f) descriptor de volumen y bloque de carga

Para las *cintas magnéticas:*

Las cintas magnéticas aún representan un medio económico de almacenar y transferir grandes volúmenes de datos y programas. Sin embargo, la limitación de su acceso secuencial permite un manejo más simple. Básicamente, su organización puede darse en tres formas:

a) archivos sin etiquetas
b) archivos con etiquetas estándar (ANSI)
c) archivos con etiquetas del fabricante del equipo.

En general los principales procesos contenidos en la componente de control son: borrar, crear, conformar, inicializar, pedir espacio y renombrar

Borrar: Este proceso se realiza eliminando lógicamente el archivo, es decir, se le marca "borrado", pero físicamente no se limpian los bloques que antes ocupaba.

Buscar: Para buscar un archivo se recorren los directorios hasta encontrarlo o hasta encontrar una marca de fin de directorio.

Crear: Busca un lugar en el directorio que corresponda, tomando el primer lugar marcado libre por una operación de borrado o después la marca de fin de directorio.

Conformar: Es el proceso de estructuración de un disco para que pueda recibir archivos.

Inicializar: Es el proceso de hacer reconocible al disco por el sistema de archivos.

Pedir espacio: Cuando se va a crear un nuevo bloque de datos en un archivo, debe pedirse. El proceso de asignación incluye la visita al bit map para localizar el segmento apropiado y la actualización de los bloques requiere de bloques

adicionales, para nuevos apuntadores a bloques de datos.
Renombrar: Este es un proceso sencillo y basta con localizar el archivo y cambiarle el nombre en el índice y los directorios.
Uso compartido de archivos
El uso de archivos cae en una de las categorías siguientes:

a) archivos de un solo usuario (privados)
b) archivos compartidos
    b1) para leer, únicamente
    b2) para leer y escribir
        b21) en forma concurrente
        b22) en forma concurrente

4. Desempeño de un sistema operativo

- ***Algoritmos de schedulling***

Se puede definir el scheduling -algunas veces traducido como -planificación- como el conjunto de políticas y mecanismos construidos dentro del sistema operativo que gobiernan la forma de conseguir que los procesos a ejecutar lleguen a ejecutarse.
El **scheduling** está asociado a las cuestiones de:

- Cuándo introducir un nuevo proceso en el Sistema.
- Determinar el orden de ejecución de los procesos del sistema.
- El scheduling está muy relacionado con la gestión de los recursos.
    - Planificador de la CPU o a corto plazo.
    - Planificador a medio plazo.
    - Planificador a largo plazo.

Planificación a largo plazo
Este planificador está presente en algunos sistemas que admiten además de procesos interactivos trabajos por lotes. Usualmente , se les asigna una prioridad baja a los trabajos por lotes, utilizándose estos para mantener ocupados a los recursos del sistema durante períodos de baja actividad de los procesos interactivos. Normalmente, los trabajos por lotes realizan tareas rutinarias como el cálculo de nóminas; en este tipo de tareas el programador puede estimar su gasto en recursos, indicándoselo al sistema. Esto facilita el funcionamiento del planificador a largo plazo. El objetivo primordial del planificador a largo plazo es el de dar al planificador de la CPU una mezcla equilibrada de trabajos, tales como los limitados por la CPU (utilizan mucho la CPU) o la E/S. Así, por ejemplo, cuando la utilización de la CPU es baja, el planificador puede admitir más trabajos para aumentar el número de procesos listos y, con ello, la probabilidad de tener algún trabajo útil en espera de que se le asigne la CPU. A la inversa, cuando la utilización de la CPU llega a ser alta, y el tiempo de respuesta comienza a reflejarlo, el planificador a largo plazo puede optar por reducir la frecuencia de admisión de trabajos. Normalmente, se invoca al planificador a largo plazo siempre que un proceso termina. La frecuencia de invocación depende, pues, de la carga del sistema, pero generalmente es mucho menor que la de los otros dos planificadores. Esta baja frecuencia de uso hace que este planificador pueda permitirse utilizar algoritmos complejos, basados en las estimaciones de los nuevos trabajos.
Planificación a Medio Plazo
En los sistemas de multiprogramación y tiempo compartido varios procesos residen en la memoria principal. El tamaño limitado de ésta hace que el número de procesos que residen en ella sea finito. Puede ocurrir que todos los procesos en memoria estén bloqueados, desperdiciándose así la CPU. En algunos sistemas se intercambian procesos enteros (swap) entre memoria principal y memoria secundaria (normalmente discos), con esto se aumenta el número de procesos, y, por tanto, la probabilidad de una mayor utilización de la CPU. El planificador a medio plazo es el encargado de regir las transiciones de procesos entre memoria principal y secundaria, actúa intentando maximizar la utilización de los recursos. Por ejemplo, transfiriendo siempre a memoria secundaria procesos bloqueados, o transfiriendo a memoria principal procesos bloqueados únicamente por no tener memoria.

5. Dispositivos y servicios especiales

- ***Dispositivos de entrada/salida***

Su control adecuado es indispensable para el funcionamiento de la interfaz y del sistema de archivos y para la

comunicación de la máquina con el exterior, leyendo y escribiendo datos a través de diversos dispositivos. Los componentes de E/S en un S.O está constituida básicamente por los manejadores de dispositivos de los cuales existe uno para cada tipo.

Dispositivos

A cada dispositivo físico reconocido le asigna un nombre interno. Un dispositivo físico puede tener varios nombres y algunos nombres pueden ser asignados dinámicamente a diferentes dispositivos. Al conjunto de nombres internos de los dispositivos se les denomina dispositivos lógicos. El manejo de dispositivos lógicos permite cambiar el destino de un archivo, sin modificar el programa que lo genera, asignándole el nombre a otro archivo. En un sistema pueden existir dispositivos especiales, tal como el dispositivo del sistema (aquel donde reside el sistema operativo). Por otra parte, los dispositivos pueden agruparse en tres tipos, de acuerdo a su uso:

a) asignados o dedicados
b) compartidos y
c) virtuales

a) Dispositivos asignados. Estos son aquellos que únicamente pueden atender a una tarea, por un tiempo relativamente largo. Por ello se asigna a la tarea que los requiere y son de su uso exclusivo hasta que los libere o concluya su ejecución.

b) Dispositivos compartidos. Existen dispositivos que por su naturaleza permiten que las tareas los empleen por tiempos muy cortos, de tal forma que el efecto real es de que varias tareas los comparten (como el caso de los discos magnéticos y los discos ópticos).

c) Dispositivos virtuales. Emulador. Es la simulación de un dispositivo asignado en otro compartido.

Controladores

La mayoría de los dispositivos se conectan a la computadora a través de un controlador. Existen controladores que soportan un solo dispositivo. Cada controlador recibe ordenes del procesador, generadas por el SO u otro programa; los datos los extrae directamente de memoria. El controlador realiza sus instrucciones haciendo trabajar a los dispositivos que tiene asociados, y supervisando su operación. Los resultados se envía de nuevo al procesador generalmente a través de interrupciones. Desde el punto de vista del S.O. un controlador puede verse como un conjunto de registros de uso especial, que realizan una o más de las siguientes funciones: transmisión de órdenes, envío y recibo de datos y control del estado de los dispositivos conectados. Cuando se desea realizar una operación de entrada o salida, es necesario averiguar el estado del dispositivo, enviar las órdenes necesarias y enviar o recibir los datos.

Relación con el resto del sistema operativo

Los manejadores correspondientes a dispositivos, son los programas que realizan el proceso de transferencia de datos y órdenes con el controlador. Su acción es iniciada desde algún otro programa, especialmente del subsistema de archivos. En la entrada-salida, como se maneja usualmente desde un lenguaje de alto nivel, las ordenes siempre pasan por el sistema de archivos, donde se determina el dispositivos y el archivo a que se refiere la orden.

Estructura de datos

En el manejo de los dispositivos existen una cuantas estructuras básicas: los buffers que recibe del sistema de archivos y que regresa al mismo; los registros del controlador y los dispositivos, que debe consultar y modificar, y las que le son propias: descriptores de dispositivos y cola de solicitudes pendientes. Los registros de controladores y dispositivos son partes de estos que sirven para comunicarse con el resto de la computadora. Su consulta y actualización se realiza por medio de subrutinas primitivas de la parte central del SO. Estos registros contienen los siguientes elementos: orden específica recibida, resultado de una operación, parámetros recibidos, estado del dispositivo y códigos de error. Los descriptores son tablas que contienen información acercar de los dispositivos. Los elementos mínimos de un descriptor de dispositivos son los siguientes:

- nombre del dispositivo (lógico)
- controlador del que depende
- estado actual (listo, en falla)
- apuntadores a lista de solicitudes pendientes

La cola de dispositivos pendientes se refiere a aquellas solicitudes originadas en las tareas, a través del sistema de archivos o de otras partes del sistema operativo, y que deben ser satisfechas por algún dispositivo.

Manejadores

Un manejador es una rutina que se encarga de programar a un controlador, y de esta manera realizar la entrada salida de datos. Un manejador básicamente recibe ordenes generales de entrada salida, como "envía" el contenido del buffer a la impresora, y luego las convierte a secuencias de instrucciones que envía al controlador

correspondiente, esperando sus respuesta. Los manejadores pueden agruparse en dos categorías:

- Aquellas que manejan registros más o menos grandes, que son enviados o recibidos en memoria, (ej. disco o cinta magnética).
- Aquellos que manejan cadenas de caracteres enviados (o recibidos) uno a uno (ej. las terminales).

Otra diferencia se da entre los dispositivos asignados y compartidos. En el primer caso la activación del manejador queda bajo la responsabilidad de la tarea solicitante, mientras que en el segundo el manejador permanece activo mientras haya solicitudes en la cola asignada a él, venga de donde sea. Un manejador puede verse como un paquete que contiene varias rutinas. La principales son:

- atención a la interrupción del dispositivo
- inicio de atención a solicitud
- tiempo límite del dispositivo
- cancelación de una solicitud

II. Sistemas operativos especializados

1. Tipos especiales de sistemas operativos

- ***Sistemas operativos de red: servidores de archivos y de impresión, arquitectura cliente-servidor, arquitectura par a par.***

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.

Ventajas

Aumento de la productividad:

Los usuarios pueden utilizar herramientas que le son familiares, como hojas de cálculo y herramientas de acceso a bases de datos. Mediante la integración de las aplicaciones cliente/servidor con las aplicaciones personales de uso habitual, los usuarios pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.
Una interfaz gráfica de usuario consistente reduce el tiempo de aprendizaje de las aplicaciones.

Menores costes de operación:

Permiten un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. Por ejemplo, la compartición de servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes permite un mejor rendimiento del conjunto.

Proporcionan un mejor acceso a los datos:

La interfaz de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que se produzcan en él y de la ubicación de la información. El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

Mejora en el rendimiento de la red:

Las arquitecturas cliente/servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso. Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente. Entonces, la máquina cliente presenta los datos al usuario mediante interfaces amigables. Todo esto reduce el tráfico de la red, lo que facilita que pueda soportar un mayor número de usuarios. Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones. Las UCPs utilizadas en los respectivos equipos pueden dimensionarse a partir de las aplicaciones y el tiempo de respuesta que se requiera. La existencia de varias UCPs proporciona una red más fiable: un fallo en uno de los equipos no significa necesariamente que el sistema deje de funcionar. En una arquitectura como ésta, los clientes y los servidores son independientes los unos de los otros con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente, sin afectar al resto del sistema. La arquitectura modular de los sistemas cliente/servidor permite el uso de ordenadores especializados (servidores de base de datos, servidores de ficheros, estaciones de trabajo para CAD, etc.). Permite centralizar el control de sistemas que estaban descentralizados, como por ejemplo la gestión de los ordenadores personales que antes estuvieran aislados.

Inconvenientes

Hay una alta complejidad tecnológica al tener que integrar una gran variedad de productos. Requiere un fuerte rediseño de todos los elementos involucrados en los sistemas de información (modelos de datos, procesos, interfaces, comunicaciones, almacenamiento de datos, etc.). Además, en la actualidad existen pocas herramientas que ayuden a determinar la mejor forma de dividir las aplicaciones entre la parte cliente y la parte servidor. Es más difícil asegurar un elevado grado de seguridad en una red de clientes y servidores que en un sistema con un único ordenador centralizado. A veces, los problemas de congestión de la red pueden degradar el rendimiento del sistema por debajo de lo que se obtendría con una única máquina (arquitectura centralizada). También la interfaz gráfica de usuario puede a veces ralentizar el funcionamiento de la aplicación. El quinto nivel de esta arquitectura (bases de datos distribuidas) es técnicamente muy complejo y en la actualidad hay muy pocas implantaciones que garanticen un funcionamiento totalmente eficiente. Existen multitud de costes ocultos (formación en nuevas tecnologías, licencias, cambios organizativos, etc.) que encarecen su implantación.

Relación con otros conceptos

Arquitectura cliente/servidor y downsizing

Muchas organizaciones están transportando sus aplicaciones a plataformas más pequeñas (downsizing) para conseguir la ventaja que proporcionan las nuevas plataformas físicas más rentables y la arquitectura cliente/servidor. Este transporte siempre supone un coste, debido a la necesidad de rediseñar las aplicaciones y de re-entrenar a los usuarios en los nuevos entornos.

Independencia de Bases de Datos

Las arquitecturas cliente/servidor permiten aprovechar los conceptos de cliente y servidor para desarrollar aplicaciones que accedan a diversas bases de datos de forma transparente. Esto hace viable cambiar la aplicación en la parte servidora, sin que la aplicación cliente se modifique. Para que sea posible desarrollar estas aplicaciones es necesaria la existencia de un estándar de conectividad abierta que permita a los ordenadores personales y estaciones de trabajo acceder de forma transparente a bases de datos corporativas heterogéneas.

Relación con los Sistemas Abiertos

Las arquitecturas cliente/servidor se asocian a menudo con los sistemas abiertos, aunque muchas veces no hay una relación directa entre ellos. De hecho, muchos sistemas cliente/servidor se pueden aplicar en entornos propietarios. En estos entornos, el equipo físico y el lógico están diseñados para trabajar conjuntamente, por lo que, en ocasiones se pueden realizar aplicaciones cliente/servidor de forma más sencilla y fiable que en los entornos que contienen plataformas heterogéneas. El problema surge de que los entornos propietarios ligan al usuario con un suministrador en concreto, que puede ofrecer servicios caros y limitados. La independencia del suministrador que ofrecen los entornos de sistemas abiertos, crea una competencia que origina mayor calidad a un menor precio. Pero, por otra parte, debido a la filosofía modular de los sistemas cliente/servidor, éstos se utilizan muchas veces en entornos de diferentes suministradores, adecuando cada máquina del sistema a las necesidades de las tareas que realizan. Esta tendencia está fomentando el crecimiento de las interfaces gráficas de usuario, de las bases de datos y del software de interconexión. Debido a esto, se puede afirmar que los entornos cliente/servidor facilitan el movimiento hacia los sistemas abiertos. Utilizando este tipo de entornos, las organizaciones cambian sus viejos equipos por nuevas máquinas que pueden conectar a la red de clientes y servidores. Los suministradores, por su parte, basan uno de los puntos clave de sus herramientas cliente/servidor en la interoperabilidad.

Relación con Orientación a Objetos

No hay una única forma de programar aplicaciones cliente/servidor; sin embargo, para un desarrollo rápido de aplicaciones cliente/servidor y para obtener una reducción importante de costes, la utilización de la tecnología cliente/servidor puede considerarse en conjunción con la de orientación a objetos.

CARACTERÍSTICAS:

Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes:

· El servidor presenta a todos sus clientes una interfaz única y bien definida.

· El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

· El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

· Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Todos los sistemas desarrollados en arquitectura Cliente/Servidor poseen las siguientes características distintivas de otras formas de software distribuido:

· Servicio. El servidor es un proveedor de servicios; el cliente es un consumidor de servicios.

· Recursos compartidos. Un servidor puede atender a muchos clientes al mismo tiempo y regular su acceso a recursos compartidos.

· Protocolos Asimétricos. La relación entre cliente y servidor es de muchos a uno; los clientes solicitan servicios, mientras los servidores esperan las solicitudes pasivamente.

· Transparencia de ubicación. El software Cliente/Servidor siempre oculta a los clientes la ubicación el servidor.

· Mezcla e igualdad. El software es independiente del hardware o de las plataformas de software del sistema operativo; se puede tener las mismas o diferentes plataformas de cliente y servidor.

· Intercambio basados en mensajes. Los sistemas interactúan a través de un mecanismo de transmisión de mensajes: la entrega de solicitudes y respuestas del servicio.

· Encapsulamiento de servicios. Los servidores pueden ser sustituidos sin afectar a los clientes, siempre y cuando la interfaz para recibir peticiones y ofrecer servicios no cambie.

· Facilidad de escalabilidad. Los sistemas Cliente/Servidor pueden escalarse horizontal o verticalmente. Es decir, se pueden adicionar o eliminar clientes (con apenas un ligero impacto en el desempeño del sistema); o bien, se puede cambiar a un servidor más grande o a servidores múltiples.

· Integridad. El código y los datos del servidor se conservan centralmente; esto implica menor costo de mantenimiento y protección de la integridad de los datos compartidos. Además, los clientes mantienen su individualidad e independencia.

· Interoperabilidad. Capacidad de los equipos de informática de diferentes fabricantes para comunicarse entre sí en una red.

La arquitectura Cliente/Servidor es una infraestructura versátil modular y basada en mensajes que pretende mejorar la portabilidad, la interoperabilidad y la escalabilidad del cómputo; además es una apertura del ramo que invita a participar a una variedad de plataformas, hardware y software del sistema.

- ***Sistemas operativos distribuidos: sistema de archivos distribuidos, memoria distribuida, balanceo de carga***

Sistemas distribuidos

Antes de hablar de sistemas operativos distribuidos primero hay que hablar un poco acerca de sistemas distribuidos. No hay un consenso en la literatura acerca de la definición de un sistema distribuido. Sin embargo, lo que es aceptado por todos son dos aspectos que lo caracterizan:

- Esta formado por un conjunto de máquinas autónomas conectadas entre sí.
- Los usuarios perciben el sistema como si fuera un solo computador.
- El surgimiento de los sistemas distribuidos fue posible gracias a la invención de las redes de computadores junto con el desarrollo tecnológico de enlaces de comunicación de alta velocidad. Esto permitió la construcción de las redes tipo LAN y WAN.

Dentro de las principales motivaciones que conlleva a la construcción de sistemas distribuidos se encuentran:

- Economía.
- Velocidad
- Distribución Inherente
- Confiabilidad
- Crecimiento incremental o escalabilidad

Estas motivaciones surgen de un análisis comparativo entre la utilización de sistemas centralizados ver sus la utilización de sistemas distribuidos. Un sistema centralizado se entiende como un sistema que posee una sola CPU con sus respectivos recursos al cual se le pueden conectar terminales. Por otro lado, el uso de sistemas distribuidos frente al uso de computadores autónomos trabajando aisladamente presenta otro conjunto de ventajas:

- Información compartida
- Dispositivos compartidos
- Facilidad de comunicación
- Flexibilidad

Hasta el momento solo hemos hablado de las maravillas que proporcionan los sistemas distribuidos ahora mencionaremos cuales son sus principales desventajas.

- Software
- Redes
- Seguridad
- Aspectos del Hardware

Desde el punto de vista del Hardware presente en un sistema distribuido debemos considerar las diversas formas en que es posible interconectar varios computadores o bien varias CPUs. Desde esta perspectiva existe una taxonomía general propuesta por Flyn (1972). Flyn propuso 4 diferentes categorías en que se podían clasificar los sistemas hardware existentes de acuerdo a dos parámetros. Número de flujos de instrucciones y número de flujos de datos. La siguiente es la clasificación que propuso.

SISD. Un flujo de instrucción con un flujo de datos.
SIMD. Un flujo de instrucción varios flujos de datos.
MISD. Varios flujos de instrucciones un flujo de datos. (No se usa)
MIMD. Varios flujos de instrucciones y varios flujos de datos.

De acuerdo a estas categorías un sistema distribuido se encuentra dentro de la última mencionada. Y en particular dentro de esta categoría se pueden encontrar dos tipos distintos de la forma en que se puede interconectar el hardware. Es así, como tenemos los siguientes grandes grupos. Dentro de cada uno de ellos se distinguen dos maneras distintas de interconexión.

- Multiprocesadores.
- Multicomputadores.

Cada uno de estos tipos, permite una interconexión de sus componentes tipo bus y tipo conmutada. Desde el punto de vista de las velocidades de comunicación que pueden alcanzar estos dos grandes grupos se tienen dos conceptos asociados: sistemas fuertemente acoplados y sistemas débilmente acoplados

Multiprocesador

Los multiprocesadores corresponden a un conjunto de CPUs conectadas entre sí que utilizan un espacio de direccionamiento virtual común.

Multiprocesadores con conexión de bus

En este caso las CPUs están interconectadas entre sí, mediante un bus. Cada vez que una CPU quiere realizar un acceso de lectura o escritura debe accesar el bus. Con la existencia de un único bus compartido por múltiples CPUs, se puede producir un problema de rendimiento en el sentido que no es posible que mas de una CPU accese el canal de comunicaciones simultáneamente. Este problema por lo general se ataca mediante la utilización de memoria caché. Sin embargo, el uso de memoria caché también produce otros problemas relacionados con la coherencia de la información. Para solucionar este último problema normalmente se utilizan caches monitores.

Multiprocesadores con conexión conmutada

En este caso, la memoria se divide en diversos bancos, y las CPUs se interconectan con ellas no mediante un canal tipo bus sino de otras manera, una de ellas se le conoce como conexión crossbar y la otra como red omega. En el primer caso, en cada vértice de la matriz formada se ubica un conmutador, los cuales se cierran de acuerdo a los requerimientos de acceso entre CPU y banco de Memoria. La red Omega en cambio contiene conmutadores 2x2, cada uno de los cuales tiene dos entradas y dos salidas. Los principales problemas que enfrentan los multiprocesadores están limitados en la cantidad de CPUs que puede interconectar, los de conexión bus por una parte no puede recargar mucho el uso del canal; y los de conexión mediante conmutadores permiten mas CPUs, pero son mas lentos y caros.

Multicomputadores

Cuando se habla de multicomputadores, es para referirse a sistemas de cómputo con memoria distribuida, en este caso cada computador posee su propia memoria local. Por lo tanto, se comunican con los otros computadores mediante algún canal que los conecte usando primitivas de comunicación.

Multicomputadores con conexión de bus

Este esquema es similar al caso de los multiprocesadores con conexión tipo bus, sólo que no se requiere que el canal de comunicación sea tan alto como es en el caso de los multiprocesadores. Por lo general este esquema corresponde a una LAN conectada con velocidades de 10M, 100Mb/seg. Claro que si el canal ofrece mayor velocidad mejor es el desempeño del sistema.

Aspectos del Software

El software, por su naturaleza es difícil de clasificarlo, a diferencia del hardware. Sin embargo, es posible hacer una clasificación mas bien cualitativa. Se dice que el software puede ser fuertemente o débilmente acoplado, de acuerdo al grado de interacción distribuida que permita. Desde el punto de vista del software los siguientes son los mas

conocidos.

- Sistemas operativos de redes
- Sistemas distribuidos reales
- Sistemas de multiprocesadores con tiempo compartido
- Sistemas operativos de redes

Este caso de software se dice que es débilmente acoplado en un hardware también débilmente acoplado. Este esquema corresponde a una LAN en donde cada computador opera en forma autónoma posee su propio sistema operativo. Sin embargo, este esquema permite la comunicación e interacción entre los distintos computadores mediante algunos comandos de red. Un ejemplo de esto, es el comando rlogin de Unix, el cual permite abrir una sesión en una máquina remota, antes de hacerse efectiva la conexión remota se realiza la desconexión de la máquina original, este comando permite usar la máquina original como terminal de la máquina remota. Otro comando de Unix similar al rlogin es el rcp, el cual permite copiar un archivo de una máquina a otra.

Sistemas distribuidos reales

Este esquema corresponde a un sistema de software fuertemente acoplado en un sistema de hardware débilmente acoplado. En este caso, el software utiliza los recursos del hardware distribuido de la misma manera como un sistema operativo convencional utiliza los recursos del computador donde es montado. Esta perspectiva es percibida por los usuarios como si la red de computadores fuera un solo sistema de tiempo compartido. Es en este sentido que se habla de sistemas operativos distribuidos. Un sistema operativo distribuido contempla varias funciones, entre las que se encuentran:

- Mecanismos de comunicación global entre procesos
- Mecanismos de protección global
- Interfaz de llamadas a sistema homogénea
- Distribución adecuada de los procesos en el sistema
- Sistemas de multiprocesadores con tiempo compartido

En este caso se tiene un software fuertemente acoplado con un hardware fuertemente acoplado. Este esquema contempla un sistema operativo para una máquina que contienen múltiples procesadores trabajando con memoria compartida. Por lo general, cada CPU tiene una memoria caché. Una de las principales características de estos sistemas es en cuanto a la ejecución de proceso existe una cola de procesos listos, que pueden ser atendidos por cualquier CPU, respecto a este punto es necesario garantizar el acceso adecuado a dicha cola. Además el uso de cachés, puede influir en la decisión del planificador respecto a la CPU que sería mejor ejecutase un proceso determinado. El sistema operativo, en este caso, puede considerar la posibilidad de realizar espera ocupada cuando se efectúa una operación de E/S, a diferencia de un sistema operativo que opera sobre un solo procesador en donde no se realiza espera ocupada en el caso de una operación de E/S.

Aspectos de diseño

Dentro de los principales aspectos que se deben considerar a la hora de diseñar un sistema operativo distribuido se encuentran:

- Transparencia.
- Flexibilidad.
- Confiabilidad.
- Desempeño.
- Escalabilidad.
- Transparencia

La transparencia es un concepto que esta directamente relacionado con las acciones que realiza el sistema para ocultar al usuario y al programador que dicho sistema consiste en un conjunto de procesadores (o máquinas) que operan en tiempo compartido. Es decir, los usuarios, por ejemplo no necesitan saber si están conectados en la máquina X o en la Y, luego un comando del tipo rlogin no corresponde a un esquema de transparencia. El concepto de transparencia puede clasificarse en varios tipos diferentes, los cuales se presentan a continuación:

- Transparencia de localización.
- Transparencia de migración.
- Transparencia de copia o réplica.
- Transparencia de concurrencia.
- Transparencia de paralelismo.

La transparencia, también puede no ser requerida siempre, por ejemplo, un usuario puede en algunos casos preferir

imprimir en una impresora particularmente (porque le queda mas cerca, por ejemplo).

Flexibilidad

Este aspecto tiene relación con que tan flexible puede ser un sistema. Dentro de este aspecto existen dos grandes distintas líneas de diseño. Entre ellas se tiene:

- Sistemas Monolíticos.
- Sistemas de Microkernel.

UNIX como sistema operativo no distribuido es un sistema monolítico, y los sistemas operativos distribuidos que se han construido en base a él han adoptado la misma filosofía. En cambio los sistemas operativos diseñados a partir de cero, de preferencia, han sido diseñados como sistemas de microkernel. La tendencia en el diseño de sistemas operativos distribuidos es la utilización de la visión de microkernel. El sistema Amoeba, es de tipo microkernel y el Sprite es monolítico.

Confiabilidad

La confiabilidad es un aspecto muy importante que debe considerarse en el diseño de sistemas distribuidos. Se supone que si el sistema distribuido esta compuesto por un conjunto de recursos que cooperan entre sí, entonces si un componente del sistema se descompone otro componente podría reemplazarlo. Un aspecto muy relacionado con la confiabilidad respecto a los descrito tiene que ver con la Tolerancia a fallas. Este aspecto suena mucho en el ambiente de investigación. Otro aspecto importante de mencionar respecto a la confiabilidad es la disponibilidad, este aspecto tiene que ver con la fracción de tiempo en que se puede utilizar el sistema. Unos de los mecanismos utilizados para aumentar la disponibilidad del sistema es la redundancia. Sin embargo, este mecanismo debe ser controlado muy bien para no caer en inconsistencias. La seguridad es otro aspecto clave relacionado con la confiabilidad, pues deben existir los mecanismos de protección adecuados para el control de la información de los usuarios y del propio sistema. En un sistema distribuido la seguridad es mas vulnerable que un sistema centralizado o de conmutadores aislados o en red.

Desempeño

El desempeño es una motivación muy importante que conlleva el uso de sistemas distribuidos. El desempeño puede ser evaluado en base al tiempo de respuesta, al grado de flexibilidad, al grado de confiabilidad, etc. Sin embargo, el desempeño depende mucho de las capacidades de los canales de comunicación que interconectan a los diversos componentes del sistema. En un sistema multiprocesador, las velocidades de comunicación son mas altas respecto a las que se manejan en una LAN. Sin embargo, el desarrollo de tecnologías de comunicación de altas velocidades están apareciendo con gran fuerza en los últimos tiempos, como es el caso de ATM.

Escalabilidad

La escalabilidad tiene relación con la proyección del crecimiento de las redes, en donde día a día se incorporan nuevos servicios y nuevos usuarios que requieren los servicios. La escalabilidad tienen relación con la perspectiva que debe tener el sistema a dichos incrementos.

Ventajas de los Sistemas Distribuidos

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

Economía: El cociente precio/desempeño de la suma del poder de los procesadores separados contra el poder de uno solo centralizado es mejor cuando están distribuidos.

Velocidad: Relacionado con el punto anterior, la velocidad sumada es muy superior.

Confiabilidad: Si una sola máquina falla, el sistema total sigue funcionando.

Crecimiento: El poder total del sistema puede irse incrementando al añadir pequeños sistemas, lo cual es mucho más difícil en un sistema centralizado y caro.

Distribución: Algunas aplicaciones requieren de por sí una distribución física.

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

Compartir datos: Un sistema distribuido permite compartir datos más fácilmente que los sistemas aislados, que tendrían que duplicarlos en cada nodo para lograrlo.

Compartir dispositivos: Un sistema distribuido permite accesar dispositivos desde cualquier nodo en forma transparente, lo cual es imposible con los sistemas aislados. El sistema distribuido logra un efecto sinergético.

Comunicaciones: La comunicación persona a persona es factible en los sistemas distribuidos, en los sistemas aislados no. _ Flexibilidad: La distribución de las cargas de trabajo es factible en el sistema distribuidos, se puede incrementar el poder de cómputo.

Desventajas de los Sistemas Distribuidos
Así como los sistemas distribuidos exhiben grandes ventajas, también se pueden identificar algunas desventajas, algunas de ellas tan serias que han frenado la producción comercial de sistemas operativos en la actualidad. El problema más importante en la creación de sistemas distribuidos es el software: los problemas de compartición de datos y recursos es tan complejo que los mecanismos de solución generan mucha sobrecarga al sistema haciéndolo ineficiente. El checar, por ejemplo, quiénes tienen acceso a algunos recursos y quiénes no, el aplicar los mecanismos de protección y registro de permisos consume demasiados recursos. En general, las soluciones presentes para estos problemas están aún en pañales. Otros problemas de los sistemas operativos distribuidos surgen debido a la concurrencia y al paralelismo. Tradicionalmente las aplicaciones son creadas para computadoras que ejecutan secuencialmente, de manera que el identificar secciones de código `paralelizable' es un trabajo arduo, pero necesario para dividir un proceso grande en sub-procesos y enviarlos a diferentes unidades de procesamiento para lograr la distribución. Con la concurrencia se deben implantar mecanismos para evitar las condiciones de competencia, las postergaciones indefinidas, el ocupar un recurso y estar esperando otro, las condiciones de espera circulares y , finalmente, los "abrazos mortales" (deadlocks). Estos problemas de por sí se presentan en los sistemas operativos multiusuarios o multitareas, y su tratamiento en los sistemas distribuidos es aún más complejo, y por lo tanto, necesitará de algoritmos más complejos con la inherente sobrecarga esperada.

- ***Sistemas en tiempo real.***

Un sistema operativo en tiempo real procesa las instrucciones recibidas al instante, y una vez que han sido procesadas muestra el resultado. Este tipo tiene relación con los sistemas operativos monousuarios, ya que existe un solo operador y no necesita compartir el procesador entre varias solicitudes. Su característica principal es dar respuestas rápidas; por ejemplo en un caso de peligro se necesitarían respuestas inmediatas para evitar una catástrofe.

- ***Sistemas para computadoras paralelas (memoria compartida, memoria distribuida, memoria distribuida/compartida)***

Tiempo Compartido.
El tiempo compartido en ordenadores o computadoras consiste en el uso de un sistema por más de una persona al mismo tiempo. El tiempo compartido ejecuta programas separados de forma concurrente, intercambiando porciones de tiempo asignadas a cada programa (usuario). En este aspecto, es similar a la capacidad de multitareas que es común en la mayoría de los microordenadores o las microcomputadoras. Sin embargo el tiempo compartido se asocia generalmente con el acceso de varios usuarios a computadoras más grandes y a organizaciones de servicios, mientras que la multitarea relacionada con las microcomputadoras implica la realización de múltiples tareas por un solo usuario.

C. UTILERÍAS Y MANEJADORES

I. Orientados al usuario

1. Sistemas de respaldo y recuperación

- ***Compactación y descompactación***

Compactación. La creación de áreas libres (huecos) entre particiones se denomina fragmentación externa, y aparece habitualmente en sistemas con asignación dinámica de bloques de memoria de tamaño variable. Si la memoria resulta seriamente fragmentada, la única salida posible es reubicar algunos o todas las particiones en un extrema de la memoria a así combinar los huecos para formar una única área libre grande, a esta solución se le llama *compactación.*

- ***Respaldos increméntales, periodicidad y confiabilidad***

Respaldos increméntales.- La forma mas simple de estos vaciados es mediante un vaciado completo en forma periódica, por ejemplo, una vez al mes o a la semana y hacer un vaciado diario solo de los archivos modificados desde el ultimo vaciado total. Para la implementación de este método, debe mantenerse en el disco una lista de los tiempos de vaciado de cada archivo. El archivo de vaciado verifica entonces cada uno de los archivos del disco. Si un archivo fue modificado desde su ultimo vaciado, se le vacía de nuevo y su tiempo del ultimo vaciado cambia por el

tiempo actual. MS-Dos ayuda un poco en la realización de respaldar. A cada archivo se le asocia un bit de atributo llamado bit de biblioteca. Al hacer un respaldo del sistema de archivos, los bit de biblioteca de todos los archivos toman el valor de 0. Después, cuando se modifica un archivo, el sistema activa en forma automática su bit de biblioteca. Cuando es tiempo del siguiente respaldo, el programa de respaldo revisa todos los bits de biblioteca y solo respalda aquellos archivos cuyos bits de biblioteca están activos. También limpia todos esos bits para hacer una revisión posterior del uso de los archivos.

- ***Herramientas para reparación y recuperación***

Una de las herramientas de recuperación son las bitácoras, esta técnica nos permite garantizar la disponibilidad continua de los datos. Esto es posible creando un archivo de bitácora de todas las transacciones copiándolas en otro disco. Esta redundancia puede resultar costosa, pero en caso de fallos es posible reconstruir todo el trabajo perdido. Otra herramienta de recuperación son los respaldos increméntales, que mencionamos anteriormente.

Tratamiento de virus

Prevención, detección y erradicación

Las medidas preventivas mas comunes incluyen el filtrado preventivo de todo software de recién adquisición, las copias de respaldo frecuentes y una combinación de utilidades tales como comprobadores de integridad, programas de vigilancia y supresores de virus.

a) Comprobadores de integridad.- Son programas que intentan detectar modificaciones en otros programas, mediante el calculo y almacenamiento de sumas de chequeo en los programas ejecutables y en los archivos de ordenes. Se pretende que sean ejecutados con bastante frecuencia, por ejemplo en cada arranque. Puesto que la infección del virus modifica el tamaño del programa afectado.
b) Programas supervisores.- Son programas permanentemente residentes (demonios) que comprueban continuamente ciertas operaciones de memoria y de E/S para detectar comportamientos sospechosos que puedan ser atribuibles al virus. Los programas de vigilancia son rutinas adicionales al S.O que tratan de sellar los agujeros de seguridad conocidos o que puedan ser probablemente explorados por virus. Pueden ser capaces de detectar algunas formas de infección virica antes de que alcancen la etapa de activación.
c) Los supresores de virus.- Son programas que examinan el almacenamiento del sistema buscando firmas y patrones que pertenecen a un virus conocido por el autor del programa. Pueden detectar y eliminar virus conocidos incluso en la etapa de letargo. Los supresores de virus no pueden tratar a virus desconocidos, estos se deben actualizar frecuentemente con el fin de adaptar su trabajo a los nuevos tipos de virus.

2. Tratamiento de virus

- ***Prevención, detección y erradicación***

a. Generalidades

Antes de profundizar en este tema, debemos aclarar que los virus de computadoras son simplemente programas, y como tales hechos por programadores. Son programas que debido a sus características particulares son especiales. Para hacer un virus de computadora no se requiere capacitación especial, ni una genialidad significativa, sino conocimientos de lenguajes de programación para el público en general y algunos conocimientos puntuales sobre el ambiente de programación y arquitectura de las PC's. Nuestro trabajo capta el problema del virus, desde el punto de vista funcional. En la vida diaria, cuando un programa invade inadvertidamente el sistema, se replica sin conocimiento del usuario y produce daños, pérdida de información o fallas del sistema, reconocemos que existe un virus. Los virus actúan enmarcados por "debajo" del sistema operativo, como regla general, y para actuar sobre los periféricos del sistema, tales como disco rígido, disketteras, ZIP's CD-R's, hacen uso de sus propias rutinas aunque no exclusivamente. Un programa normal, por llamarlo así, utiliza las rutinas del sistema operativo para acceder al control de los periféricos del sistema, y eso hace que el usuario sepa exactamente las operaciones que realiza, teniendo control sobre ellas. Los virus, por el contrario, para ocultarse a los ojos del usuario, tienen sus propias rutinas para conectarse con los periféricos de la computadora, lo que les garantiza cierto grado de inmunidad a los ojos del usuario, que no advierte su presencia, ya que el sistema operativo no refleja su actividad en la PC. Una de las principales bases del poder destructivo de estos programas radica en el uso de funciones de manera "sigilosa", oculta a los ojos del usuario común. El virus, por tratarse de un programa, para su activación debe ser ejecutado y funcionar dentro del sistema al menos una vez. Demás esta decir, que los virus no surgen de las computadoras espontáneamente, sino que ingresan al sistema inadvertidamente para el usuario, y al ser ejecutados, se activan y actúan con la computadora huésped.

b. Definiciones

1. “Un virus es simplemente un programa. Una secuencia de instrucciones y rutinas creadas con el único objetivo de alterar el correcto funcionamiento del sistema y, en la inmensa mayoría de los casos, corromper o destruir parte o la totalidad de los datos almacenados en el disco.”
2. Un virus es una porción de código ejecutable, que tiene la habilidad única de reproducirse. Se adhieren a cualquier tipo de archivo y se diseminan con los archivos que se copian y envían de persona a persona. Además de reproducirse, algunos virus informáticos tienen algo en común: una rutina dañina, que el virus descarga como una bomba, mientras que las descargas pueden ser simples mensajes o imágenes, éstas también pueden borrar archivos, reformatear el disco duro o causar otro tipo de daño. Si el virus no contiene una rutina dañina, aún puede causar problemas, como tomar espacio libre del disco y de la memoria, y también disminuir el rendimiento de la computadora. Los virus de las computadoras no son más que programas; y estos virus casi siempre los acarrean las copias ilegales o piratas. Provocan desde la pérdida de datos o archivos en los medios de almacenamiento de información (diskette, disco duro, cinta), hasta daños al sistema y, algunas veces, incluyen instrucciones que pueden ocasionar daños al equipo.

c. Características

Hay que recordar que un virus no puede ejecutarse por sí solo, pues necesita un programa portador para poder cargarse en memoria e infectar; asimismo, para poder unirse en un programa portador, el virus precisa modificar la estructura de aquél, posibilitando que durante su ejecución pueda realizar una llamada al código del virus. Las particularidades de los virus:

- Son muy pequeños.
- Casi nunca incluyen el nombre del autor, ni el registro o copyright, ni la fecha de creación.
- Se reproducen a sí mismo.
- Toman el control o modifican otros programas.
- Es dañino: El daño es implícito, busca destruir o alterar, como el consumo de memoria principal y tiempo de procesador.
- Es autorreproductor: A nuestro parecer la característica más importante de este tipo de programas es la de crear copias de sí mismo.
- Es subrepticio: Esto significa que utilizará varias técnicas para evitar que el usuario se de cuenta de su presencia.

d. ¿Quiénes hacen los virus?

Los virus informáticos son hechos por personas con conocimiento de programación, pero que no son necesariamente genios de las computadoras. Tienen conocimiento de lenguaje ensamblador y de cómo funciona internamente la computadora. A diferencia de los virus que causan resfriados y enfermedades en humanos, los virus computacionales no ocurren de forma natural, cada uno es programado. No existen virus benéficos. Algunas veces son escritos como una broma, desplegando un mensaje humorístico. En estos casos, el virus no es más que una molestia. Muchas veces son creados por personas que se sienten aburridas, con coraje, como reto intelectual; cualquiera que sea el motivo, los efectos pueden ser devastadores.

e. Síntomas Más Comunes de Virus

La mejor forma de detectar un virus es, obviamente un antivirus, pero en ocasiones los antivirus pueden fallar en la detección. Puede ser que el escaneo no detecte nada y sí el análisis huerístico. Puede ser que no detectemos nada y aún seguir con problemas. En estos casos debemos notar algunos síntomas posibles:

- Los programas comienzan a ocupar más espacio de lo habitual. Se reduce el espacio libre en la memoria RAM. El virus al entrar en el sistema, se sitúa en al memoria RAM, ocupando una porción de ella. El tamaño útil y operativo de la memoria se reduce en la misma cuantía que tiene el código del virus. Siempre en el análisis de una posible infección es muy valioso contar con parámetros de comparación antes y después de la posible infección. Por razones prácticas casi nadie analiza detalladamente su PC en condiciones normales y por ello casi nunca se cuentan con patrones antes de una infección, pero sí es posible analizar estos patrones al arrancar una PC en la posible infección y analizar la memoria arrancando el sistema desde un disco libre de infección.
- Aparecen o desaparecen archivos. En mayor o menor medida, todos los virus, al igual que programas residentes comunes, tienen una tendencia a “colisionar” con otras aplicaciones, lo que provoca también aparición de mensajes de error no comunes.
- Cambia el tamaño de un programa o un objeto. Programas que normalmente funcionaban bien, comienzan a fallar y generar errores durante la sesión.
- Aparecen mensajes u objetos extraños en la pantalla. El código viral, ocupa parte de la RAM y debe quedar “colgado” de la memoria para activarse cuando sea necesario. Esa porción de código que queda en RAM, se llama residente y con algún utilitario que analice el RAM puede ser descubierto.

• El disco trabaja más de lo necesario. Tiempos de cargas mayores y es debido al enlenticimiento global del sistema, en el cual todas las operaciones se demoran más de lo habitual.
• Los objetos que se encuentran en la pantalla aparecen ligeramente distorsionados. Las operaciones se realizan con más lentitud, ya que los virus son programas y como tales requieren de recursos del sistema para funcionar y su ejecución al ser repetitiva, lleva a un enlentecimiento y distorsión global en las operaciones.
• Se modifican sin razón aparente el nombre de los ficheros.
• No se puede acceder al disco duro.

f. Clasificación

A continuación esbozamos una clasificación que tiende a catalogar los virus actuales, sin intentar crear una clasificación académica, sino una orientación en cuanto a funcionalidad de los virus:
• Virus de Macros/Código Fuente: Se adjuntan a los programas fuente de los usuarios y, a las macros utilizadas por: Procesadores de Palabras (Word, Works, WordPerfect), Hoja de Cálculo (Excell, Quattro, Lotus).
• Virus Mutantes: Son los que al infectar realizan modificaciones a su código, para evitar ser detectados o eliminados (NATAS o SATÁN, Miguel Angel, por mencionar algunos).
• Gusanos: Son programas que se reproducen a sí mismo y no requieren de un anfitrión, pues se "arrastran" por todo el sistema sin necesidad de un programa que los transporte. Los gusanos se cargan en la memoria y se posesionan en una determinada dirección, luego se copian en otro lugar y se borran del que ocupaban, y así sucesivamente. Esto hace que queden borradas los programas o la información que encuentran a su paso por la memoria, lo que causa problemas de operación o pérdidas de datos.
• Caballos de Troya: Son aquellos que se introducen al sistema bajo una apariencia totalmente diferente a la de su objetivo final; esto es, que se presentan como información perdida o "basura", sin ningún sentido. Pero al cabo de algún tiempo, y esperando la indicación programada, "despiertan" y comienzan a ejecutarse y a mostrar sus verdaderas intenciones.
• Bomba de Tiempo: Son los programas ocultos en la memoria del sistema o en los discos, o en los archivos de programas ejecutables con tipo COM o EXE. En espera de una fecha o una hora determinadas para "explotar". Algunos de estos virus no son destructivos y solo exhiben mensajes en las pantallas al llegar el momento de la "explosión". Llegado el momento, se activan cuando se ejecuta el programa que las contiene.
• Autorreplicables: Son los virus que realizan las funciones más parecidas a los virus biológicos, ya que se autoreproducen e infectan los programas ejecutables que se encuentran en el disco. Se activan en una fecha u hora programadas o cada determinado tiempo, contado a partir de su última ejecución, o simplemente al "sentir" que se les trata de detectar. Un ejemplo de estos es el virus del viernes 13, que se ejecuta en esa fecha o se borra (junto con los programas infectados), evitando así ser detectado.
• Infectores del área de carga inicial: Infectan los diskettes o el disco duro, alojándose inmediatamente en el área de carga. Toman el control cuando se enciende la computadora y lo conservan todo el tiempo.
• Infectores del sistema: Se introducen en los programas del sistema, por ejemplo COMMAND.COM y otros se alojan como residentes en memoria. Los comandos del Sistema Operativo, como COPY, DIR o DEL, son programas que se introducen en la memoria al cargar el Sistema Operativo y es así como el virus adquiere el control para infectar todo disco que sea introducido a la unidad con la finalidad de copiarlo o simplemente para ver sus carpetas (también llamadas: folders, subdirectorios, directorios).
• Infectores de programas ejecutables: Estos son los virus más peligrosos porque se diseminan fácilmente hacia cualquier programa (como hojas de cálculo, juegos, procesadores de palabras). La infección se realiza al ejecutar el programa que contiene al virus, que en ese momento se posesiona en la memoria de la computadora y a partir de entonces infectará todos los programas cuyo tipo sea EXE o COM, en el instante de ejecutarlos, para invadirlos autocopiándose en ellos.

Todos estos programas tienen en común la creación de efectos perniciosos, sin embargo, no todos pueden ser considerados como virus propiamente dichos. La barrera entre virus puros y el resto de programas malignos es muy difusa, prácticamente invisible, puesto que ya casi todos los virus incorporan características propias de uno o de varios de estos programas.

g. Ciclo de Infección

Como mencionamos con anterioridad, para que un virus se active en memoria, se debe ejecutar el programa infectado en primer término, para que el virus inicie sus actividades dentro de nuestro sistema. En este caso, no es necesario arrancar ningún programa, sino simplemente abrir un archivo de Word o Excel infectado.

El ciclo completo de infección de un Macro Virus sería así:
• Se abre el archivo infectado, con lo cual se activa en memoria.
• Infecta sin que el usuario se dé cuanta al NORMAL.DOT, con eso se asegura que el usuario sea un reproductor del virus sin sospecharlo.

• Si está programado para eso, busca dentro de la PC los archivos de Word, Excel, etc. que puedan ser infectados y los infecta.
• Si está programado, verifica un evento de activación, que puede ser una fecha, y genera el problema dentro de la pc (borrar archivos, destruir información, etc.).
• Ahora bien, en el caso de mails vía internet. Los mails no son programas. Algunos no poseen macros (los que sí poseen macros son los mails de Microsoft Outlook). Aquellos que no tienen lenguaje de macros (NO PUEDEN CONTENER VIRUS).
• Los archivos adjuntos asociados al mail pueden llevar virus (siempre que sean susceptibles de ser infectados). Bajen el adjunto, y verifíquenlo. Asegúrense que el antivirus chequee los zipeados o comprimidos si lo adjuntado es un archivo de ese tipo. Si el adjunto es un documento que puede tener macros, desactiven las macros del programa Word antes de abrirlo. Si el adjunto es un archivo de texto plano, pueden quedarse tranquilos.

h. Medidas de Protección Efectivas

Obviamente, la mejor y más efectiva medida es adquirir un antivirus, mantenerlo actualizado y tratar de mantenerse informado sobre las nuevas técnicas de protección y programación de virus. Gracias a internet es posible mantenerse al tanto a través de servicios gratuitos y pagos de información y seguridad. Hay innumerables boletines electrónicos de alerta y seguridad que advierten sobre posibles infecciones de mejor o menor calidad. Existen herramientas indispensables para aquellos que tienen conexiones prolongadas a internet, que tienden a proteger al usuario no sólo detectando posibles intrusos dentro del sistema, sino chequeando constantemente el sistema, a modo de verdaderos escudos de protección.

- ***Reparación de archivos***

Antivirus

a. Generalidades

El programa de antivirus debe ser completo, preciso y rápido. Si no es así, usted simplemente no lo utilizará, y dejará a un lado esta actividad, lo cual es muy peligroso. Por ejemplo, en cualquier mes determinado hay de 200 a 300 virus circulando por el mundo. Esa cifra proviene de la WildList, una lista mensual reconocida internacionalmente, que tiene los virus que se dispersan en el mundo "en estado salvaje". El método principal de un analizador de antivirus para atrapar los virus es comparar un código sospechoso con las bases de datos de conocidas firmas de virus. Estas bases de datos incluyen nombres actuales o anteriores en la WildList, así como decenas de miles "zoovirus", que en su mayoría existen en laboratorios, pero que utilizan trucos que pueden ser empleados por futuros virus. Con el propósito de adquirir un buen antivirus, lo primero es verificar el tipo de tecnología aplicada en su desarrollo, actualmente los antivirus utilizan dos técnicas de verificación:
• La conocida técnica de escaneo, consistente en tener una gran base de datos con fragmentos víricos para comparar los archivos con esa inmensa biblioteca del wildlist.
• La tecnología heurística es fundamental en estos momentos, y en mi opinión, los antivirus han de ofrecer como alternativa al escaneo común (aún necesario) la búsqueda heurística. Esta técnica permite detectar virus que aún no estén en la base de datos scaning, y es muy útil cuando padecemos la infección de un virus que aún no ha sido estudiado ni incorporado a los programas antivirus.

b. Definición de Antivirus

• Es el programa que se encarga de analizar el contenido de los ficheros y, en caso
de detectar un virus en su interior, proceder a su desinfección. También realizan búsquedas heurísticas, esto es, buscar funciones que puedan resultar nocivas para tu ordenador, sin que sean virus reconocidos.
• Es una aplicación o programa dedicada a detectar y eliminar virus informáticos. La forma en que protege es la siguiente, el sistema de protección del Antivirus depende del sistema operativo en que se esté trabajando. En DOS se utiliza TSR (Terminate and Stay Resident, programas que terminan y se quedan residentes en memoria), en Windows 95/98 VxD (Virtual Driver) y en NT drivers en modo Kernel. Por término general se puede pensar en un programa que vigila todos y cada uno de los accesos que se realizan a ficheros y discos y antes de autorizarlos avisa de la existencia virus y, en su caso, desinfecta el fichero en cuestión. Si eres muy cuidadoso con los programas que utilizas, la información que introduces a tu computadora y con los lugares que visitas en el Internet, así como intercambiar tus discos en el trabajo o discos de amigos (duda procedencia) es muy posible que nunca tengas problemas con virus, lo que sí, es indispensable que tengas instalado un buen Antivirus.

c. Los Antivirus Más Buscados

Actualmente, los virus no sólo son más potentes que sus predecesores, sino que se propagan más rápido. En los años 80, los virus del sector de arranque se multiplicaban a través del intercambio de discos flexibles. A finales de los 90, el correo electrónico era quien transportaba virus de macros que venían en documentos anexos de Microsoft Word. Ahora el peligro viene principalmente de los gusanos de envíos masivos, se autoreplican y son capaces de

secuestrar los libros de direcciones de correo electrónico y autoenviarse a múltiples destinatarios. Por ejemplo, LoveLetter era un virus de guión en Visual Basic. Hoy, la mayoría de los gusanos de correo masivo son programas Win32 independientes, como en el caso de SirCam y Klez. Estos programas son lo peor de todas las infecciones de virus. Por su parte, los virus de los macros están en un distante segundo lugar y los de guión vienen pegados en un tercer lugar. Ahora los virus del sector arranque sólo representan cerca del 1% de las infecciones. Al elegir un antivirus, tomamos en cuenta tres aspectos fundamentales: facilidad de adquisición de las actualizaciones, menor costo posible y facilidad de uso. Atendiendo a estos tres requisitos, recomendamos en el mismo orden Scan de McAffee que es un producto gratuito y se puede conseguir fácilmente en el internet o Norton Antivirus para el cual tendrá que invertir algunos dólares.

d. Antivirus al Rescate

A juzgar por la evaluación de siete productos que llevan a cabo, los fabricantes de antivirus, los mismos han respondido bastante bien ante las amenazas: Estos productos son: Etrust EZ Antivirus 5.4, de Computer Associates; Anti-Virus Personal Por 4, de Kaspersky Lab; McAfee Virus Scan 6.02, de Network Associates; Virus Control 5.2, de Norman; Antivirus Platinum 6.25, de Panda; Norton AntiVirus 2002, de Symantec; y PC-cillin 2002, de Trend Micro. Los productos de Norton, Kaspersky y McAfee fueron los que mejor erradicaron los virus, pero Norton se llevó el premio a la mejor compra gracias a su interfaz intuitiva. Además de la clase de virus que el analizador descubre, también es importante la ubicación de éste, por ejemplo, el protector antivirus debe ser capaz de meterse bien dentro de los archivos zip y otros archivos comprimidos, incluso hasta en los archivos zip que estén ubicados dentro de otros archivos zip. También debe revisar los anexos al correo electrónico, y donde quiera que descubra una infección, debe eliminarla sin destruir archivos valiosos. Kaspersky, McAfee, Norton, Panda y PC-cillin interceptan y analizan los anexos al correo electrónico antes de que lleguen a la unidad de disco duro. Pero Norton y PC-cillin sólo funcionan con programas de correo electrónico que cumplan con los requisitos de POP3, mientras que Kaspersky sólo funciona con los clientes de Outlook, Outlook Express y Exchange, de Microsoft, Panda, a su vez, analiza anexos de POP3, Exchange e incluso de AOL. Cuando estos productos encontraron un virus, la mayoría realizó un buen trabajo al quitarles sin dañar archivos, pero Norton fue el único que lo hizo a la perfección.

e. Conozca Bien su Antivirus

Debido a que en todo momento aparecen nuevos virus, es necesario actualizar con facilidad las definiciones. Todos los programas probados, excepto Etrust, ofrecen actualizaciones automáticas programadas. Sin embargo, nuestro tanto a favor es para Norton, que revisa si hay actualizaciones de manera prefijada, al momento de ser instalado y después, cada 4 horas. Norton también gana puntos por tener la interfaz más lógica, de fácil dominio. En virtud de lo anterior, al hacer una evaluación es importante tratar de verificar hasta que punto los diversos antivirus cumplen con las siguientes características:

1. Deben actualizar los patrones o firmas, por lo menos una vez por semana.
2. La empresa que los promueve debe contar con un equipo de soporte técnico con acceso a un laboratorio especializado en códigos maliciosos y un tiempo de respuesta no mayor a 48 horas, el cual me pueda orientar, en mi idioma, en caso de que yo contraiga una infección.
3. Deben contar con distintos métodos de verificación y análisis de posibles códigos maliciosos, incluyendo el heurístico, el cual no se basa en firmas virales sino en el comportamiento de un archivo, y así poder detener amenazas incluso de posibles virus nuevos.
4. Se deben poder adaptar a las necesidades de diferentes usuarios.
5. Deben poder realizar la instalación remota tanto en una red LAN como en una WAN.
6. Deben constar de alguna consola central en donde se puedan recibir reportes de virus, mandar actualizaciones y personalizar a distintos usuarios.
7. Deben ser verdaderamente efectivos para efectos de detección y eliminación correcta y exacta de los distintos virus que puedan amenazar a los sistemas.
8. Deben de permitir la creación de discos de emergencia o de rescate de una manera clara y satisfactoria.
9. No deben de afectar el rendimiento o desempeño normal de los equipos, y de ser preferible lo que se desea es que su residente en memoria sea de lo más pequeño.
10. El número de falsos positivos que se den tanto en el rastreo normal como en el heurístico debe de ser el mínimo posible.
11. Su mecanismo de auto-protección debe de poder alertar sobre una posible infección a través de las distintas vías de entrada, ya sea Internet, correo electrónico, red o discos flexibles, etc.
12. Deben de tener posibilidad de chequear el arranque, así como los posibles cambios en el registro de las aplicaciones.

En base a estos parámetros, uno mismo puede poner a prueba los distintos productos que hay en el mercado y de acuerdo a nuestras prioridades sacar conclusiones.

f. Importancia del Antivirus

Actualmente, no es difícil suponer que cada vez hay mas personas que están conscientes de la necesidad de hacer uso de algún antivirus como medida de protección básica.

• Desde el punto de vista del administrador, este desea primordialmente tener resultados al problema de administración centralizada. Es decir, desea que un antivirus posea una consola que permita la instalación remota tanto en una red LAN como en una WAN y no verse obligado a instalar el producto a pie en cada una de las estaciones de trabajo.

• Desde el punto de vista del usuario final, a quien le interesa no infectarse por ningún motivo y que la protección en memoria del producto sea de lo más eficaz, tanto para detectar y remover cualquier virus que pueda presentarse.

Basados en estas necesidades, podemos darles los siguientes tips:

f.1. Controles

• Control de acceso físico a los equipos.
• Control de entradas a los programas de la computadora a través de claves de acceso (passwords).
• Registro, verificación y control de los diskettes, cd's que se introducen a la computadora.
• Se recomienda algún programa de tipo menú que restrinja los programas que se pueden ejecutar a sólo los autorizados a cada usuario.

f.2. Bloqueos

• Cerradura para floppies "drive lock"
• Uso del candado o llave de encendido, si la computadora lo tiene.
• Deshabilitar el arranque desde la unidad de diskette.
• Deshabilitar completamente las unidades de diskette.
• Habilitación de la facilidad de palabra clave (password).
• Activar la protección anti-virus en BIOS.

f.3. Diskettes

Estos son puntos muy importantes, ¡prácticamente todos los virus se introducen a una computadora por medio de diskettes! Y en caso de un desastre, las copias de respaldo en diskette serán la salvación de nuestros datos.

• Verificar contra virus todos los diskettes que se introduzcan en la computadora, aunque sólo sean de datos.
• No ejecutar programas de origen dudoso.
• No meter diskettes extraños.
• Nunca arranque desde diskette en la operación normal de su computadora.
• Nunca dejar puestos diskettes al apagar la computadora.
• Tenga un diskette de arranque que esté libre de virus y protegido contra escritura.
• Si es necesario arrancar desde diskette, utilice únicamente este diskette.
• Proteja contra escritura sus discos del sistema, así como sus discos de programas de aplicación.
• Que los usuarios sólo manejen diskettes de datos y nunca de programas.
• Instalar nuevos paquetes en una máquina que sirva de conejillo de Indias y que esté un tiempo en observación.
• Mantener copias respaldo, tanto de los programas, como de los datos.
• Hacer por separado los respaldos de datos y de programas.

f.4. Vacunas Antivirus

• Tener varios programas antivirus, preferentemente con diferentes enfoques.
• Utilizar o activar las diversas opciones de protección.
• Comprar las versiones actualizadas de las vacunas.
• Leer la documentación y manuales de los antivirus.

f.5. Servicios en Línea

• Verificar contra virus todo programa que se transfiera.
• Verificar contra virus todo archivo autodescomprimible (aunque sea de datos).

f.6. OTROS

• Capacitar a los usuarios en protección contra virus.
• Desarrollar un plan de emergencia contra virus que prevea procedimientos o máquinas alternas para el proceso de los datos.
• Mantenerse informado, o sea leer sobre el tema.

II, Orientados al sistema

1. Cargadores y ligadores

- ***Relocalización***

Resolución de direcciones y referencias externas

Un programa llamado cargador realiza las dos funciones de carga y edición de enlace (ligadura). El proceso de carga consiste en tomar el código de máquina relocalizable, modificar las direcciones relocalizables y ubicar las instrucciones y los datos modificados en las posiciones apropiadas de la memoria. El editor de enlace permite formar un solo programa a partir de varios archivo de código maquina relocalizable. Estos archivos pueden haber sido el resultado de varias compilaciones distintas, y uno o varios de ellos pueden ser archivos de biblioteca de rutinas proporcionadas por el sistema y disponibles para cualquier programa que las necesite. Si los archivos se van a usar juntos de manera útil, puede haber algunas referencias externas, en las que el código de un archivo hace referencia a una posición de otro archivo. Esta referencia puede ser a una posición de datos definida en un archivo y utilizada en otro, o puede ser el punto de entrada de un procedimiento que aparece en el código de un archivo y se llama de otro. El archivo con el código de maquina relocalizable debe conservar la información de la tabla de símbolos para cada posición de datos o etiqueta de instrucción a la que se hace referencia externamente.

2. Administración y vigilancia.

- ***Detección de errores físicos***

Una fuente frecuente de errores físicos son los dispositivos periféricos electromecánicos, que proveen entrada o salida a la CPU. Los componentes mecánicos de estos dispositivos periféricos son probables de llevar a cabo un error resultado de una presión de uso. Algunos de estos dispositivos son frágiles, consecuentemente una fuente potencial de errores. El debilitamiento de estos medios puede no ser fatal, porque las unidades periféricas pueden ser incapacitas. Los componentes modernos son diseñados para monitorear su propio desempeño y constantemente realizan un test por ellos mismos, para asegurar que cada operación sea ejecutada correctamente.

- ***Herramientas para diagnóstico***

Se cuenta con dos tipos generales de herramientas de diagnostico asistidas por computadora para ayudar a detectar y aislar problemas que podrían ocurrir en el sistema

1.- Herramienta de diagnostico interno que corre en marcha, la puesta en cero o la autoevaluación.

2.- Un disquete de prueba de diagnostico que corre una serie de programas de prueba desde el disquete.

- ***Puntos de reinicio***

Las características de punto de reinicio/verificación disponibles en muchos sistemas facilitan la suspensión/reanudación solo con la perdida de trabajo desde el punto de verificación (la ultima grabación del estado del sistema). Pero muchos sistemas de aplicación se diseñan sin aprovechar las ventajas de las funciones de reinicio/verificación.