La Soluzione Finale: UNICODE

- Chiaramente, l'aggiunta di codifiche su codifiche, ciascuna di al più 256 caratteri (8 bit), non può andare lontano...
- Una soluzione radicale è quella proposta dallo **standard UNICODE**:
  - abbandonare gli standard 8 bit
  - codificare ogni carattere con **32 bit**
    - oltre **4 miliardi di caratteri** diversi codificabili!

UNICODE

- UNICODE assegna ad ogni carattere un proprio codice numerico
  - indipendentemente dalla piattaforma
  - indipendentemente dal programma
  - indipendentemente dal linguaggio
- Con oltre 4 miliardi di codici assegnabili, si può “scialare”
  - finezze tipografiche
  - lingue morte o rarissime

UNICODE

- Definito inizialmente nel 1991, UNICODE ha avuto varie versioni
- L'ultima, UNICODE 5.0, è del Luglio 2006
- Ogni versione “include” le precedenti, e solitamente aggiunge altri caratteri
- UNICODE definisce inoltre:
  - mappature (mappings)
  - codifiche (encodings)
  - regole di ordinamento (collation)

UNICODE

- UNICODE supporta praticamente tutte le lingue e i sistemi di scrittura esistenti al mondo (e alcuni non esistenti), fra cui ovviamente le lingue occidentali e:

Arabic
Armenian
Bengali
Braille embossing patterns
Canadian Aboriginal Syllabics
Cherokee
Coptic
Cyrillic
Devanāgarī
Ethiopic
Georgian
Greek
Gujarati
Gurmukhi (Punjabi)
Han (Kanji, Hanja, Hanzi)
Hangul (Korean)
Hebrew
Hiragana and Katakana (Japanese)
International Phonetic Alphabet (IPA)
Khmer (Cambodian)
Kannada
Lao
Latin
Malayalam
Mongolian
Myanmar (Burmese)
Oriya
Syriac
Tamil
Telugu
Thai
Tibetan
Tifinagh
Yi
Zhuyin (Bopomofo)

UNICODE

- Sono previste poi:
  - alcune lingue morte (per la scrittura di testi accademici)
    - Cuneiforme (Assiro-Babilonese), Deseret, Lineare B (Cretese), Ogham, Etrusco, Fenicio, Runico, Shavian, Ugaritico, ...
  - alcune lingue fantastiche
    - Klingon, Ferengi (proposti ma ritirati)
    - Elfico, Tengwar e Cirth (tuttora in considerazione)
  - alcune altre notazioni comuni
    - simboli matematici
    - simboli musicali

Struttura di UNICODE

- Un **carattere** UNICODE è caratterizzato dal suo codice numerico, detto **code point**, solitamente rappresentato con 8 cifre esadecimali
  - Per esempio, fi (legatura di f e i) è rappresentato dal codice 0000FB01, mentre il simbolo “do doppio diesis strumentale” della notazione musicale greca antica (simile a una lambda maiuscola con una gambetta) è 0001D235
- È uso omettere gli “0” iniziali...

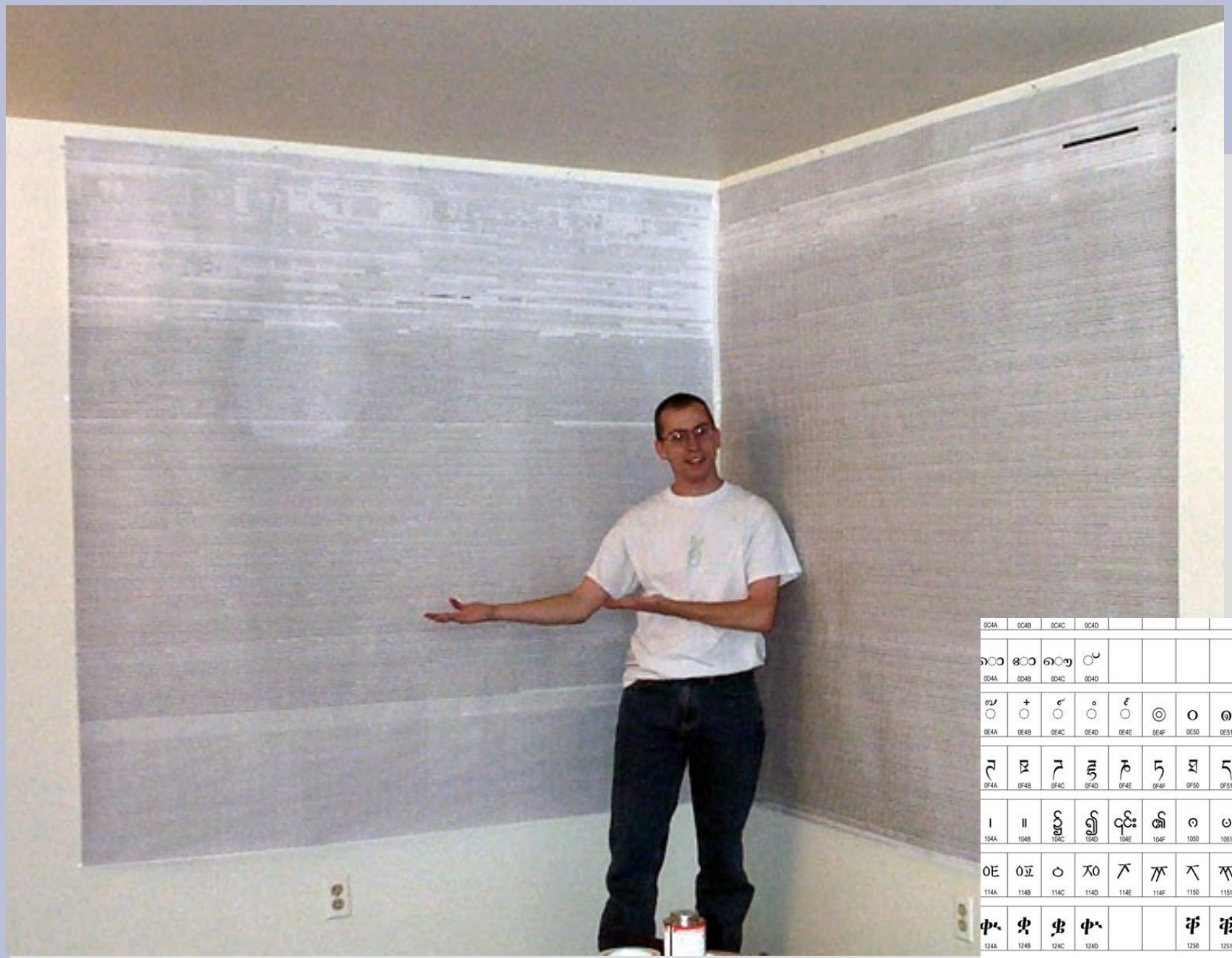
Struttura di UNICODE

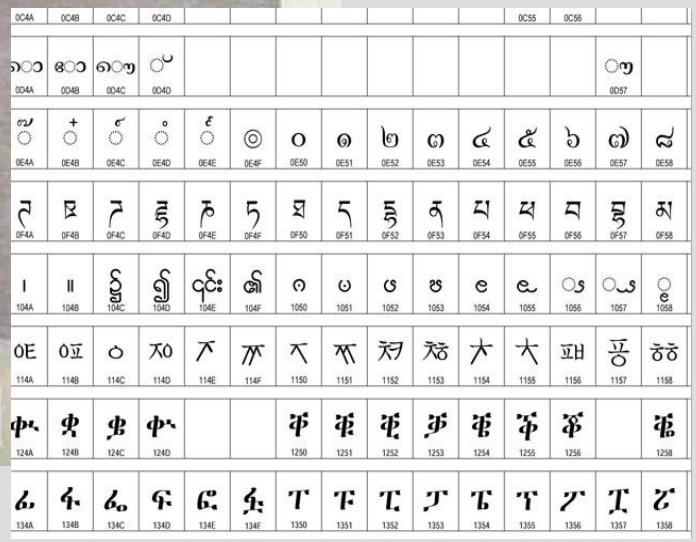
- I primi 256 code point, da 0 a FF, sono identici ai caratteri ISO-8859-1; ciò semplifica grandemente la conversione fra UNICODE e ISO-Latin-1 (e -15)
- UNICODE ha spazio per 16.777.216 “pagine” di 256 caratteri; molti standard precedenti sono inclusi tali e quali per comodità
- Moltissimi caratteri sono quindi ripetuti più volte in pagine diverse

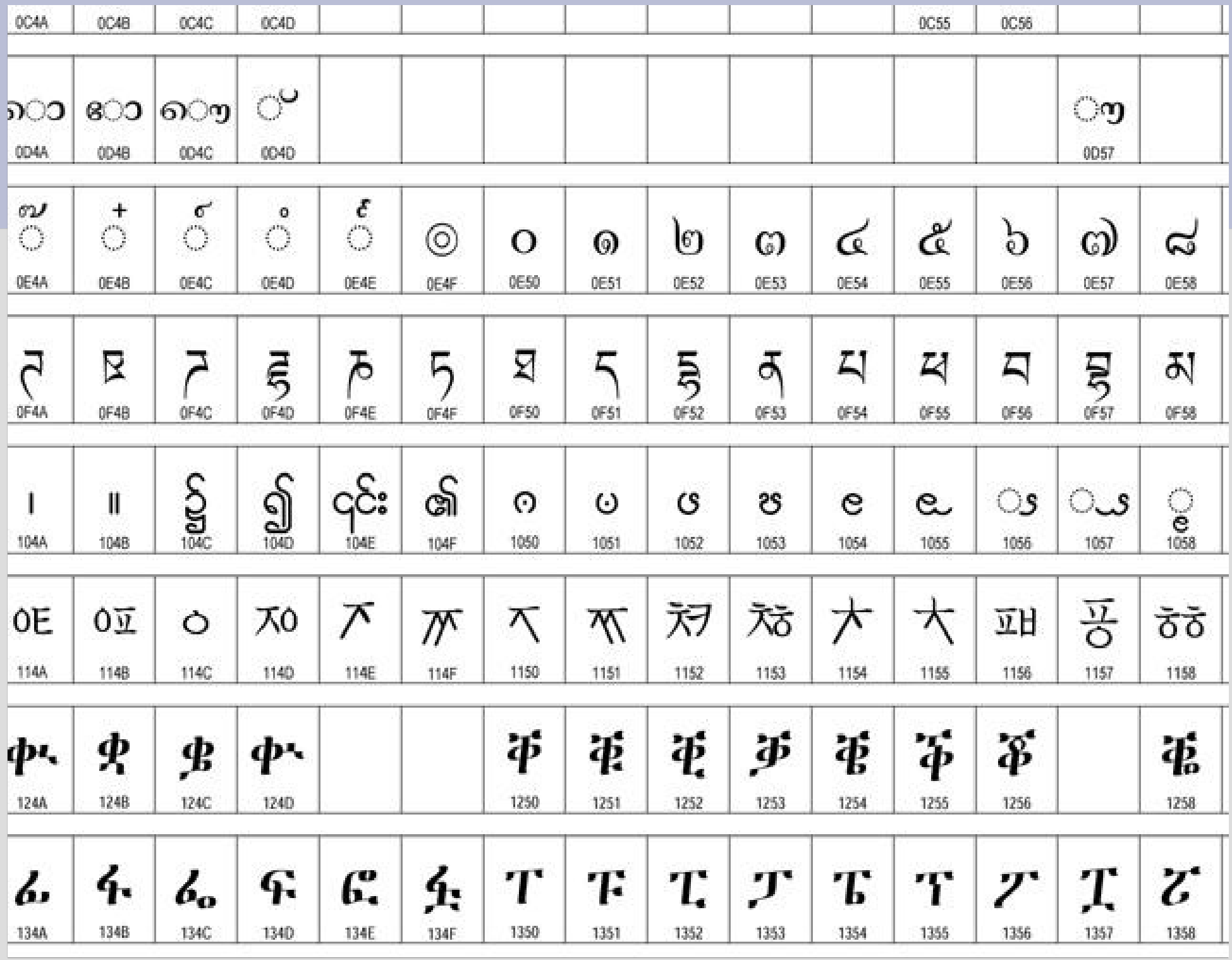
Struttura di UNICODE

- Similmente, benché UNICODE contempli la possibilità di “legare” caratteri (per esempio, lettere e accenti), molte versioni precombinate sono disponibili per maggiore comodità
- In alcune lingue (arabo, hindi, ecc.) le regole per le legature sono molto complesse: le versioni precombinate risparmiano al software di rendering decisioni penose

Dimensioni di UNICODE






0C4A 0C4B 0C4C 0C4D 0C55 0C56
0D4A 0D4B 0D4C 0D4D 0D57
0E4A 0E4B 0E4C 0E4D 0E4E 0E4F 0E50 0E51 0E52 0E53 0E54 0E55 0E56 0E57 0E58
0F4A 0F4B 0F4C 0F4D 0F4E 0F4F 0F50 0F51 0F52 0F53 0F54 0F55 0F56 0F57 0F58
104A 104B 104C 104D 104E 104F 1050 1051 1052 1053 1054 1055 1056 1057 1058
114A 114B 114C 114D 114E 114F 1150 1151 1152 1153 1154 1155 1156 1157 1158
124A 124B 124C 124D 1250 1251 1252 1253 1254 1255 1256 1258
134A 134B 134C 134D 134E 134F 1350 1351 1352 1353 1354 1355 1356 1357 1358

Applicazioni di UNICODE

- Con UNICODE, è possibile creare e gestire senza troppa pena documenti multilingue:
  - A, Δ, Й, ק ,م ,ม, あ , 叶 , 葉
- In particolare, tutti gli standard W3C (incluso HTML) supportano UNICODE; è quindi possibile usare qualunque carattere su una pagina web
  - riga precedente: A, Δ, Й, ק, م, ๗, あ, 叶, 葉
  - occorre naturalmente che il browser abbia accesso ai font corrispondenti...

Il problema della codifica

- Abbiamo visto che UNICODE può codificare 4.294.967.296 caratteri distinti
- Ogni carattere occupa 32 bit (contro gli 8 delle altre codifiche); i documenti richiedono quindi 4 volte lo spazio
- Ma la stragrande maggioranza (quasi totalità) dei documenti usa da 60 a 1000 caratteri, per cui basterebbero da 6 a 10 bit... quanto spazio sprecato!

Il problema della codifica

- Per ovviare a questo problema, e garantire maggiore compatibilità con S.O. e applicazioni che non sono in grado di gestire facilmente 32 bit per carattere, UNICODE definisce vari formati di codifica più o meno compatti
- UTF-7, UTF-8, CESU-8, UTF-16, UTF-32, UTF-EBCDIC, SCSU, UCS
- Vediamo alcuni dei più comuni...

UTF-8

- UTF-8 (8-bit UCS/Unicode Transformation Format) è una **codifica a lunghezza variabile** fra **una sequenza di valori a 8 bit** e **una sequenza di caratteri UNICODE**
- Creata da Ken Thompson and Rob Pike (due degli autori originali di UNIX)
- È una codifica diffusa, con molte proprietà utili

UTF-8

- I primi 128 caratteri di UNICODE (0-7F), equivalenti ai caratteri ASCII, sono codificati con il loro codice “naturale”
- Tutti gli altri caratteri sono codificati con due, tre o quattro valori a 8 bit (byte)
  - lo schema garantisce che in questi casi tutti i byte abbiano un valore superiore a 127, in modo che non ci sia confusione con i codici precedenti

UTF-8

- In dettaglio:

Range caratteri UNICODE	Code point (in binario)	Codifica UTF-8 (in binario)	Note
000000–00007F 128 caratteri	0zzzzzzz sette bit z	0zzzzzzz sette bit z	range equivalente all'ASCII; tutti i byte iniziano con un bit a 0
000080–0007FF 1920 caratteri	00000yyy yyzzzzzz tre bit y; due bit y, sei bit z	110yyyyy 10zzzzzz cinque bit y, sei bit z	il primo byte inizia con 110, il successivo con 10.
000800–00FFFF 63488 caratteri	xxxxyyyy yyzzzzzz quattro bit x, quattro bit y; due bit y, sei bit z	1110xxxx 10yyyyyy 10zzzzzz quattro bit x; sei bit y; sei bit z	il primo byte inizia con 1110, i successivi con 10.
010000–10FFFF 1048576 caratteri	000wwwxx xxxxyyyy yyzzzzzz tre bit w, due bit x; quattro bit x, quattro bit y; due bit y, sei bit z	11110www 10xxxxxx 10yyyyyy 10zzzzzz tre bit w; sei bit x; sei bit y; sei bit z	il primo byte inizia con 11110, i successivi con 10

- Proprietà interessante: il codice si auto-sincronizza
  - guardando i primi bit di un byte, so a che punto sono e come gestire i byte successivi

UTF-8: esempio di codifica

- Consideriamo il carattere א (aleph), la prima lettere dell'alfabeto ebraico
  - Il suo code point è 5D0 (000005D0), quindi ricade nel secondo caso della tabella precedente:

Code point	Code point (in bin)	UTF-8 (in bin)	UTF-8 (in hex)
00 00 05 D0	00000101 11010000	11010111 10010000	D7 90
	00000yyy yyzzzzzz	110yyyyy 10zzzzzz	

- Per la decodifica si applica in direzione opposta lo stesso meccanismo

Usi notevoli di UTF-8

- Nel linguaggio di programmazione Java (e derivati), le stringhe sono codificate con UTF-8; i programmi Java sono quindi in grado di gestire nativamente UNICODE
  - C'è qualche piccola differenza; per esempio il code point 0 non è codificato con 00000000 ma con 11000000 10000000
- I file system Macintosh, DVD, e alcuni su UNIX usano UTF-8 per i nomi dei file
- Gli standard relativi al Web e alla e-mail richiedono che un programma compatibile supporti *almeno* UTF-8 come standard di codifica
- Quasi tutti i programmi che trattano testi ASCII sono UTF-8 compatibili!

UTF-16

- UTF-16 (16-bit UCS/Unicode Transformation Format) è una **codifica a lunghezza variabile** fra **una sequenza di valori a 16 bit** e **una sequenza di caratteri UNICODE**
- I primi 65.536 valori di UTF-16 coincidono con i primi 65.536 caratteri di UNICODE, e con la codifica UCS-2

UTF-16

- In dettaglio:
  - caratteri fra 0000 e FFFF vengono rappresentati, su due byte, tali e quali
  - caratteri fra 10000 e 10FFFD (il limite superiore attuale di UNICODE) vengono scomposti e rappresentati con due parole a 16 bit:
    - La prima è data dall'OR fra D800 e i 10 bit alti del code point
    - La seconda è data dall'OR fra DC00 e i 10 bit bassi del code point

UTF-16

- In altre parole:
  - se il valore binario del code point è
    00000000 00000000 xxxxxxxx yyyyyyyy
  - il suo codice UTF-16 è
    xxxxxxxx yyyyyyyy
  - se invece il valore binario del code point è
    00000000 0000xxxx xxxxxxyy yyyyyyyy
  - il suo codice UTF-16 è
    110110xx xxxxxxxx 110111yy yyyyyyyy

UTF-16: esempio di codifica

code point	carattere	codifica UTF-16	glifo
122 (hex 7A)	z (alfabeto latino)	007A	z
27700 (hex 6C34)	acqua (ideogramma cinese)	6C34	水
119070 (hex 1D11E)	chiave di Sol (musica)	D834 DD1E	(simbolo della chiave di Sol)

- Anche in questo caso, per la decodifica si applica lo stesso meccanismo in direzione inversa
  - Si noti che però il codice non si auto-sincronizza (i byte intermedi di un codice su due parole potrebbero a loro volta contenere le sequenze speciali)

Usi notevoli di UTF-16

- Windows (NT, 2000, XP, CE) usa UTF-16 internamente per rappresentare le stringhe
- La macchine virtuali Java e .NET e il S.O. Symbian (quello dei telefonini) fanno altrettanto
- I motori grafici di Cocoa (Macintosh) e QT (Linux) usano UTF-16

UTF-16 e UCS-2

- UCS-2 (Universal Character Set – 2 bytes) è una **codifica a lunghezza fissa** fra **valori a 16 bit** e **i primi 65536 caratteri** di UNICODE
  - caratteri fra 0000 e FFFF vengono rappresentati, su due byte, tali e quali
  - tutti gli altri caratteri **non sono rappresentabili** – punto e basta!
- Codifica superata e resa obsoleta da UTF-16

Usi notevoli di UCS-2

- Le versioni di Windows NT sviluppate prima di Windows 2000 supportavano solo UCS-2, e potevano quindi gestire solo alcune decine di migliaia di caratteri
- Per tutti gli altri usi (es., Cinese) si usavano sistemi ad-hoc

UCS-4

- UCS-4 (Universal Character Set –4 bytes) è una **codifica a lunghezza fissa** fra **valori a 32 bit** e **tutti i caratteri** di UNICODE
  - Ogni carattere viene rappresentato dal suo code point, espresso su 4 byte
  - Si tratta in effetti di una “non-codifica” che lascia tutto come sta
  - Grande spreco di spazio!
- Noto anche come UTF-32

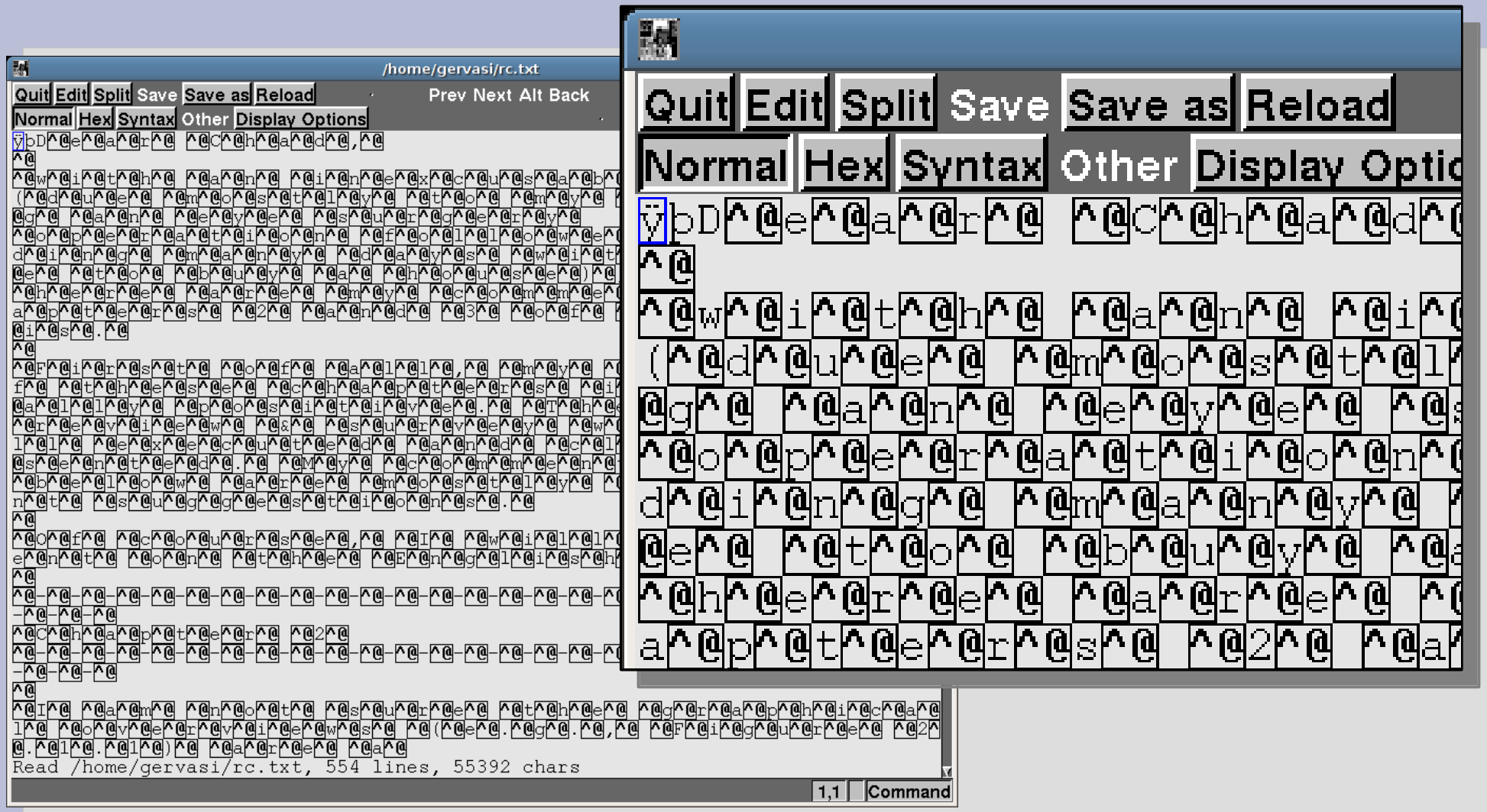
Alcune considerazioni pratiche

- Cosa accade quando la codifica non viene riconosciuta correttamente?
  - Se il testo contiene solo caratteri ASCII (codici 0-127) e la codifica è compatibile con l'ASCII (es. UTF-8, ISO-8859-1/15), si potrebbe non notare nulla
  - Ma se il testo contiene lettere accentate, simboli matematici, lingue straniere... cominciano i guai!
  - Alcuni caratteri appariranno “sbagliati”

Alcune considerazioni pratiche

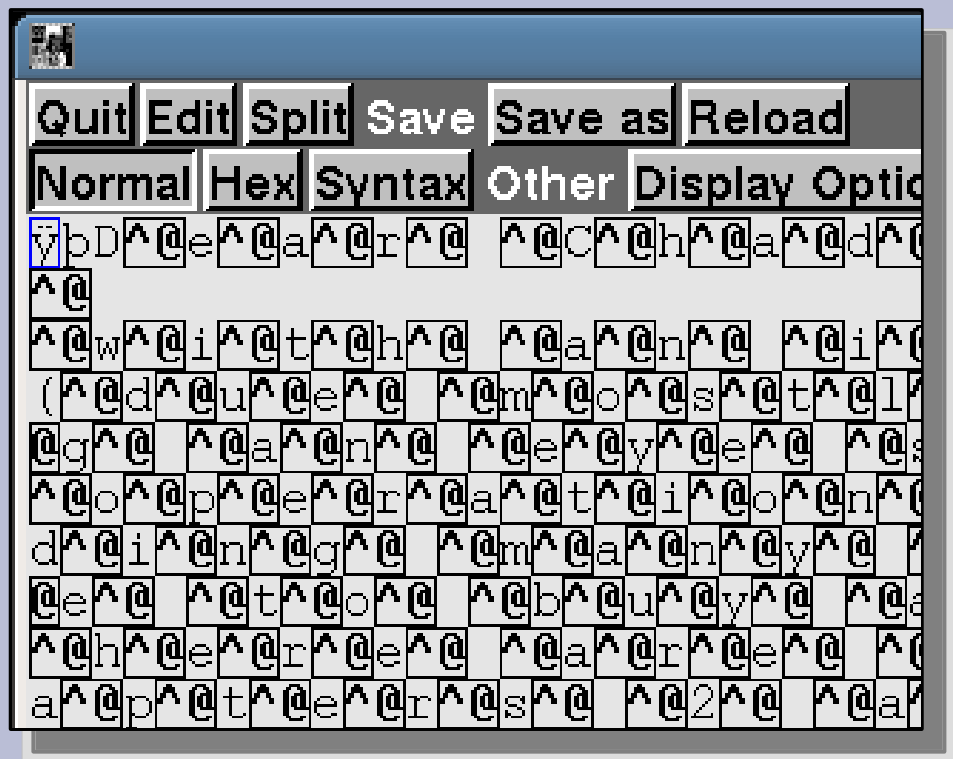
- Cosa accade quando la codifica non viene riconosciuta correttamente?
  - Se la codifica non è compatibile ASCII (come UTF-16, UCS-4), non c'è salvezza!
  - Anche il documento più semplice apparirà irrimediabilmente “rovinato”
  - In realtà, basterebbe aprirlo con un programma che sia in grado di gestire la codifica corretta

Un esempio



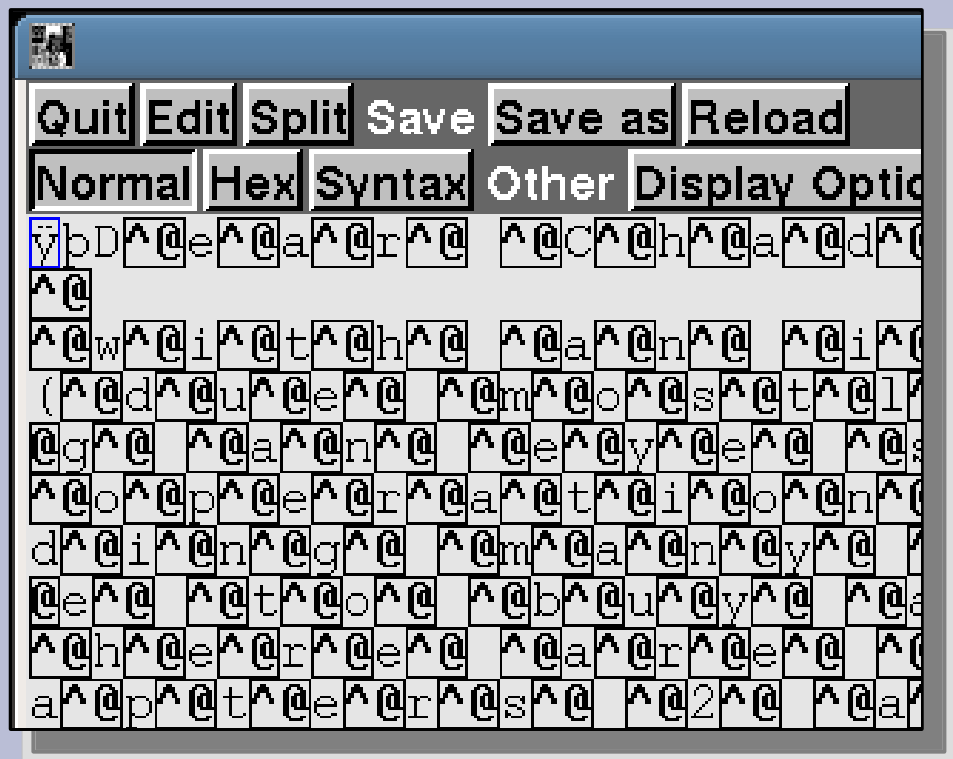
- Aita! Soccorso! Ch'accade?? Che fu??

Un esempio



- Notiamo che il documento contiene caratteri “normali”, intervallati da ^@
- Ma ^@ è il simbolo ASCII di NUL, ovvero un byte di valore 0...
- Sembra quindi che **ogni** carattere sia codificato in **due** byte anziché uno...

Un esempio

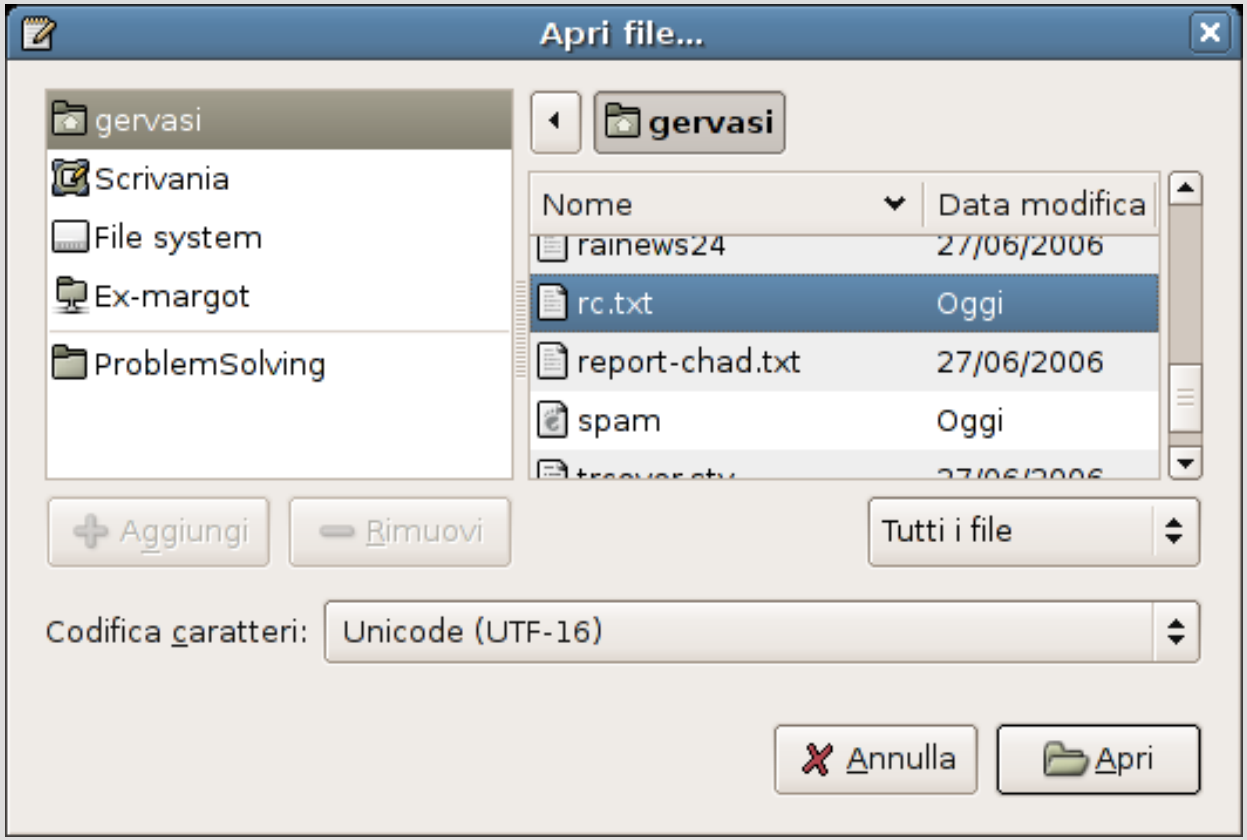


Terribile sospetto: sarà mica che il documento è codificato in UTF-16 ? O forse UCS-2?

- Notiamo che il documento contiene caratteri “normali”, intervallati da ^@
- Ma ^@ è il simbolo ASCII di NUL, ovvero un byte di valore 0...
- Sembra quindi che **ogni** carattere sia codificato in **due** byte anziché uno...

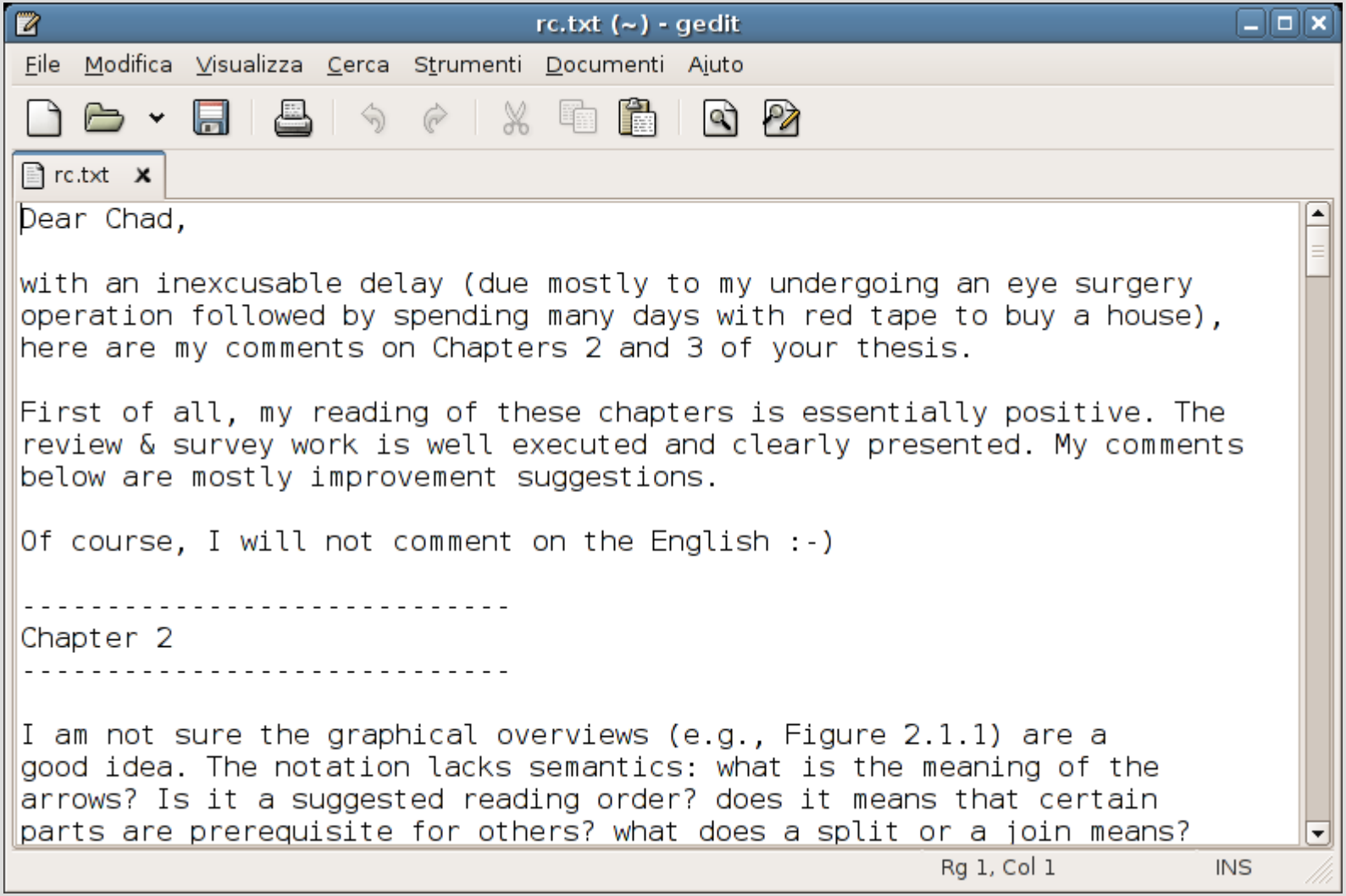
Un esempio

- Apriamo lo stesso file con un'applicazione che ci permette di scegliere la codifica da usare...



Un esempio

- Et voilà!



Esercizio

- Riuscite a riconoscere la codifica e il contenuto del testo descritto dalla sequenza di codici (in hex)

  C3 88 20 6C 27 6F 72 61 21 0A

Approfondimenti

- Il sito principale su UNICODE è http://www.unicode.org; la pagina http://unicode.org/charts/ è particolarmente affascinante
- L'ultima versione dello standard è pubblicata come *“The Unicode Standard, Version 5.0”*, Addison-Wesley Professional, ISBN: 0321480910 (1472 pagine, $40)
- Alcuni alfabeti registrati “per uso privato” (inclusi Klingon, Elfico, lingua dei segni, ecc.) sono elencati all'URL http://www.evertype.com/standards/csur
- Un confronto fra le varie codifiche di UNICODE è presentato alla pagina "Comparison of Unicode encodings” di Wikipedia (http://en.wikipedia.org)