Niveles del sistema

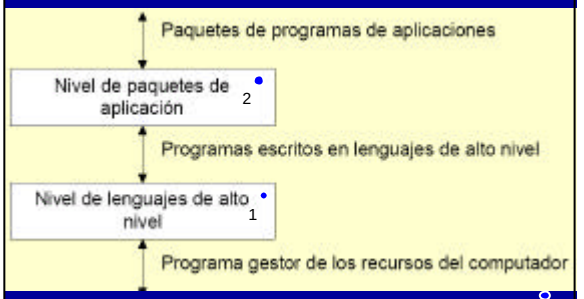
Usuarios y programas de aplicaciones

Componentes de un sistema de computación

- Usuarios :
  personas, máquinas u otros computadores
- Aplicaciones:
  consumen los recursos para resolver los problemas de computación de los usuarios

Usuarios

Usuarios y programa de aplicaciones



Lenguaje

- Lenguaje escrito utilizado por los seres humanos

  basado en convenciones :

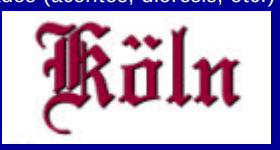
- La 1era Grupo de personas (tribu, clan, comunidad, etc.),

Siempre los mismos signos escritos para

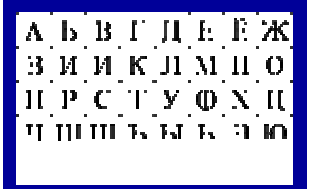
representar los sonidos utilizados en el lenguaje oral.

Lenguaje

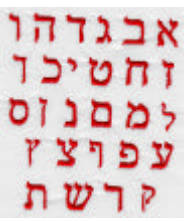
- Grupo humano, que en general habita América, el oeste y norte de Europa y partes de Oceanía,
- utiliza principalmente los símbolos básicos del alfabeto latino (o romano)
- O con algunas pequeñas modificaciones o agregados (acentos, diéresis, etc.) o también en su grafía.

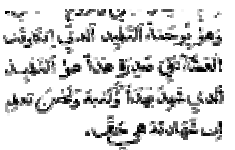


- Pero a medida que nos desplazamos hacia el este europeo observamos algunas diferencias sustanciales en los símbolos utilizados convencionalmente.



Lenguaje





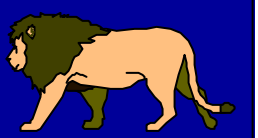
Lenguaje

• -> el Este Asiático, tenemos aún + diferencias, puesto que los símbolos ya no representan sonidos sino ideas.

Lenguaje

• La 2da. convención que se hace es de qué manera se combinan esos símbolos para formar palabras que denominan cosas.

• Así la combinación de símbolos: " L E O N ", convencionalmente representa a:



• Pero la combinación de símbolos: " L O N E ", no lo representa

Lenguaje

La 3.era convención que se hace es:

• de qué manera se combinan esas palabras para formar expresiones, frases y oraciones

que permiten la comunicación efectiva entre los integrantes del grupo.

Lenguaje

• Por ejemplo si decimos:

El pastaba blanco el verde en caballo prado,

es una expresión que no sirve para comunicar cosas; pero si decimos:

El caballo blanco pastaba en el prado verde,

esta sí es una expresión que permite comunicar una idea a otra persona.

Lenguaje

Un conjunto de caracteres, convenciones y reglas utilizado para comunicar información.

Las tres dimensiones de un lenguaje son:

La pragmática (práctica).

La semántica (significado).

La sintaxis (estructura).

Lenguaje

Una dimensión de un lenguaje es:

-La semántica, que tiene que ver con el significado de las palabras.

Y por ello debemos utilizar la palabra exacta para indicar lo que queremos comunicar;

teniendo en cuenta que la misma palabra puede tener significados ⇔ según el contexto en la cual se use.

Lenguaje

Por ejemplo :

La palabra CAJA tendrá significados diversos según el contexto :

- Contabilidad
- Música
- Anatomía
- Carpintería, etc.

Lenguaje

Otra dimensión de un lenguaje es:

- La sintaxis, que tiene que ver con la estructura de las frases.

Y esto nos indica si la frase está bien construída gramáticalmente,

o si por el contrario hay que modificar el orden de las palabras en la oración.

Lenguaje

Y por último otra dimensión de un lenguaje es la pragmática, que tiene que ver con la intención que subyace al efectuar la comunicación.

Por ejemplo: si se dice Guarden silencio !!! , la intención subyacente en esta comunicación es justamente que cesen las conversaciones en un lugar determinado (aula, oficina, hogar, etc.)

Lenguaje

Por otra parte, el lenguaje utilizado al realizar una comunicación siempre debe ser orientado;

y la orientación debe ser => el receptor de la comunicación,

teniendo en cuenta su capacidad de entendimiento

(términos que puede o no puede comprender, conocimientos previos que pudiera tener, etc.)

Lenguaje

En definitiva para efectuar exitosamente una comunicación, necesitamos:

a) Un emisor

b) Un receptor

c) Un lenguaje a través del cual se instrumentará la comunicación

Lenguaje

Y cuando el emisor es una persona y el receptor es una computadora; para efectuar la comunicación se utiliza un lenguaje de programación.

Lenguajes de programación

Un conjunto de caracteres, expresiones y símbolos, así como las reglas para combinarlos en instrucciones interpretables (directa o indirectamente) y ejecutables por el computador.

Un lenguaje que consta de todos los símbolos, los caracteres y las reglas de utilización que permite a las personas comunicarse con una computadora.

Lenguajes de programación

Un lenguaje de programación, es una escritura formal que permite indicar un modelo lógico de la resolución de un problema.

Lenguajes de programación

Todos los lenguajes de programación tienen una orientación a algo: ya sea al computador, o a los problemas del usuario.

También presentan las tres dimensiones de todo lenguaje:

a) la semántica

b) la sintaxis

c) la pragmática

Lenguajes de programación

Con respecto a la semántica se observa que en los lenguajes de programación generalmente existen determinados códigos o palabras que no pueden ser alterados y no pueden ser utilizados para expresar otra cosa.

Lenguajes de programación

Entre ellos tenemos las denominadas palabras reservadas que como su nombre lo indica son utilizadas exclusivamente para indicar a la computadora que haga una tarea determinada.

Lenguajes de programación

En relación a la sintaxis se observa que en todos los lenguajes de programación existen reglas de construcción de las instrucciones que conforman un programa, las que no pueden ser alteradas porque de lo contrario el equipo no sabría cómo ejecutar la instrucción.

Lenguajes de programación

Y con respecto a la pragmática observamos que la intención subyacente de todo programa de computación es la resolución de los problemas del usuario, por lo que el lenguaje de programación con el cual se escribe deberá contemplar esta dimensión.

¿Cómo diseñar un lenguaje?

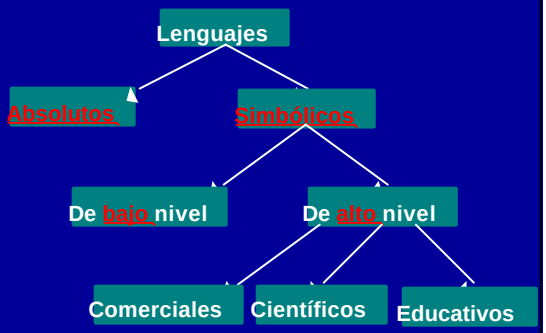
Se tiene que tomar en cuenta tres influencias principales:

1. La computadora subyacente en donde se van a ejecutar los programas escritos en el lenguaje.
2. El modelo de ejecución, o computadora virtual, que apoya a ese lenguaje en el equipo real.
3. El modelo de computación que el lenguaje implementa.

Paradigmas de los lenguajes de programación

- En ciencias de la computación un paradigma se puede definir como un conjunto de conceptos que permiten modelar el mundo.
- Un paradigma es usado para formular una solución de cómputo a un problema.

Clasificación de los lenguajes



Lenguaje absoluto

Un lenguaje dependiente del computador que éste interpreta directamente a través de la activación de sus circuitos mediante códigos binarios.

También llamado lenguaje máquina u objeto.

Lenguaje simbólico

- Un lenguaje de programación orientado al computador cuyas instrucciones tienen cierta correspondencia con las instrucciones en lenguaje absoluto, y cuenta con la posibilidad de utilizar códigos de operación y direcciones simbólicas mnemónicas, comentarios y algunas macroinstrucciones.

Lenguaje simbólico

• Se dividen en lenguajes de bajo nivel y lenguajes de alto nivel

• Deben ser traducidos al lenguaje máquina antes de ser ejecutados.

• También son llamados lenguaje fuente o código fuente

Lenguaje Simbólico clasificación x la proximidad al lenguaje natural

1. Lenguajes de bajo nivel: Lenguajes de máquina, microinstrucción, Ensambladores y Macroensambladores.

3. Lenguajes de alto nivel, como Pascal, Fortran, C, C++, Lisp, Basic, Prolog, Algol, etc.

Lenguajes de Bajo Nivel

Lenguajes simbólicos cuyas instrucciones tienen correspondencia unívoca (1 a 1) con las instrucciones en lenguaje absoluto.

Generalmente dependen de un tipo de computador.

Están totalmente vinculados al hardware, a la estructura interna del computador.

Lenguajes de Bajo Nivel

Utilizan programas ensambladores para la traducción, y están generalmente más orientados al equipo que al usuario. (ejemplo lenguaje assembler).

Cada problema tiene que ser dividido en instrucciones muy simples que se correspondan directamente con una instrucción en código máquina.

Estos lenguajes son únicos para cada microprocesador particular (ej. 8086, 80286, 80386).

Lenguajes de Alto Nivel

Lenguaje simbólico que permite a los usuarios (programadores) escribir instrucciones en una notación orientada a la resolución del problema más que al computador.

Cada instrucción (sentencia) en un lenguaje de alto nivel corresponde a varias instrucciones en lenguaje máquina.

Lenguajes de alto nivel x tipo de problema a resolver

1. Aplicaciones científicas, en donde predominan operaciones numéricas propias de algoritmos numéricos. Aquí tenemos a Fortran y Pascal, donde particularmente destaca Fortran .
2. Procesamiento de datos, como COBOL y SQL.
3. Tratamiento de textos como C.
4. Inteligencia artificial, como aplicaciones en sistemas expertos, juegos y visión artificial. Aquí tenemos a LISP y PROLOG.
5. Programación de Sistemas: Software que permite la interfaz entre el hardware y el usuario. Tenemos a ADA, MODULA-2 y C.

FORTRAN

```
    INTEGER I
    REAL X(10), SUM
    SUM = 0.0
    DO 100 I=1,10
100 SUM=SUM+ X(I)**2
    IF (SUM .GT. 1E+5) STOP
    WRITE(6,200) I,SUM
200 FORMAT ('SUMA', I, 'VALORES E',E15.7)
    END
```


Pascal

```
Program demouno;
Uses
  Crt;
Var             {Declaración de variables}
  A,b,c:integer;
                {Declaración de procedimientos}
procedure suma(a , b:Integer; var c:integer);
begin
  c:=a + b;
end;

Begin           {Cuerpo principal del programa}
  Clrscr;
  A:=5;
  B:=10;
  Suma(a,b,c);
  Writeln( c ) ;
End.            {punto obligatorio final programa}
```

Cobol

```
$ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  FragmentA.
AUTHOR.  Michael Coughlan.

DATA DIVISION.

WORKING-STORAGE SECTION.
01  Num1                    PIC 9  VALUE ZEROS.
01  Num2                    PIC 9  VALUE ZEROS.
01  Result                  PIC 99 VALUE ZEROS.

PROCEDURE DIVISION.
Calc-Result.
    ACCEPT Num1.
    MULTIPLY Num1 BY Num2 GIVING Result.
    ACCEPT Num2.
    DISPLAY "Result is = ", Result.
    STOP RUN.
```

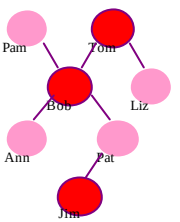

Lenguaje C

LISP

```
; Factorial

(defun fact (n)
   (if (= n 1)   ; caso de terminación -> 1! = 1
       1
       (* n (fact (- n 1)))  ; relación recursiva: n! = n * (n-1)!
   )
)
```


PROLOG

```
/* Esta linea es un comentario */
/* BASE DE HECHOS */
padre(pam, bob).
padre(tom, bob).
padre(tom, liz).
padre(bob, ann).
padre(bob, pat).
padre(pat, jim).
femenino(pam).
femenino(liz).
femenino(ann).
femenino(pat).
masculino(bob).
masculino(tom).
masculino(jim).

/* REGLAS */
es_hijo(X,Y):- padre(Y,X), masculino(X).
tia(X,Y):- padre(PY,Y), hermanos(X,PY), femenino(X).
hermanos(X,Y):- padre(PX,X), padre(PX,Y), no_es_igual(X,Y).
abuelo(X,Y):- padre(PY,Y), padre(X,PY), masculino(X).
no_es_igual(X,X):- !,fail.
no_es_igual(X,Y):- true.

/* PREGUNTA */
?- abuelo(tom,liz).
```



Ada

```
procedure DECOD_MENSAJE
   task GENERAR_CODIGOS;
   task DECODIFICAR is
       entry ENVIAR_CODIGO (C: in CHARACTER);
       entry RECIBIR_CAR (C: out CHARACTER);
   end;
   task IMPR_MENSAJES;
   task body GENERAR_CODIGOS is
       CODIGO_SIGUIENTE: CHARACTER;
   begin
            loop
       --setencias para recibir datos
       --y generar un valor para CODIGO_SIGUIENTE
       DECODIFICAR.ENVIAR_CODIGO (CODIGO_SIGUIENTE);
   end;
```



```
    task body DECODIFICAR is
        CODIGO, CAR : CHARACTER;
    begin
        loop
                              accept ENVIAR_CODIGO (C: in CHARACTER) do
                    CODIGO := C;
             end;
        --setencias para decodificar el valor de CODIGO
        --y presentar el valor decodificado en CAR
        accept     RECIBIR_CAR (C: out CHARACTER) do
                C:= CAR;
        end;
    end loop;
    end;

task body IMPR_MENSAJE is
    TAM_LINEA := 1;
    CAR_SIGUIENTE: CHARACTER;
    POSICION_LINEA: INTEGER;
    LINEA: STRING (1..TAM_LINEA);
begin
     POSICION_LINEA := 1;
    loop
        DECODIFICAR.RECIBIR_CAR (CAR_SIGUIENTE);
        LINEA(POSICION_LINEA):=CAR_SIGUIENTE;
        if POSICION_LINEA < TAM_LINEA then
             POSICION_LINEA := POSICION_LINEA + 1;
        else
             IMPRIMIR(LINEA);
             POSICION_LINEA := 1;
        end if;
    end loop;
end;

begin
    PUT ("SE HAN ACTIVADO LAS TAREAS PARA DECODIFICAR MENSAJES")
end;
```


Paradigmas de los lenguajes de programación

Existen 8 modelos que describen los lenguajes de programación:

- Imperativos o de procedimientos
- Aplicativos o funcionales
- Lenguajes con base en reglas o lógicos
- Orientados a objetos
- Concurrentes
- Markup – Lenguaje de Marcado o etiquetación
- Scripting – Guiones de texto
- Event-Driven –Disparadores de eventos

1. Lenguajes imperativos o de procedimientos

Se caracterizan por ser claros, formales y elegantes.

- Son controlados por enunciados imperativos

  enunciado 1;
  enunciado 2;
  ...

- La ejecución de un enunciado hace que el interprete cambie el valor de una o mas localidades en memoria
- Ejemplos: Fortran, Pascal, C, Algol, Ada, PL/1

Paradigmas de programación

1. Lenguajes imperativos o de procedimiento

La memoria es un conjunto de cajas

Ejs: C, Fortran, Pascal, COBOL, etc.

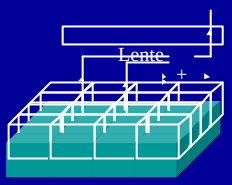
2. Lenguajes aplicativos o funcionales

Se caracterizan por ser muy eficientes, expresivos y semanticamente elegantes.

- Los lenguajes aplicativos hacen uso de las funciones puras con composición funcional, recursión y expresiones condicionales
- Tienen 4 componentes:
  - un conjunto de funciones primitivas
  - un conjunto de formas funcionales
  - la operación de aplicación
  - un conjunto de objetos de datos
- Ejemplos: LISP, ML

2. Lenguajes aplicativos o funcionales

Función(...(función2(función1(datos))...)...



Cambia la forma de acceder a datos desde la memoria

Ejs.: LISP, ML

3. Lenguajes con base en reglas o lógicos

Se caracterizan por ser eficaces y veloces

- Se ejecutan verificando una condición, que cuando se satisface ejecutan una acción:

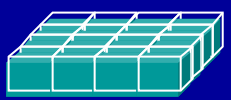
  condición 1 entonces acción 1

  condicion 2 entonces accion 2

  ...
- Ejemplo: Prolog

3. Lenguajes con base en reglas

Condición habilitadora 1
Condición habilitadora 2
...



Usar filtros para habilitar cambio de estado

Ejs.: Prolog

4. Lenguajes orientados a objetos

Se caracterizan por trabajar con entes abstractos (objetos) que reflejan las propiedades y características de objetos o entes reales

- Las actividades a realizarse se tornan alrededor de los objetos mediante métodos (funciones)
- La comunicación con el objeto se da a través de mensajes
- Un aspecto fundamental es el concepto de herencia que se da cuando los objetos pertenecen a la misma clase
- Ejemplos: Simula, Smalltalk, Java.

4. Lenguajes orientados a objetos

Objetos complejos de datos
+ funciones que operan esos datos

Objetos de datos
-> eficiencia de lenguajes imperativos

Funciones asociadas
-> flexibilidad y confiabilidad mod. Aplicativos

Los Objectos Tienen:
- Atributos: Campos
- Métodos: Operaciones

Funciones Virtuales en C++

```
// Archivo descrip1.hpp
class description {
protected:
  char * information
public:
  description(char *info):
    information(info){}
  void print(){
    printf("%s\n", information);
  }
};

// Archivo descrip2.hpp
class sphere: public description {
private:
  float radius;
public:
  sphere(char * info, float rad):
    (info),radius(rad){}
  void print(){
    printf("%s\n", information);
    printf("radio=%g\n", radius);
  }
};
```

```
// Archivo descrip.cpp
sphere small_ball("mini",1.0),
       beach_ball("plastico", 24.0),
       planetoid("luna", 1e24);

description *shapes[] = {
  &small_ball,
  &beach_ball,
  &planetoid
}

main() {
double tam;
tam = sizeof(shapes)/sizeof(shapes[0]);
small_ball.print();
beach_ball.print();
planetoid.print();

for(int=0;i<tam; i++)
  shapes[i]->print();
}
```

```
mini
radio=1.0
plastico
radio=24.0
luna
radio=1e+024
mini
plastico
luna
```

Funciones Virtuales en C++

```
// Archivo descrip1.hpp
class description {
protected:
  char * information
public:
  description(char *info):
    information(info){}
  virtual void print(){
    printf("%s\n", information);
  }
};

// Archivo descrip2.hpp
class sphere: public description {
private:
  float radius;
public:
  sphere(char * info, float rad):
    (info),radius(rad){}
  virtual void print(){
    printf("%s\n", information);
    printf("radio=%g\n", radius);
  }
};
```

```
// Archivo descrip.cpp
sphere small_ball("mini",1.0),
       beach_ball("plastico", 24.0),
       planetoid("luna", 1e24);

description *shapes[] = {
  &small_ball,
  &beach_ball,
  &planetoid
}

main() {
double tam;
tam = sizeof(shapes)/si
small_ball.print();
beach_ball.print();
planetoid.print();

for(int=0;i<tam; i++)
  shapes[i]->print();
}
```

```
mini
radio=1.0
plastico
radio=24.0
luna
radio=1e+024
mini
radio=1.0
plastico
radio=24.0
luna
radio=1e+024
```

Java

```
class TestTh extends Thread {
    private String nombre;
    private int retardo;
    // Constructor para almacenar nuestro nombre
    // y el retardo
    public TestTh( String s,int d ) {
        nombre = s;
        retardo = d;
    }
    // El metodo run() es similar al main(), pero para
    // threads. Cuando run() termina el thread muere
    public void run() {
        // Retasamos la ejecución el tiempo especificado
        try {
            sleep( retardo );
        } catch( InterruptedException e ) {
            ;
            }
        // Ahora imprimimos el nombre
        System.out.println( "Hola Mundo! "+nombre+" "+retardo );
    }
}
```

Java

```
public class MultiHola {
    public static void main( String args[] ) {
        TestTh t1,t2,t3;
        // Creamos los threads
        t1 = new TestTh( "Thread 1",(int)(Math.random()*2000) );
        t2 = new TestTh( "Thread 2",(int)(Math.random()*2000) );
        t3 = new TestTh( "Thread 3",(int)(Math.random()*2000) );
        // Arrancamos los threads
        t1.start();
        t2.start();
        t3.start();
        }
    }
```


5. Lenguajes Concurrentes

Su principal objetivo es mejorar la velocidad de computo, compartir recursos y distribuir la carga de trabajo :

- Diferentes tipos de arquitecturas:
  - Redes de cobertura amplia
  - Redes Locales
  - Multiprocesadores (*Clusters*)

5. Lenguajes Concurrentes

- Comunicación y cooperación entre aplicaciones a través de :
  - Envío y recepción de mensajes
  - Llamado a procedimiento remoto
  - Comunicación de grupo
  - Memoria Virtual Distribuida
- Ejemplos: PVM, CSP, Ada

6. Definiciones de marcado

- Documentos de papel:
  - Se refiere a la manera en la que el editor anota los manuscritos con especificaciones tipográficas.
- Documentos electrónicos:
  - Describe los códigos o etiquetas que añadidos al texto definen su estructura y formato.
- El lenguaje de marcado:
  - Define un conjunto de signos y reglas con los que etiquetar los documentos.

6. Lenguajes de marcado I

- Poseen un estructura lógica y una estructura física:
  - La estructura **lógica** está formada por las distintas partes que lo componen y por sus relaciones.
  - La estructura **física** indica la apariencia del documento, ya sea en papel o en la pantalla.

6. Lenguajes de marcado II

- En el medio impreso no se pueden separar las dos estructuras (lógica y física).
- En el electrónico se pueden almacenar independientemente. Esto es posible gracias a los lenguajes de *marcado genérico*.
- Otro tipo de marcado es el denominado *específico*, p. e. Script, Tex.

6. **Un Documento HTML**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
        <HEAD>
        <!-- esto es la cabecera del documento -->
        </HEAD>
        <BODY>
        <!-- este es el cuerpo del documento -->
        </BODY>
</HTML>
```


6. El lenguaje SGML I

- **Standard Generalized Markup Language**. Norma ISO 8879.
- Tiene su origen en la **industria editorial** necesitados de tener una manera normalizada de transmitir los documentos en su formato adecuado en los procesos de edición e impresión.
- En cierto sentido SGML es un **metalenguaje**, pues se considera como un sistema para la especificación de lenguajes de marcado.

6. El lenguaje SGML II

- Utiliza una *definición del tipo de documento* o **DTD** que permite especificar la estructura lógica de una clase de escrito.
- Una DTD es una definición formal que indica:
  - que elementos se incluyen como contenido de los documentos
  - y en que orden.
- Cada elemento del documento se marca
  - mediante una etiqueta de comienzo
  - otra de final.
  - Cada etiqueta se especifica mediante
    - un identificador genérico
    - y unos atributos.

6. SGML (ventajas)

- Existe una separación entre
  - la estructura
  - y la presentación del documento.
- Es independiente del idioma, al poderse definir el alfabeto a utilizar.
- El documento puede incluir cualquier tipo de información multimedia.
- Es independiente del tipo de ordenador y de la aplicación, permitiendo el intercambio entre plataformas diferentes.

6. SGML (desventajas)

- Es muy complejo al tener que hacer frente a una gran variedad de tipos de documentos.
- Al no imponer limitaciones a la hora de estructurar documentos, se podría dar el caso de que dos DTD destinados al mismo documento fueran incompatibles entre sí.
- Los contenidos multimedia sólo se pueden utilizar como entidades externas a los incluidos en el documento.

HTML
6. Hiperdocumentos en Internet

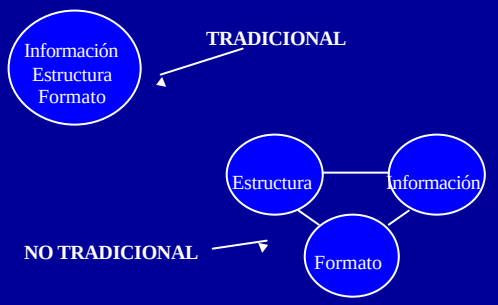
- ISO 15445
- **HyperText Markup Language**, es un lenguaje de marcado para el intercambio de hiperdocumentos.
- Se usa en combinación con **HTTP**, un protocolo de red para la transferencia de información de diversa naturaleza a través de Internet.
- Está basado en SGML y se puede considerar una DTD.

6. El lenguaje XML

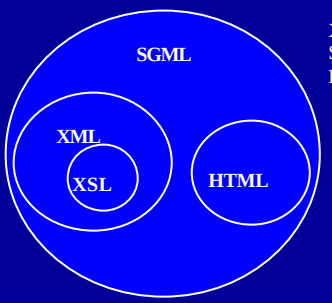
6. Características generales

- XML (e**X**tended **M**arkup **L**anguage)
- Es un subconjunto simplificado de SGML.
- Es un formato de texto estandarizado que sirve para representar información estructurada en la Web.
- XML es una especificación que sirve para especificar lenguajes de marcado, es por tanto un Metalenguaje.

6. Documentos tradicionales y no tradicionales



6. Las familias de SGML

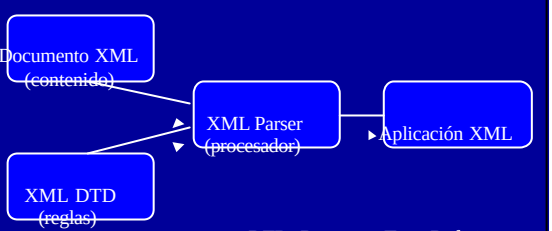


XSL: XML Stylesheet Language

6. Introducción a XML

- XML es un meta-lenguaje que nos permite definir lenguajes de marcado, adecuados a usos determinados
- XML es un estándar de Internet, aprobado por la W3C.
- Los proveedores tratan de ajustarse a las especificaciones que W3C dicta.

6. El sistema XML



DTD: Document Type Definition

6. Documento XML

- Está basado en entidades que pueden consistir de Contenidos y Marcas.
- Los Contenidos es la información real, también se denomina como “character data”.
- Los Contenidos se enmarcan entre Marcas. Es decir etiquetas al estilo de HTML.

6. XML DTD

- Se utilizan para asegurar que la información está estructurada adecuadamente.
- Las DTDs son series de expresiones que definen la estructura lógica del documento XML.
- Las DTD son opcionales.

6. XML Parser

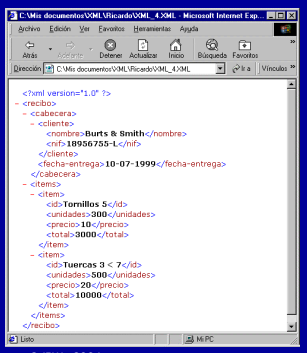
- Es un motor software que chequea el documento XML para asegurar que es correcto sintácticamente.
- Si se utiliza DTD el parser chequea también el documento frente al DTD para asegurar que está estructurado correctamente.

6. Aplicación XML

- Las aplicaciones XML procesan la información incluida en los documentos XML.
- No hay límites virtuales acerca de lo que una aplicación XML puede hacer.

http://www.w3schools.com/xml/default.asp

Código XML



Código XSL

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  <!-- Edited with XML Spy v4.2  -->
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/xhtml1/strict">
<body style="font-family:Arial,helvetica,sans-serif;font-size:12pt; background-color:#EEEEEE">
<xsl:for-eachselect="breakfast_menu/food">
<div style="background-colorteal;color:white;padding:4px">
<span style="font-weight:bold;color:white">
<xsl:value-of select="name" />
</span>
-
<xsl:value-of select="price" />
</div>
<div style="margin-left:20pxmargin-bottom:1em;font-size:10pt">
<xsl:value-of select="description" />
<span style="font-style:italic">
(
<xsl:value-of select="calories" />
calories per serving)
</span>
</div>
</xsl:for-each>
</body>
</html>
```

Código en explorer

Resultado programa XML

- **Belgian Waffles** - $5.95
- two of our famous Belgian Waffles with plenty of real maple syrup *(650 calories per serving)*
- **Strawberry Belgian Waffles** - $7.95
- light Belgian waffles covered with strawberries and whipped cream *(900 calories per serving)*
- **Berry-Berry Belgian Waffles** - $8.95
- light Belgian waffles covered with an assortment of fresh berries and whipped cream *(900 calories per serving)*
- **FrenchToast** - $4.50
- thick slices made from our homemade sourdough bread *(600 calories per serving)*
- **Homestyle Breakfast** - $6.95
- two eggs, bacon or sausage, toast, and our ever-popular hash browns *(950 calories per serving)*

Ejemplo en Explorer

¿Qué es XML?

- Es un subconjunto de SGML (*Standard Generalized Mark-up Language*), simplificado y adaptado a Internet
- XML (*eXtensible Mark-up Language*) no es un lenguaje de marcado
- XML es un **meta-lenguaje** que permite definir lenguajes de marcado adecuados a usos

```
<libro>
      <autor>Antonio Muñoz Molina</autor>
      <titulo>El Jinete Polaco</titulo>
      <precio moneda="EURO">20</precio>
</libro>
```

Ventajas de XML

- Procesable por humanos y por software
- Separa la **información** o el **contenido** de su **presentación** o **formato**

- Ideal para transacciones B2B
- Permite poderosas técnicas de extracción de información y *data-mining*
- Fácil análisis sintáctico

Lenguajes de marcado genérico

- **ODA**: Open Document Architecture, es una norma internacional que se utiliza para representar e intercambiar documentos electrónicos.
- **PostScript**: Pertenece al grupo de lenguajes de descripción de página (PDL). Este formato es originario de Adobe. En la actualidad ha sido sustituido por Acrobat y su formato PDF.

Interaccion de las tecnologías:

•XML: Intercambio de Información
•XML+XSL: Formato de publicación
•XML+XSL+XLL: Navegación
•XML+XSL+XLL+ script: DXML (aplicación)

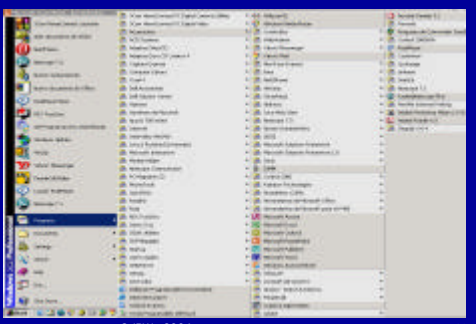
7. Lenguaje Script

- Instrucciones registradas en y como guiones de texto (scripts)
- Un guión de texto (Script) requiere de una aplicación host que interprete and ejecute cada línea como guiónes de texto (script) durante su ejecución.
- Ejemplo : los Guiones de texto Java (Java Script)

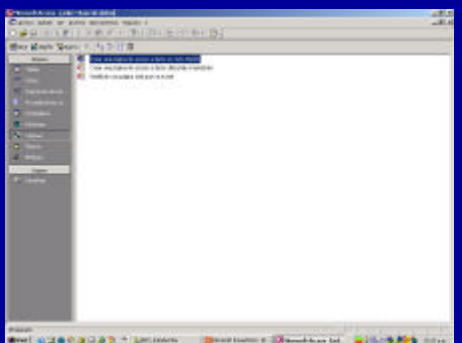
8. Disparadores de Eventos

- Los códigos son añadidos a Objetos (Attach code to Objects).
- La Aplicación es disparada (iniciada) por eventos

Programas escritos en lenguajes de alto nivel



Paquetes de programas de aplicaciones



Paquetes de programas de aplicaciones

REFERENCIAS:

PETER wegner, Programing with ADA: an introduction by means of graduates examples, Prentice hall 1980, Englewood Cliffs N.Y.

Vowels Robin A. , Algol 60 and Fortran IV ,IW & SA 1974.

http://www.berts.docs/tutorials

SQL
http://www.jegsworks.com/Lessons -sp/lesson6/lesson9-2.htm

C++
http://www.jegsworks.com/Lessons -sp/lesson9/lesson9-2.htm

COBOL
http://www.jegsworks.com/Lessons-sp/lesson9/lesson9-2.htm

FORTRAN
http://www.geocities.com/SiliconValley/Horizon/2536/ejemplos.htm

JAVA
http://www.jegsworks.com/Lessons-sp/lesson9/lesson9-2.htm

PASCAL
http://members.tripod.com.ar/pascal/estructura_de_un_programa_en_pas.htm

REFERENCIAS:

PROLOG
http://www.uam.edu.ni/uam99/ingieneria/prolog/

Smalltalk
http://www.dc.uba.ar/people/materias/plp/Smalltalk.html

APL (A Programming Language)
http://www.engin.umd.umich.edu/CIS/course.des/cis400/index.html

XML (EXTENDED MARKUP LANGUAGE)
http://www.programacion.net/html/xml/htmdsssl/capitulo3/capitulo3.htm

PERL (Practical Extraction and Report Language)
http://132.248.71.81/lacertus/tutorial_perl.html

LENGUAJE C
http://habitantes.elsitio.com/cafcocar/Lista_programas.html

SNOBOL
http://www.engin.umd.umich.edu/CIS/course.des/cis400/snobol/word.htm

ML
http://cs.wwc.edu/Environment/SML-Tutorial.html

PL/1
http://www.engin.umd.umich.edu/CIS/course.des/cis400/pl1/pl1bubble.h