1.1 Codificare le parole

1.2 Un nuovo, grande problema

Usiamo tutti il computer per “gestire” testi. Ma cosa vuol dire “gestire”?

Naturalmente, leggere e scrivere, ma anche trasmettere, archiviare e fare copie: tutte cose che si fanno da millenni. Ma l’uso del computer, se apre la strada a straordinarie potenzialità, comporta una grande, inaspettata complessità.

Dall’inizio della Storia e fino a poco tempo fa, la scrittura prevedeva un supporto fisico: argilla, pietra, pelle, stoffa, papiro e, finalmente, carta. Erano proprio i supporti la fonte principale dei problemi: per trasmettere un testo bisognava trasferire il supporto.

E per archiviarlo servivano ambienti appositi: archivi (appunto) o biblioteche. E poi c’era il grande problema delle copie: fare una copia equivaleva a riscrivere il testo, con gran dispendio di energie e rischio di errori. Le cose sono migliorate con la fotocopiatrice, ma anche qui, la copia è sempre diversa dall’originale.

Ma da quando è arrivato il computer, tutto è cambiato. Archiviazione immateriale, trasmissione istantanea, copie che non sono copie ma nuovi originali…

Tutto risolto, quindi?

Non proprio, perché è nato un nuovo, grande problema…

1.3 Come scrive un computer?

Sappiamo tutti che i computer conoscono solo due segni: zero e uno.

La loro memoria è composta di celle, ciascuna delle quali contiene una singola informazione binaria. O zero o uno. Contiene, quindi, un "bit", da "binary" e "digit", cifra binaria. Per i primissimi computer, nati per macinare numeri, tanto bastava. E il codice binario, con un minimo di elaborazione, era usato anche per scrivere programmi.

Piano piano, però, qualcuno pensò di programmare descrivendo i comandi con parole come "PRINT", "GOTO", "IF". E, sulla loro scia, altri chiedevano di lavorare con le parole per scrivere lettere, articoli e racconti. E nel frattempo, ai computer si collegavano rudimentali tastiere e telescriventi (proprio telescriventi, non monitor!).

Da qui il problema: come rappresentare un carattere nella memoria del computer?

È evidente che un bit non basta, ce ne vogliono tanti.

Sì, ma quanti?

E poi, con quale criterio associare a ogni gruppo di bit una lettera?

Ogni produttore di computer decideva per conto suo. Così, nel 1969, quando Arpanet, l'antenata di Internet, mise in rete i primi quattro computer, si scoprì che le codifiche dei testi avvenivano in quattro modi diversi.

1.4 Il codice ASCII

Per far parlare tra loro i computer di Arpanet si decise di utilizzare un nuovo standard, presentato per la prima volta in quegli anni: la codifica ASCII (American Standard Code for Information Interchange). Il codice ASCII usava sette bit per rappresentare 128 caratteri diversi, infatti una sequenza di sette bit può contenere i numeri da 0 a 127.

Ma 128 caratteri non sono troppi per le 26 lettere della lingua inglese?

No, perché, ci sono maiuscole e minuscole, le dieci cifre, i segni di interpunzione, gli operatori matematici e molti altri simboli, dalla & commerciale al dollaro, dalla percentuale alla chiocciola. Sono in tutto 95 caratteri "stampabili", quelli che si usano per scrivere. E poi ci sono 32 caratteri "speciali", che non sono simboli, ma comandi per una telescrivente: il codice 10 significa "vai a capo", il 9 è un tabulatore e così via.

Questo sistema permise finalmente ai computer di comunicare tra loro, scambiarsi messaggi di posta elettronica e, in generale, di elaborare testi. In più la codifica ASCII si sposava bene con l'unità di memoria che andava affermandosi: il byte, composto da un insieme di otto bit.

A dire il vero, sette bit in un byte ci stanno piuttosto larghi.

C'è anche un bit in più…

1.5 L'ottavo bit e i caratteri "particolari"

Per molti il codice ASCII era (ed è ancora) la soluzione perfetta. Ma solo per la lingua inglese! Perché in italiano, per esempio, ci sono le vocali accentate, minuscole maiuscole. E caratteri particolari, tutti diversi tra loro, si trovano in spagnolo, francese, tedesco, svedese. Praticamente in ogni lingua europea. E che dire del greco?

Emerse molto presto che, se l'uso del computer doveva far comunicare il mondo, di caratteri stampabili ne servivano altri. Fu così che qualcuno si ricordò dell'ottavo bit che compone il byte. Perché un bit in più consentiva di arrivare a un totale di 256 caratteri. Così si aggiunse alla primitiva codifica una "seconda pagina".

Ma, come accade spesso nella nostra storia, risolto un problema se ne apre un altro…

1.6 Le seconde pagine

Per le lingue europee, almeno quelle dell'Europa occidentale, altri 128 caratteri possono bastare. Ma per quelle dell'Europa orientale ne servono altri. E altri ancora per il greco e i caratteri cirillici. Per evitare una nuova Babele, intervenne l'ISO, International Standard Organization, l'ente non governativo che emana norme tecniche, adottate in tutto il mondo, su qualunque aspetto della vita comune, dalla costruzione di manufatti alla qualità delle merci.

L'ISO produsse una serie di norme anche sui caratteri ASCII, lasciando i primi 128 (la "prima pagina") uguale per tutti e la "seconda pagina" variabile in funzione della lingua. Nacquero così quindici estensioni, tra cui:

- ISO 8859-1, chiamata "Latino 1", per l'Europa Occidentale;
- ISO 8859-2, "Latino 2", per l'Europa Orientale;
- ISO 8859-5, "Latino/Cirillico";
- ISO 8859-6, "Latino/Arabo"; e
- ISO 8859-7, "Latino/Greco".

1.7 Usare l'ASCII esteso

Usando l'ASCII esteso, basta scrivere un testo adottando l'estensione adatta alla propria lingua e fare in modo che chi riceve il messaggio usi la stessa estensione per decodificarlo. Così, in teoria, si può comunicare in molte lingue.

In pratica, se uso il "Latino 1" per scrivere una mail a un amico americano e voglio avere la

certezza che venga visualizzata correttamente, devo far precedere il testo da una "dichiarazione" come questa…

Praticamente, avverto il computer ricevente che sto usando proprio "Latino 1". Altrimenti lo interpreterà usando la sua estensione di default, cioè quella della lingua in cui è stato impostato. E tutte le volte che scrivo "perché", il mio amico leggerà per esempio così…

È questo il motivo per cui ogni tanto leggiamo pagine web piene di segni incomprensibili!

1.8 Quanti caratteri esistono?

Nel mondo esistono dalle 6.000 alle 7.000 lingue. Ciascuna lingua usa un alfabeto, cioè un repertorio di caratteri, che possono essere molto diversi: basti pensare all'arabo, al coreano, al cinese. Per fortuna ci sono lingue che condividono lo stesso alfabeto, ma altre, come il serbo-croato, ne possono usare due (latino e cirillico).

Se contare gli alfabeti è difficile, per il numero complessivo di caratteri una stima c'è: sono più di 165.000!

Come possiamo rappresentarli tutti senza problemi di decodifica? La prima idea che viene in mente è usare più di un byte per ciascun carattere. Se un byte contiene 256 valori diversi (da 0 a 255), con due arriviamo a 65.366 (256 per 256). Non basta ancora: ne servirebbe uno in più. Con tre byte per un singolo carattere abbiamo più di 16 milioni di valori diversi. È molto più di quello che ci serve!

Sarebbe una buona idea se non fosse per l'enorme mole di dati già codificati in ASCII (compresi i programmi per computer). E per lo spreco di memoria che si avrebbe usando tre byte quando, quasi sempre, ne basta uno. Ci vuole un'altra idea. E questa idea si chiama Unicode…

1.9 Come funziona Unicode

La codifica Unicode appare come un colpo di genio, che colpisce per la sua semplicità concettuale, anche se sul piano tecnico serve qualche artificio per renderla applicabile. In Unicode a ogni possibile carattere è associato un numero chiamato "code point": alla "A" maiuscola corrisponde il 65, alla "B" maiuscola il 66 e al simbolo dell'Euro il code point 8.364.

Non c'è limite: in Unicode sono codificati gli alfabeti di tutte le lingue conosciute, compresa quella degli Indiani (d'America) Osage, il Tangut, parlato in Tibet fino a qualche secolo fa, e l'Etrusco. E non si tratta solo di alfabeti: tra gli oltre 140.000 code point della versione 11.0 di Unicode, c'è spazio per decine di "emoticon", le faccine che usiamo nelle nostre mail!

Ma definire l'elenco di code point è solo il primo passo, perché restano due problemi da risolvere.

Il primo è quello dei testi (e dei programmi) scritti prima della nascita di Unicode, che non

possono andare perduti. La soluzione è stata quella di usare per i primi 128 code point il codice ASCII originale e di inserire anche le “seconde pagine” più diffuse.

Molto più complesso il secondo problema: come evitare di usare una quantità eccessiva di memoria…

1.10 Dal code point al code value

Per trasformare senza sprechi ciascun carattere Unicode (espresso da un code point, cioè da un numero) in una sequenza di byte, chiamata “code value”, nella memoria di un computer ci sono diversi schemi, detti UTF, Unicode Transformation Format. Il più diffuso, noto come UTF-8, adotta un metodo efficace e raffinato: usare code value di lunghezza variabile!

In pratica funziona così:

- Per i caratteri più diffusi, compresi nella vecchia codifica ASCII, basta un byte.
- Per gli altri, ce ne vogliono due, tre o anche quattro, come accade per il cinese e il giapponese.

Naturalmente, per evitare ambiguità, nel codice c’è il modo di segnalare al computer di quanti byte è composto il carattere in questione.

Ma non è tutto!

1.11 Script e categorie

All’interno di Unicode la maggior parte dei code point sono raggruppati in “script”, che grosso modo, corrispondono agli alfabeti, fatti di lettere e altri segni. Alcuni script, come il Latin vengono usati da decine o centinaia di lingue diverse. Altri come il Khmer solo da una. Inoltre, Unicode assegna a ciascun code point delle proprietà che definiscono, per esempio, di che tipo di carattere si tratta: una lettera, una cifra, un segno di interpunzione, e così via.

Alcune proprietà sono comuni a tutti i code point che fanno parte di uno script. Tra queste c’è il modo in cui vanno rappresentati a video o stampati. L’arabo e l’antico aramaico, per esempio, si leggono da destra a sinistra!

È evidente che ci vuole una grande capacità di elaborazione per gestire ogni carattere di un testo: capire di quanti byte è composto, leggere il code value e da quello risalire al code point e poi allo script di riferimento e alle categorie. Ma è un lavoro che i computer fanno benissimo!

1.12 Unicode standard aperto e "democratico"

Da alcuni anni, Unicode è lo standard di riferimento per la codifica dei testi. Tutti i browser e tutti i sistemi operativi lo hanno ormai pienamente adottato, per i suoi enormi vantaggi:

- Nel web non ci sono più pagine rese incomprensibili da problemi di codifica.
- Si possono usare nello stesso documento alfabeti diversi. Come sarebbe possibile, altrimenti, scrivere in italiano un manuale di cinese?
- Tutte le lingue del mondo, comprese quelle estinte o parlate da esigue minoranze, vengono rappresentate con lo stesso diritto di cittadinanza.

E poi, la stessa organizzazione di Unicode è “democratica”. Perché lo standard è gestito dall’Unicode Consortium, un’organizzazione non governativa e non profit, in stretta collaborazione con l’International Standard Organization e il World Wide Web Consortium. E perché la configurazione dei singoli script è delegata a comitati locali, formati da persone che quell’alfabeto lo “vivono” quotidianamente.

È per questo che Unicode è uno standard davvero universale, aperto e inclusivo.