UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

AUTORES: L. I. ERROL ROBERTO FABREGAT TINAJERO

L.I. y MTRA. RITA AURORA FABREGAT TINAJERO

Informática VII (Programación e implementación de sistemas)		Clave: 1866
Plan: 2005		Créditos: 8
Licenciatura: Informática		Semestre: 8º
Área: Informática		Hrs. Asesoría: 2
Requisitos: Informática II (estructura de datos estáticas y dinámicas en memoria principal)		Hrs. Por semana: 4
Tipo de asignatura:	Obligatoria (x)	Optativa ( )

Objetivo general de la asignatura

Al finalizar el curso, el alumno conocerá la definición y forma de construcción del software que hace posible el funcionamiento de las computadoras en diferentes niveles de operación.

Temario oficial (horas sugeridas 64)

1. Introducción (4 h)
2. Lenguajes de programación (16 h)
3. Compiladores (16 h)
4. Sistemas operativos (28 h)

Introducción

El avance en las tecnologías de la información requiere generar sistemas de información de mayor complejidad y por demás eficientes, generando así reducción en tiempos y costos para las empresas; por lo que a lo largo de este material tú encontrarás la ayuda suficiente para comprender la base de un sistema de información y las diversas etapas que lo conforman.

En esta asignatura estudiaremos la definición de los lenguajes de programación más utilizados, su origen y aplicación en el campo de sistemas, diferentes compiladores para esos lenguajes y la definición y estructuración de los sistemas operativos.

TEMA 1. INTRODUCCIÓN

Objetivo particular

Al finalizar el tema, el alumno:

a) reconocerá lo que es un programa de sistema y un lenguaje de programación,
b) diferenciará un intérprete de un compilador, e
c) identificará las características generales de un sistema operativo.

Temario detallado

1.1 Definición de programas de sistema
1.2 Lenguajes de programación
1.3 Intérpretes
1.4 Compiladores
1.5 Sistemas operativos

Introducción

A través del tiempo el uso de los equipos de cómputo y comunicación ha sufrido un cambio considerable debido principalmente a que el software y los programas que han permitido su desarrollo se han visto modificados. En este tema abarcaremos los fundamentos esenciales sobre los lenguajes de programación, intérpretes, compiladores y sistemas operativos con el fin de generalizar los elementos necesarios para el desarrollo de sistemas de información.

1.1 Definición de programas de sistema

Un sistema es un conjunto de elementos que sirven para llevar a cabo alguna tarea. Dentro de estos elementos encontramos al software que son todos aquellos programas que vamos a emplear para llevar a cabo nuestro sistema.

Un Programa puede ser definido como un conjunto de instrucciones organizadas de acuerdo con el lenguaje que se vaya a operar, que permite realizar funciones o tareas específicas.[1]

1.2 Lenguajes de programación

Un lenguaje de programación es un conjunto de reglas, signos y palabras que permite ejecutar programas con funciones o tareas particulares. Los programas para ser entendidos por la computadora se codifican en lenguaje binario (ceros y unos); esta codificación es conocida como lenguaje máquina.

En un inicio los lenguajes de programación se acercaban mucho al lenguaje máquina pero a medida que ha pasado el tiempo, han ido evolucionando y se han transformado en un pseudo-inglés reducido con un abundante aparato formal o bien en elementos gráficos o visuales que han facilitado su uso por parte del programador.

De acuerdo con su nivel de abstracción: “entre más lejano es un lenguaje a la computadora se le denomina de alto nivel, de lo contrario es de bajo nivel”.[2]

Si deseas profundizar en el tema se recomienda que visites la siguiente dirección: http://www.slideshare.net/jrojas/tema1-lenguajes-de-programacion.

---

[1] Joyanes Aguilar, Luis; L. Rodríguez Baena; Matilde Fernández Azuela, *Fundamentos de Programación: Libro de Problemas*. México, McGraw Hill, 1996. p. 10.

[2] Con base en http://foro.latinohack.com/showthread.php?t=17882, consultado el 08/10/09.

Ahora bien, a lo largo de la historia de los lenguajes de programación, se reconocen 5 generaciones, en donde cada una mejora a sus predecesoras en cuanto a las facilidades que otorga al programador.[3]

1. Primera generación: lenguaje máquina.
2. Segunda generación: lenguajes ensambladores.
3. Tercera generación: lenguajes de alto nivel. Ej. C, Pascal, Cobol…
4. Cuarta generación: lenguajes capaces de generar código por sí solos (Desarrollo de Aplicaciones Rápidas -RAD) y los lenguajes orientados a objetos
5. Quinta generación: lenguajes orientados a la inteligencia artificial.

Lenguaje Máquina

El lenguaje máquina, el de más bajo nivel; Consiste en la combinación de 0's y 1's para formar las órdenes entendibles por el hardware. Son mucho más rápidos que un lenguaje de alto nivel pero su desventaja es que generan enormes códigos fuente donde encontrar un error es casi imposible además de requerir un conocimiento profundo de la arquitectura del equipo que se utilice.

Lenguaje Ensamblador

Es un derivado del lenguaje máquina y en lugar de usar ceros y unos, está formado por abreviaturas de letras y números llamadas mnemotécnicos. Para usar un lenguaje ensamblador se requiere un lenguaje traductor, es decir, un lenguaje que permita pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Los lenguajes ensambladores generan códigos fuentes más cortos que los generados en lenguaje máquina pero su desventaja es que la programación sigue siendo tediosa, repetitiva y propensa al error.

[3] Véase, José Miguel Marinez Albañil: “Características de los lenguajes”, 04/03/04, material electrónico disponible en: http://mx.geocities.com/michaelneo_matrix/CARACTERISTICAS.doc, recuperado el 04/05/09.

Ejemplo de código ensamblador para el famoso programa ¡Hola Mundo!:

```
; HOLA.ASM
; Programa clásico de ejemplo. Despliega una leyenda en pantalla.
STACK    SEGMENT STACK              ; Segmento de pila
           DW    64 DUP (?)         ; Define espacio en la pila
STACK    ENDS

DATA     SEGMENT                    ; Segmento de datos
SALUDO     DB   "Hola mundo!!",13,10,"$" ; Cadena
DATA     ENDS

CODE     SEGMENT                    ; Segmento de Codigo
       ASSUME CS:CODE, DS:DATA, SS:STACK

INICIO:                           ; Punto de entrada al programa
       MOV  AX,DATA                ; Pone direccion en AX
       MOV   DS,AX                      ; Pone la direccion en los
registros
       MOV    DX,OFFSET  SALUDO       ; Obtiene direccion del
mensaje
       MOV  AH,09H                 ; Funcion: Visualizar cadena
       INT     21H                 ; Servicio: Funciones alto nivel
DOS
       MOV  AH,4CH                 ; Funcion: Terminar
       INT    21H
CODE     ENDS
       END  INICIO                 ; Marca fin y define INICIO[4]
```

Lenguaje de alto nivel

Surge a partir de la aparición de las macroinstrucciones que son instrucciones escritas en un lenguaje diferente al lenguaje máquina pero entendidas por este.[5] El lenguaje de alto nivel es un lenguaje muy similar al lenguaje humano, que usa palabras o comandos del lenguaje natural, como por ejemplo del inglés.

[4] Eduardo René Rodríguez Ávila: "Lenguaje ensamblador": Principia, material electrónico disponible en: http://homepage.mac.com/eravila/asmix862.html, recuperado el 04/05/09.

5 Véase: José R. Rojas: "Lenguajes de programación" diapositivas disponibles en: http://www.slideshare.net/jrojas/tema1-lenguajes-de-programacion, recuperado el 04/05709.

La gran ventaja que significaron estos lenguajes es que pueden ser utilizados por diferentes marcas de computadoras, lo cual reduce el costo de la reprogramación, los códigos son más rápidos, se pueden documentar con mayor facilidad y son menos propensos al error.

Entre los lenguajes de esta generación se encuentran ADA, C, Basic, Pascal, COBOL, APL, entre otros.

Lenguaje de muy alto nivel

Son conocidos también como lenguajes de cuarta generación (4GL's), o lenguajes declarativos. Se trata esencialmente de lenguajes de programación que no siguen órdenes procedimentales sino que definen *qué* queremos obtener, y el compilador se encarga de los detalles relativos a *cómo* obtenerlo de una manera eficiente.

Un lenguaje de muy alto nivel es muy fácil de leer, comprender y programar pues contiene macroinstrucciones, además de que no se requiere de grandes conocimientos de computación ni de arquitectura computacional lo cual los hace altamente transportables.

Lenguaje Natural[6]

En la quinta generación se encuentran los llamados Lenguajes naturales, por su acercamiento a la lengua escrita. Su gran característica es que permiten la implementación de aplicaciones que simulan comportamientos inteligentes.

En el siguiente tema, se verá con mayor detalle cada uno de los lenguajes de programación que corresponden a estas etapas.

[6] Ibid.

1.3 Intérpretes

Un intérprete es un programa de software de sistema que traduce un programa de alto nivel a lenguaje máquina pero de forma distinta a un compilador; el intérprete no traduce un programa fuente en un solo paso sino que va traduciendo y ejecutando instrucción por instrucción; así, hasta que traduce y ejecuta una instrucción pasa a la siguiente.

1.4 Compiladores

Un compilador es un programa que traduce las instrucciones de un lenguaje de alto nivel (lenguaje fuente), a instrucciones en lenguaje máquina (lenguaje destino) que la computadora puede leer, interpretar y ejecutar.[7]

Una de las funciones más importantes que tienen los compiladores es la de reportar cualquier error en el programa fuente que se detecte durante el proceso de traducción.

Generalmente los compiladores tienen dos grandes partes:

1. **Back end**: es específico de la plataforma y es la parte que se encarga de generar el código máquina a partir de los resultados de la fase de análisis. Dicho código normalmente no se puede ejecutar de manera directa, para ello requiere de otro programa llamado enlazador.

2. **Front end**: es independiente de la plataforma y es la parte encargada de analizar el código fuente, comprobar la validez, generar el árbol de derivación y rellenar la tabla de símbolos.

[7] Alfred B., Aho; *Compiladores. Principios, técnicas y herramientas*. México, Pearson Educación, México, 2008. p. 1.

Existen categorías de compiladores que tienen que ver con la manera en que generan y hacen uso del código. Las categorías no son excluyentes, es decir, un compilador puede pertenecer a una o más categorías. Las categorías son:

- Compilador optimizador: realiza cambios en el código para hacerlo más eficiente pero mantiene la funcionalidad del programa original.
- Compilador cruzado: genera código para una plataforma distinta de la que se está operando.
- Compilador JIT (Just In Time): compila partes de código según se necesite, ya que forma parte de un intérprete.
- Compilador de una sola pasada: genera código máquina a partir de la primera lectura del código fuente.
- Compilador de varias pasadas: produce el código máquina después de haber leído varias veces el código fuente.[8]

Diferencias entre compilador e intérprete

Existen diferencias entre compiladores e intérpretes que derivan en ventajas y desventajas entre unos y otros. Una primera diferencia es que los intérpretes señalan de inmediato al programador errores en la sintaxis de una instrucción lo que le permite al programador corregir el programa durante su desarrollo.

Otra diferencia es que los intérpretes no utilizan recursos de computación de manera tan eficiente como un programa ya compilado. El intérprete al no producir un programa objeto, debe realizar la traducción cada vez que se ejecuta el programa.

[8] Con base en http://es.wikipedia.org/wiki/Compilador, consultado el 11/05/09.

1.5. Sistemas operativos

La evolución de las computadoras va de la mano con su software, que son todos aquellos programas que no podemos ver físicamente, pero sí los podemos usar para redactar un texto, navegar por Internet, programar, etc.

Un sistema de computación está formado por 6 elementos: software, hardware, bases de datos, documentación, procedimientos y usuarios. El Software son todos aquellos programas que vamos a emplear para llevar a cabo nuestro sistema; el Hardware son los dispositivos físicos; la Base de datos es una colección de datos que vamos a tener y a la cual se accede a través del software; la Documentación son todos aquellos manuales e información que describen cómo funciona el sistema; los Procedimientos son los pasos que definen el uso específico de cada elemento del sistema y finalmente, los Usuarios son todos aquellos Individuos que interactúan durante el desarrollo y funcionamiento del sistema.

El software se divide en dos grandes grupos: programas de aplicación y programas de sistema. El principal programa de sistema es el llamado Sistema Operativo, que es el responsable del control de todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación.

Existen diversas definiciones de lo que es un Sistema Operativo, pero no hay una definición exacta, es decir una que sea estándar. Se podría decir que los Sistemas Operativos son un conjunto de programas que crean la interfaz del hardware con el usuario[9], y que tiene dos funciones primordiales:

[9] Harvey Deitel. *Introducción a los Sistemas Operativos,* México, Addison-Wesley Iberoamericana, 1993. pp. 10.

A. Gestionar el hardware.- Se refiere al hecho de administrar de una forma más eficiente los recursos de la máquina.

- Facilitar el trabajo al usuario.- Permite una comunicación con los dispositivos de la máquina.

El Sistema Operativo se encuentra almacenado en la memoria secundaria. Primero se carga y ejecuta un pedazo de código que se encuentra en el procesador, el cual carga el BIOS, y este a su vez carga el Sistema Operativo que carga todos los programas de aplicación y software variado.

Características de los Sistemas Operativos

Las características deseables que debe poseer un Sistema Operativo son las siguientes:

A. Permitir la ejecución de procesos simultáneos administrando el tiempo de entrada/salida (E/S), el de cálculo y la memoria.
B. Efectiva gestión recursos.
C. Efectiva gestión de tiempos de procesamientos.
D. Fiabilidad respecto a errores y posibles situaciones.
E. Ser de tamaño pequeño.
F. Posibilitar y facilitar la intermediación entre computadora y usuario de la misma.
G. Permitir a los usuarios compartir datos entre ellos, en caso necesario.
H. Gestión eficaz de los dispositivos de E/S de la computadora.[10]

[10] Con base en http://entren.dgsca.unam.mx/introduccion/so_carac.html, consultado el 11/05/09.

Bibliografía del tema 1

- Aho, Alfred B.; *Compiladores. Principios, técnicas y herramientas* México, Pearson Educación, 2008.

- Joyanes Aguilar, Luis; Luis Rodríguez Baena y Matilde Fernández Azuela, *Fundamentos de Programación: Libro de Problemas*. México, McGraw-Hill, 1996.

- Deitel Harvey, M. l*., Introducción a los Sistemas Operativos*. México, Addison-Wesley Iberoamericana, 1993.

Sitios web

- http://es.wikipedia.org/wiki/Compilador
- http://foro.latinohack.com/showthread.php?t=17882
- http://mx.geocities.com/michaelneo_matrix/CARACTERISTICAS.doc
- http://homepage.mac.com/eravila/asmix862.html
- http://entren.dgsca.unam.mx/introduccion/so_carac.html

Actividades de aprendizaje

A.1.1. Elabora un mapa conceptual de lo estudiado en el tema uno, con el propósito de que tengas un panorama global de la programación e implementación de sistemas, apoyándote con el temario propuesto y la bibliografía sugerida.

A.1.2. Elabora un resumen de hasta dos páginas en un documento Word, en donde resaltes los orígenes y las características de al menos tres etapas de los lenguajes de programación y sus principales objetivos.

A.1.3. Elabora en una diapositiva en PowerPoint, un cuadro sinóptico en donde resaltes las características de los sistemas operativos y sus principales componentes.

Cuestionario de autoevaluación

1. Define qué es un lenguaje de programación.
2. ¿Qué es un intérprete?
3. ¿Qué es un compilador?
4. ¿Qué es el *front end* de un compilador?
5. ¿Qué es el *back end* de un compilador?
6. Menciona una ventaja del compilador.
7. Menciona una ventaja del intérprete.
8. ¿Cuáles son los componentes de un sistema operativo?
9. ¿Cuáles son las funciones principales de un sistema operativo?
10. Menciona 3 características que posee cualquier sistema operativo

Examen de autoevaluación

Subraya el inciso con la respuesta correcta

1. Programa capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel.

a) Intérprete

b) Compilador

c) Sistema operativo

d) Lenguaje de programación

2. Parte que analiza el código fuente, comprueba su validez, genera el árbol de derivación y rellena los valores de la tabla de símbolos.

a) *Back End*

b) Analizador sintáctico

c) *Front End*

d) Analizador lexicográfico

3. Forman parte de un intérprete y compilan partes del código según se necesita.

a) Compiladores cruzados

b) Compiladores de una sola pasada

c) Compiladores de varias pasadas

d) Compiladores JIT

4. Son funciones básicas de un sistema operativo.

a) Traducir e interpretar

b) Gestionar recursos, brindar una interfaz de usuario

c) Procesar y almacenar

d) Presentar datos

5. Tipo de compilador que realiza cambios en el código para hacerlo más eficiente pero mantiene la funcionalidad del programa original.

a) Compiladores cruzados
b) Compiladores de una sola pasada
c) Compiladores de optimizador
d) Compiladores JIT

6. Tipo de compilador que genera código máquina a partir de la primera lectura del código fuente.

a) Compiladores cruzados
b) Compiladores de una sola pasada
c) Compiladores de optimizador
d) Compiladores JIT

7. Tipo de compilador que genera código para una plataforma distinta de la que se está operando.

a) Compiladores cruzados
b) Compiladores de una sola pasada
c) Compiladores de optimizador
d) Compiladores JIT

8. Parte de un compilador que es específica de la plataforma que se esté operando.

a) *Back End*
b) Analizador sintáctico
c) *Front End*
d) Analizador lexicográfico

9. Tipo de compilador que produce el código máquina después de haber leído varias veces el código fuente

a) Compiladores cruzados

b) Compiladores de una sola pasada

c) Compiladores de varias pasadas

d) Compiladores JIT

10. Parte de un compilador que es independiente de la plataforma que se esté operando.

a) *Back End*

b) Analizador sintáctico

c) *Front End*

d) Analizador lexicográfico

TEMA 2. LENGUAJES DE PROGRAMACIÓN

Objetivo particular

El alumno reconocerá las diversas generalidades de los lenguajes de programación, su sintaxis y aplicación en los sistemas de información a través del tiempo.

Temario detallado

2.1 Historia de los lenguajes de programación
2.2 Sintaxis de los lenguajes de programación
2.3 Etapas de traducción
2.4 Modelos formales de traducción
2.5 Tipos de datos
2.6 Tipos de datos abstractos
2.7. Gestión de almacenamiento
2.8 Control de secuencia

Introducción

La manera en que se realiza la interacción usuario-máquina es vital en el desarrollo de sistemas de información, por tal motivo es necesario comprender el desarrollo de los lenguajes de programación a través del tiempo, y su estructura gramática y funcionamiento lógico.

2.1. Historia de los lenguajes de programación[11]

A continuación se mencionan algunos de los lenguajes de programación más importantes por año de aparición:

Plankalkül[12]

Es un lenguaje teórico desarrollado por Konrad Zuse en 1946 pero que no fue implementado sino hasta 1995. Se dice que es teórico porque en él se muestra la forma de organización de datos para posteriores lenguajes de programación.

Plankalkül utilizaba solo valores atómicos (bits) y los demás tipos de valores se construían a partir de estos.

Fortran[13]

Es el acrónimo de “Formula Translator” o Traductor de Fórmulas y se desarrolló en 1955 por *John Backus*. Fue el lenguaje científico más popular, desafortunadamente tenía un limitado procesamiento alfanumérico y sus capacidades de manejo de datos lo hacían complicado cuando se aplicaba en programación para negocios.

- **LISP[14]**

Acrónimo de "LIStProcessing” o Procesamiento de Listas”. Lisp es un lenguaje de programación de alto nivel creado en 1958 por John McCarthy y sus colaboradores en el MIT. Fue desarrollado basándose en las ideas de listas y árboles de longitud variable lo que lo hace muy adecuado para la Inteligencia Artificial.

---

[11] Con base en Luis, Joyanes, et.al., *Fundamentos de Programación: Libro de Problemas,* México, McGraw-Hill, 1996, pp. 10 y en http://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n, consultado el 11/05/09.

[12] Con base en http://es.wikipedia.org/wiki/Plankalk%C3%BCl, consultado el 11/05/09.

[13] Con base en Larry Long, *Introducción a las computadoras y al procesamiento de información*, Prentice Hall, Nueva Jersey, 1995, p. 252

[14] Con base en http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp16.pdf, consultado el 11/05/09.

- **ALGOL**[15]

  Acrónimo de “Algorithmic Language” o Lenguaje algorítmico. Algol es uno de varios lenguajes de alto nivel, diseñado específicamente por un grupo de expertos europeos y americanos para la programación de los cálculos científicos. Apareció a finales de los años 50 y se formalizó en un reporte titulado ALGOL 58; luego avanzó a través de otros reportes ALGOL 60 y ALGOL 68. Fue diseñado por un comité internacional para ser un lenguaje universal.

  ALGOL nunca alcanzó el nivel de popularidad comercial de FORTRAN y COBOL, pero se consideró el lenguaje más importante de su era en términos de su influencia en el desarrollo de posteriores lenguajes de programación.

- **COBOL**[16]

  Es el primer lenguaje para programación de negocios, fue desarrollado y presentado en 1959 por *Grace Hooper;* su nombre es acrónimo de “COmmon Business - Oriented Language”, o Lenguaje Común Orientado a Negocios.

  La intención original de quienes lo desarrollaron fue crear un lenguaje cuyas instrucciones se parecieran al inglés. Aún cuando COBOL es muy poderoso existen programadores que consideran que utiliza un número excesivo de palabras en su sintaxis.

[15] Con base en http://elvex.ugr.es/etexts/spanish/pl/algol.htm, consultado el 11/05/09.
[16] Véase, Larry Long, *Introducción a las computadoras y al procesamiento de información*”, Prentice Hall, Nueva Jersey 1995, pp. 254; pero está disponible en: Wikipedia: “COBOL”, en línea: http://es.wikipedia.org/wiki/COBOL, consultado el 11/05/09.

- **APL**

  Acrónimo de “A Programing Language” o Un lenguaje de programación. Es un lenguaje simbólico interactivo de programación presentado en 1968 por Kenneth Iverson y utilizado principalmente por ingenieros, matemáticos y científicos.

  En este lenguaje se requiere de una terminal especial de exhibición en video para escribir los programas. Su teclado cuenta con un conjunto especial de caracteres que incluye los símbolos para escribir las instrucciones APL. Estos símbolos proporcionan una escritura abreviada que acelera el proceso de codificar un programa.

  Se le conoce por su enorme potencia ya que entre otras cosas permite manipular a escala de bits y cuenta con procesadores auxiliares.[17]

- **BASIC**

  En 1965, BASIC acrónimo de “Beginners All-purpose Symbolic Code” (en español ‘código de instrucciones simbólicas de propósito general para principiantes’) se utilizó como herramienta para enseñar los principios fundamentales de la programación pero durante su desarrollo adquirió tal popularidad que dejó de ser considerado un lenguaje para principiantes y cubrió las demandas de programadores con mayores capacidades.

  Es considerado el lenguaje principal de las microcomputadoras y fue desarrollado por Thomas E. Kurtz. [18]

[17] Ibid, pp. 253.
[18] Ibid, pp. 258.

- **PL/1**

  PL/1, acrónimo de “Programming Language 1” o Lenguaje de Programación 1, fue desarrollado y presentado en 1964 por IBM, pero su versión inicial no era eficiente. Una vez eliminados sus errores ocultos, comenzó a ganar popularidad. A partir de ese momento, 1970, muchas compañías lo adoptaron como su único lenguaje orientado a los procedimientos. [19]

  Su lenta aceptación se debió no a la falta de calidad o capacidad, sino a que las compañías habían invertido grandes recursos con COBOL y FORTRAN y no deseaban reinvertir.

  PL/1 fue probablemente el primer lenguaje comercial cuyo compilador estaba escrito en el lenguaje que compilaba [20]

- **BCPL**

  Acrónimo inglés “Basic Combined Programming Language” o Lenguaje de Programación Básico Combinado. BCPL es un lenguaje que fue diseñado en 1966 por Martin Richards y aplicado por primera vez en el MIT en la primavera de 1967.

  BCPL es un lenguaje que contiene muchas características que más tarde aparecen en C más o menos modificadas. Además contiene otras que han sido utilizadas posteriormente en otros lenguajes, entre ellas podemos destacar que es un lenguaje sin tipos ("Typeless"); el tipo de las variables no es determinado en el momento de su declaración, de hecho, cada variable puede ser considerada como de cualquier tipo. Al aplicarle un operador es cuando se decide el tipo adecuado en ese instante.

[19] Ibid, pp. 261.
[20] Con base en http://es.wikipedia.org/wiki/PL/I, consultado el 11/05/09.

Otras características, en cambio, son idénticas o muy parecidas a las que estamos habituados a ver en C/C++, por ejemplo: existencia de constantes manifiestas; variables carácter y cadenas alfanuméricas (strings); muchos de sus operadores, así como los conceptos de asociatividad y precedencia de los mismos cuando se utilizan juntos en una expresión; uso de funciones y paso de argumentos (por valor), entre otras.

- **B**[21]

  B es el nombre de un lenguaje de programación desarrollado en los laboratorios Bell, predecesor del lenguaje de programación C. Fue mayoritariamente un trabajo de Ken Thompson con contribuciones de Dennis Ritchie. Apareció primero sobre el año 1969 y es un lenguaje de computadora destinado a ser recursivo, principalmente no numérico y caracterizado por las aplicaciones de su sistema de programación.

- **Pascal**[22]

  Recibe su nombre en honor al matemático francés del siglo XVII, Blaise Pascal. El lenguaje se dio a conocer en 1968 y tenía como objetivo facilitar el aprendizaje de la programación en alumnos.

  En Pascal, el código se divide en funciones o procedimientos y cuenta con una potencia, flexibilidad y estructura de autodocumentación muy sencilla de implementar.

[21] Con base en Gottfried, Byron S. *"Programación en C"*. McGraw-Hill Interamericana, México, 2ª dición, 2005, pp. 22.
[22] Con base en Larry Long, "*Introducción a las computadoras y al procesamiento de información*", Prentice Hall, New Jersey 1995, pp. 262

- **C**[23]

  Lenguaje de programación diseñado por Dennis Ritchie durante el decenio de 1970 e inmediatamente utilizado para reimplementar Unix; se llama así porque muchas de sus características están derivadas de un primer compilador llamado 'B'.

  C se convirtió en inmensamente popular fuera de los laboratorios Bell después de 1980 y es ahora el lenguaje dominante en los sistemas y las aplicaciones de programación de microcomputadoras.

  C es descrito a menudo como un lenguaje que combina toda la elegancia y el poder del lenguaje ensamblador con toda la legibilidad y facilidad de mantenimiento de un lenguaje de cuarta generación.

- **Smalltalk**[24]

  Smalltalk es un lenguaje de programación estándar normalizado en 1998 por el Instituto Nacional de Estándares Americanos - ANSI. Tiene como objetivo realizar tareas de computación mediante la interacción con un entorno de objetos virtuales.

- **Prolog**[25]

  Prolog es un lenguaje de programación simple pero poderosa, desarrollado en la Universidad de Marsella como una herramienta práctica para la programación lógica. Desde el punto de vista del usuario, la ventaja principal es la facilidad para programar, ya que se pueden escribir rápidamente y con pocos errores, programas claramente legibles.

---

[23] Con base en Gottfried, Byron S. *Programación en C*, 2ª edición, México, McGraw-Hill Interamericana, 2005, pp. 23.
[24] Con base en http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp12.pdf, consultado el 11/05/09.
[25] Con base en W. D. Burnham, "*Prolog: programación y aplicaciones",* Limusa, México, 1989, pp5.

Su nombre proviene del francés *Programation et Logique*, y bastante popular en el medio de investigación en Inteligencia Artificial.

- **ML**

  Acrónimo de “Meta Lenguaje”, es un lenguaje funcional que a diferencia de lenguajes convencionales de procesos, es matemáticamente "puro". Fue desarrollado por Robin Milner en la Universidad de Edimburgo a finales de los años 70.

  Entre sus características se incluye el álgebra de funciones, el polimorfismo y el análisis de tipos, patrones y excepciones. Actualmente es usado para diseñar y manipular lenguajes de programación, en sistemas financieros y en bioinformática.

- **SQL**[26]

  Es el acrónimo de “Structured Query Language”, o Lenguaje de Consulta Estructurado. SQL se constituyó como un estándar de la ANSI para acceder a y manipular sistemas de bases de datos.

  SQL está basado en el álgebra relacional del Dr. Codd y es considerado un lenguaje de cuarta generación (4GL).

- **ADA**[27]

  Es un lenguaje de aplicación múltiple desarrollado por el Departamento de Defensa de los Estados Unidos. Recibió su nombre en honor a Lady Augusta Ada, pionera del siglo XIX, considerada la primera programadora.

[26] Con base en Judy Bishop, *Java: fundamentos de programación*, Madrid, Addison Wesley, 1999, pp. 428.
[27] Con base en Larry Long, op. cit., p. 262.

ADA se volvió importante y ganó aceptación no sólo en el sector militar sino en el privado. Se considera un lenguaje complejo orientado a procedimientos.

- **C++**[28]

El Comité para el estándar ANIS C fue formado en 1983 con el objetivo de crear un lenguaje uniforme a partir del C original, desarrollado por Kernighan y Ritchie en 1972 en la AT&T. C++ se comenzó a desarrollar en 1980. Su autor fue B. Stroustrup también de la AT&T. Su nombre hace referencia al carácter del operador incremento de C.

El C++ es un lenguaje de programación versátil, potente y general. Mantiene las ventajas de C en cuanto a riqueza de expresiones, operadores, flexibilidad, concisión y eficiencia; y abarca tres paradigmas de programación: estructurada, genérica y orientada a objetos.

- **Eiffel**[29]

Eiffel es un lenguaje de Programación Orientada a Objetos puro. En el entorno académico está altamente difundido y es fuente continua de nuevas investigaciones. Fue creado en 1985 por Bertrand Mayer y usa una sintaxis parecida a la de Pascal. Entre sus principales características destacan el uso de pre- y pos-condiciones, bucles y clases, objetos y mensajes.

[28] Con base en Bjarne Stroustrup, *"El lenguaje de programación C++,* Madrid, Addison Wesley, 1993, pp. 15.

[29] Con base en http://www.diclib.com/cgi-bin/d1.cgi?l=es&base=es_wiki_10&page=showid&id=5623, consultado el 11/05/09.

- **Perl**[30]

  Perl es un lenguaje de programación que significa "Practical Extraction and Report Language". Es un lenguaje de programación medianamente nuevo, diseñado por Larry Wall en 1987, y surgió de otras herramientas de UNIX como son: sed, grep, awk y c-shell.

  Principalmente sirve para labores de procesamiento de texto, lo cual lo hace en una forma fácil y clara, no como es en C o Pascal; también sirve para la programación de software de sistemas; y ahora se ha consolidado como un lenguaje para programar aplicaciones para WWW. También es muy usado para manejo y gestión de procesos (estado de procesos, conteo y extracción de parámetros característicos, entre otros).

- **Python**

  Python es un lenguaje de programación creado en 1990 por Guido Van Rossum. Recibe su nombre por la afición de su creador a los humoristas británicos.

  Python es un lenguaje de programación orientado a objetos notablemente dinámico.

  Algunas de sus principales características son: sintaxis muy clara y legible, fuerte capacidad de introspección, orientación a objetos intuitiva, expresión natural del código de procedimiento, modularidad, excepción basada en el manejo de errores, amplias bibliotecas, entre otras.

  El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de diseño.[31]

[30] Con base en http://www.perl.org/, consultado el 11/05/09.
[31] Con base en http://es.wikipedia.org/wiki/Python, consultado el 11/05/09.

- **Java**[32]

  Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros.

- **Ruby**[33]

  Es un lenguaje de programación dinámico y de código abierto enfocado a la simplicidad y productividad. Fue creado por Yukihiro Matsumoto quien mezcló partes de sus lenguajes favoritos (Perl, Smalltalk, Eiffel, Ada y Lisp) para formar un nuevo lenguaje que incorporara tanto la programación funcional como la programación imperativa.

  Desde su liberación pública en 1995, Ruby ha atraído devotos desarrolladores de todo el mundo. En el 2006, Ruby alcanzó reconocimiento masivo, formándose grupos de usuarios activos en las ciudades más importantes del mundo y llenando las capacidades de las conferencias relacionadas a Ruby.

2.2 Sintaxis de los lenguajes de programación[34]

La sintaxis son las reglas de construcción de los programas. Cada lenguaje de programación utiliza su propia sintaxis, pero en general, la mayoría de los lenguajes cuentan con los siguientes elementos:

- Metalenguaje B.N.F (Forma de Backus-Naur)
  - Notación para especificar una gramática generativa: define el conjunto de cadenas que son programas del Lenguaje de Programación sujeto, junto con su estructura sintáctica.

[32] Con base en Judy Bishop, op, cit., p. 3.
[33] Con base en http://www.ruby-lang.org/es/, consultado el 11/05/09.
[34] Con base en Augie Hansen, *¡Aprenda C Ya!, Madrid,* Anaya Multimedia, 1990, p. 32.

- Permite describir lenguajes con una sintaxis “independiente del contexto”.

- Gramáticas de atributos

a) Extensión de B.N.F. mediante atributos y reglas de evaluación de dichos atributos.

b) Permite describir lenguajes con hechos sintácticos “dependientes del contexto”.

c) Símbolos no-terminales (uno de ellos distinguido)
-Para describir los constructores sintácticos del LP sujeto

d) Símbolos terminales
-Para describir los símbolos (texto) del LP sujeto

Reglas de producción

A. Una regla con alternativas para cada símbolo no-terminal

B. Cada alternativa: cadena de terminales y/o no-terminales

Ejemplos:

<sent-cond>: if <comp> then <serie-instr> fi |
if <comp> then <serie-instr> else <serie-instr> fi
<programa>: <serie-declar> <serie-instr>

- Se permite recursión en las reglas de producción
  – a izquierdas: <serie-instr> <instr> | <serie-instr> ;<instr>
  – a derechas: <serie-instr>: <instr> | <instr>; <serie-instr>

- Estructura sintáctica => árbol sintáctico

  Ejemplo (parte de un árbol):

  <sent-cond>

  if <comp> then <serie-instr> fi

  <serie-instr>; <instr>

- Gramática ambigua:

  – Para una cadena terminal hay más de un árbol sintáctico

  Ejemplo:

  <expr>: x | y | z | (<expr>) | <expr> + <expr> | <expr> * <expr>

  (recursión a izquierda y derecha para un mismo símbolo no-term.)

  Dos árboles sintácticos para la cadena x + y * z :

  <expr> <expr>

  <expr> + <expr> <expr> * <expr>

  x <expr> * <expr> <expr> + <expr> z

  y z x y

- Evitar la doble recursividad:

  – Introduciendo nuevos símbolos no-terminales (dejando * y + al mismo nivel)

  <expr>: <elem> | <expr> + <elem> | <expr> * <elem>

  <elem>: x | y | z | (<expr>)

  - misma prioridad para * y + , y asociatividad a izquierdas

  – Introduciendo nuevos símbolos no-terminales (dejando * y + a distinto nivel)

  <expr>: <term> | <expr> + <term>

<term> ::= <elem> | <term> * <elem>

<elem> ::= x | y | z | (<expr>)

- Prioridad de * respecto de +, y asociatividad a izquierdas para ambas operaciones.

Ejemplo:

Características del lenguaje Eva:

- Identificadores de distintos tipos (char, string, proc)
- Estructura de datos compuesta (string) con operaciones asociadas (head, tail, cons)
- Declaraciones
- Estructura de bloques
- Procedimientos (recursivos), paso de parámetros (por valor)

Asumiremos en cada programa escrito en Eva que:

1. Se asociará un valor inicial por defecto (cadena vacía) a cada variable de tipo string en el momento de su declaración; mientras que las variables de tipo char quedan indefinidas
2. No hay conversiones de tipo (string __char)

Descripción sintáctica completa de Eva:

- Gramática BNF + CONDICIONES de CONTEXTO:
    1. Los <name> distintos dentro de cada <declaration sequence> en cada <block> y dentro de cada <parameter list>

    2. Cada <name> contenido en una <statement> S tiene que haber sido declarado en un <block> que contenga a S o bien pertenecer a una <parameter list> de un proc que contenga a S

3. Si hay más de uno _ “el más interno”

4. En input <name> _ <name> debe ser de tipo char

5. En call <name> (<expression list>) _ <name> debe ser de tipo proc y el número y tipo de los argumentos de la llamada deben coincidir con lo declarado para <name>

6. En cons <char expression>,<name> _ <name> debe ser de tipo string

7. <name> debe ser de tipo char en <char expression> y de tipo string en <string expression>

Begin EJEMPLO DE PROGRAMA en Eva

```
char c
proc printword (string word)= (
neq tail word,“”: call printword (tail word)
output head word )
proc control =
begin string w
proc readword = ( cons c,w
input c
neq c,space:call readword )
call skipblaks
call readword
neq w,”zz” : ( call control
output space
call printword (w))
end
proc skipblancs = ( input c
```

```
eq c, space: call skipblanks )
call control
end
```

Variaciones de BNF

- Herramientas notacionales añadidas para:

– reducir el tamaño de la gramática

– incrementar la claridad

- No suponen mayor poder de expresión

2.3 Etapas de traducción

La traducción se refiere a la transformación de un programa fuente en programa objeto. Básicamente se puede hablar de dos grandes etapas en este proceso:

- Análisis del programa fuente
- Síntesis del programa objeto

Análisis del programa fuente

La fase de análisis supone la división del programa fuente en componentes, y a cada uno de ellos le impone una estructura gramatical. Si durante el análisis se detecta que el programa fuente está mal formado en cuanto a la sintaxis, o que no tiene una semántica consistente, entonces debe proporcionar mensajes informativos para que el programador pueda corregirlo. En la fase del análisis también se recolecta información sobre el programa fuente y se almacena en una estructura de datos llamada tabla de símbolos, la cual se pasa junto con la representación intermedia a la fase de la síntesis.

Síntesis del programa objeto

Durante la fase de la síntesis, se construye el programa objeto deseado a partir de la representación intermedia y de la información en la tabla de símbolos. Como ya se mencionó en la parte introductoria de compiladores, a la *fase del análisis* se le llama comúnmente el *Front end* del compilador y la fase de la *síntesis*, propiamente la traducción es lo que se conoce como *Back end*.

2.4 Modelos formales de traducción

Los modelos formales de traducción han emergido como enfoques analíticos en donde el desarrollo de software puede ser verificado por medio de poderosas teorías matemáticas, presentando ventajas en la calidad del software, tiempo de desarrollo, tamaño y complejidad de las pruebas. Sin embargo, aún existen muchos inconvenientes para la correcta implantación de los métodos formales como una metodología de desarrollo de software comúnmente aceptada en ingeniería de software.

Hasta el momento se reconocen como modelos formales de traducción los siguientes:

A. BNF (Forma Backus-Naur)
B. Autómatas
C. Algoritmo de análisis sintáctico eficiente
D. Modelado semántico

BNF (Forma Backus-Naur)

Es una notación creada por John Backus y Peter Naur para describir las gramáticas de contexto libre que definen un lenguaje de programación. La gramática describe cómo formar instrucciones a partir de palabras, es decir, nos da las reglas para construir oraciones que puedan ser entendidas por un lenguaje. Una gramática de contexto libre está formada

por 4 elementos: símbolos terminales, símbolos no terminales, símbolo inicial y conjunto de reglas.

BNF se originó en los años 60 como una notación formal para describir la sintaxis de ALGOL 60, y ha sido usado hasta la fecha para describir la mayoría de lenguajes formales de programación.[35]

Autómatas

Los autómatas son mecanismos que generan gramáticas formales. La manera en que realizan esto es mediante la noción de *reconocimiento*.

Se considera a los autómatas como *analizadores léxicos* de las gramáticas a que corresponden.

Un analizador léxico lee los caracteres de entrada del programa fuente, los agrupa en secuencias conocidas como lexemas y produce como salida una secuencia de tokens para cada lexema en el programa fuente. Un token es una estructura formada por el nombre del token y su valor.

Podemos encontrar seis tipos de autómatas: regulares, de pila, lineales, de pila no deterministas, máquinas de Turing y autómatas por jerarquía de Chomsky.[36]

Algoritmo de análisis sintáctico eficiente

Un analizador sintáctico es una herramienta que nos permite determinar la estructura sintáctica de un programa fuente. Existen tres tipos generales de analizadores para las gramáticas: universales, descendentes y ascendentes. Estos analizadores pueden ser creados de manera manual o

[35] Con base en: http://www.itculiacan.edu.mx/apuntes/maestros/Ram%F3n%20Zatara%EDn/LenguajesAutomatas/LENGUAJESYAUTOMATAS.ppt, recuperado el 11/05/09.
[36] Con base en http://delta.cs.cinvestav.mx/~gmorales/ta/ta.html, consultado el 11/05/09.

de manera automática. Para los creados de manera automática es necesario contar con algoritmos, ya que éstos inducen representaciones a partir de ejemplos donde el conocimiento ya ha sido aplicado.

Un algoritmo es un listado definido, ordenado y finito de operaciones para hallar solución a un problema. Para determinar estructuras sintácticas de programas fuente contamos con algoritmos reconocidos como: el Cocke-Younger-Kasami, el algoritmo de Early, el Left-Corner, Recursive-Descent, entre otros.[37]

Modelado semántico

Consiste en estudiar los datos que se pretenden almacenar antes de elegir el modelo de datos concreto que se va a usar para su manipulación. El modelado semántico permite separar el análisis (¿qué?) del diseño (¿cómo?).

2.5. Tipos de datos

Un tipo de datos define la forma en que el compilador almacena información en la memoria. Se puede definir un **tipo de dato** a partir de los valores permitidos y las operaciones que se puedan llevar a cabo sobre estos valores.[38]

Los datos se clasifican en simples y compuestos. Los datos simples no se derivan de otro tipo de datos, soportan literales que les dan valor, son indivisibles y ocupan solo una casilla de memoria.

[37] Si deseas profundizar en este tema, se recomienda leer el artículo “Un analizador sintáctico eficiente para gramáticas en Español” de Nora La Serna, disponible en http://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/risi/N1_2004/a04.pdf, consultado el 11/05/09.

[38] Véase, Tipos de dato en: http://tricia-tecnologia.blogspot.com/2007_09_01_archive.html, consultado el 11/05/09.

Por su parte, los datos compuestos están formados de datos simples o de otros datos compuestos, son reusables, y a través de la declaración de una variable pueden hacer referencia a diferentes casillas de memoria. Los datos compuestos pueden ser arreglos, cadenas y registros.

a) Un arreglo es una colección finita, homogénea y ordenada de elementos en la que se hace referencia a cada elemento por medio de un índice.
b) Una cadena es un tipo de dato estructurado compuesto por caracteres.
c) Un registro es una colección de elementos finita y heterogénea.

2.6. Tipos de datos abstractos

Un TDA es un tipo de dato definido por el programador, que está formado por un conjunto válido de elementos y un número de operaciones primitivas que se pueden realizar sobre ellos.

Dentro de los TDA podemos encontrar diversas estructuras:

- Listas
- Colas
- Pilas

Lista

Una lista es una serie de N elementos ordenados en la cual el elemento i+1 viene a continuación del elemento i. Si la lista contiene 0 elementos se denomina como *lista vacía*.[39]

[39] Con base en http://www.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/, consultado el 11/05/09.

Cola

Denominada también listas FIFO, del inglés “First In - First Out”, el primero que entra es el primero que sale. Una cola es una estructura de datos de dos extremos, que señalan el primer elemento y el último. Los elementos se añaden por el final y se sacan por el principio. Entre sus propiedades básicas se encuentran las siguientes:

A. Los elementos se disponen de manera secuencial
B. Hay dos extremos, el primero y el último
C. Los elementos se añaden detrás del último
D. Los elementos se extraen desde el primero
E. La cola se puede recorrer desde el primero hasta el último extremo.[40]

Cola de prioridad

Es un tipo de cola que posee para sus elementos un criterio de orden dado.

Pila

Denominada también lista LIFO, del inglés “Last-In,Fisrt-Out”, el último en entrar es el primero en salir. Una pila es un apilamiento de elementos, como su nombre lo sugiere, que se añaden y retiran por arriba. Entre sus propiedades básicas se encuentran las siguientes:

F. Los elementos se disponen de manera secuencial
G. Hay un extremo especial denominado, la cima
H. Los elementos se añaden a la cima
I. Los elementos se extraen de la cima
J. La pila no se puede recorrer. Sólo es visible el elemento situado en la cima.[41]

[40] Con base en Judy Bishop, *Java: fundamentos de programación*, ed. cit., p. 461.
[41] Ibid, p. 459.

La interfaz la pila provee las siguientes operaciones:

- **apilar**: inserta el elemento en el tope de la pila.
- **desapilar**: regresar el elemento que se encuentre en la cima de la pila y eliminarlo.
- **tope**: regresar el elemento que se encuentre en la cima de la pila, pero sin eliminarlo.
- **estaVacia**: regresar *verdadero/falso* si la pila contiene o no elementos.[42]

2.7 Gestión de almacenamiento

El sistema operativo es entre otras cosas un administrador del almacenamiento[43]; dependiendo del tamaño de espacio para almacenar, se utiliza la memoria o algún dispositivo de almacenamiento secundario.

Al almacenamiento en memoria principal se le conoce como gestión de memoria o asignación de memoria, al almacenamiento en memoria secundaria, en cintas, discos magnéticos u otros dispositivos se le conoce como almacenamiento secundario o gestión de almacenamiento.[44]

En el almacenamiento secundario se suelen encontrar los datos permanentes y los que no caben en la memoria principal.[45]

---

[42] Con base en http://www.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/, consultado el 11/05/09.
[43] Gabriela García: "Administración de la memoria", material disponible en línea: http://www.angelfire.com/bc3/gabrielagarcia0/Unidad1.htm, consultado el 08/10/09.
[44] Gary Bronson, Jorge Alberto Velázquez Arellano tr., C++ para ingeniería y ciencias, México, International Thomson, 2007, pp. 24.
[45] http://neo.lcc.uma.es/staff/francis/fundcomp/tema8.ppt, recuperado el 08/10/2009.

Jerarquía del almacenamiento

La jerarquía en el almacenamiento se da en el siguiente orden: cintas magnéticas, discos ópticos, disco magnético (disco duro), memoria flash, memoria principal y memoria caché.

- Para poder ser referenciados o ejecutados los programas y datos tienen que estar en la memoria principal.
- Se mantienen en almacenamiento secundario los programas y datos que no son necesarios de inmediato.
- El almacenamiento principal es más costoso pero de acceso más rápido que el secundario.

Estrategias de gestión del almacenamiento

Están dirigidas a la obtención del mejor uso posible del recurso del almacenamiento principal[46]. Se dividen en[47]:

- Estrategias de búsqueda anticipada: consisten en determinar la parte del programa o fragmento de datos que se almacenará en la memoria principal, esto puede producir un mejor rendimiento del sistema.
- Estrategias de búsqueda por demanda: se realiza cuando la siguiente parte del programa o fragmento de datos se carga al almacenamiento principal cuando algún programa en ejecución lo referencia.
- Estrategias de colocación: están relacionadas con la determinación del espacio en memoria donde se cargará un programa nuevo.

[46] Con base en Deitel Harvey. M. l. *Introducción a los Sistemas Operativos,* México, Addison-Wesley Iberoamericana, pp.7
[47] Con base en http://ingenieria.uatx.mx/~patrick/so/unidad2.pdf, recuperado el 08/10/2009

- Estrategias de reposición: están relacionadas con la determinación de qué fragmento de programa o de datos desplazar para dar lugar a los programas nuevos

Ahora bien, ya que mencionamos dentro del almacenamiento secundario a los dispositivos, es importante revisar de manera general como funcionan y están constituidos algunos de ellos.

Cintas magnéticas

Es un dispositivo de almacenamiento magnético que graba los datos en pistas de plástico mediante material ferromagnético, generalmente óxido de hierro. La principal diferencia que existe con este un medio de acceso secuencial, mientras que el disco duro es un medio de acceso aleatorio[48].

Discos ópticos

Los discos ópticos (CD y DVD) son unidades físicas de almacenamiento comúnmente construidas en policarbonato. Se les denomina ópticos porque funcionan mediante un láser que crea surcos microscópicos para almacenar, guardar y codificar la información[49]. Los datos son almacenados en una capa dentro del policarbonato y otra capa metálica, refleja la luz del láser de vuelta hacia un sensor.

Por lo general los discos ópticos se componen de 2 capas, una que se halla dentro del policarbonato y que sirve para almacenar los datos y otra metálica que refleja la luz del láser en un sensor y que sirve para leer los datos del disco.

En un CD la capa de los datos está cerca de la parte de arriba del disco, del lado de la etiqueta. En un DVD existen discos de una o dos capas de datos que se

[48] Con base en http://es.wikipedia.org/wiki/Cinta_magn%C3%A9tica_de_almacenamiento_de_datos, consultado el 0810/09
[49] Con base en http://es.wikipedia.org/wiki/Disco_%C3%B3ptico, consultado el 08/10/09

encuentran en el medio del disco y que pueden ser accedidos desde un lado o desde ambos lados.

Para mayor información véase el siguiente enlace: http://www.slideshare.net/fundamentosinformaticos_a/discos-opticos-presentation

El disco duro

El disco duro es un disco magnético que puede llegar a almacenar más de 100 gigabytes. Se le denomina duro para diferenciarlo de los discos flexibles o disquetes.

El disco duro consiste en uno o varios discos o platos que tienen pistas donde es almacenada la información[50], al conjunto de pistas de diferentes discos en la misma posición, se le conoce como cilindro y a las diferentes partes de una pista se le conoce como sector. Para leer las pistas se utilizan cabezales de lectura/escritura.

Para la gestión del espacio de almacenamiento en un disco duro, existen tres métodos[51]:

a) **Asignación continua**: En este método cada archivo ocupa bloques con direcciones lógicas del dispositivo. La asignación es definida por la dirección en disco del bloque inicial y la longitud del área asignada al archivo (Número de bloques)

b) **Asignación Enlazada**: Es un método que mejora el rendimiento en el disco al unir bloques que pueden estar dispersos por todo el disco. Cada bloque contiene un apuntador a otro bloque y la asignación se va dando a través de saltos. Los últimos bits apuntan los siguientes apuntadores del archivo.

[50] Con base en http://www.masadelante.com/faqs/disco-duro, consultado el 08/10/09

[51] Con base en http://www.infor.uva.es/~fjgonzalez/apuntes_aso/Tema6.pdf, consultado el 08/10/09

c) **Asignación Indexada**: Es un método que funciona como una asignación enlazada pero el apuntador de cada bloque referencia hacia un bloque índice.

2.8 Control de secuencia

Las secuencias de los programas se basan en las instrucciones de saltos que hacen las máquinas a nivel de hardware. Estas instrucciones funcionan porque utilizan programas almacenados en memoria; la Unidad Central de Proceso tiene un registro que apunta a la próxima instrucción que se debe ejecutar y esta dirección puede ser alterada por diferentes estructuras llamadas estructuras de secuencia.[52]

Las estructuras de secuencia se clasifican en tres grupos:

A. Las que se usan entre expresiones, por ejemplo:

3*4-5*1 =7

No es igual que

(3*4)-(5-1)=8

B. Las que se usan entre enunciados, por ejemplo:

```
If calificación > 6
then
printf “Aprobado”;
```

C. Las que se usan entre subprogramas, por ejemplo:

En C, un parámetro formal de un dato local dentro de un subprograma sería llamado así:

```
SUB(int X, char Y)
```

[52] Con base en http://weblogs.inf.udp.cl/inf9019-1/files/2006/09/tareacontrol.pdf, consultado el 08/octubre de 2009.

Un ejemplo de estructuras de control secuencial combinadas lo encontramos en el siguiente trozo de programa:

```
if (x <= 1.0) then
  printf("x too small\n");
else {
  y = 1.0;
  while (y <= 10.0) {
    y = y*x;
    x = x+0.5;
  }
}
```

El cual se ejecutará según el control como una de las secuencias de instrucciones siguientes:

```
printf("x too small\n");
y = 1.0;
y = 1.0; y = y*x; x = x+0.5;
y = 1.0; y = y*x; x = x+0.5; y = y*x; x = x+0.5;
y = 1.0; y = y*x; x = x+0.5; y = y*x; x = x+0.5; y = y*x; x = x+0.5;
```

El control en sí mismo depende de las variables diferenciadas. Un cambio pequeño de la precedencia de los valores puede dar lugar a un cambio de la secuencia del control.

Bibliografía del tema 2

Bronson Gary, trad. de Jorge Alberto Velázquez Arellano., C++ para ingeniería y ciencias, México, International Thomson, 2007, 826pp.

Bishop, Judy, *Java: fundamentos de programación*, Madrid, Addison Wesley, 1999.

Burnham, W. D., *Prolog: programación y aplicaciones,* México*,* Limusa, 1989.

Deitel Harvey. M. l*. Introducción a los Sistemas Operativos,* México, Addison-Wesley Iberoamericana, 1993.

Gottfried, Byron S. *Programación en C*. 2ª ed., México, McGraw-Hill Interamericana, 2005.

Hansen, Augie. *¡Aprenda C Ya!*, Madrid, Anaya Multimedia, 1990.

Joyanes Aguilar, Luis; Luis Rodríguez Baena; Matilde Fernández Azuela, *Fundamentos de Programación: Libro de Problemas*. México, McGraw-Hill, 1996.

Long, Larry, trad. de María de Lourdes Fournier García, *Introducción a las computadoras y al procesamiento de información*, Prentice Hall Hispanoamericana, Nueva Jersey, 1995.

Stroustrup, Bjarne, *El lenguaje de programación C++,* Madrid, Addison-Wesley, 1993.

Sitios web

- http://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n
- http://es.wikipedia.org/wiki/Plankalk%C3%BCl
- http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp16.pdf
- http://es.wikipedia.org/wiki/COBOL
- http://es.wikipedia.org/wiki/PL/I
- http://es.wikipedia.org/wiki/Python
- http://www.itculiacan.edu.mx/apuntes/maestros/Ram%F3n%20Zatara%EDn/LenguajesAutomatas/LENGUAJESYAUTOMATAS.ppt
- http://elvex.ugr.es/etexts/spanish/pl/algol.htm
- http://delta.cs.cinvestav.mx/~gmorales/ta/ta.html
- http://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/risi/N1_2004/a04.pdf
- http://www.diclib.com/cgibin/d1.cgi?l=es&base=es_wiki_10&page=showid&id=5623
- http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp12.pdf
- http://www.ruby-lang.org/es/
- http://www.perl.org/
- http://tricia-tecnologia.blogspot.com/2007_09_01_archive.html
- http://www.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/
- http://www.angelfire.com/bc3/gabrielagarcia0/Unidad1.htm
- http://neo.lcc.uma.es/staff/francis/fundcomp/tema8.ppt
- http://ingenieria.uatx.mx/~patrick/so/unidad2.pdf
- http://www.masadelante.com/faqs/disco-duro
- http://www.infor.uva.es/~fjgonzalez/apuntes_aso/Tema6.pdf
- http://es.wikipedia.org/wiki/Disco_%C3%B3ptico

- http://es.wikipedia.org/wiki/Cinta_magn%C3%A9tica_de_almacenamiento_de_ datos
- http://weblogs.inf.udp.cl/inf9019-1/files/2006/09/tareacontrol.pdf

Actividades de aprendizaje

A.2.1 Elabora un mapa conceptual de lo estudiado en el tema dos, con el propósito de que tengas un panorama global de la programación e implementación de sistemas, apoyándote con el temario propuesto y la bibliografía sugerida.

A.2.2 Elabora en un documento Word, un resumen no mayor de tres páginas donde resaltes qué y cuáles son los lenguajes de programación y las etapas por las que pasa un programa desde su creación para poder ser ejecutado.

A.2.3 Elabora en una diapositiva en PowerPoint, un cuadro sinóptico en donde resaltes las características de los diferentes tipos de datos existentes para programación.

Cuestionario de autoevaluación

1. Menciona algunos de los lenguajes de programación más importantes.
2. Explica brevemente la creación del lenguaje C.
3. Da algunas características del lenguaje EVA.
4. ¿Cuáles son las etapas de traducción de un programa?
5. ¿Qué son los tipos de datos?
6. ¿Cuáles son los principales tipos de datos?
7. ¿Qué son los tipos de datos abstractos?
8. ¿Qué es una cola?
9. ¿Qué es el control de secuencia?
10. ¿Qué es una pila?

Examen de autoevaluación

Subraya el inciso con la respuesta correcta

1. Lenguaje de programación que se utiliza principalmente en aplicaciones científicas y análisis numérico desarrollado en los años 50.

a) C++
b) FORTRAN
c) JAVA
d) LISP

2. Lenguaje de programación que fue creado en el año 1960 con el objetivo de crear un lenguaje de programación universal que pudiera ser usado en cualquier computadora y que estuviera orientado principalmente a los negocios.

a) Pascal
b) C
c) COBOL
d) LISP

3. Lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. Es el lenguaje de programación más popular para crear software de sistemas.

a) C
b) C++
c) Pascal
d) JAVA

4. Es el lenguaje de consulta estructurado, declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

a) JAVA
b) COBOL
c) SQL
d) C++

5. Lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

a) SQL
b) C++
c) JAVA
d) Pascal

6. Es un lenguaje de programación dinámico y de código abierto que incorporara tanto la programación funcional como la programación imperativa y que fue liberado en 1995

a) C++
b) JAVA
c) ADA
d) RUBY

7. Es una lista de elementos de la cual sólo se puede extraer el último elemento insertado

a) Pila
b) Control de secuencia
c) Autómata
d) Cola

8. Es una herramienta que nos permite determinar la estructura sintáctica de un programa fuente

a) Analizador léxico
b) Analizador gramatical
c) Analizador verbal
d) Analizador sintáctico

9. Son mecanismos que generan gramáticas formales

a) Autómata
b) Control de secuencia
c) Pila
d) Cola

10. Son tipos de datos a los que se añaden propiedades de reutilización y de compartición de código.

a) TDA
b) Tipos definidos
c) Tipos de datos
d) Objetos

TEMA 3. COMPILADORES

Objetivo particular

Al finalizar el tema, el alumno identificará qué es un compilador, cómo está estructurado y sus diversas etapas para realizar el proceso de compilación de un programa.

Temario detallado

3.1 Fases de la compilación
3.2 Análisis lexicográfico
3.3 Análisis sintáctico
3.4 Análisis semántico
3.5 Optimización
3.6 Preparación para la generación del código
3.7 Generación del código
3.8 Ejemplo práctico de las fases de compilación en un lenguaje de programación.

Introducción

La compilación de la gramática de un lenguaje es por mucho una de las etapas más sobresalientes en el desarrollo de programas, por lo que el conocer en qué consiste este mecanismo es de suma importancia en la creación de sistemas de información.

3.1. Fases de la compilación

Un compilador opera como una secuencia de fases, cada una de las cuales transforma una representación del programa fuente en otro.

En la práctica las fases pueden agruparse en análisis y síntesis, pero si nos adentramos más en el tema podemos estructurar el proceso de compilación en 5 grandes etapas: analizador léxico, analizador sintáctico, analizador semántico, generador de código intermedio y optimizador de código.[53]

Durante el análisis se divide el programa fuente en componentes y se genera una estructura gramatical. Después, esa estructura gramatical se utiliza para crear una representación intermedia del programa fuente. Si durante el análisis se detecta que el programa fuente está mal respecto a la sintaxis o a la semántica, se genera un mensaje para que el programador pueda corregirlo. También, durante el análisis se recolecta información sobre el programa fuente que es almacenado en una estructura denominada tabla de símbolos, la cual se pasa junto con la representación intermedia a la parte de la síntesis y es utilizada durante todas las fases de la compilación. En la parte de la síntesis se construye el programa objeto.

Al análisis se le conoce como *front end* del compilador; a la síntesis, que es la traducción propiamente dicha, se le denomina *back end.*

3.2. Análisis lexicográfico

Es la primera fase de un compilador. Consiste en leer el flujo de caracteres que componen el programa fuente y agruparlos en secuencias significativas conocidas como lexemas. Para cada lexema, el analizador produce un token,

[53] Cf., Universidad de Huelva: “El proceso de compilación, del código fuente al código máquina”, disponible en línea: http://www.uhu.es/04004/material/Transparencias3.pdf, recuperado el 14/09/09.

formado por el nombre del token[54] y su valor (que es la dirección en la tabla de símbolos asignada a este token) y será usada para el análisis semántico y la generación de código.[55]

El analizador léxico realiza otras tareas aparte de identificar lexemas, entre éstas se encuentran:

- Eliminar los comentarios y espacios en blanco.
- Correlacionar con el programa fuente los mensajes de error generados por el compilador

Cuando se habla de análisis léxico es necesario comprender tres términos muy importantes que durante esta fase están relacionados:

- **Token**: es una estructura compuesta por un nombre de token y un valor de atributo. El nombre es un símbolo de entrada que representa un tipo de unidad léxica, es decir, es como un sinónimo de un Id.

- **Patrón**: es la descripción de la forma que pueden tomar los lexemas en un token.

- **Lexema**: es la secuencia de caracteres en el programa fuente, que coincide con el patrón para un token y que el analizador léxico identifica como una instancia de ese token.

[54] Símbolo abstracto que se utiliza durante el análisis sintáctico. Estos símbolos hacen referencia a una secuencia de caracteres de entrada y son procesados más tarde para construir la estructura de datos.
[55] Cf., *Revista Ciencias Matemáticas*: "Análisis de modelos para la representación de lenguajes", vol. 20, N° 2, 2002, material disponible en línea: http://www.dict.uh.cu/Revistas/CM2002_2003/CM02202k.doc, recuperado el 14/09/09.

3.3 Análisis sintáctico

La sintaxis de construcción de un lenguaje de programación puede especificarse mediante gramáticas libres de contexto o mediante la notación BNF (**Backus-Naur form**) vista en el tema 2.2. Los beneficios de las gramáticas son muy importantes pues entre otras cosas:

- Proporcionan una especificación precisa, pero fácil de entender, de un lenguaje de programación.
- A partir de ciertos tipos de gramáticas se puede construir automáticamente analizadores sintácticos.
- Son útiles para traducir programas fuente en un adecuado código objeto.
- Detectan errores.
- Permiten a los lenguajes evolucionar o desarrollarse de manera interactiva.

En análisis sintáctico es la segunda fase de un compilador y es conocido también como *Parsing*. El *parser* o analizador sintáctico, propiamente dicho, utiliza los primeros componentes de los tokens producidos por el analizador léxico para crear una representación intermedia en forma de árbol que describa la estructura gramatical del flujo de tokens. Cada nodo interior de ese árbol sintáctico representa una operación y los hijos del nodo representan los argumentos de la operación. El árbol sintáctico se pasa al resto del compilador para que se siga procesando.

Un analizador sintáctico tiene, entre sus objetivos:

- Reportar la presencia de errores con claridad y precisión.
- Recuperarse de cada error tan rápido como para poder detectar el siguiente error.

Los errores de programación comunes ocurren en diferentes niveles: errores léxicos, errores sintácticos, errores semánticos y errores lógicos.

- Los errores semánticos incluyen los conflictos de tipos entre operadores y operandos. Por ejemplo una estructura *for* que tenga mal escrita la variable de inicialización.
- Los errores léxicos incluyen la escritura incorrecta de los operadores, de los identificadores o de las palabras clave.
- Los errores sintácticos incluyen la colocación incorrecta de signos de puntuación como el punto y coma (;) y las llaves ({ }).
- Los errores lógicos incluyen razonamientos incorrectos del programador en el uso de operadores.[56]

3.4 Analizador semántico

El analizador semántico es la tercera fase del compilador. Entre sus principales funciones se encuentran:

- Utilizar el árbol sintáctico y la información en la tabla de símbolos para comprobar la consistencia semántica del programa fuente contra la definición del lenguaje.
- Recopilar información sobre el tipo en el árbol sintáctico o en la tabla de símbolos, para usarla más tarde durante la generación del código intermedio.
- Verificar que cada operador tenga operandos que coincidan.
- Producir un código intermedio.[57]

[56] Cf., Brian Kernighan y Dennis Ritchie; *El Lenguaje de Programación C*, 2ª ed., México, Prentice-Hall, 1991, p. 187.
[57] Con base en Mitotecnológico, "Analizador semántico", material en línea, disponible en: http://www.mitecnologico.com/Main/AnalizadorSemantico, recuperado el 14/09/09.

3.5. Optimización

Es una fase independiente del procesador y tiene como objetivo mejorar el código intermedio, de manera que se produzca un mejor código destino. Mejorar en términos de tiempo, de requerimientos de memoria y de uso de recursos.

Para obtener un mejor código destino, en ocasiones se usan algoritmos simples de generación de código intermedio.

La optimización de código varía mucho entre cada compilador. Hay algunos que invierten mucho tiempo en esta fase y realizan una mayor optimización, a estos se les conoce como “compiladores optimizadores”. Hay otros que solo mejoran el tiempo de ejecución sin repercutir demasiado con la velocidad de compilación y se les conoce como “compiladores simples”.[58]

3.6 Preparación para la generación del código

La preparación de código consiste en que el compilador verifique línea por línea todas y cada una de las expresiones escritas dentro del editor del lenguaje, verificando en las librerías la existencia de cada palabra reservada, tipo de dato u operador matemático y lógico; es decir, se verifica la sintaxis gramatical del lenguaje, la cual debe coincidir con las oraciones y sentencias escritas por el programador.

3.7 Generación de código[59]

Fase del compilador que recibe una representación intermedia del programa fuente y la asigna al código destino. Si el código destino es código máquina, se seleccionan registros o localidades de memoria para cada una de las variables que utiliza el programa. Después las instrucciones intermedias se traducen en

[58] Véase, Wikipedia: “Optimización de software”, actualizado el 13/02/09, disponible en línea: http://es.wikipedia.org/wiki/Optimizaci%C3%B3n_de_software, recuperado el 14/09/09.

[59] Véase, Aho Alfred V.; *Compiladores. Principios, técnicas y herramientas*. México, Pearson Educación, 2008. p 10.

secuencias de instrucciones de máquina que realizan la misma tarea. Un aspecto crucial de la generación de código es la asignación juiciosa de los registros para guardar las variables.

> La generación de código es usada para construir de manera automática programas, evitándoles a los programadores el trabajo manual. Esta generación de código puede llevarse a cabo en tiempo de ejecución, en tiempo de carga, o bien, en tiempo de compilación.[60]

3.8. Ejemplo práctico de las fases de compilación en un lenguaje de programación

Ejemplo en C++
Desde el block de notas podemos escribir en el archivo ejemplo.c:

```
#include <stdio.h>

int main(){
   printf("ejemplo !\n");
   return 0;
}
```

1. Compilación

En Linux llamamos directamente a gcc (-W y –Wall permiten mostrar más mensajes para verificar si el código es “limpio”, -o ejemplo.exe indica que el ejecutable que será creado debe llamarse ejemplo.exe):

```
gcc – W – Wall –o ejemplo.exe ejemplo.c
```

Implícitamente el compilador hace tres etapas (preprocesado, compilación y enlazado).

[60] Wikipedia: “Generación de código”, actualizado el 13/12/08, material en línea, disponible en: http://es.wikipedia.org/wiki/Generaci%C3%B3n_de_c%C3%B3digo, recuperado el 14/09/09.

1) El preprocesado

/*Todo lo que es definido por <stdio.h>, incluyendo printf() */

int main(){

    printf(“ejemplo!\n");

---

    return 0;

}

2) Compilación

Encuentra sin problemas printf ya que éste es declarado en <stdio.h>

3) Enlazado

Encuentra sin problemas printf en el binario de la lib c. También lo podemos verificar bajo linux con ldd:

ldd ejemplo.exe

Lo que da:

linux-gate.so.1 => (0xbtf2b000)

    libc.so.6 => /lib/i686/cmov/livc.so.6 (0xb7dbb000)

    /lib/ld-linux.so.2 (0xb7f2c000)

- En la segunda línea podemos ver que utiliza la lib c.

Luego crea ejemplo.exe

Por otra parte comprobamos que no hay error ni warning.

---

2. Ejecución

Tan solo queda ejecutarlo:

./ejemplo.exe

Lo que da como resultado:

ejemplo ![61]

Bibliografía del tema 3

Kernighan, Brian W.; Ritchie, Dennis M., *El Lenguaje de Programación C*. 2ª edición, México, Prentice-Hall, 1991.

Aho, Alfred, B.; *Compiladores. Principios, técnicas y herramientas*. México, Pearson Educación, 2008.

Sitios web

- http://www.uhu.es/04004/material/Transparencias3.pdf
- http://www.dict.uh.cu/Revistas/CM2002_2003/CM02202k.doc
- http://www.mitecnologico.com/Main/AnalizadorSemantico
- http://es.wikipedia.org/wiki/Optimizaci%C3%B3n_de_software
- http://es.wikipedia.org/wiki/Generaci%C3%B3n_de_c%C3%B3digo
- http://es.kioskea.net/faq/sujet-2821-la-compilacion-y-los-modulos-en-c-y-c

[61] Kioskea "La compilación y los módulos en C y C++", actualizado el 03/06/09, disponible en línea: http://es.kioskea.net/faq/sujet-2821-la-compilacion-y-los-modulos-en-c-y-c, recuperado el 29/09/09.

Actividades de aprendizaje

A.3.1 Elabora un mapa conceptual de lo estudiado en el tema tres, con el propósito de que tengas un panorama global de la programación e implementación de sistemas, apoyándote con el temario propuesto y la bibliografía sugerida.

A.3.2 Elabora un resumen no mayor de dos páginas en un documento Word, en donde resaltes las características de cada una de estas fases hasta la generación del código.

A.3.3 Elabora en una diapositiva en PowerPoint, un cuadro sinóptico en donde resaltes las características y diferencias del análisis sintáctico y semántico.

Cuestionario de autoevaluación

1. ¿Cuáles son las fases para la compilación de un programa?
2. ¿Qué es un programa fuente?
3. ¿Qué es un programa ejecutable?
4. ¿Qué es un token?
5. ¿Cuáles son los tipos de errores que se pueden generar en la fase de compilación?
6. ¿Qué es el análisis lexicográfico?
7. ¿Qué es el análisis sintáctico?
8. ¿Qué es el análisis semántico?
9. ¿Qué es la optimización de software?
10. ¿Qué es la generación de código?

Examen de autoevaluación

1. Programa escrito en un lenguaje de alto nivel (texto ordinario que contiene las sentencias del programa en un lenguaje de programación). Necesita ser traducido a código máquina para poder ser ejecutado.

a) Programa objeto
b) Compilador
c) Programa fuente
d) Programa ejecutable

2. Programa encargado de traducir los programas fuentes escritos en un lenguaje de alto nivel a lenguaje máquina y de comprobar que las llamadas a las funciones de librería se realizan correctamente.

a) Programa fuente
b) Programa ejecutable
c) Compilador
d) Analizador lexicográfico

3. Es el programa encargado de insertar al programa objeto el código máquina de las funciones de las librerías (archivos de biblioteca) usadas en el programa.

a) Programa fuente
b) Compilador
c) Programa objeto
d) *Linker*

4. Son errores que se producen antes de la ejecución del programa, durante el proceso de compilación del programa.

a) Errores en tiempo de ejecución
b) Errores en tiempo de compilación
c) *Warnings*
d) Errores fatales

5. Indican que hay líneas de código sospechosas que, a pesar de no infringir ninguna regla sintáctica, el compilador las encuentra susceptibles de provocar un error.

a) Errores de compilación
b) Errores de tiempo de ejecución
c) Errores de sintaxis
d) *Warnings*

6. Análisis que revisa la forma de estructuras de las sentencias en un lenguaje de programación, es determinista y rechaza aquellos lenguajes potencialmente ambiguos.

a) Análisis sintáctico
b) Análisis semántico
c) Análisis lexicográfico
d) Análisis de archivos

7. Análisis basado en el árbol sintáctico de un lenguaje, es decir está sustentado en las reglas gramaticales del programa; permitiendo comprobar restricciones de tipo y otras limitaciones, como suele ocurrir en lenguaje C

a) Sintáctico
b) Semántico
c) Lexicográfico
d) *Lookahead*

8. Notación llamada *polaca inversa*, se usa para representar expresiones sin necesidad de paréntesis.

a) Prefija.
b) Infija
c) Sufija
d) De cuádruplas

9. Fase del compilador que trata de convertir programas existentes en otros programas que realicen las mismas tareas en menos tiempo, con menos requerimientos de memoria.

a) Optimización
b) Reingeniería
c) Orientación a objetos
d) Cliente-servidor

10 Fase mediante la cual un compilador convierte un programa sintácticamente correcto en una serie de instrucciones para ser interpretadas por una máquina.

a) Generación de código
b) Compilación
c) Optimización
d) Manipulación de datos

TEMA 4. SISTEMAS OPERATIVOS

Objetivo particular

Al finalizar el tema el alumno reconocerá la importancia, la aplicación y el funcionamiento que tienen los sistemas operativos en cuanto a la administración de los recursos de las computadoras.

Introducción

En la actualidad los sistemas operativos no solo están implícitos en una computadora, también se encuentran en agendas electrónicas, celulares y en muchos otros artículos, con el fin de permitir la interacción usuario-máquina.

Por mucho se dice que un sistema operativo es un traductor que permite plasmar nuestras ideas en un ordenador y obtener resultados maravillosos.

Temario detallado

4.1 Historia de los sistemas operativos
4.2 Estructura de los sistemas operativos
4.3 Procesos
    4.3.1 Introducción a los procesos
    4.3.2 Hilos
    4.3.3 Sincronización de los procesos
4.4 Bloqueos
4.5 La gestión de Entrada/Salida
    4.5.1 Principios de entrada/salida de hardware
    4.5.2 Principios de entrada/salida de software
4.6 Gestión de la memoria
4.7 Gestión de archivos

4.8 Sistemas distribuidos

4.9 Seguridad

4.10. Ejemplo práctico de un sistema operativo (UNIX, Linux, Windows)

4.1 Historia de los Sistemas Operativos[62]

Para tratar de comprender los requisitos de un Sistema Operativo y el significado de las principales características de un Sistema Operativo contemporáneo, es útil considerar como han ido evolucionando éstos con el tiempo.

Existen diferentes enfoques o versiones de cómo han ido evolucionando los Sistemas Operativos.

La primera de estas versiones podría ser:

En los años 40, se introducen los programas bit a bit, por medio de interruptores mecánicos y después se introdujo el lenguaje máquina que trabajaba por tarjetas perforadas.

Con las primeras computadoras, desde finales de los años 40 hasta la mitad de los años 50, el programador interactuaba de manera directa con el hardware de la computadora, no existía realmente un Sistema Operativo; las primeras computadoras utilizaban bulbos, la entrada de datos y los programas se realizaban a través del lenguaje máquina (bits) o a través de interruptores.

A principios de los 50, la compañía General's Motors implantó el primer sistema operativo para su IBM 170. Empiezan a surgir las tarjetas perforadas las cuales permiten que los usuarios (que en ese tiempo eran programadores, diseñadores, capturistas, etc.), se encarguen de modificar sus programas. Establecían o apartaban tiempo, metían o introducían sus programas, corregían y depuraban sus programas en su tiempo. A esto se le llamaba trabajo en serie. Todo esto se traducía en pérdida de tiempo y tiempos de programas excesivos.

[62] Véase, Andrew Tanenbaum, *Sistemas operativos modernos*, 2ª ed., México, Pearson Prentice-Hall, 2003, p 6.

En los años 60 y 70 se genera el circuito integrado, se organizan los trabajos y se generan los procesos Batch (por lotes), lo cual consiste en determinar los trabajos comunes y realizarlos todos juntos de una sola vez. En esta época surgen las unidades de cinta y el cargador de programas, el cual se considera como el primer tipo de Sistema Operativo.

En los 80, inició el auge de la INTERNET en los Estados Unidos de América. A finales de los años 80 comienza el gran auge y evolución de los Sistemas Operativos. Se descubre el concepto de multiprogramación que consiste en tener cargados en memoria a varios trabajos al mismo tiempo, tema principal de los Sistemas Operativos actuales.

Los 90 y el futuro, entramos a la era de la computación distribuida y del multiprocesamiento a través de múltiples redes de computadoras, aprovechando el ciclo del procesador.

Se tendrá una configuración dinámica con un reconocimiento inmediato de dispositivos y software que se añada o elimine de las redes a través de procesos de registro y localizadores.

La conectividad se facilita gracias a estándares y protocolos de sistemas abiertos por organizaciones como la Organización Internacional de normas, fundación de software abierto, todo estará más controlado por los protocolos de comunicación OSI y por la red de servicios digital ISDN.[63]

4.2. Estructura de los Sistemas Operativos

Las computadoras han evolucionado de una manera impresionante, hoy en día son las herramientas más empleadas por el hombre y cada día se tratan de hacer más eficientes, precisas y veloces. Esta evolución de las computadoras va de la mano con el software, que es todo aquel programa que no podemos ver físicamente (componente lógico de la computadora), pero sí los podemos usar para redactar un texto, navegar por Internet, programar, etc.

[63] Instituto Tecnológico de Campeche: "Historia de los sistemas operativos", material en línea, disponible en: www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r134.DOC, recuperado el 14/09/09.

El software se puede clasificar de dos formas: los programas de aplicación (todos aquellos que usan los usuarios como Word, Excel, etc.) y los programas de sistema (son los encargados de controlar las operaciones que realiza la computadora). El principal programa del sistema es el llamado Sistema Operativo, el cual es el responsable del control de todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación.

> Un Sistema Operativo es un programa que actúa como intermediario entre el usuario y el hardware de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas. El objetivo principal de un Sistema Operativo es, entonces, lograr que el sistema de computación se use de manera cómoda, y el objetivo secundario es que el hardware del computador se emplee de manera eficiente.

Clasificación de los Sistemas Operativos[64]

Los Sistemas Operativos fueron clasificándose con el paso del tiempo, dependiendo de la gestión de memoria, de la planificación del procesador, de la gestión de entrada/salida, de la gestión de archivos o de la aplicación que se les daba.

Sistemas Operativos por lotes[65]

Fueron introducidos alrededor de 1956 para procesar grandes cantidades de información con poca o ninguna interacción entre el usuario y los programas en ejecución.

La planificación en este tipo de sistemas es muy sencilla ya que estos sistemas juntan todos los trabajos del mismo tipo o con necesidades similares y los procesan al mismo tiempo, como un grupo. Si son bien planeados la secuencia y el tiempo de ejecución de trabajos son efectivos.

---

[64] Para este tema, Cf., Harvey Deitel, *Introducción a los Sistemas Operativos*, México, Addison-Wesley Iberoamericana, 1993, pp. 105-110.

[65] Cf., Milan Milenkovic, *Sistemas Operativos: conceptos y diseños*, Madrid, McGraw Hill, 1994, pp. 12.

Algunas características con que cuentan los Sistemas Operativos por lotes son:

- Permiten poca o ninguna interacción usuario/programa en ejecución.
- Gestión de memoria sencilla, generalmente se divide en dos: parte residente del S.O. y programas transitorios.
- No requieren gestión crítica de dispositivos en el tiempo.
- Suelen proporcionar gestión sencilla de manejo de archivos: se requiere poca protección y ningún control de concurrencia para el acceso.

Sistemas Operativos de tiempo real[66]

Los Sistemas Operativos de tiempo real son aquellos cuyo parámetro importante es el tiempo. Estos sistemas se dividen a su vez en rigurosos y no rigurosos. Los primeros son aquellos en los que es indispensable que un evento se efectúe en cierto momento o dentro de cierto intervalo; los segundos son aquellos en los que es aceptable no cumplir de vez en cuando con un plazo.

Un objetivo importante de este tipo de sistemas es proporcionar rápidos tiempos de respuesta a eventos y satisfacer así los plazos de planificación.

Algunos ejemplos de Sistemas Operativos de tiempo real son: VxWorks, y QNX. Los Sistemas Operativos de tiempo real cuentan con las siguientes características:[67]

- Procesos explícitos definidos y controlados por el programador
- Cada proceso tiene asignado un cierto nivel de prioridad
- El proceso se activa tras la ocurrencia de un suceso
- El Proceso de mayor prioridad expropia recursos de los procesos de prioridad inferior.

[66] Cf., Andrew Tanenbaum, *Sistemas operativos modernos*, 2ª ed., México, Pearson Prentice-Hall, 2003, p. 19

[67] Cf., Milan Milenkovic, *Sistemas Operativos: conceptos y diseños*, Madrid, McGraw-Hill, 1994, pp. 15-16.

- La gestión de memoria es menos exigente que en otros tipos de sistemas multiprogramación.
- La población de procesos es estática en gran medida.
- La gestión de archivos se orienta más a velocidad de acceso que a utilización eficiente del recurso.

Sistemas Operativos de mainframe[68]

Son sistemas operativos utilizados por computadoras gigantes ubicadas en importantes centros corporativos de datos. Están claramente orientados al procesamiento de varios trabajos a la vez, que requieren enormes cantidades de E/S.

Entre sus características se encuentran:

- Ofrecen servicios por lotes, procesamiento de transacciones y tiempo compartido
- Procesan trabajos rutinarios sin que haya un usuario interactivo presente
- Manejan numerosas solicitudes pequeñas
- Permiten a múltiples usuarios remotos ejecutar trabajos de manera simultánea
- Un ejemplo de este tipo de sistema operativo es OS/390

[68] Cf., Andrew Tanenbaum, *Sistemas operativos modernos*, México, Pearson Prentice Hall, 2ª ed., 2003, p. 18

Sistemas Operativos de multiprogramación (Sistemas Operativos de multitarea)[69]

Tipo de sistemas operativos que llevan generalmente el prefijo multi en sus nombres (multitarea, multiacceso, multiusuario, multiprocesamiento). Su característica principal es que además de soportar la multitarea, proporciona formas de protección de memoria más sofisticadas que otros tipos de sistemas y obliga el control de la concurrencia cuando los procesos acceden a archivos compartidos y dispositivos de E/S

Las características de un Sistema Operativo de multiprogramación o multitarea son las siguientes:

- Generalmente soportan múltiples usuarios (multiusuarios).
- Multiplexan los recursos de un sistema informático entre una multitud de programas activos.
- Gestionan la operación de sistemas informáticos que incorporan varios procesadores.
- Proporcionan facilidades para mantener el entorno de usuarios individuales.
- Proporcionan contabilidad del uso de los recursos por parte de los usuarios.
- Son multiprocesadores y multitareas por definición ya que soportan la ejecución simultánea de múltiples tareas sobre diferentes procesadores.
- Se caracterizan por tener múltiples programas activos compitiendo por los recursos del sistema: procesador, memoria, dispositivos periféricos.

[69] Cf., Milan Milenkovic, *Sistemas Operativos: conceptos y diseños*, Madrid, McGraw Hill, 1994, p. 13.

Sistemas Operativos de tiempo compartido[70]

Los sistemas operativos de tiempo compartido permiten a varios usuarios compartir la computadora simultáneamente. Estos sistemas hacen uso de la planificación del CPU y la multiprogramación para proporcionar a cada usuario una pequeña porción de la computadora.

Este tipo de sistemas son más complejos que los de multiprogramación pues incluyen procesos más sofisticados para la administración y protección de memoria.

En un sistema de tiempo compartido, el CPU ejecuta múltiples trabajos de manera conmutada pero los cambios se realizan de manera tan ágil que los usuarios pueden interactuar con cada programa mientras está en ejecución.

Algunas características de los Sistemas Operativos de tiempo compartido son[71]:

- Dan la ilusión de que cada usuario tiene una máquina para sí.
- La mayoría utilizan algoritmo de reparto circular del tiempo para la planificación.
- Los programas se ejecutan con prioridad rotatoria que se incrementa con la espera y disminuye después de concedido el servicio.
- La gestión de memoria proporciona aislamiento y protección a programas residentes.
- La gestión de E/S es sofisticada pues trata con múltiples usuarios y dispositivos
- Proporcionan mecanismos para la ejecución concurrente, sincronización y comunicación de los trabajos

[70] Cf., Abraham Silberschatz, et. al., *Sistemas Operativos*, México, Limusa Wiley, 2002, pp. 8-9.
[71] Con base en Milan Milenkovic, *Sistemas Operativos: conceptos y diseños*, Madrid, MacGraw-Hill, 1994, pp. 14.

Sistemas Operativos distribuidos[72]

Sistemas informáticos autónomos capaces de cooperar y comunicarse mediante interconexiones de hardware y software. Evolucionaron a partir del crecimiento de las redes de computadoras, especialmente la Internet y la WWW.

Algunas características de los Sistemas Operativos distribuidos son:

- Cada procesador tiene su memoria local
- Proporcionan una abstracción de máquina virtual a sus usuarios
- La distribución de componentes y recursos es transparente para los usuarios
- Proporcionan medios para la gestión de recursos del sistema
- Facilita el acceso a recursos remotos, la comunicación con procesos remotos y la distribución de los cálculos.

Sistemas Operativos de red[73]

Son aquellos sistemas que proporcionan características como la compartición de archivos de la red, e incluyen un esquema de comunicación que permite que procesos diferentes en computadoras diferentes, intercambien mensajes. Algunos autores los mencionan como un subtipo de sistema operativo distribuido.

[72] Ibid, pp. 17.
[73] Con base en Abraham Silberschatz, et. al., *Sistemas Operativos*, México, Limusa Wiley, 2002, pp. 15.

Sistemas Operativos paralelos[74]

Son sistemas que tienen más de un procesador compartiendo el bus de la computadora, el reloj y en ocasiones la memoria y dispositivos periféricos[75]. Se les conoce también como sistemas fuertemente acoplados.

Algunas características de los Sistemas Operativos paralelos son:

- Realizan más trabajo en menos tiempo
- Ejecutan programas atendiendo de manera concurrente varios procesos de un mismo usuario
- Proporcionan servicios de manera proporcional al nivel de hardware
- Son tolerantes a fallas
- Cada procesador cuenta con su propia memoria local
- Se mantienen copias de cada proceso
- Permiten compartir de manera dinámica procesos y recursos entre los diferentes procesadores.

4.3 Procesos

Todos los programas cuya ejecución solicitan los usuarios, se ejecutan en forma de procesos, de ahí la importancia para el informático de conocerlos en detalle. El proceso se puede definir como un **programa** gestionado por el sistema operativo. Durante su elección el proceso se va modificando **en ejecución**.

El sistema operativo mantiene por cada proceso una serie de estructuras de información que permiten identificar las características de este, así como los recursos que tiene asignados. En esta última categoría entran los descriptores de los segmentos de memoria asignados, los descriptores de los archivos abiertos, los descriptores de los puertos de comunicaciones, etc.

[74] Ibid, pp. 11-13.

[75] Abraham Silberschatz, et. al., *Sistemas Operativos*, México, Limusa Wiley, 2002, pp. 11.

Una parte muy importante de esta información se encuentra normalmente como en el llamado **bloque de control de procesos (BCP)**. El sistema operativo mantiene una tabla de procesos con todos los BCP de los procesos. Por razones de eficiencia, la tabla de procesos se construye normalmente como una estructura estática, que tiene un determinado número de BCP, todos ellos del mismo tamaño. La información que compone un proceso es la siguiente:

- Contenido de los segmentos de memoria en los que residen el código y los datos del proceso. A esta información se le denomina imagen de memoria o core image.
- Contenido de los registros del modelo de programación
- Contenido del BCP.[76]

4.3.1 Introducción a los procesos

El término proceso fue utilizado por primera vez por los diseñadores del sistema Multics en los años 60's. Desde entonces, el término proceso, utilizado a veces como sinónimo de tarea, ha tenido muchas definiciones. A continuación se presentan algunas:

- Un programa en ejecución
- Una actividad asíncrona
- El "espíritu animado" de un procedimiento
- El "centro de control" de un procedimiento en ejecución
- Lo que se manifiesta por la existencia de un "bloque de control del proceso" en el sistema operativo
- La entidad a la que se asignan los procesadores
- La unidad "despachable"

Aunque se han dado muchas otras definiciones, no hay una definición universalmente aceptada, pero el concepto de "Programa en ejecución" parece ser el que se utiliza con más frecuencia. Un programa es una entidad inanimada; sólo cuando un procesador le "infunde vida" se convierte en la entidad "activa" que se denomina proceso.

[76] Universidad Nacional de Chimborazo: “Procesos: concepto de proceso”, material en línea, disponible en: http://www.unach.edu.ec/Virtualizacion/Sistemas_Operativos/paginas/Unidad%202.htm, recuperado el 04/05/09.

Un proceso pasa por una serie de datos discretos. Se dice que un proceso se está ejecutando (estado de ejecución), si tiene asignada la CPU. Se dice que un proceso está listo (estado listo) si puede utilizar una CPU en caso de haber una disponible. Un proceso está bloqueado (estado bloqueado) si está esperando que suceda algún evento antes de poder seguir la ejecución.[77]

Jerarquía de procesos

La secuencia de creación de procesos genera un árbol de procesos. Para referirse a las relaciones dentro los procesos de la jerarquía se emplean los términos padre, hijo, hermano o abuelo. Cuando el proceso **A** solicita al sistema operativo que cree el proceso **B**, se dice que **A** es padre de **B** y que **B** es hijo de **A**. Bajo esta óptica, la jerarquía de procesos puede considerarse como un árbol genealógico.

Algunos sistema operativos, como UNIX, mantienen de forma explícita esa estructura jerárquica de procesos – un proceso sabe quién es su padre -, mientras que otros sistemas operativos, como Windows NT, no lo mantiene.[78]

Estados de los procesos

Los procesos pueden caer en alguno de estos 5 estados:

- **Nuevo**: El proceso es creado.
- **Ejecución**: Se ejecutan instrucciones.
- **Espera**: El proceso está en espera por la ocurrencia de algún evento.
- **Listo**: El proceso está esperando a que le asignen el procesador.
- **Terminado**: El proceso finaliza su ejecución.

Y estos procesos van a tener siempre la siguiente información asociada a ellos:

- Estado del proceso
- Program counter
- Registros del CPU
- Información de planificación del CPU
- Memoria

[77] Instituto Tecnológico de Campeche: "Administración de procesos: concepto de proceso", material electrónico disponible en línea, en: www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r139.DOC, recuperado el 14/09/09.

[78] Universidad Nacional de Chimborazo: "Procesos: concepto de proceso, jerarquía", material disponible en línea: http://www.unach.edu.ec/Virtualizacion/Sistemas_Operativos/paginas/Unidad%202.htm, recuperado el 04/05/09.

- Información para administración
- Información de estatus de E/S[79]

Creación del proceso

Al crear un proceso se le asigna memoria (para cargado de código, datos y stack), recursos, información del PCB, y se carga. Además se inicializan registros para protección del espacio de memoria reservada a este proceso.

El proceso padre puede crear procesos hijos, los cuales a su vez pueden crear otros procesos, formando así un árbol de procesos.

Existen diversas formas de compartir los recursos entre los procesos:
- Los padres e hijos comparten todos los recursos.
- El hijo comparte un subconjunto de los recursos del padre.
- El hijo y el padre no comparten recursos.

Ahora bien, para que el sistema operativo seleccione el proceso a ejecutarse, es necesario llevar a cabo una planificación de procesos entre cuyos objetivos se encuentran: ocupación del CPU al 100%, oportunidad, minimización de tiempos y número de tareas procesadas.[80]

La selección del proceso a ejecutar se basa en algún algoritmo de planificación.

Terminación de procesos

La terminación de un proceso puede ser ejemplificada con el siguiente algoritmo:

- El proceso ejecuta su última instrucción y pide al sistema que lo elimine (**exit**).
  - Los datos de salida van de hijo a padre (por **fork**).
  - Los recursos del proceso son desalojados por el sistema operativo.

[79] Monografías: "Administración de procesos", material en línea disponible en: http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml, recuperado el 19/09/09.

[80] Monografías: "Administración de procesos", material en línea disponible en: http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml, recuperado el 19/09/09.

- El padre puede terminar la ejecución de un hijo (**abort**).
  - El proceso hijo se ha excedido en los recursos alojados.
  - La tarea asignada al proceso hijo ya no es requerida.
  - El proceso padre termina.
    - El sistema operativo no permite que el hijo continúe su ejecución si el proceso padre termina.
    - Terminación en cascada.

4.3.2 Hilos

> Un hilo de ejecución es una característica que permite a una aplicación realizar varias tareas al mismo tiempo. Los hilos de ejecución que comparten los mismos recursos conforman un proceso y al ser modificados pueden modificar el proceso completo.[81]

Siempre que un proceso se ejecute, el hilo estará activo, cuando el proceso termine, también sus hilos lo harán. La diferencia entre hilos y procesos radica en la dependencia. Los hilos son dependientes de los procesos y los procesos son independientes de otros procesos e información.

4.3.3 Sincronización de procesos.

> Cuando dos o más procesos llegan al mismo tiempo a ejecutarse, se dice que se ha presentado una concurrencia de procesos. Es importante mencionar que para que dos o más procesos sean concurrentes, es necesario que tengan alguna relación entre ellos como puede ser la cooperación para un determinado trabajo o el uso de información o recursos compartidos, por ejemplo: en un sistema de un procesador, la multiprogramación es una condición necesaria pero no suficiente para que exista concurrencia, ya que los procesos pueden ejecutarse de forma totalmente independiente[82].

[81] Wikipedia: "Hilos de ejecución", actualizado el 05/05/09, material en línea, disponible en: http://es.wikipedia.org/wiki/Hilo_de_ejecuci%C3%B3n, recuperado el 14/09/09.
[82] Universidad de los Andes, "Procesamiento en paralelo: semáforos", disponible en línea: http://agamenon.uniandes.edu.co/~c21437/e-valenc/inf_semaforos.html, recuperado el 30/09/09.

Por otro lado en un sistema de varios procesos se puede presentar la concurrencia siempre y cuando las actividades necesiten actuar entre sí ya sea para utilizar información en común o para cualquier otra cosa. Existen tres formas modelos de computadora en los que se puede pueden ejecutar procesos concurrentes:[83]

1. Multiprogramación con un único procesador
En este modelo todos los procesos concurrentes se ejecutan sobre un único procesador. El sistema operativo se encarga de ir repartiendo el tiempo del procesador entre los distintos procesos, intercalando la ejecución de los mismos para dar así una apariencia de ejecución simultánea.

2. Multiprocesador
Un multiprocesador es una máquina formada por un conjunto de procesadores que comparten memoria principal. En este tipo de arquitecturas, los procesos concurrentes no sólo pueden intercalar su ejecución, sino también superponerla. En este caso si existe una verdadera ejecución simultánea de procesos, al coincidir las fases de procesamiento de distintos procesos. En un instante dado se pueden ejecutar de forma simultánea tantos procesos como procesadores haya.

3. Multicomputadora
Una multicomputadora es una máquina de memoria distribuida, en contraposición con el multiprocesador que es de memoria compartida. Está formada por una serie de computadoras completas con su CPU, memoria principal y, en su caso, periferia. Cada uno de estos procesadores completo se denomina nodo. Los nodos se encuentran conectados y se comunican entre sí a través de una red de interconexión, empleando el método de paso de mensajes. En este tipo de arquitecturas también es posible la ejecución simultánea de los procesos sobre los distintos procesadores.

En general, la concurrencia será aparente siempre que el número de procesos sea mayor que el de procesadores disponibles, es decir, cuando haya más de un proceso por procesador. La concurrencia será real cuando haya un proceso por procesador.[84]

[83] "Modelo cliente servidor", material en línea, disponible en: http://www.fortunecity.es/imaginapoder/memata/529/hoja/SIST.DISTRI1.htm, recuperado el 19/09/09.
[84] Mitotecnológico: "Concurrencia y secuencialidad", material en línea, disponible en: http://www.mitecnologico.com/Main/ConcurrenciaYSecuenciabilidad, recuperado el 14/09/09.

Beneficios de la concurrencia

1. Trata de evitar los tiempos muertos de la UCP
2. Comparte y optimiza el uso de recursos
3. Permite la modularidad en las diferentes etapas del proceso
4. Acelera los cálculos
5. Da mayor comodidad
6. Facilita la programación
7. Posibilita el uso interactivo
8. Permite un mejor aprovechamiento de los recursos.[85]

Desventajas de la concurrencia

1. Inanición e interrupción de procesos
2. Ocurrencia de bloqueos
3. Que dos o más procesos requieran el mismo recurso (no apropiativo)

Tipos de procesos concurrentes

Los procesos que ejecutan de forma concurrente en un sistema se pueden clasificar como procesos independientes y cooperantes.

Un proceso **independiente** es aquel que ejecuta sin requerir la ayuda o cooperación de otros procesos. Un claro ejemplo de procesos independientes son los diferentes intérpretes de mandatos que se ejecutan de forma simultánea en un sistema.

Los procesos **cooperantes,** cuando están diseñados para trabajar conjuntamente en alguna actividad, para lo que deben ser capaces de comunicarse e interactuar entre ellos en el ejemplo del compilador que se vio anteriormente, los dos procesos que lo conforman son procesos cooperantes.

Tanto si los procesos son cooperantes como independientes, puede producirse una serie de interacciones entre ellos. Estas interacciones pueden ser de dos tipos:

- Interacción motivada porque los procesos **comparten o compiten** por el acceso a recursos físicos o lógicos. Esta situación aparece en los distintos tipos de procesos anteriormente comentados. Por ejemplo, dos procesos totalmente independientes pueden competir por el acceso al disco. En este caso el sistema operativo deberá encargarse de que los dos procesos accedan ordenadamente sin que se cree ningún conflicto. Esta situación también aparece cuando varios procesos desean modificar el contenido de un registro de una base de

[85] James Palmer y David Perlman, *Introducción a los Sistemas Digitales*, México, McGraw-Hill, 1995, p. 90.

datos. Aquí es el gestor de la base de datos el que se tendrá que encargar de ordenar los distintos accesos al registro.

- Interacción motivada porque los procesos se **comunican y sincronizan** entre sí para alcanzar un objetivo común. Por ejemplo, los procesos compilador y ensamblador descritos anteriormente son dos procesos que deben comunicarse y sincronizarse entre ellos con el fin de producir código en lenguaje máquina.

Virtualmente todos los sistemas de tiempo-real son inherentemente concurrentes (los dispositivos operan en paralelo en el mundo real).[86]

4.4 Bloqueos

Un bloqueo es un conjunto de procesos que se encuentran esperando un evento que solo puede ser provocado por otro proceso o evento del mismo conjunto.

En algunos casos, el costo de liberar bloqueos puede ser muy alto y permitirlos podría resultar catastrófico.

Podemos encontrar diversos tipos de bloqueo[87]:

- Deadlock: que ocurre cuando 2 o más procesos comparten recursos entre sí.
- Bloqueo de tráfico: en el cual un bloqueo solo puede ser desactivado por un externo.
- Bloqueo de recursos simples: el cual tiene su origen en la contención normal de los recursos dedicados o reutilizables en serie, y
- Bloqueo en sistemas de *spool*: que ocurre en las capacidades de ejecución de ciertos dispositivos.

[86] Universidad Carlos III de Madrid, Grupo de Arquitectura de Computadores, Comunicaciones y Sistemas, Práctica de Sistemas Operativos: "Comunicación y sincronización de procesos", material electrónico disponible en: http://www.arcos.inf.uc3m.es/~ssoo-va/ssoo-prac/libro/cap04.pdf, recuperado el 14/09/09.
[87] Hervey Deitel, op. cit., p.7.

Detección de Bloqueos

La detección del bloqueo es el proceso de determinar si existe o no un bloqueo e Identificar cuáles son los procesos y recursos implicados en éste.

Las estrategias utilizadas para enfrentar los bloqueos son:

- Ignorar todo el problema.
- Detección y recuperación.
- Evitarlos dinámicamente mediante una cuidadosa asignación de recursos.[88]

El Sistema Operativo no evita bloqueos, los detecta cuando han ocurrido y toma acciones para recuperarse después del hecho.

Existen algoritmos de detección de bloqueos como el **de Ostrich**, "Un recurso de cada tipo", "Varios recursos de cada tipo", etc.

4.5 La gestión de entrada/salida

El sistema operativo tiene, entre sus principales funciones, controlar y gestionar los dispositivos y el software de entrada/salida. Con ello asegura un adecuado manejo de errores, transferencias, almacenamientos y sobre todo facilidad y transparencia de interfaz para el usuario.

Existen principios tanto de hardware como de software que le ayudan al sistema operativo a llevar una mejor gestión de entrada/salida. A continuación vamos a conocerlos.

[88] Andrew S. Tanenbaum, op. cit., p.34.

4.5.1. Principios de entrada/salida de hardware

Las unidades de E/S constan de un componente mecánico y uno electrónico. Al componente electrónico se le conoce como adaptador de dispositivo o controlador. El componente mecánico es el dispositivo como tal.

Los dispositivos de E/S pueden dividirse en dos categorías: dispositivos de bloques y dispositivos de caracteres.

Un dispositivo de bloque almacena información de tamaño fijo con su propia dirección. Los bloques van desde 512 hasta 32,768 bytes.

- **El principio de dispositivos de bloques** consiste en la posibilidad de leer o escribir cada bloque con independencia de todos los demás.

El otro tipo de dispositivo es el de caracteres, el cual suministra o acepta un flujo de caracteres sin estructurarlos en bloques, es decir, es secuencial, no es direccionable ni tiene operación de desplazamiento.

Ahora bien, respecto al adaptador o controlador, este tiene como principal función convertir el flujo de bits en una serie en un bloque de bytes y corregir errores para poder comunicarse con la Unidad Central de Proceso.

El sistema operativo puede, a través del controlador, ordenar a un dispositivo mecánico que suministre o acepte datos, se encienda o se apague o realice alguna otra acción. Para poder llevar a cabo esta comunicación, el controlador hace uso de registros de control y de búferes de datos que el sistema operativo puede leer y escribir.

Para que la Unidad Central de Proceso se comunique con los registros y los *buffers*, existen dos alternativas:

- En la primera a cada registro se le asigna un número de puerto de E/S que es leído por la CPU y almacenado en otro registro interno de ésta. Los registros de E/S y de almacenamiento son distintos.

- En la segunda alternativa se establece una correspondencia entre todos los registros de control y los registros de almacenamiento. A cada registro de control se le asigna una dirección de memoria única y se le conoce como entrada/salida con correspondencia en memoria.[89]

[89] Exa UNNE: “Gestión de E/S y planificación de discos”, material en línea, disponible en: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTES02.htm, recuperado el 14/09/09.

4.5.2 Principios de entrada/salida de software

Un adecuado software de entrada/salida debe cumplir con varios principios:

- **Independencia del dispositivo:** consiste en poder escribir programas capaces de acceder a cualquier dispositivo de Entrada/Salida sin tener que especificar por adelantado de qué dispositivo se trata.
- **Nombres uniformes:** el nombre del archivo o dispositivo debe ser una cadena o un entero y no depender en absoluto del dispositivo.
- **Manejo de errores:** los errores deben manejarse tan cerca del hardware como sea posible, si un dispositivo descubre un error deberá tratar de corregirlo él mismo si puede, de no ser así, entonces el software deberá tratar de corregirlo.
- **Transferencias síncronas y asíncronas**: las transferencias síncronas son aquellas controladas por bloqueo, las asíncronas son las controladas por interrupciones. Casi toda la E/S física es síncrona y los programas son mucho más fáciles de escribir si las operaciones son bloqueadoras; en este principio el sistema operativo debe hacer que, operaciones que en realidad son controladas por interrupciones, parezcan bloqueadoras desde la perspectiva de los programas del usuario.
- **Manejo de buffers**: es común que los datos que provienen de un dispositivo no puedan almacenarse en forma directa en su destino final sino hasta haber sido manipulados por el sistema operativo. Este principio consiste en almacenar datos en un buffer de salida mediante software, para reducir insuficiencias en el desempeño de los búferes.
- **Dispositivos compartidos y dedicados:** el sistema operativo debe tener la capacidad de manejar dispositivos de forma tal que se eviten problemas como bloqueos.[90]

4.6 Gestión de la memoria

La memoria principal es tal vez el recurso más importante de la computadora, pues se encarga de almacenar y gestionar los procesos y datos que ésta maneja.

La memoria es una gran tabla de palabras o bytes que se referencian cada una mediante una dirección única.

[90] Véase, Andrew Tanenbaum, *Sistemas operativos modernos*, México, Pearson Prentice-Hall, 2ª ed., 2003, p. 283, y Exa UNNE: "Gestión de E/S y planificación de discos", material en línea, disponible en: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTES02.htm, recuperado el 14/09/09.

Casi todas las computadoras cuentan con un administrador de memoria, que es la parte del sistema operativo que asigna, administra y libera, en otras palabras, jerarquiza memoria para diversas aplicaciones.

Los administradores de memoria se dividen en dos clases: los que traen y llevan procesos entre la memoria principal y el disco en tiempo de ejecución (intercambio y paginación) y los que ejecutan solo un programa a la vez, repartiendo la memoria entre ese programa y el sistema operativo (segmentación).

Mecanismos de asignación de memoria

Un mecanismo de asignación determina la cantidad de particiones que serán administrados en la memoria. Existen tres mecanismos:

1. Asignación de una partición. Considera la existencia de una sola partición, es decir, la capacidad de ejecutar solo un proceso. La partición será toda la memoria y la administración la debe llevar a cabo el usuario, no hay relación con el sistema operativo.

2. Asignación de dos particiones. La memoria se divide en 2 bloques. El primero se carga el Sistema Operativo y el segundo del programa del usuario.

3. Asignación de múltiples particiones. Consiste en dividir la memoria en varios bloques, un primer bloque se usa para cargar el Sistema Operativo, un segundo bloque carga los procesos de usuario y un último bloque carga las funciones especiales del Sistema Operativo.

Métodos de asignación de memoria

Un método de asignación de memoria es la forma mediante la cual el Sistema Operativo lleva el control eficiente de la memoria. Los métodos de asignación más comunes son:

1. Segmentación

Este método consiste en proporcionar a la máquina espacios de direcciones independientes por completo llamados segmentos. El tamaño de cada segmento o bloque dependerá de la petición requerida, aunque su tamaño máximo está determinado por el número bits de los que dispone el hardware para almacenar una dirección.

El acceso a cada segmento se hace mediante una dirección que se integra por dos elementos: una dirección de segmento y una de desplazamiento, a esta dirección se le conoce como dirección de memoria.

Para poder asegurarse de que cada dirección de memoria esté dentro del rango de direcciones, el sistema operativo implementa una tabla de segmentos que es un arreglo de registros base.

2. Paginación

Este método consiste en considerar el espacio de direcciones de memoria de cada proceso como un conjunto de bloques llamados páginas. Cada dirección de memoria manejada para un proceso estará conformada por dos elementos: una dirección de página y una de desplazamiento.

El tamaño de una página debe ser igual al tamaño de un 'marco', el cual se refiere a un bloque de tamaño consistente que se implementa para administrar la memoria física.

3. Segmentación/Paginación

Consiste en combinar los mecanismos de segmentación y paginación. Se utiliza cuando se tiene un gran número de procesos, o bien, se requiere de mayor rapidez de procesamiento.

Memoria virtual

Es un método en el que el tamaño combinado del programa, sus datos y su pila, pueden exceder la cantidad de memoria física que le es asignada, es decir, el sistema operativo simula tener más memoria principal que la que existe físicamente.

Para poder ser implementada, se utiliza un medio de almacenamiento secundario de alta velocidad de acceso, generalmente un disco duro. [91]

4.7 Gestión de archivos[92]

Las aplicaciones que utilizan las computadoras almacenan y recuperan información. El almacenamiento está relacionado con el tamaño de espacio de dirección válido asignado; para algunas aplicaciones ese tamaño es suficiente, pero para otras es demasiado pequeño.

[91] Departamento de Sistemas y Computación del Instituto Tecnológico de La Paz: "Administración de la memoria", material electrónico disponible en: http://sistemas.itlp.edu.mx/tutoriales/sistemasoperativos2/unidad1.htm; y Gabriela García: "Administración de la memoria", material disponible en línea: http://www.angelfire.com/bc3/gabrielagarcia0/UNIDADI.htm, recuperado el 14/09/09.

[92] Véase, Wikipedia: "Gestión de archives", actualizado el 18/03/09, material en línea disponible en: http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_archivos, recuperado el 14/09/09.

Cuando se habla de almacenamiento y recuperación de información se requieren tres requisitos fundamentales:

- Se debe permitir almacenar grandes volúmenes de información.
- La información debe estar disponible, aún cuando el proceso que la emplea haya concluido.
- Múltiples procesos deben poder acceder a la misma información de manera concurrente.

La respuesta a estos requisitos se encuentra en los archivos. La forma en que se usan, se nombran, se tiene acceso a ellos, se protegen y se implementan, es una función del sistema operativo; él es quien gestiona los archivos a través de lo que se conoce como *sistema de archivos.*

Los archivos son una estructura de abstracción que permiten el almacenamiento de información en disco para su posterior recuperación; esta abstracción es transparente al usuario, es decir, el usuario no tiene por qué enterarse de cómo se gestionan o almacenan los archivos.
Los archivos se clasifican en archivos normales, directorios, archivos especiales de caracteres y archivos especiales de bloques, y pueden estructurarse de varias maneras: por sucesión de bytes, de registros o en forma de árboles.

- La sucesión de bytes consiste en que el sistema operativo reconozca un archivo mediante su secuencia de bytes; el sistema operativo no sabe lo que contiene el archivo, lo único que ve son bytes y de esta forma los va leyendo.

- La sucesión de registros consiste en estructuras formadas por registros (conjunto de bytes) que tienen una longitud fija.

- Los árboles consisten en conjuntos de registros de diferente longitud, cada uno de los cuales contiene un campo clave para poder ser identificado.

Para poder acceder a los archivos, los sistemas operativos cuentan con dos procesos básicos, el *acceso secuencial* y *el acceso aleatorio.*

El *acceso secuencial* consiste en un proceso que permite leer todos los bytes o registros de un archivo en orden, comenzando por el principio, pero no puede efectuar saltos ni leerlos en otro orden.

El *acceso aleatorio* es un proceso que consiste en dos métodos para especificar dónde comenzar la lectura; en el primero cada operación da la posición en el archivo donde debe comenzarse a leer, en el segundo se cuenta con una operación especial para establecer la posición actual, teniendo la posición el archivo puede ser leído de forma secuencial.

4.8. Sistemas distribuidos[93]

Es una colección de elementos que permiten distribuir trabajos, tareas o procesos, entre un conjunto de procesadores. Puede ser que este conjunto de procesadores esté en uno o en diferentes equipos, y es transparente para el usuario. En un sistema distribuido, "los usuarios no deben ser conscientes del lugar donde su programa se ejecute o de lugar donde se encuentren sus archivos; eso debe ser manejado en forma automática y eficaz por el sistema operativo"[94].

Existen dos esquemas básicos de éstos: los débilmente acoplados y los fuertemente acoplados.

Un sistema fuertemente acoplado es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores. En

[93] Véase, material en línea, disponible en: http://www.angelfire.com/droid/sql/teoria.htm, consultado el 14/09/09.
[94] Andrew Tanenbaum, op. cit., p.16.

un sistema débilmente acoplado los procesadores no comparten ni memoria ni reloj, ya que cada uno cuenta con su memoria local.

Algunas características que encontramos en los sistemas distribuidos son las siguientes:

- Colección de sistemas autónomos capaces de comunicación y cooperación mediante interconexiones hardware y software.
- Gobierna la operación de un Sistema de cómputo y proporciona la abstracción de máquina virtual a los usuarios.
- Su objetivo clave es la transparencia.
- Generalmente proporcionan medios para la compartición global de recursos.
- Proporciona servicios añadidos: denominación global, sistemas de archivos distribuidos, facilidades para distribución de cálculos (a través de comunicación de procesos inter-nodos, llamadas a procedimientos remotos, etc.).

Ventajas:

1. Procesadores más poderosos
2. Respuesta rápida
3. Ejecución concurrente, procesos
4. Crecimiento modular

Desventajas:

1. Requerimientos de mayores controles de acceso y procesamiento
2. Velocidad de propagación de información
3. Administración más compleja
4. Costos.

Modelos de construcción de Sistemas Distribuidos

Arquitectura Cliente/Servidor

Es un modelo para el desarrollo de sistemas en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Éste modelo cuenta con dos elementos básicos: el cliente que es el proceso que inicia el diálogo o solicita los recursos y el servidor, que es el proceso que responde a las solicitudes.

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.

Ventajas

- Aumento de la productividad:
- Los usuarios pueden utilizar herramientas que le son familiares y pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.
- Menores costos de operación.
- Permiten un mejor aprovechamiento de los sistemas existentes al compartir recursos.
- Proporcionan un mejor acceso y control a los datos.
- Es escalable.

Desventajas

- Alta complejidad tecnológica al tener que integrar una gran variedad de productos.
- Dificultad para asegurar un elevado grado de seguridad.
- Congestionamientos.

Redes de comunicación

Conjunto de elementos llamados nodos que se comunican a través de reglas conocidas como protocolos y cuyo objetivo es el intercambio de información sobre vías conocidas como medios de transmisión.

Las redes pueden clasificarse de acuerdo a su **cobertura** en LAN (Redes de Área Local), MAN (Redes de Área Metropolitana) y WAN (Redes de Área Extensa); de acuerdo a su **topología** en bus, estrella, anillo y malla; y de acuerdo a su **uso** en públicas y privadas.

Lo que es necesario tener en cuenta es que una red puede ser vista como un sistema distribuido, cuyos objetivos son compartir recursos de hardware y de software y el intercambio de información. [95]

4.9 Seguridad

Actualmente, cualquier computadora conectada a Internet es potencialmente blanco de un ataque. Definir la seguridad en un sistema es difícil, no existe una definición de seguridad en cómputo universalmente aceptada, esto se debe a que es un tema relativamente nuevo y un tanto ambiguo al momento de definir sus fronteras.

[95] Véase, Juansa: "Introducción a Redes: Arquitectura Cliente/Servidor", material en línea, disponible en: http://www.juansa.net/Admin2003/cliser.htm, recuperado el 19/09/09.

Podemos solo decir que un sistema de cómputo es seguro si se puede confiar en que se comportará como se espera que lo haga.

Respecto al Sistema Operativo, éste solo es una parte del total de software que contiene un sistema y es propenso a fallos, los cuales pueden afectar la seguridad general del sistema. Actualmente, dependiendo de las fuentes de amenazas, la seguridad puede dividirse en seguridad lógica y seguridad física. En estos momentos la seguridad informática es un tema de dominio obligado por cualquier usuario de la Internet, para no permitir que su información sea robada, mal empleada o modificada.

Existen distintos tipos de seguridad por considerar:

- Confidencialidad. Consiste en proteger la información de ser accedida por cualquier persona que no esté autorizada para ello.
- Integridad. Consiste en la protección de la información (incluyendo programas) por ser borrada o alterada sin el permiso del dueño de la información.
- Disponibilidad. La información debe ser utilizable cuando es requerida por el usuario autorizado.
- Consistencia. Asegurarse que el sistema se comporta como se espera que lo haga. Si el software o hardware repentinamente empieza a comportarse de manera diferente a como lo hace normalmente, esto podría representar un problema.
- Control. Consiste en regular el acceso al sistema. Se debe cuidar que no se encuentren en el sistema individuos o programas no autorizados.
- Auditoría. La auditoría reside en tener un registro de las actividades dentro del sistema que identifique actores y acciones. En aplicaciones críticas, la auditoría será suficiente para poder revertir los posibles cambios a un sistema y restablecerlo por completo.

Aunque todos estos aspectos de seguridad son importantes, cada organización determina el orden de prioridad para cada uno. Esta variación se debe a que cada organización tiene diferentes preocupaciones de seguridad.

En el ámbito informático, los principios básicos de seguridad son los siguientes:

- Autenticidad: que la persona que se firme sea quien dice ser.
- Integridad: la información no puede ser modificada por quien no está autorizado.
- Consistencia: el sistema debe comportarse como se espera que lo haga.
- Confidencialidad: la información solo debe ser legible para los autorizados.
- Disponibilidad: la información debe estar disponible cuando se necesita.
- Irrefutabilidad (no-rechazo o no repudio): que no se pueda negar la autoría. [96]

Es imposible lograr una seguridad absoluta, pero podemos proteger nuestros sistemas con herramientas de seguridad, políticas, monitoreo, métodos y técnicas de encriptación, entre otras cosas.

- Una herramienta de seguridad es un programa diseñado para ayudar al administrador a mantener su sistema seguro, pero debemos asumir que si la herramienta está disponible para el administrador del sistema, también lo está para el atacante. Las herramientas de seguridad pueden clasificarse en:[97]

[96] Kioskea: "Introducción a la seguridad informática", actualizado el 16/10/08, disponible en línea: http://es.kioskea.net/secu/secuintro.php3, recuperado el 19/09/09.
[97] Véase, Linux Data: "Recomendaciones de seguridad en una instalación de Linux", material en línea, disponible en: http://www.linuxdata.com.ar/index.php?idmanual=recomendacionesdsa.htm&manuale=1, recuperado el 19/09/09.

1. Sistemas detectores de Intrusos: Son también llamados IDS por sus siglas en ingles “Intrusión Detection System”.

Un IDS es un software que monitorea el tráfico de una red y los sistemas de una organización en busca de señales de intrusión, actividades de usuarios no autorizados y la ocurrencia de malas prácticas, como en el caso de los usuarios autorizados que intentan sobrepasar sus límites de restricción de acceso a la información.

Algunas de las características deseables para un IDS son:

- Deben estar continuamente en ejecución con un mínimo de supervisión.
- Se debe recuperar de las posibles caídas o problemas con la red.
- Debe poderse analizar él mismo y detectar si ha sido modificado por un atacante.
- Debe utilizar los mínimos recursos posibles.
- Debe estar configurado acorde con la política de seguridad seguida por la organización.
- Debe de adaptarse a los cambios de sistemas y usuarios y ser fácilmente actualizable.

2. Analizadores por red: herramientas que exploran la red y escanean puertos para el análisis, administración y auditoria de sistemas. Ejemplos de éstas son Nmap, Nessus, Tara, etc.

3. Analizadores locales: herramientas que permiten revisar rupturas de seguridad respecto a contraseñas y problemas en archivos de sistema garantizando la disponibilidad, los cambios y los respaldos de información. Ejemplos de estos son Tripwire Tiger, Cops, etc.

4. Herramientas para el administrador: son todas aquellas que facilitan la tarea de gestionar equipos e información. Ejemplos de éstas son: npasswd, logcheck, passwd+, etc.

- El monitoreo consiste en una constante revisión de los elementos dentro del sistema por parte del administrador del mismo, para identificar posibles cambios que puedan ser un indicio de una intrusión. Todos los movimientos del sistema son registrados en archivos, que los administradores deben revisar diariamente.

- Las políticas ayudan a definir lo que es valioso y especifican qué pasos se deberían seguir para asegurar los bienes o servicios. Las políticas pueden ser formuladas de diferentes maneras. Se pueden escribir políticas muy generales (de unas cuantas páginas) que cubran muchas posibilidades. También se pueden escribir políticas para diferentes conjuntos de bienes o servicios; el tipo de política establecida está en función de la organización de que se trate.

- Los métodos y técnicas de encriptación comprenden un conjunto de técnicas que proporcionan los siguientes servicios:

  Cifrado. Proceso que transforma los datos del emisor a una forma ilegible para cualquier receptor no autorizado, para asegurar la privacidad o confidencialidad de los mismos.

Las principales técnicas de cifrado se clasifican en los siguientes grupos:

- Sistemas de cifrado simétrico o de llave secreta. Sistemas que involucran a algoritmos que usan la misma llave en las dos etapas del algoritmo (cifrado y descifrado, creación de firmas, y verificación de firmas). También es llamada *Criptografía de Llave Secreta* porque las dos entidades comparten la misma llave[98].

- Sistemas de cifrado asimétrico o de llave pública. Sistemas que usan un par de llaves para el envío de mensajes. Las dos llaves pertenecen a la persona que ha enviado el mensaje. Una llave es pública y se puede entregar a cualquier persona. La otra llave es privada y el propietario debe guardarla de modo que solo él tenga acceso a ella. El remitente usa la llave pública del destinatario para cifrar el mensaje, y una vez cifrado, sólo la llave privada del destinatario podrá descifrar este mensaje[99].

- Sistemas híbridos. Son sistemas que combinan dos o más algoritmos de cifrado, por lo general simétrico y asimétrico.
- Firmas Digitales. Información incluida en un paquete o que se envía por separado para identificar y autentificar al emisor y la información del mensaje. También puede confirmar que un mensaje o programa no haya sido alterado.[100]

---

[98] Cf., UNAM. DGSCA. Seguridad en cómputo, Usuario Casero. Diccionario http://www.seguridad.unam.mx/usuario-casero/secciones/diccionario.dsc?dicc=5184 recuperado el 19/09/09.

[99] Cf., UNAM. DGSCA. Seguridad en cómputo, Usuario Casero. Diccionario http://www.seguridad.unam.mx/usuario-casero/secciones/diccionario.dsc?dicc=5184 recuperado el 19/09/09.

[100] Cf., UNAM. DGSCA. Seguridad en cómputo, Usuario Casero. Diccionario http://www.seguridad.unam.mx/usuario-casero/secciones/diccionario.dsc?dicc=5184 recuperado el 19/09/09.

- Funciones hash (compendios y resúmenes criptográficos). Funciones que generan claves que sirven para garantizar la integridad de los textos[101].

Descifrado. Es el proceso inverso al cifrado, esto es, transformar datos cifrados a su forma original.

Autenticación. Proceso para identificar a un usuario, una máquina en la red, una organización, un documento o un software que desea tener acceso a un servicio de cómputo determinado. La autenticación se lleva a cabo comúnmente mediante el uso de contraseñas.

Firmas digitales. Equivalente a las firmas en papel, ligan un documento con el propietario de una llave particular. Información incluida en un paquete que se envía por separado para identificar y autenticar al emisor y la información del mensaje. También puede confirmar que un mensaje o programa no haya sido alterado.

Recomendaciones

El activo más importante que se posee es la información y, por lo tanto, deben existir técnicas que la aseguren, más allá de la seguridad física que se establezca sobre los equipos en los cuales se almacena. Estas técnicas las brinda la seguridad lógica que consiste en la aplicación de *barreras y procedimientos* que resguardan el acceso a los datos y sólo permiten acceder a ellos a las personas autorizadas para hacerlo. Existe un viejo dicho que dicta: *"lo que no está permitido debe estar prohibido"* y esto es lo que respecta a la seguridad lógica.

[101] Cf., UNAM DGSCA Seguridad en Internet. Funciones Resumen (Hash) http://www.eumed.net/cursecon/ecoinet/seguridad/resumenes.htm recuperado el 19/09/09

Algunas recomendaciones son:

- Restringir el acceso (de personas de la organización y de las que no lo son) a los programas y archivos.
- Asegurar que los operadores puedan trabajar pero que no puedan modificar los programas ni los archivos que no correspondan (sin una supervisión minuciosa).
- Asegurar que se utilicen los datos, archivos y programas correctos en/y/por el procedimiento elegido.
- Asegurar que la información transmitida sea la misma que reciba el destinatario al cual se ha enviado y que no le llegue a otro.
- Asegurar que existan sistemas y pasos de emergencia alternativos de transmisión entre diferentes puntos.
- Organizar a cada uno de los empleados por jerarquía informática, con claves distintas y permisos bien establecidos, en todos y cada uno de los sistemas o aplicaciones empleadas.
- Actualizar constantemente las contraseñas de accesos a los sistemas de cómputo.

4.10. Ejemplo práctico de un sistema operativo (UNIX, Linux, Windows)

UNIX / LINUX

Todos los operadores, sintaxis y atributos de los comandos se conocen escribiendo el comando de ayuda **man** seguido del comando del que requerimos información.

Ejemplo: man [finger] (y teclee enter)

A continuación se establecen los principales comandos que pueden ser ejecutados desde línea de comando en un sistema operativo Unix o Linux.

TABLA de COMANDOS UNIX/LINUX

login	Pide nombre y clave del usuario
logout, exit	Sale del sistema
date	Muestra la fecha
who	Muestra información de los usuarios del servidor
whoami	Muestra información acerca de nosotros
man,	Muestra ayuda sobre un comando
ls	Muestra lista de archivos
cat	Muestra contenido de uno o varios archivos
cp	Copia un archivo
mv	Cambia nombre de un archivo
more	Muestra el contenido por pantallas
rm	Borra un archivo
diff	Compara y muestra diferencias entre archivos
find	Localiza los archivos
finger	Muestra información de usuarios en toda la red
cd	Cambia de directorio
mkdir	Crea un directorio
rmdir	Borra un directorio
pwd	Muestra la ruta del directorio actual
alias	Asigna otro nombre a un comando
Ps	Muestra los procesos que se están ejecutando
kill	Elimina procesos
clear	Limpia la pantalla
passwd	Cambia el password del usuario
mail	Envía un correo electrónico

Entra a Unix o a Linux (es indistinto) y ejecuta los comandos que se señalan en el siguiente cuadro (para la sintaxis completa ejecuta el comando man; en el

espacio en blanco describe lo que tecleaste en la línea de comando y el resultado(s) que obtuviste.

Comando	**Resultado**
date	
whoami	
ls	
cp	
mv	
rm	
diff	
find	
cd	
mkdir	
rmdir	
pwd	
alias	
Ps	
kill	
clear	
passwd	
logout, exit	

Bibliografía del tema 4

Deitel Harvey. M. l. *Introducción a los Sistemas Operativos,* México, Addison-Wesley Iberoamericana, 1993.

Milenkovik Milan, *Sistemas Operativos: conceptos y diseños*, Madrid, MacGraw-Hill, 1994.

Tanenbaum Andrew, *Sistemas Operativos Modernos*, 2ª ed., México, Pearson, 2003.

Palmer, James E.; Perlman, David E., *Introducción a los Sistemas Digitales,* México, McGraw Hill, 1995.

Silberschatz, Abraham, et. al., *Sistemas Operativos*, México, Limusa Wiley, 2002.

Sitios web

- http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r134.DOC
- http://sectec.ilce.edu.mx/sectec40/intruduc_cur.html
- http://www.unach.edu.ec/Virtualizacion/Sistemas_Operativos/paginas/Unidad%202.htm
- http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r139.DOC
- http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml
- http://es.wikipedia.org/wiki/Hilo_de_ejecuci%C3%B3n
- http://agamenon.uniandes.edu.co/~c21437/e-valenc/inf_semaforos.html
- http://www.fortunecity.es/imaginapoder/memata/529/hoja/SIST.DISTRI1.htm
- http://www.mitecnologico.com/Main/ConcurrenciaYSecuenciabilidad
- http://www.arcos.inf.uc3m.es/~ssoo-va/ssoo-prac/libro/cap04.pdf

- http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTES02.htm
- http://sistemas.itlp.edu.mx/tutoriales/sistemasoperativos2/unidad1.htm
- http://www.angelfire.com/bc3/gabrielagarcia0/UNIDADI.htm
- http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_archivos
- http://www.angelfire.com/droid/sql/teoria.htm
- http://www.juansa.net/Admin2003/cliser.htm
- http://es.kioskea.net/secu/secuintro.php3
- http://www.linuxdata.com.ar/index.php?idmanual=recomendacionesdsa.htm&manuale=1
- http://www.seguridad.unam.mx/usuario-casero/secciones/diccionario.dsc?dicc=5184
- http://www.eumed.net/cursecon/ecoinet/seguridad/resumenes.htm

Actividades de aprendizaje

A.4.1 Elabora un mapa conceptual de lo estudiado en el tema cuatro, con el propósito de que tengas un panorama global de la programación e implementación de sistemas, apoyándote con el temario propuesto y la bibliografía sugerida.

A.4.2 Elabora en una diapositiva en PowerPoint, un cuadro sinóptico en donde resaltes las características de los procesos e hilos y sus diferencias

A.4.3 Elabora, en una diapositiva en PowerPoint, un cuadro sinóptico en donde resaltes los puntos más importantes del tema de seguridad.

A.4.4 Elabora un cuadro comparativo en donde expliques como con el Sistema Operativo Windows realizas los comandos de Unix/Linux descritos en el tema 4.10

Cuestionario de autoevaluación

1. ¿Qué es la multiprogramación?
2. ¿Qué es el tiempo compartido?
3. ¿Qué es el tiempo real?
4. Explica brevemente la evolución de los sistemas operativos.
5. Enlista al menos cinco sistemas operativos diferentes.
6. ¿Qué es una máquina virtual?
7. ¿Qué es un proceso?
8. ¿Qué son los hilos?
9. ¿Qué son los bloqueos?
10. ¿Cómo se hace la asignación de múltiples particiones?
11. ¿Qué es un IDS?
12. Menciona al menos 2 tipos de seguridad a considerar
13. ¿Qué es la autenticación?
14. ¿Qué es el cifrado?
15. ¿Qué es la firma electrónica?

Examen de autoevaluación

1. En este tipo de sistema la memoria principal alberga a más de un programa de usuario.

    a) Multitarea
    b) Multiprogramado
    c) Tiempo compartido
    d) Tiempo real

2. Sistema operativo que se convirtió en un componente más de Windows, como lo conocemos ahora:

a) MS-DOS
b) LINUX
c) OS2
d) UNIX

3. Sistema operativo que presenta una interface de cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente:

a) Multitarea
b) Máquina virtual
c) Monolítico
d) Jerárquico

4. Es un programa en ejecución:

a) Archivo
b) Programa
c) Hilo
d) Proceso

5. Permite, a una aplicación dada, realizar varias tareas concurrentemente:

a) Hilo
b) Proceso
c) Programa
d) Sistema operativo

6. Un proceso está en este estado si se encuentra esperando un evento determinado que no ocurrirá.

a) Compilación

b) Excepción

c) Interrupción

d) Bloqueo

7. En este tipo de bloqueo, cada proceso está esperando al otro para liberar uno de los recursos:

a) Recurso simple

b) Tráfico

c) Sistemas de spool

d) Deadlock

8. Es una de las funciones básicas que debe implementar un SO para tener un control sobre los lugares donde están almacenados los procesos y datos que actualmente se están utilizando:

a) Manejo de archivos

b) Gestión de la memoria

c) Sistema de E/S

d) Traducción de programas

9. La memoria es dividida en varias particiones, una para el Sistema Operativo y las demás para los procesos de usuarios u otras funciones especiales del Sistema Operativo, este es el principio de la asignación:

a) De múltiples particiones

b) De dos particiones

c) De una partición

d) Particiones diferentes

10 Proceso que transforma los datos del emisor a una forma ilegible para cualquier receptor no autorizado, para asegurar la privacidad o confidencialidad de los mismos

a) Autenticación

b) Integridad

c) Cifrado

d) Descifrado

Bibliografía básica

Aho, Alfred B.; Compiladores. Principios, técnicas y herramientas México, Pearson Educación, 2008.

Bishop, Judy, *Java: fundamentos de programación*, Madrid, Addison Wesley, 1999.

Bronson Gary, trad. de Jorge Alberto Velázquez Arellano., C++ para ingeniería y ciencias, México, International Thomson, 2007, 826pp.

Burnham, W. D., Prolog: programación y aplicaciones, México, Limusa, 1989.

Deitel Harvey, M. l., Introducción a los Sistemas Operativos. México, Addison-Wesley Iberoamericana, 1993.

Gottfried, Byron S. *Programación en C*. 2ª ed., México, McGraw-Hill Interamericana, 2005.

Hansen, Augie. *¡Aprenda C Ya!*, Madrid, Anaya Multimedia, 1990.

Joyanes Aguilar, Luis; Luis Rodríguez Baena y Matilde Fernández Azuela, Fundamentos de Programación: Libro de Problemas. México, McGraw-Hill, 1996.

Kernighan, Brian W.; Ritchie, Dennis M., *El Lenguaje de Programación C*. 2ª edición, México, Prentice-Hall, 1991.

Long, Larry, trad. de María de Lourdes Fournier García, *Introducción a las computadoras y al procesamiento de información*, Prentice Hall Hispanoamericana, Nueva Jersey, 1995.

Milenkovik Milan, *Sistemas Operativos: conceptos y diseños*, Madrid, MacGraw-Hill, 1994.

Palmer, James E.; Perlman, David E., *Introducción a los Sistemas Digitales,* México, McGraw Hill, 1995.

Silberschatz, Abraham, et. al., *Sistemas Operativos*, México, Limusa Wiley, 2002.

Stroustrup, Bjarne, *El lenguaje de programación C++,* Madrid, Addison-Wesley, 1993.

Tanenbaum Andrew, *Sistemas Operativos Modernos*, 2ª ed., México, Pearson, 2003.

Sitios web

- http://agamenon.uniandes.edu.co/~c21437/e-valenc/inf_semaforos.html
- http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp12.pdf
- http://computacion.cs.cinvestav.mx/~acaceres/courses/itesm/lp/clases/lp16.pdf
- http://delta.cs.cinvestav.mx/~gmorales/ta/ta.html
- http://elvex.ugr.es/etexts/spanish/pl/algol.htm
- http://es.kioskea.net/faq/sujet-2821-la-compilacion-y-los-modulos-en-c-y-c
- http://es.kioskea.net/secu/secuintro.php3
- http://es.wikipedia.org/wiki/Cinta_magn%C3%A9tica_de_almacenamiento_de_ datos
- http://es.wikipedia.org/wiki/COBOL
- http://es.wikipedia.org/wiki/Compilador
- http://es.wikipedia.org/wiki/Disco_%C3%B3ptico
- http://es.wikipedia.org/wiki/Generaci%C3%B3n_de_c%C3%B3digo
- http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_archivos
- http://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n
- http://es.wikipedia.org/wiki/Hilo_de_ejecuci%C3%B3n
- http://es.wikipedia.org/wiki/Optimizaci%C3%B3n_de_software

- http://es.wikipedia.org/wiki/PL/I
- http://es.wikipedia.org/wiki/Plankalk%C3%BCl
- http://es.wikipedia.org/wiki/Python
- http://entren.dgsca.unam.mx/introduccion/so_carac.html
- http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTES02.htm
- http://foro.latinohack.com/showthread.php?t=17882
- http://homepage.mac.com/eravila/asmix862.html
- http://ingenieria.uatx.mx/~patrick/so/unidad2.pdf
- http://mx.geocities.com/michaelneo_matrix/CARACTERISTICAS.doc
- http://neo.lcc.uma.es/staff/francis/fundcomp/tema8.ppt
- http://sectec.ilce.edu.mx/sectec40/intruduc_cur.html
- http://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/risi/N1_2004/a04.pdf
- http://sistemas.itlp.edu.mx/tutoriales/sistemasoperativos2/unidad1.htm
- http://tricia-tecnologia.blogspot.com/2007_09_01_archive.html
- http://weblogs.inf.udp.cl/inf9019-1/files/2006/09/tareacontrol.pdf
- http://www.angelfire.com/bc3/gabrielagarcia0/Unidad1.htm
- http://www.angelfire.com/droid/sql/teoria.htm
- http://www.arcos.inf.uc3m.es/~ssoo-va/ssoo-prac/libro/cap04.pdf
- http://www.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/
- http://www.diclib.com/cgibin/d1.cgi?l=es&base=es_wiki_10&page=showid&id=5623
- http://www.dict.uh.cu/Revistas/CM2002_2003/CM02202k.doc
- http://www.eumed.net/cursecon/ecoinet/seguridad/resumenes.htm
- http://www.fortunecity.es/imaginapoder/memata/529/hoja/SIST.DISTRI1.htm
- http://www.infor.uva.es/~fjgonzalez/apuntes_aso/Tema6.pdf

- http://www.itculiacan.edu.mx/apuntes/maestros/Ram%F3n%20Zatara%EDn/LenguajesAutomatas/LENGUAJESYAUTOMATAS.ppt
- http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r134.DOC
- http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r139.DOC
- http://www.juansa.net/Admin2003/cliser.htm
- http://www.linuxdata.com.ar/index.php?idmanual=recomendacionesdsa.htm&manuale=1
- http://www.masadelante.com/faqs/disco-duro
- http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml
- http://www.mitecnologico.com/Main/AnalizadorSemantico
- http://www.mitecnologico.com/Main/ConcurrenciaYSecuenciabilidad
- http://www.perl.org/
- http://www.ruby-lang.org/es/
- http://www.seguridad.unam.mx/usuariocasero/secciones/diccionario.dsc?dicc=5184
- http://www.uhu.es/04004/material/Transparencias3.pdf
- http://www.unach.edu.ec/Virtualizacion/Sistemas_Operativos/paginas/Unidad%202.htm

RESPUESTAS A LOS EXÁMENES DE AUTOEVALUACIÓN
INFORMÁTICA 7

Tema 1		Tema 2		Tema 3		Tema 4	
1.	A	1.	B	1.	C	1.	B
2.	C	2.	C	2.	C	2.	A
3.	D	3.	A	3.	D	3.	B
4.	B	4.	C	4.	B	4.	D
5.	C	5.	B	5.	D	5.	A
6.	B	6.	D	6.	C	6.	D
7.	A	7.	A	7.	A	7.	A
8.	A	8.	D	8.	C	8.	C
9.	C	9.	A	9.	A	9.	A
10.	C	10.	D	10.	A	10.	C