LLMSGI

UD1. Introducció als llenguatges de marques.

Índex

Índex 2
1. Ordenador e informació 3
Maneres de representar informació a l'ordinador 3
Dades en forma de text y dades binàries 3
Dades binàries 3
Text 4
2. Codificando de texto a binario 4
El codi ASCII 4
Unicode 7
Fitxers binaris i fitxers de text 9
Avantatges dels arxius binaris 9
Avantatges dels fitxers de text 9
3. Exportar/importar dades 9
El problema de compartir dades 9
El text com el format més versàtil 10
4. Llenguatges de marques 11
Introducció històrica 11
Aparició dels llenguatges de marques 11
Charles Goldfarb i el GML 11
TeX i LaTeX 12
RTF 13
SGML 15
PostScript 15
HTML 16
XML 16
Markdown 17
JSON 17
Tipus de llenguatges de marques 18
5. Webgrafia 19

1. Ordenador e informació

Maneres de representar informació a l'ordinador

L'ordinador és una màquina digital, per tant només és capaç de representar informació fent servir el sistema binari de numeració. Això obliga que, per poder emmagatzemar informació en un ordinador, prèviament calga **codificar**-la en forma de números binaris.

El problema dels números binaris és que estan molt allunyats de l'ésser humà; és a dir, que les persones no estem capacitades per manejar informació en binari. Nosaltres fem servir un sistema numèric decimal per als números i sistemes molt més complexos per representar altres tipus d'informació (com el text, les imatges, la música,…)

Al principi els ordinadors només eren capaços de manipular números (de fet encara, la CPU, el cervell dels ordinadors segueix treballant en binari), però actualment no hi ha pràcticament cap mena d'informació que no estiguem manipulant a través de l'ordinador: així fem servir text, imatges, música, vídeo,... etc. a través de l'ordinador. Això és possible perquè s'ha aconseguit que gairebé qualsevol tipus d'informació es pugui transformar a binari.

Els éssers humans tenim la capacitat de diferenciar clarament allò que és un text d'una imatge, allò que és un nombre d'una cançó, però en un ordinador tot és més complicat, perquè tot és binari.

Aquest procés de transformar la informació humana al seu equivalent màquina, es coneix com a **codificació**. El principal problema és que no hi ha una sola manera de codificar, és a dir, una mateixa informació (una fotografia per exemple) es pot codificar a binari de diferents maneres. Tot i que avui dia ja tenim nombrosos estàndards, encara continua sent un dels aspectes problemàtics de la informàtica.

Fonamentalment la informació que un ordinador maneja són números i text. Però, curiosament, a nivell formal només es consideren dades binàries a la informació representable a l'ordinador que no és text (imatge, so, vídeo,…); encara que, com ja hem comentat, en realitat tota la informació que maneja un ordinador és binària, inclòs el text.

Dades en forma de text y dades binàries

Dades binàries

Com s'ha comentat abans, a qualsevol dada codificada en un ordinador que no sigui text, se'l considera dada binària. Exemples de dades binàries són: una cançó, un vídeo, una imatge, una aplicació informàtica o un document creat amb el programari Microsoft Word.

La forma de codificar aquest tipus de dades a la forma binària és diferent en cada cas. Per exemple, en el cas de les imatges, cada punt (**píxel**) de la imatge es codifica utilitzant el nivell de vermell, verd i blau. Així per exemple el codi 11111111 00000000 000000, es correspondria a un píxel de color roig pur (té de gom a gom els nivells de roig i totalment apagats els nivells de verd i blau). De manera que una sola imatge requereix milions de dígits binaris (és a dir, milions de bits).

En qualsevol cas, sigua quina sigua la informació que estem codificant en binari, per poder accedir a aquesta informació, l'ordinador necessita un programari que sàpiga com **descodificar**-la. Això significa què deu conèixer què significa cada dígit binari per traduir-lo a la seva forma original. Això només és possible utilitzant el mateix programari amb què es va codificar, o bé un altre programari capaç de reconèixer aquest format.

Per exemple, el programari *Microsoft Word* enregistra els documents que l'usuari crea en format *docx*. Si obrim un document així creat amb un editor de text (com el Bloc de Notes), no aconseguirem veure el fitxer original. Però si l'obrim amb el Word sí que ho veurem. Word sap com interpretar la informació binària del fitxer. També podem obrir el fitxer amb un altre programari com *Google Docs*, ja que és capaç d'entendre aquesta informació.

Text

El text és potser la manera més humana de representar informació. Abans de l'arribada de l'ordinador, la informació es transmetia mitjançant documents o llibres en paper. Aquesta forma de transmetre és mil·lenària i continua sent la forma més habitual de transmetre informació entre humans; fins i tot amb la tecnologia actual, aplicacions com *twitter* o *whatssap* segueixen usant el text com a format fonamental per transmetre informació.

Quan va aparèixer la informàtica com la ciència que es va ocupar de la informació digital, va aparèixer també el problema de com codificar text en forma de dígits binaris per fer-ho representable a l'ordinador. La forma habitual ha estat codificar cada caràcter en una sèrie de números binaris. Així exemple el caràcter A es codifica com a *01000001* i la B el *01000010* (així es codifiquen usant el codi ASCII).

El problema va sorgir per la falta d'estandardització: la lletra A es codificava de manera diferent depenent del sistema que s'utilitzi. El problema passa quan volem traslladar la informació d'un ordinador a un altre.

Per això, a poc a poc van aparèixer estàndards que pretenien aconseguir que tot el maquinari i el programari codificaren els caràcters de la mateixa manera, independentment del sistema que utilitzem.

Aquest problema continua passant actualment. Així, si escrivim al Bloc de Notes de Windows un text que contingui la lletra **enye** (com España), i després marxem al terminal del sistema i mostrem aquest arxiu, veurem que l'enye no es llegeix bé al terminal. La raó és que Windows utilitza la codificació anomenada Windows 1252 i el terminal clàssic (procedent del vell sistema **MS-DOS**) utilitza normalment (a Espanya) la pàgina de codis CP 850.

2. Codificando de texto a binario

El codi ASCII

El problema de la codificació de text que feia incompatibles els documents de text entre diferents sistemes, es va pal·liar quan es va idear el 1967 un codi estàndard per part de l'ANSI, l'agència d'estàndards nord-americana, aquest codi és l'anomenat ASCII (American Standard Code for

Information Interchange, codi estàndard americà per a l'intercanvi d'informació). El codi utilitza l'alfabet anglès (que utilitza caràcters llatins) i per codificar tots els possibles caràcters necessaris per escriure en anglès es va idear un sistema de 7 bits (amb 7 bits es poden representar 128 símbols, suficients per a totes les lletres de l'alfabet anglès, en minúscules i majúscules, caràcters de puntuació, símbols especials i fins i tot símbols de control).

El codi ASCII és el següent:

Núm.	Significado	¿Control?	Núm	Sign.	¿Control?
0	Carácter nulo	Sí	64	@	No
1	Inicio de Encabezado	Sí, ctrl-A	65	A	No
2	Inicio de Texto	Sí, ctrl-B	66	B	No
3	Fin de Texto	Sí, ctrl-C	67	C	No
4	Fin de Transmisión	Sí, ctrl-D	68	D	No
5	Petición	Sí, ctrl-E	69	E	No
6	Confirmación	Sí, ctrl-F	70	F	No
7	Timbre	Sí, ctrl-G	71	G	No
8	Retroceso	Sí, ctrl-H	72	H	No
9	Tabulación horizontal	Sí, ctrl-I	73	I	No
10	Alimentación de línea	Sí, ctrl-J	74	J	No
11	Tabulación Vertical	Sí, ctrl-K	75	K	No
12	Alimentación de carro	Sí, ctrl-L	76	L	No
13	Retorno de carro	Sí, ctrl-M	77	M	No
14	Quitar mayúsculas	Sí, ctrl-N	78	N	No
15	Poner mayúsculas	Sí, ctrl-O	79	O	No
16	Data Link Escape	Sí, ctrl-P	80	P	No
17	Control Disp-1	Sí, ctrl-Q	81	Q	No
18	Control Disp-2	Sí, ctrl-R	82	R	No
19	Control Disp-3	Sí, ctrl-S	83	S	No
20	Control Disp-4	Sí, ctrl-T	84	T	No
21	Confirmación negativa	Sí, ctrl-U	85	U	No
22	Idle síncrono	Sí, ctrl-V	86	V	No
23	Fin de bloque de transmisión	Sí, ctrl-W	87	W	No
24	Cancelar	Sí, ctrl-X	88	X	No
25	Fin de mitad	Sí, ctrl-Y	89	Y	No
26	Sustituto	Sí, ctrl-Z	90	Z	No
27	Escape	Sí, ctrl-[	91	[	No
28	EOF	Sí, ctrl-\	92	\	No
29	Separador de Grupo	Sí, ctrl-]	93	]	No
30	Separador de registro	Sí, ctrl-^	94	^	No
31	Separador de unidad	Sí, ctrl-_	95	_	No
32	Espacio	No	96	`	No
33	!	No	97	a	No

34	"	No	98	b	No
35	#	No	99	c	No
36	$	No	100	d	No
37	%	No	101	e	No
38	&	No	102	f	No
39	'	No	103	g	No
40	(	No	104	h	No
41	)	No	105	i	No
42	*	No	106	j	No
43	+	No	107	k	No
44	,	No	108	l	No
45	-	No	109	m	No
46	.	No	110	n	No
47	/	No	111	o	No
48	0	No	112	p	No
49	1	No	113	q	No
50	2	No	114	r	No
51	3	No	115	s	No
52	4	No	116	t	No
53	5	No	117	u	No
54	6	No	118	v	No
55	7	No	119	w	No
56	8	No	120	x	No
57	9	No	121	y	No
58	:	No	122	z	No
59	;	No	123	{	No
60	<	No	124	|	No
61	=	No	125	}	No
62	>	No	126	~	No
63	?	No	127	Borrado	No

Però, a països amb llengües diferents de l'anglès, va sorgir el problema que part dels símbols dels seus alfabets quedaven fora de l'ASCII (com passava amb la lletra enye a Espanya).

Per això es van dissenyar codis de 8 bits que afegien 128 símbols més i així van aparèixer els anomenats codis ASCII estesos. Els 128 primers símbols són els mateixos de la taula ASCII original i els 128 següents es corresponen a símbols extra. Així, per exemple, el sistema MS-DOS utilitzava l'anomenat codi 437, que incloïa símbols i caràcters de llengües d'Europa Occidental, a més de caràcters útils, com ara els que permetien fer marcs i vores en pantalles de text. Per la seva banda, Windows utilitza el codi 1252 i la seva consola clàssica el 850.

8 bits segueixen sent insuficients per codificar tots els alfabets del planeta, per la qual cosa cada zona feia servir la seva pròpia taula ASCII estesa. Davant del caos consegüent, la ISO va decidir normalitzar aquestes taules de codis per aconseguir-ne versions estàndards. Ho va fer mitjançant les següents normes (cadascuna de les quals definia una taula de 256 caràcters; sempre els 128 primers són l'ASCII original i els següents 128 són els que s'utilitzen per als símbols de la zona geogràfica concreta)

- **8859-1.** ASCII estès per a Europa Occidental (inclou símbols com ñ o β)
- **8859-2**. ASCII estès per a Europa Central i de l'Est (inclou símbols com Ź o č)
- **8859-3.** ASCII estès per a Europa del Sud (inclou símbols com Ġ o Ï)
- **8859-4.** ASCII estès per a Europa del Nord (inclou símbols com ø o å)
- **8859-5.** ASCII estès per alfabet ciríl·lic (inclou símbols com д o Ж)
- **8859-6**. ASCII estès per a alfabet àrab (inclou símbols com ن o ي)
- **8859-7.** ASCII estès per a alfabet grec modern (inclou símbols com ф o α)
- **8859-8**. ASCII estès per alfabet hebreu (inclou símbols com ץ o ק)
- **8859-9.** ASCII estès, versió 8859-1 que inclou símbols turcs en lloc d'altres poc utilitzats
- **8859-10.** ASCII estès, versió de 8859-4 que inclou símbols més utilitzats en les llengües nòrdiques actuals
- **8859-11.** ASCII estès per alfabet tailandès (inclou símbols com ๗ o ๘)
- **8859-12.** ASCII estès per alfabet devanagari de l'Índia i el Nepal que ja no es fa servir
- **8859-13.** ASCII estès per a alfabets bàltics amb símbols que no estaven en 8859-4
- **8859-14.** ASCII estès per a alfabet cèltic (inclou símbols com ŵ o Ẃ)
- **8859-15.** ASCII estès, versió 8859-1 que inclou el símbol de l'euro i símbols de llengües bàltiques. És el recomanat actualment per a Europa Occidental.
- **8859-16.** ASCII estès, versió de 8859-1 pensada per als països del sud-est d'Europa
- **2022-JP.** Símbols japonesos (part 1)
- **2022-JP-2.** Símbols japonesos (part 2)
- **2022-KR.** Símbols coreans

Aquest problema continua existint ara, de manera que als documents de text cal indicar el sistema de codificació utilitzat (el cas més evident són les pàgines web), per saber com interpretar els codis del fitxer. Així en 8859_1 el codi 245 és el caràcter õ i en 8859_2 és el caràcter õ

Unicode

La complicació de les taules de codi s'intenta resoldre gràcies al sistema **Unicode**. Aquest sistema pretén aglutinar en una mateixa taula de codis tots els caràcters de qualsevol llengua del planeta. A canvi, cada caràcter ja no ocupa ni un sol byte.

A Unicode a cada caràcter se li assigna un número. Els 128 primers continuen sent els originals de l'ASCII per mantenir la compatibilitat amb els textos ja codificats. Els 128 següents es corresponen amb els de la taula ISO-8859_1, de manera que els textos codificats en aquesta taula (l'habitual de les llengües d'Europa Occidental) són compatibles amb Unicode.

Per això un organisme, també anomenat Unicode (http://unicode.org/), participat per nombroses i influents empreses informàtiques i coordinat per la pròpia ISO, s'encarrega de definir la taula de codis i a més ha definit tres formes fonamentals de codificar els caràcters:

- **UTF-8.** És la més utilitzada (i la més complexa de processar per a l'ordinador). Per a cada caràcter s'usa d'un a quatre bytes, de manera que:
    - Els caràcters que pertanyen al codi ASCII original ocupen un byte.
    - Dos bytes ocupen els pertanyents a llengües llatines, ciríl·liques, gregues, àrabs, hebrees i altres d'Europa, Àsia Menor i Egipte.
    - Tres per a símbols dels alfabets en ús diferents dels de l'apartat anterior com el xinès, el tailandès, el coreà o el japonès.
    - Quatre per a altres símbols: per exemple, els matemàtics i símbols de llengües mortes com el fenici o l'assiri o símbols asiàtics d'ús poc freqüent.
- **UTF-16.** Utilitza per a cada caràcter dos (per als dos primers grups del punt anterior) o quatre caràcters (per a la resta). És més senzill que l'anterior
- **UTF-32.** La més senzilla de totes. Cada caràcter independentment del grup a què pertanyi ocupa 4 caràcters. No es fa servir.

Exemple, el text: 取得cigüeña𐤀 es codificaria d'aquesta manera:

- **A UTF-8**:
    - Com que són símbols xinesos, ocupen 3 bytes cadascun
    - Els caràcters c i g e ocupen un sol byte cadascun en ser part de l'ASCII original
    - Els símbols ü i ñ ocupen dos bytes
    - El símbol 𐤀 és persa antic i ocupa 4 bytes
    - En total el text ocupa 19 bytes
- **A UTF-16**:
    - Els símbols i 𐤀 ocupen 4 bytes cadascun
    - La resta 2 bytes
    - En total ocupen 26 bytes
- **A UTF-32**
    - Tots ocupen 4 bytes
    - Total: 40 bytes

Tot i que UTF-8 és més complexa de processar, actualment és el clar estàndard per codificar text. Tots els sistemes i dispositius actuals tenen capacitat per codificar així, per la qual cosa sembla que serà l'estàndard definitiu en els propers anys.

Fitxers binaris i fitxers de text

Avantatges dels arxius binaris

1. Ocupen menys espai que els fitxers de text, ja que optimitzen millor la seva codificació a binari (per exemple el número 213 ocupa un sol byte i no tres com passaria si fos un text).

2. Són més ràpids de manipular per part de l'ordinador (s'assemblen més al llenguatge de la màquina)

3. Permeten l'accés directe a les dades. Els fitxers de text sempre es manegen de manera seqüencial, més lenta

4. Les dades no són fàcilment interpretables, la qual cosa aporta certa ocultació al contingut. El contingut dels fitxers de text és fàcilment interpretable.

5. Els arxius binaris són ideals per emmagatzemar contingut xifrat. És possible xifrar el text també, però els algorismes de xifrat són més segurs si es fan servir tècniques no textuals.

Avantatges dels fitxers de text

1. Són ideals per emmagatzemar dades per exportar i importar informació a qualsevol dispositiu electrònic.

2. Són directament modificables, sense haver d'acudir a programari específic.

3. La seva manipulació és més senzilla que la dels arxius binaris.

4. Els dispositius de xarxa i programari client permeten el pas d'arxius de text ja que no són susceptibles de contenir virus informàtics.

3. Exportar/importar dades

El problema de compartir dades

Els problemes relacionats amb l'intercanvi d'informació entre aplicacions i màquines informàtiques és tan vell com la pròpia informàtica.

El problema parteix del fet d'haver realitzat una determinada feina amb un programari en un determinat ordinador i després voler passar aquesta feina a un altre programari en aquest o altre ordinador.

Els fitxers binaris tenen la complicació que per fer aquest procés, l'origen i el destí de les dades han de comprendre com codificar i descodificar la informació. Això, en molts casos, ha estat un

gran problema que ha obligat que tots els treballadors i les treballadores s'hagin hagut d'adaptar al programari de l'empresa. D'altra banda, dificulta que les empreses migren altres sistemes per la por de perdre les dades.

A la informàtica actual això és encara més problemàtic en tenir una necessitat de disponibilitat global del treball a través de dispositius molt diferents com tauletes, smartphones i altres dispositius portàtils.

Per això, a poc a poc han aparegut formats binaris de fitxer que han estat estàndards de facto (encara que la majoria no han estat reconeguts per cap organisme d'estàndards) com ara el format documental **PDF**, el format d'imatge **JPEG** o el format d'àudio **MP3**.

Però el problema fonamental continua estant al programari empresarial. El cas típic podria ser una empresa que utilitza a les seves oficines el programari **Apache Open Office** i després una persona de l'empresa vol obrir el document on està treballant en un dispositiu que té instal·lat Microsoft Office. És probable que no pugueu veure la feina en tots dos dispositius.

Aquest problema pot passar fins i tot amb el mateix programari però amb versions diferents (per exemple, intentar obrir un document creat amb **Microsoft Word 2016** en una màquina amb **Microsoft Word 2001**).

Per això moltes vegades l'opció per exportar i importar dades és utilitzar **convertidors**. Programari amb capacitat de convertir les dades d'un format a un altre (per exemple de **Word** a **Open Office**; de **MP3** a **MOV** d'Apple, etc.).

El text com el format més versàtil

Com ja hem explicat, hi ha un format de fitxer que qualsevol dispositiu és capaç d'entendre: el text. La qüestió és que els fitxers anomenats de text, només són capaços d'emmagatzemar text pla; és a dir, només text sense indicar cap format o afegir informació no textual.

A causa de la facilitat de ser llegit amb qualsevol aparell, s'intenta que el text mateix servisca per emmagatzemar altres dades, és a dir informació que no és text sense més ni més. Evidentment hi ha tipus d'informació pràcticament impossibles de representar en un fitxer de text, però sí que hi ha trucs per poder representar informació de diversos tipus.

Per això, dins del fitxer hi haurà contingut que no s'interpretarà com a text sense més ni més, sinó que dins del fitxer hi haurà **text especial**, marcat d'una manera que permeti donar-li un altre significat. És el que es coneix com a **metadades**: dades que serveixen per descriure altres dades. En el cas dels fitxers de text, són paraules marcades de forma especial que serveix per descriure el text al qual acompanya.

Des de fa molts anys hi ha tres camps en què aquesta idea ha funcionat molt bé: a les bases de dades, als processadors de text i, especialment, a les pàgines web. L'èxit d'Internet ha permès impulsar aquesta tecnologia a molts altres camps.

Cal recordar un problema fonamental amb el text: en ser format tan universal, i ser-ne el contingut tan accessible, és perillós com a font per emmagatzemar dades confidencials, ja que queda fàcilment exposat a qualsevol persona.

4. Llenguatges de marques

Introducció històrica

Aparició dels llenguatges de marques

Com s'ha comentat al punt anterior, el problema de l'exportació de dades ha posat en dubte els fitxers binaris com a font per exportar i importar informació.

En canvi, sembla que els fitxers de text tenen menys problemes. Per això, s'ha intentat que els fitxers de **text pla** (arxius que només contenen text i no altres dades binàries) pogueren servir per emmagatzemar altres dades com ara detalls sobre el format del propi text o altres indicacions.

Els **processadors de text** van ser el primer programari a trobar-se amb aquest dilema. Com que són programes que serveixen per escriure text semblava que el més lògic era que les seves dades s'emmagatzemessin com a tal. Però necessiten guardar dades referides al format del text, mida de la pàgina, color, tipografia, marges, etc. La solució clàssica ha estat guardar la informació de forma binària, cosa que provoca els ja comentats problemes.

Alguns processadors de text van optar per desar tota la informació com a text, fent que les indicacions de format no s'emmagatzemin de manera binària sinó textual. Aquestes indicacions són caràcters marcats de manera especial perquè així un programa adequat pugui traduir aquests caràcters no com a text sinó com a operacions que finalment produiran mostrar el text del document de forma adequada.

La idea del marcat procedeix de l'anglès *marking up* terme amb què es referien a la tècnica de marcar manuscrits amb llapis de color per fer anotacions com ara la tipografia a emprar a les impremtes. Aquest mateix terme s'ha utilitzat per als documents de text que contenen ordres o anotacions.

Les possibles anotacions o indicacions incloses en els documents de text han donat lloc a llenguatges (entenent que en realitat són formats de document i no llenguatges en el sentit dels llenguatges de programació d'aplicacions) anomenats llenguatges de marques, llenguatges de marcatge o llenguatges de etiquetes.

Charles Goldfarb i el GML

Es considera Charles Goldfarb com el pare dels llenguatges de marques. La raó per a aquesta consideració és, precisament, la seva ajuda a la creació del llenguatge GML (Generalized Markup Language).

Golfarb era un investigador d'IBM que va proposar idees perquè els documents de text que incloguessin la possibilitat de marcar-ne el format. Al final va ajudar a realitzar el llenguatge GML d'IBM el qual va posar els fonaments del futur SGML (pare d'HTML i XML) ideat pel mateix Goldfarb i pare de la majoria de llenguatges de marques actuals.

Exemple de codi GML

```
:h0.El reino de los animales
:h1.Mamíferos
:p.Los mamíferos (:hp1.Mammalia:ehp1.) son una clase de
vertebrados :hp2.amniotas homeotermos:ehp2. que poseen
glándulas mamarias productoras de leche con las que
alimentan a las crías
:h1.Aves
:p. Las aves son animales vertebrados, de sangre
caliente, que caminan, saltan o se mantienen solo sobre
las extremidades posteriores
```

Aquest codi renderitzat per un programari que interprete aquest codi obtindria com a resultat:

El reino de los animales

Mamíferos

Los mamíferos (**Mammalia**) son una clase de vertebrados ***amniotas homeotermos***. que poseen glándulas mamarias productoras de leche con las que alimentan a las crías

Aves

Las aves son animales vertebrados, de sangre caliente, que caminan, saltan o se mantienen solo sobre las extremidades posteriores

Ilustración 1-1. Renderizado de código GML (al estilo de HTML)

La idea és que els elements arcats amb símbols “:” i “.” delimiten marques de format. Així: h1. significa títol principal i :p. significa paràgraf.

TeX i LaTeX

A la dècada dels 70 Donald Knuth (un dels enginyers informàtics més importants de la història, pare de l'anàlisi d'algorismes i premi Turing 1974) va crear el llenguatge TeX per produir documents científics utilitzant una tipografia i capacitats que fossin iguals a qualsevol ordinador, assegurant a més una gran qualitat en els resultats.

Per això va donar suport a TeX amb tipografia especial (fonts **Modern Computer**) i un llenguatge de definició de tipus (**METAFONT**). TeX ha tingut cert èxit a la comunitat científica gràcies a les seves 300 ordres que permeten crear documents amb tipus de gran qualitat. Requereix de programari capaç de convertir el fitxer TeX a un format d'impressió.

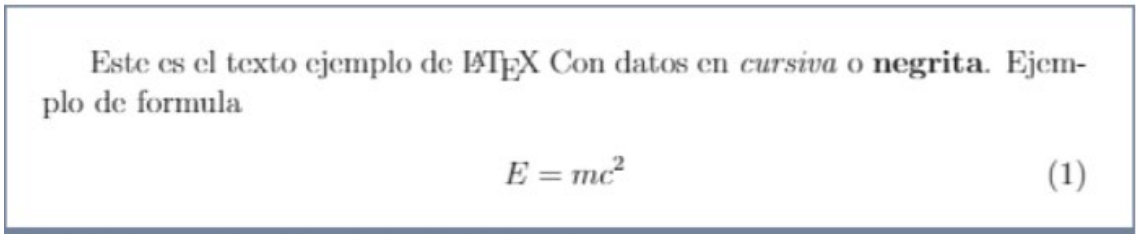
L'èxit de TeX va produir nombrosos derivats dels quals el més popular és **LaTeX**. LaTeX va ser definit el 1984 per Leslie Lamport (premi Turing 2003), encara que després ha estat nombroses vegades revisat. En utilitzar ordres de TeX i tota la seva estructura tipogràfica, va adquirir ràpidament notorietat i segueix sent utilitzat per produir documents amb expressions

científiques, de gran qualitat. La idea és que els científics se centrin en el contingut i no en la presentació.

Exemple de codi LaTeX:

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{\Ejemplo}
\begin{document}
Este es el texto ejemplo de \LaTeX{}
Con datos en \emph{cursiva} o \textbf{negrita}.
Ejemplo de f\'ormula
\begin{align}
E &= mc^2
\end{align}
\end{document}
```

Que amb un traductor donaria lloc al resultat:



Este es el texto ejemplo de LaTeX Con datos en *cursiva* o **negrita**. Ejemplo de formula

$$E = mc^2 \quad (1)$$

Ilustración 1-2. Renderizado de código Latex

RTF

RTF és l'acrònim **de Rich Text Format** (Format de Text Enriquit) un llenguatge ideat per Richard Brodie, Charles Simonyi i David Luebbert (membres de l'equip de desenvolupament de Microsoft Word) el 1987 per produir documents de text que incloguin anotacions del format. És un format propietat de **Microsoft** però reconegut per la majoria d'aplicacions de procés de text.

Actualment és un format acceptat per a documents de text que continguin informació sobre l'estil del text. Es fa servir molt com a format d'intercanvi entre diferents processadors per la seva potència. El processador de text **Word Pad** incorporat dins del sistema operatiu Windows l'utilitza com a format nadiu.

Codifica el text usant 8 bits, per a caràcters fora de l'ASCII requereix seqüències d'escapament el que, pràcticament, el fa inviable com a format llegible de text a la majoria de llengües del planeta. En les últimes versions de RTF ja sí que s'ofereix un suport més gran a Unicode.

El seu èxit procedeix que les indicacions de format són potents i són més llegibles per les persones que el format nadiu dels processadors de textos, encara que és, com a llenguatge de marcatge, un dels més críptics.

```
{\rtf\ansicpg1252\deff0\deflang3082
{\fonttbl
{\f0\fcharset0\froman Times New Roman}
{\f1\fcharset0\fswiss Arial Black}
}
{\pard \f1\fs48
El reino de los animales
\par}
{\pard \f1\fs40
Mamíferos
\par}
{\pard \f0\fs25
Los mamíferos ({\b Mammalia}) son una clase de
vertebrados {\i amniotas homeotermos} que poseen
glándulas mamarias productoras de leche con las que
alimentan a las crías
\par}
{\pard \f1\fs40
Aves
\par}
{\pard \f0\fs25
Las aves son animales vertebrados, de sangre caliente,
que caminan, saltan o se mantienen solo sobre las
extremidades posteriores
\par}
}
```

Produeix el resultat:

Ilustración 1-3. Renderizado de código GML (al estilo de HTML)

SGML

Es tracta d'una millora molt notable del llenguatge de **GML** que estandarditzava el llenguatge de marcatge i que va ser definida finalment per **ISO** com a estàndard mundial en documents de text amb etiquetes de marcatge. El seu responsable va ser **Charles Goldfarb**.

La seva importància rau que és el **pare del llenguatge XML** i la base sobre la qual se sosté el llenguatge **HTML**, dos dels llenguatges de marques més populars de la història.

A SGML els elements que contenen indicacions per al text es col·loquen entre símbols < i >. Les etiquetes es tanquen amb el signe /. És a dir, les regles fonamentals dels llenguatges d'etiquetes actuals ja les havia definit SGML.

En realitat (com XML) no és un llenguatge amb unes etiquetes concretes, sinó que és un llenguatge que serveix per definir llenguatges. Entre els llenguatges definits mitjançant SGML, sens dubte HTML és el més popular.

Exemple:

```
<articulo>
        <titulo1>El reino de los animales</titulo1>
        <titulo2>Mamíferos</titulo2>
        <normal>Los mamíferos (<negrita>Mammalia</negrita>) son una clase de vertebrados <cursiva>amniotas homeotermos</cursiva>.
        <titulo2>Aves</titulo2>
        <normal>Las aves son animales vertebrados, de sangre caliente, que caminan, saltan o se mantienen solo sobre las extremid
</articulo>
```

Com veurem més endavant, aquest document és molt semblant a un document realitzat amb XML, de fet XML és un subconjunt de SGML més restrictiu (és un llenguatge que té normes més estrictes).

SGML necessitarà definir com s'han de mostrar els elements titulo1, titulo2, etc. Ja que són noms d'elements que caldrà definir. Aquesta és la prova que és un llenguatge per definir tipus de document.

SGML va aportar les etiquetes tal com les coneixem actualment gràcies a l'èxit de HTML.

PostScript

És un llenguatge de descripció de pàgines. De fet és el més popular per a aquest fi, sent el llenguatge més utilitzat pels sistemes d'impressió d'alta gamma.

Permet crear documents en què es donen indicacions potentíssimes sobre com mostrar informació al dispositiu final. Es va iniciar el seu desenvolupament el 1976 per John Warnock i dos anys més tard es va continuar amb l'empresa Xerox, fins que el 1985 el mateix Warnock funda Adobe Systems i des d'aquesta empresa es continua el seu desenvolupament.

És en realitat tot un llenguatge de programació que indica la manera com s'ha de mostrar la informació que pot incloure text i el tipus de lletra del mateix, píxels individuals i formes vectorials (línies, corbes). Les possibilitats són molt àmplies.

Exemple:

```
%colocar el cursor
100 100 moveto
%dibuja cuadrado
100 200 lineto
200 200 lineto
200 100 lineto
100 100 lineto
%relleno
stroke
```

HTML

Tim Bernes Lee va utilitzar SGML per definir un nou llenguatge d'etiquetes que va anomenar Hypertext Markup Language (llenguatge de marcatge d'hipertext) per crear documents transportables a través d'Internet on fos possible l'hipertext; és a dir, la possibilitat que determinades paraules marcades de manera especial permetessin obrir un document relacionat amb elles.

Tot i trigar a ser acceptat, HTML va ser un èxit rotund i la causa indubtable de lèxit dInternet. Avui dia gairebé tot a Internet es veu a través de documents HTML, que popularment s'anomenen pàgines web.

Inicialment aquests documents es veien amb ajuda d'intèrprets de text (com per exemple el Lynx d'Unix) que simplement pintaven el text i remarcaven l'hipertext. Després el programari es va millorar i van aparèixer navegadors amb capacitat més gràfica per mostrar formats més avançats i visuals.

Exemple (usant el mateix contingut dels exemples anteriors):

```
<!DOCTYPE html>
<html lang="es">
<head>
        <meta charset="UTF-8">
        <title>Document</title>
</head>
<body>
        <h1>El reino de los animales</h1>
        <h2>Mamíferos</h2>
        <p>Los mamíferos (<strong>Mammalia</strong>) son una clase de vertebrados <em>amniotas homeotermos</em>
        <h2>Aves</h2>
        <p>Las aves son animales vertebrados, de sangre caliente, que caminan, saltan o se mantienen solo sobre
</body>
</html>
```

XML

Es tracta d'un subconjunt de SGML ideat per millorar el propi SGML i definir llenguatges de marcatge amb sintaxi més estricta, però més comprensible.

Ha estat enormement popular des de finals dels 90 i ha aconseguit incorporar nombrosos llenguatges al seu voltant per aconseguir documents molt dinàmics i amb una gran capacitat de format. És un dels formats de documents més populars per a exportació i importació de dades.

Actualment està sent sobrepassat a la majoria dels seus usos per JSON

```
<?xml version="1.0" encoding="UTF-8"?>
<nombre>Jorge</nombre>
<apellido1>Sánchez</apellido1>
<dirección>
  <calle>C/ Falsa nº 0</calle>
  <localidad>Palencia</localidad>
  <código_Postal>34001</código_Postal>
  <pais>España</pais>
</direccion>
<teléfonos>
  <teléfono tipo="fijo">999 999 999</teléfono>
  <teléfono tipo="móvil">666 666 666</teléfono>
</telefono>
```

Markdown

Es tracta d'un format de marcatge simple que permet crear documents senzills i convertir-los en documents HTML.

Va ser creat per John Gruber amb l'ajuda d'Aaron Shwartz. La pretensió daquest llenguatge és definir unes normes molt senzilles per crear documents semblants als que es creen mitjançant el llenguatge HTML.

Ha tingut un èxit molt notable, especialment des que va ser adoptat per llocs tan populars com GitHub, Reddit o StackExchange perquè els usuaris publiquessin contingut amb format.

Exemple de text amb format Markdown:

```
# El reino de los animales
## Mamíferos
Los mamíferos (**Mammalia**) son una clase de vertebrados *amniotas homeotermos*. que poseen glándulas mamarias productoras de leche con las que alimentan a las crías
## Aves
Las aves son animales vertebrados, de sangre caliente, que caminan, saltan o se mantienen solo sobre las extremidades posteriores
```

Aquest codi genera el mateix resultat que el codi mostrat a l'apartat d'HTML, però se'n percep clarament la senzillesa.

JSON

Abreviatura de JavaScript Object Notation, Es tracta d'una notació de dades procedent del llenguatge JavaScript estàndard (concretament a la versió ECMAScript de 1999). L'any 2002 se li

donava suport des de molts dels navegadors i la seva fama ha estat tan gran que ara s'ha convertit en una notació independent de JavaScript que competeix clarament amb XML en funcionalitat.

Les raons del seu èxit es deuen a la seva versatilitat, ja que permeten definir dades complexes, com ara arrays o codi de funcions, elements pertanyents al món de la programació d'aplicacions. L'èxit de JavaScript juntament amb la versatilitat comentada, l'han convertit en el llenguatge de marcatge més popular per emmagatzemar dades.

A JSON, el text es divideix en dades i metadades. De manera que el símbol dels dos punts separa la metadada de la dada. D'altra banda, els símbols de clau i claudàtor permeten agrupar de diverses formes les dades.

Exemple de codi JSON:

```
{
    "nombre": "Jorge",
    "apellido1": "Sánchez",
    "dirección": {
        "calle": "C/ Falsa nº 0",
        "localidad": "Palencia",
        "código Postal": 34001,
        "país":"España"
    },
    "teléfonos": [
        {
            "tipo": "fijo",
            "número": "999 999 999"
        },
        {
            "tipo": "móvil",
            "number": "666 666 666"
        }
    ]
}
```

Tipus de llenguatges de marques

- **Orientats a la presentació**. Les metadades permeten indicar el format en què s'ha de presentar el text. És el cas de **RTF**, en què les seves etiquetes especifiquen tipus de lletra, mides de pàgina, colors, etc. Les primeres versions d'**HTML** també es consideren així, ja que incloïen etiquetes com a font mitjançant la qual s'especificava el format de font.

- **Orientats a la descripció**. Les marques especials permeten donar significat al text però no indiquen com s'ha de presentar en pantalla el text. Seria el cas de **XML** (o de **SGML**), **JSON**, **Markdown** i de les versions actuals de **HTML**. En aquests llenguatges simplement

s'hi indica el significat del contingut: si el text és un títol, un paràgraf normal, un peu d'il·lustració, una adreça postal etc.

- **Orientats a procediments**. Es tracta de documents en què el text marcat s'interpreta com a ordres a seguir, i així l'arxiu en realitat conté instruccions a realitzar amb el text (girar-lo, convertir-lo en una fórmula, fer una suma, etc.). És el cas de **LaTeX** o **PostScript**.

5. Webgrafia

http://jorgesanchez.net/manuales/html/introduccion-lenguajes-de-marcas.html