Emacs Beginner's HOWTO

Jeremy D. Zawodny, *Jeremy@Zawodny.com* v1.12, 25 Marzo 2001

Questo documento introduce gli utenti Linux all'editor Emacs. Si assume una minima familiarità con `vi` o un editor similare. La versione più recente di questo documento è normalmente disponibile da *http://www.wcnet.org/jzawodn/emacs/* . Traduzione a cura di Giovanni Benedetti (bened@toglimi.tin.it). Aggiornato e mantenuto da Gianluigi Spagnuolo (spagnuologianluigi@toglimi.interfree.it), revisionato da Kriss (chgwor@toglimi.tin.it).

Indice

1 Introduzione

1.1 Copyright

1.2 Pubblico e Intenti

Questo documento è indirizzato agli utenti Linux interessati ad imparare qualcosa riguardo Emacs ed a provarlo. Il tutto di fatto è iniziato come un estratto di un breve corso che ho dovuto tenere ad un incontro a Toledo in un Linux User Group locale: *http://www.talug.org/* . È poi un po' cresciuto come risultato dell'utile scambio che ho ricevuto dalla comunità. Vedere la sezione Crediti per dettagli.

Detto questo, non c'è praticamente niente di specifico su Linux in questo documento. Tutto è in effetti applicabile a tutti i tipi di Unix e perfino ad Emacs funzionante su Microsoft Windows. Ma dato che questo documento fa parte del Linux Documentation Project, considero importante dire che è stato scritto per gli utenti Linux, anche perché di fatto, lo è stato.

E infine, quelli di voi che preferiscono il nome GNU/Linux al posto del semplice "Linux" (leggere *http://www.gnu.org/gnu/linux-and-gnu.html* per capire perché uno potrebbe preferirlo) accetteranno volentieri di sostituire mentalmente GNU/Linux in tutte le occorrenze di Linux in questo documento. Nonostante non sia in disaccordo con le ragioni e lo spirito dietro questa idea, non mi sento obbligato a scrivere GNU/Linux.

1.3 Che cos'è Emacs?

Emacs è qualcosa di differente per ciascuna persona. A seconda a chi lo chiedi, potresti ottenere una qualsiasi delle seguenti risposte:

- Un editor di testo
- Un client per la posta
- Un lettore di news
- Un Word Processor
- Una religione
- Un ambiente di sviluppo integrato
- Qualsiasi cosa tu voglia esso sia!

Ma per i nostri scopi, fingiamo che sia solo un editor di testo, un editor di testo sorprendentemente flessibile comunque. Scaveremo più a fondo nella questione più avanti. Emacs è stato scritto da Richard Stallman (fondatore della Free Software Foundation: *http://www.fsf.org/* e del progetto GNU *http://www.gnu.org/*) che ancora oggi lo mantiene.

Emacs è uno dei più popolari e potenti editor di testo usato in Linux (e Unix). È secondo in popolarità solo a **vi**. È conosciuto per il suo enorme insieme di funzionalità, la possibilità di essere facilmente personalizzato e la mancanza di bug.Le sue molteplici funzionalità e la possibilità di essere personalizzato sono di fatto il risultato di come Emacs sia stato progettato e implementato. Senza entrare in tutti i dettagli, semplicemente farò notare che Emacs non è "soltanto un editor". È un editor scritto principalmente nel linguaggio di programmazione **Lisp**. Nel cuore di Emacs c'è un'interprete Lisp con funzionalità complete scritto in C. Solo le parti più basilari e a basso livello di Emacs sono scritte in C. La maggior parte dell'editor è di fatto scritto in Lisp. Quindi, in un certo senso, Emacs ha un intero linguaggio di programmazione "incorporato" che potete usare per personalizzare, estendere e cambiare il suo ambiente.

Emacs è anche uno dei più vecchi editor in circolazione. Il fatto che sia stato usato da migliaia di programmatori negli ultimi 20 (?) anni, significa che ci sono molti pacchetti aggiuntivi (add-on) disponibili. Questi add-on vi permettono di far fare a Emacs cose che Stallman non aveva probabilmente nemmeno sognato essere possibili quando iniziò a lavorare su Emacs. Altro su questo argomento si trova in una sezione più avanti.

Ci sono molti altri siti Web e documenti che danno una migliore visione globale di Emacs, la sua storia e gli avvenimenti relativi. Piuttosto che tentare di riprodurre qui molto di tutto questo, vi suggerisco di dare un'occhiata in alcuni dei posti elencati nella sezione 6 (Altre risorse) in questo documento.

1.3.1 Piattaforme e Versioni

È importante notare che ci sono di fatto due differenti editor Emacs: GNU Emacs e XEmacs. Ambedue provengono dalla stessa eredità e condividono la maggior parte delle stesse caratteristiche. Questo documento è relativo al GNU Emacs (versione 20.3, specificatamente) ma molto di quello che leggerete qui si applica altrettanto bene a XEmacs e alle prime versioni di GNU Emacs. In questo documento mi riferirò semplicemente a "Emacs". Quando lo farò, tenete presente questo.

1.3.2 Ottenere Emacs

Ottenere Emacs è facile. Se state usando una delle diffuse distribuzioni di Linux, tipo Debian, RedHat, Slackware o qualsiasi altra, Emacs è probabilmente in un pacchetto (package) opzionale che potete installare dal supporto della vostra distribuzione (CD-Rom, floppy, etc.). Altrimenti, potete ottenere il codice sorgente di Emacs e compilarlo da soli. Visitate il sito Web di GNU per l'esatta locazione: *http://www.gnu.org/software/emacs/emacs.html*

2 Far girare Emacs

2.1 Avviare e Chiudere Emacs

Come nuovi utenti, vorrete probabilmente lanciare Emacs giusto per fare un giro al suo interno e provarlo. Una volta dentro Emacs se voleste uscire, tuttavia, potreste non essere in grado di capire che cosa fare. Quindi se non avete mai usato Emacs prima, dategli un'occhiata ora. Al prompt della vostra shell, digitate `emacs` e premete invio. Emacs dovrebbe avviarsi. Se non lo fa, potrebbe non essere installato o non essere nel vostro path.

Una volta che avete visto Emacs, avete bisogno di sapere come uscire. I tasti chiave per lasciare Emacs sono `C-x C-c`. La notazione `C-x` significa: tenere premuto il tasto `Ctrl` e premere il tasto `x`. In questo caso, dovrete poi ancora tenere premuto il tasto `Ctrl` e premere il tasto `c` per raggiungere lo scopo.

I tasti chiave usati in Emacs vi possono sembrare insoliti, strani e forse perfino scomodi all'inizio, specialmente se siete un utente di `vi`. Al contrario di `vi`, Emacs non ha modalità separate per editare il testo e inviare comandi.

Per riassumere: `emacs` avvierà Emacs. `C-x C-c` farà uscire da Emacs.

2.1.1 Che cosa vedrete

Quando Emacs si sarà avviato riempirà interamente una finestra di X (o lo schermo se lo state lanciando da una console invece che nel sistema X-Window). Vedrete dei menù in alto, del testo nella parte principale dello schermo e un paio di linee in fondo.

Sarà simile a questo disegno in ASCII:

```
+----------------------------------------------------------------------+
|Buffers Files Tools Edit Search Mule Help                             |
|                                                                      |
|Welcome to GNU Emacs, one component of a Linux-based GNU system.      |
|                                                                      |
|                                                                      |
|                                                                      |
| ...                                                                  |
|                                                                      |
|---1:---F1  *scratch*         (Lisp Interaction)--L1--All-------------|
|For information about the GNU Project and its goals, type C-h C-p.    |
+----------------------------------------------------------------------+
```

NOTA: Emacs normalmente riempie l'intera finestra/schermo. Ho ristretto l'esempio sopra per salvare spazio qui. Vedrete anche un messaggio di benvenuto in Emacs, quando lo avviate per la prima volta. L'ho omesso in questo esempio e sostituito con "`...`". Il messaggio di benvenuto semplicemente identifica l'esatta versione di Emacs che state usando e vi indirizza all'aiuto in linea e cose del genere.

La barra dei menù La linea più in alto nell'interfaccia Emacs è un menù. Se state usando X, lo riconoscerete come un normale menù a discesa a cui potete accedere usando il mouse. Altrimenti per accedere ai menù avrete bisogno di usare le scorciatoie da tastiera (non trattate qui).

La barra di stato (Status Bar) e il Mini-buffer Delle ultime due linee in basso nell'interfaccia Emacs, quella superiore è essenzialmente una barra di stato. Contiene informazioni sul buffer in cui state lavorando, in quale modalità (mode) si trova Emacs e varie altre cose. Per ora, prendete solo atto che la barra è là.

La linea inferiore è chiamata **mini-buffer**. È separato dal buffer principale dalla barra di stato di cui abbiamo appena parlato. Potete pensare al mini-buffer come la "riga di comando" (command-line) di Emacs. È dove appaiono i comandi che date a Emacs ed è dove vengono visualizzati i messaggi di stato in risposta a quello che fate.

Troverete che a quella che ho chiamato barra di stato, nella documentazione allegata a Emacs, si fa normalmente riferimento come linea di modalità (mode line). È dove Emacs mostra informazioni relative alla/alle modalità corrente/i che potreste star usando e altre cose tipo la data e l'ora corrente, il numero di riga, la dimensione del file e quasi tutto quello che potreste voler vedere là.

2.2 Un po' di terminologia

Questa sezione riguarda gli elementi più basilari della terminologia Emacs che incontrerete quando userete e leggerete di Emacs.

2.2.1 Buffer e File

A differenza di alcuni editor, quando aprite un file in Emacs questo non sta "aperto" tutto il tempo in cui lavorate con esso. Al contrario, Emacs legge il file in un **buffer** in memoria. Mentre state editando il buffer e lavorando con i dati, niente è cambiato sul disco. Solo quando di fatto salvate il buffer, allora il file sul disco viene aggiornato. Ci sono vantaggi e svantaggi con questo approccio ma è importante solo che capiate che lavora in questo modo.

Di conseguenza, vedrete il termine "buffer" usato nella documentazione Emacs, nelle modalità, nei package e così via. Considerate che buffer significa "una copia del file che si trova attualmente in memoria". Oh, è importante notare che un buffer non deve sempre essere riferito ad uno specifico file sul disco. Spesso Emacs creerà dei buffer come risultato dei comandi che lancerete. Questi buffer potranno contenere il risultato dei comandi, una lista di selezioni da cui scegliere e così via.

2.2.2 Point e Region (Punto e Regione)

Nel gergo di Emacs, sentirete o vedrete spesso riferimenti al **point**. In termini generali il point è il cursore. La reale distinzione fra il point e il cursore probabilmente non è importante quando comincerete a usare Emacs. Ma se siete curiosi, pensate al riguardo in questa maniera. Il cursore è la rappresentazione visiva del point. Il cursore è sempre "su" una particolare posizione del carattere nel buffer corrente. Il point, invece, vive nello spazio *fra i caratteri* che si trovano nel buffer. Quindi potreste dire che se il cursore si trova sulla lettera 'h' nella parola "the" allora il point è tra la 't' e la 'h'.

Come molti editor moderni, Emacs permette di effettuare operazioni (indentazione, controllo ortografico, riformattazione, taglia, copia, incolla ...) su una porzione del buffer corrente. Potete evidenziare (o "marcare") un blocco di testo usando la tastiera o il mouse e poi eseguire operazioni solo sul blocco selezionato di testo. In Emacs, quel blocco di testo è chiamato una **region** (regione).

2.2.3 Finestre

Okay, questo sarà un po' confuso per chi non abbia usato prima una interfaccia grafica (GUI). Ricordate che Emacs fu sviluppato molto prima che le interfacce GUI e i gestori di finestre (window manager) diventassero popolari.

Una **finestra** in Emacs è un area dello schermo nel quale è visualizzato un buffer. Quando Emacs viene avviato per la prima volta, avete una finestra sul vostro schermo. Alcune funzioni di Emacs (tipo l'help e la documentazione) spesso aprono (temporaneamente) una finestra aggiuntiva nella vostra schermata di Emacs.

Le finestre di Emacs non hanno niente a che fare con le finestre X nel senso delle GUI. Potete aprire finestre X addizionali per mostrare i buffer di Emacs, magari per confrontare due file fianco a fianco. Queste nuove finestre X sono chiamate **frame** (riquadri) in gergo Emacs. Continuate a leggere.

2.2.4 Riquadri (Frame)

In Emacs, un **frame** è una finestra X separata nel quale viene mostrato un buffer di Emacs. Ma entrambe fanno parte della stessa sessione di Emacs. Il comportamento è qualcosa di simile (ma non troppo) a quello che succede se premete Alt+N in Netscape Navigator.

2.3 Basi dell'uso della tastiera

Questa sezione copre l'uso basilare della tastiera per Emacs. Come con ogni editor potente, tutto quello che potete fare con Emacs è giusto qualche tasto chiave più avanti.

Se siete un utente `vi`, le nozioni sull'uso dei tasti `k, j, l, h` per spostarsi su alla riga superiore, giù di una riga, avanti di un carattere e indietro di un carattere, probabilmente ora tornano utili. In realtà, vi ci potrebbero esser volute poche ore oppure settimane di pratica, prima di poter navigare confortevolmente in un file usando le varie combinazioni di tasti disponibili in `vi`.

Emacs non è diverso. Ci sono tasti e comandi diversi da imparare. Come con `vi`, avete solo bisogno di padroneggiare le cose basilari per ritrovarvi con molto lavoro fatto. Dopo, con il passare del tempo, potrete lentamente espandere la vostra conoscenza e trovare strade più veloci per fare le cose.

2.3.1 Tasti di comando (Meta, Esc, Control e Alt)

Come imparerete presto, Emacs fa un uso intensivo di combinazioni di più tasti. Dato che non è un editor modale come `vi`, non dovete preoccuparvi di essere in "modalità comandi" o "modalità inserimento" prima di provare a muovere il cursore o eseguire un comando. Invece dovete solo premere la giusta combinazione di tasti e (normalmente) Emacs farà quello che gli è stato detto.

I tasti di cui Emacs fa maggiore uso sono normalmente abbreviati nella documentazione come `C` (per Control o Ctrl) e `M` per (Meta). Mentre le più moderne tastiere di PC hanno uno o più tasti etichettati come `Ctrl`, poche ne hanno uno etichettato come `Meta`. Potrete mentalmente sostituire sia `Esc` che `Alt` al tasto Meta. Nella maggior parte delle configurazioni standard, entrambi, Esc e Alt, faranno essenzialmente le stesse cose.

Quindi quando vedete un riferimento, in qualsiasi documentazione relativa a Emacs, a `C-x f`, significa "premere control-x e poi f". E se vedrete un riferimento a qualcosa del tipo `M-x shell` significa "premere alt-x e digitare la parola shell".

Un comando veramente utile per chi inizia è `M-x apropos` o `C-h a`. apropos cercherà, nella documentazione in linea di Emacs, tutte le funzioni e cercherà l'espressione regolare che digiterete. Questo, ad esempio, è un ottimo modo per scoprire tutti i comandi relativi a un frame. Semplicemente digitate `C-h a` e poi `frame`.

2.3.2 Muoversi in un Buffer

Ora che sapete che cosa significano tutte quelle simpatiche abbreviazioni, ecco qui una lista delle combinazioni di tasti più comuni per muoversi in un buffer:

```
Tasti          Azione
-----------------------------------
C-p          Su di una riga
C-n          Giu di una riga
C-f          Avanti di un carattere
C-b          Indietro di un carattere
C-a          Inizio di una riga
C-e          Fine di una riga
```

```
C-v         Giu di una pagina
M-v         Su di una pagina
M-f         Avanti di una parola
M-b         Indietro di una parola
M-<         Inizio del buffer
M->         Fine del buffer
C-g         Chiude l'operazione corrente
-----------------------------------
```

E, come potevate aspettarvi, i tasti cursore (o tasti freccia) funzionano normalmente come vi aspettavate. Il vostro tasto `Backspace` potrebbe non funzionare invece. Ma questa è un'altra storia. `:-(`

2.3.3 Comandi essenziali

Okay, ora che sapete come spostarvi in un buffer, che cosa ne pensate di aprire e salvare file? Ricercare? Ecco alcuni comandi base.

Prima di saltare dritto su questi comandi, ho bisogno di puntualizzare brevemente come questi lavorano.

Tutti i "tasti di comando" in Emacs (quelli che sono `M-x qualcosa` o `C-qualcosa`) sono di fatto proprio delle scorciatoie a delle funzioni che fanno parte di Emacs. Potete chiamare una qualsiasi di queste funzioni digitando `M-x funzione-nome` e premendo `Enter`. Potete anche usare le scorciatoia da tastiera per questa funzione (se ne ha una).

Per esempio, la funzione di Emacs che salva un buffer su disco è chiamata `save-buffer`. Per default è anche vincolata a `C-x C-s`. Quindi, potete usare sia la scorciatoia da tastiera per salvare il buffer corrente, sia digitare `M-x save-buffer` per raggiungere esattamente lo stesso risultato.

Tutte le funzioni più comuni hanno delle scorciatoie da tastiera per default. Alcune di esse sono elencate qui sotto.

```
Tasti          Funzione          Descrizione
-------------------------------------------------------------------
C-x C-s     save-buffer        Salva il buffer corrente su disco
C-x u       undo               Annulla l'ultima operazione
C-c C-f     find-file          Apre un file dal disco
C-s         isearch-forward    Cerca avanti una stringa
C-r         isearch-backward   Cerca indietro una stringa
            replace-string     Cerca e rimpiazza una stringa
            replace-regexp     Cerca e rimpiazza usando regexp
C-h t       help-with-tutorial Usa la guida interattiva
C-h f       describe-function  Mostra aiuto per una funzione
C-h v       describe-variable  Mostra aiuto per una variabile
C-h x       describe-key       Mostra che cosa fa una sequenza di tasti
C-h a       apropos            Cerca aiuto per una stringa/regexp
C-h F       view-emacs-FAQ     Mostra le FAQ di Emacs
C-h i       info               Legge la documentazione di Emacs
C-x r m     bookmark-set       Imposta un segnalibro. Utile nelle ricerche
C-x r b     bookmark-jump      Salta ad un segnalibro.
-------------------------------------------------------------------
```

Quando proverete molte di queste funzioni, noterete che molte vi chiedono di inserire qualcosa al prompt. Lo faranno sempre nel mini-buffer. Questo è simile all'uso dei comandi `:` in `vi` o la maggior parte dei comandi che usereste nella vostra shell Unix favorita.

Emacs ha letteralmente centinaia di funzioni incorporate disponibili. La lista riportata sopra è un campione minimo che rappresenta quelle che io uso regolarmente. Vedere l'aiuto in linea per un più completo elenco delle funzioni disponibili ed una documentazione più completa su quelle che ho menzionato sopra.

2.3.4 Completamento con Tab

Come molte shell Unix popolari (bash, csh, tcsh, ...) Emacs offre il completamento del comando tramite il tasto `Tab`. Infatti il completamento del comando in bash venne preso a modello proprio da Emacs, quindi se usate questa caratteristica in bash vi troverete subito bene.

Come esempio, provate `M-x search` e poi premete `Tab`. Emacs aggiungerà un trattino per indicare che ci sono molti possibili completamenti ma questi hanno tutti un trattino come carattere successivo. Premete `Tab` una volta ancora e Emacs mostrerà una lista delle possibili combinazioni da cui voi potrete scegliere. Notate che farà questo in una *nuova finestra*. Temporaneamente dividerà il vostro schermo in due finestre: una contenente il buffer che state editando e l'altra che contiene la lista dei possibili completamenti per "search-". Potete premere `C-g` per uscire fuori dal processo di selezione e chiudere la nuova finestra.

2.4 Tutorial, Aiuto e Info

Emacs ha un tutorial in linea che vi accompagna attraverso le caratteristiche di base dell'editing e delle funzioni che ognuno dovrebbe conoscere. Spiega anche come usare le altre funzionalità di aiuto in Emacs.

Io vi raccomando caldamente di spendere un po' di tempo nel consultare a fondo il tutorial se pensate di sforzarvi seriamente per imparare Emacs. Come mostrato nella tabella sopra riportata, potete entrare nel tutorial tramite `C-h t`. Il tutorial è una auto-guida e aiuta le persone che hanno appena iniziato con Emacs.

Se state facendo girare Emacs in X, vedrete che il menù più a destra nella barra dei menù è etichettato con Help. Come esplorerete il menù di Help noterete che certe voci hanno delle scorciatoie da tastiera e che queste sono elencate a destra nel menù.

Infine, per vedere l'intero volume della documentazione disponibile per Emacs, dovreste provare `M-x info` o `C-h i` che lancia Info, il browser per la documentazione di Emacs.

3 Modalità di Emacs

Le modalità (mode) di Emacs sono differenti ambienti e funzionalità che potete attivare o disattivare (o personalizzare, ovviamente) per usarle in circostanze diverse. Le modalità sono quello che rende un editor (Emacs) ugualmente utile per scrivere documentazione, programmare in una varietà di linguaggi (C, C++, Perl, Python, Java e molti altri), creare una home page, inviare E-Mail, leggere i newsgroup Usenet, tenere traccia dei vostri appuntamenti e perfino giocare.

Le modalità di Emacs sono semplicemente delle librerie di codice Lisp che estendono, modificano o migliorano Emacs in qualche modo.

3.1 Modalità Primarie vs. Modalità Secondarie

Ci sono fondamentalmente due tipi di modalità disponibili: Primarie (Major) e Secondarie (Minor). La distinzione non è la cosa più facile da afferrare finché non avrete lavorato con un po' di queste, ma proviamo a darne una spiegazione.

In un dato momento può essere attivata solamente una modalità primaria. Invece nello stesso momento possono essere attive molte modalità secondarie. Le modalità primarie tendono ad essere specifiche di un linguaggio o di un compito, mentre le modalità secondarie sono delle utility più piccole e meno specifiche che riguardano molti compiti.

Suona come qualcosa di astratto, quindi proviamo con un esempio. C'è una modalità che io uso abbastanza spesso quando devo scrivere dei vecchi e semplici file di testo. Si chiama `text-mode`. Questa modalità fu progettata per scrivere testo in forma libera come in un file README. Capisce come identificare parole e paragrafi e in genere si assicura di fare quello che mi aspetto quando uso i normali tasti di navigazione in un documento.

Quando sto scrivendo testo per uso umano, normalmente voglio che abbia un bell'aspetto. Dovrebbe essere allineato in maniera appropriata ad un valore ragionevole e così via. Per abilitare l'allineamento devo solo attivare la modalità secondaria `auto-fill`. Questo modalità cerca di fare la Cosa Giusta quando continuo a scrivere e raggiungo la fine della riga. Il fatto che sia una modalità secondaria significa che può lavorare con diverse modalità primarie. La mia accezione di "Cosa Giusta" da fare quando raggiungo la fine della riga è diversa quando sono in `text-mode` da quando sono in `java-mode`, per esempio. Io non voglio che il mio codice Java sia allineato come se fosse testo in inglese. Ma io *voglio* che i blocchi di commenti nel mio codice Java siano allineati! La modalità `auto-fill` è intelligente abbastanza da capire il tutto.

Gli autori delle varie modalità di Emacs hanno fatto un ottimo lavoro nell'assicurarsi che funzioni che dovrebbero operare come modalità secondarie siano effettivamente delle modalità secondarie.

Se riguardate il disegno in ASCII della schermata di Emacs, noterete che la linea di modo identifica la/le modalità in cui Emacs si trova. Nell'esempio, era in una modalità chiamata "Lisp Interaction" che è la modalità di default. È in realtà utile solamente se state per scrivere codice Lisp. (Ma dato che la maggior parte di Emacs è scritta in Lisp, perché no?)

3.2 Le Modalità di Programmazione

Innanzitutto e molto importante, Emacs è stato elaborato da un programmatore per programmatori. Ci sono modalità di alta qualità disponibili per quasi ogni linguaggio di programmazione più diffuso a cui potete pensare (e perfino per qualcuno non così diffuso). Qui descriverò solo brevemente alcuni di essi.

La maggior parte delle modalità condividono alcune caratteristiche comuni. Normalmente, alcune o tutte delle cose seguenti:

- Forniscono un'evidenziazione a colori della sintassi del linguaggio.
- Forniscono un indentazione automatica e formattazione del codice del linguaggio.
- Forniscono un aiuto (del linguaggio) sensibile al contesto.
- Si interfacciano automaticamente con il vostro debugger.
- Aggiungono dei menù specifici del linguaggio alla barra dei menù.

In più, ci sono delle modalità non specifiche di un linguaggio che aiutano per scopi che sono comuni alla programmazione in molti linguaggi. Cose del tipo interfacciamento per il vostro controllo di versione del software, aggiunta automatica di commenti al vostro codice, creazione di Makefile, aggiornamento di Change Logs e così via.

Aggiungendo tutte queste modalità e considerando la maturità e la stabilità del codice di Emacs, questi compete piuttosto bene al paragone con gli Integrated Development Environments (IDE) commerciali sul mercato per linguaggi come C++ e Java. Ed è,ovviamente, gratuito.

3.2.1 C/C++/Java

Dato che la sintassi del C, C++ e Java sono abbastanza simili, c'è una modalità di Emacs che tratta tutti e tre i linguaggi (lo stesso per Objective-C e IDL). È veramente un maturo e completo package ed è incluso nella distribuzione di Emacs. Questa modalità si chiama `cc-mode` o `CC Mode`.

Per maggiori dettagli o per scaricare la versione più recente, visitate *http://www.python.org/emacs/* .

3.2.2 Perl

Ci sono di fatto due modalità per l'editing del codice Perl in Emacs. Il primo si chiama `perl-mode` (come vi potevate aspettare) e il secondo è `cperl-mode`. Non ho una buona padronanza di questa storia e del perché ci sono due modalità (i documenti non lo dicono), ma sembrerebbe che `perl-mode` fosse la modalità originale per editare il codice Perl in Emacs. Sembra avere meno servizi del `cperl-mode` e manca l'abilità di riconoscere alcuni costrutti estrosi del linguaggio Perl.

Personalmente, io uso e raccomando `cperl-mode` il quale sembra essere abbastanza attivamente mantenuto ed ha quasi tutte le funzionalità che potrei mai volere. Potete trovare l'ultima release qui: *ftp://ftp.math.ohio-state.edu/pub/users/ilya/emacs* .

Ma non prendete le mie parole come definitive. Provateli entrambi e usate quello che meglio soddisfa le vostre necessità.

3.2.3 Python

Anche per Python (un altro linguaggio di scripting molto popolare) è disponibile una modalità Emacs. Per quel che posso dire io, *non* è distribuito con GNU Emacs ma è distribuito con XEmacs. Comunque funziona abbastanza bene in ambedue gli editor.

Potete ottenere il `python-mode` dal sito web ufficiale di Python *http://www.python.org/emacs/python-mode/* .

3.2.4 Altro

Ci sono tantissime altre modalità di editing disponibili per aiutare i programmatori. Queste modalità aiutano con cose tipo:

- Script della shell (Bash, sh, ksh, csh, ...)
- Awk, Sed, Tcl, ...
- I Makefile
- Change Logs
- Documentazione
- Debugging

E ancora di più. Guardate l'ultima sezione di questo documento per maggiori informazioni per trovare altre modalità e add-in.

3.3 Editoria

Le fantasiose modalità di Emacs *non* sono limitate solo a quelli che scrivono codice. Anche persone che scrivono documentazione (di qualsiasi tipo) possono beneficiare di un'ampia scelta di modalità di Emacs.

3.3.1 Controllo ortografico (`ispell` mode)

Gli autori di molti tipi di documenti hanno bisogno di effettuare ogni tanto il controllo ortografico. Se avete **GNU ispell** installato, potete digitare `M-x ispell` ed effettuare il controllo ortografico sul buffer corrente. Se ispell trova parole che non conosce, vi avvisa con una lista di possibili rimpiazzi e vi permette di selezionarne uno (o nessuno). Le sue funzionalità equivalgono ai controllori ortografici in molti pacchetti software diffusi, non gratuiti.

3.3.2 HTML (`html-helper` mode)

Se vi trovate a dover scrivere file HTML una volta ogni tanto (oppure molte volte), potreste provare `html-helper-mode`. È disponibile da *http://www.santafe.edu/˜nelson/tools/* oltre alla documentazione ed altro materiale correlato.

Come il suo nome suggerisce, `html-helper-mode` fornisce molte cose per aiutare quelle persone che ancora scrivono HTML a mano (alla vecchia maniera).

3.3.3 TeX (`tex-mode`)

Quando state scrivendo documenti in TeX, è spesso di aiuto avere Emacs che aggiunge alcuni colori ed evidenzia le barre inverse, parentesi graffe ed altri caratteri. `tex-mode` si occupa di questo per voi.

Sebbene non scriva più molto direttamente in TeX, quando l'ho fatto, questa modalità si è rivelata abbastanza utile nel rendere i miei sorgenti in TeX un po' più leggibili.

3.3.4 SGML (`sgml-mode`)

Il documento che state leggendo è stato scritto in SGML (e probabilmente convertito nel formato in cui lo state leggendo). `sgml-mode` fornisce tutte le basi per i documenti SGML: validazione, evidenziazione, marcatore avanti, marcatore indietro e molto altro. È una parte standard di Emacs.

3.4 Altre modalità

Ovviamente, ci sono molte altre utili modalità per rendere la vita più facile. Ecco giusto un campione delle più diffuse:

3.4.1 Controllo di versione (`vc` mode)

La modalità `vc` si interfaccia con la maggior parte dei più diffusi controllori di versione dell'ultim'ora (RCS, SCCS, CVS) per rendere veramente facile il controllo dei file dentro e fuori, gestire release e così via. È una parte standard di Emacs ed è documentata nella documentazione di Emacs.

3.4.2 Modalità Shell

Perché spostarsi in un'altra finestra X o in una console virtuale solo per far girare pochi comandi della shell? Fatelo da dentro Emacs e evitatevi il problema. `:-)`

`M-x shell` lancerà una shell dentro a un buffer di Emacs. Potete fare con questo buffer la maggior parte di quello che potreste fare al prompt di una normale shell (eccetto che lanciare programmi a tutto schermo come `vi` o `pine`) poiché Emacs dialoga con la vostra vera shell dietro le quinte.

Anche questa è una parte standard di Emacs, quindi la troverete documentata nella documentazione di Emacs.

3.4.3 Telnet e FTP

Perché spostarsi in un'altra finestra X o in una console virtuale solo per far girare telnet o FTP? Fatelo da dentro Emacs e evitatevi il problema. (Non avete ancora notato il modello?)

Similmente a lanciare una shell all'interno di Emacs, potete fare telnet e ftp. Provate `M-x telnet` o `M-x ftp` per sperimentarlo da soli. Vedere la documentazione per tutti i grumosi dettagli.

3.4.4 Man

Perché spostarsi in un'altra finestra X o in una console virtuale solo per leggere una pagina man? Fatelo da dentro Emacs e evitatevi il problema. (Smetto! Lo prometto.)

Similmente a lanciare una shell all'interno di Emacs, potete leggere le pagine man. Provate `M-x man` per sperimentarlo da soli. Vedere la documentazione per altre informazioni.

3.4.5 Ange-FTP

Citando la documentazione di `ange-ftp`:

> Questo pacchetto tenta di dare accesso a file e directory usando FTP dall'interno di GNU Emacs nel modo più semplice e trasparente possibile. Un sottoinsieme delle comuni routine di gestione dei file sono state estese per interagire con FTP.

Questo significa che potete trattare i file in una macchina remota come se fossero in locale. Quindi se avete bisogno di editare un file su un computer remoto, dite semplicemente a Emacs di aprirlo (usando una sintassi leggermente diversa per il percorso) ed egli si occuperà di tutti i dettagli di collegamento e rintracciamento del file. Dopo, quando salvate il file con `C-x C-s`, `ange-ftp` intercetta il salvataggio e scrive il file sulla macchina remota.

La sintassi leggermente diversa per il percorso è qualcosa di simile... Un file chiamato "miofile", in una directory "user", su una macchina chiamata "my.host.org" può essere aperto aprendo (`C-x f`) il file con:

```
/user@my.host.org:~user/miofile
```

Anche questo è parte della distribuzione standard di Emacs, quindi potete trovare la sua documentazione nella documentazione di Emacs.

Grazie a Etienne Grossmann (*etienne@anonimo.isr.ist.utl.pt*) per l'esempio sopra riportato.

4 Personalizzare Emacs

Di fatto tutta la personalizzazione di Emacs viene fatta tramite codice Lisp. Potete modificare variabili che influenzano il modo in cui Emacs opera o potete aggiungere nuove funzioni a Emacs (o sovrascrivere funzioni esistenti, rimpiazzandole con altre vostre).

4.1 Personalizzazione temporanea

Mentre starete sperimentando la personalizzazione di Emacs, probabilmente vorrete farlo in un modo che sia temporaneo. Se fate qualcosa di orribilmente sbagliato, avrete solo da uscire da Emacs con `C-x C-c` e rilanciarlo di nuovo. Una volta che avrete capito quali cambiamenti rendere permanenti, potrete aggiungerli al vostro file molto personale `.emacs` in modo che possano avere effetto ogni volta che avvierete Emacs. Questo viene discusso nella prossima sezione.

4.1.1 Assegnazione di variabili

Le personalizzazioni più semplici si ottengono cambiando il valore di una variabile in Emacs. Il codice listato per farlo, appare simile a questo:

```
(setq nome-variabile nuovo-valore)
```

Dove `nome-variabile` è il nome della variabile e `nuovo-valore` è il valore che vorreste dare alla variabile (in gergo Lisp, state legando una variabile ad un valore). La funzione `setq` in lisp è analoga agli operatori di assegnamento (normalmente `=`) in altri linguaggi di programmazione.

NOTA: Qui sto passando sopra a molti dettagli per amor di semplicità. Potreste anche vedere me o altri, usare le funzioni Lisp `set` e perfino `setq-default`. Se siete veramente curiosi, dateci un'occhiata nei testi di riferimento di Emacs Lisp.

Diamo un'occhiata alla riga presa dal mio file `.emacs`

```
(setq-default transient-mark-mode t)
```

La variabile `transient-mark-mode` controlla se una regione (region) viene evidenziata o no quando la marco. In molte applicazioni GUI, se cliccate e trascinate il mouse per selezionare una parte di testo, questi diventa evidenziato in video inverso o in altri colori. Emacs farà la stessa cosa se la variabile `transient-mark-mode` è impostata (ad un valore non-nil).

Un valore *COME*?

Okay. Breve digressione. La maggior parte dei linguaggi di programmazione hanno diverse opinioni sui valori vero/falso. In C/C++ un valore è considerato vero se è un valore non-zero. In Perl, un valore non-nullo o non-zero è vero. In Lisp, si applica la stessa idea ma i nomi e i simboli sono differenti.

Vero è normalmente scritto come `t` e falso (o null) è scritto come `nil`. Come in altri linguaggi, tuttavia, qualsiasi valore non-nil è considerato vero.

Per ottenere la descrizione completa di che cosa fa `transient-mark-mode`, potete usare l'aiuto in linea. Digitate `C-h v` o `M-x describe-variabile` e poi `transient-mark-mode`. Se siete pigri come me, potete avvalervi del completamento del nome della variabile usando il tasto `Tab`. Digitate solo una parte del nome della variabile e premete il tasto `Tab`. Se avete digitato abbastanza lettere del nome in modo che Emacs possa già identificarlo univocamente, vedrete l'intero nome completarsi per voi.

Un'altra variabile che le persone spesso impostano è `fill-column`. Questa dice ad Emacs quanto largo deve essere lo schermo per eseguire l'allineamento (e `auto-fill-mode` rispetta questo valore). Per impostare il valore su qualcosa di assurdo, potete digitare:

```
(setq fill-column 20)
```

Ma questo di fatto non produrrà niente. Avete bisogno di dire a Emacs di **valutare** l'espressione che avete digitato. Per fare questo, portate il puntatore (cursore) alla fine dell'espressione e poi digitate `C-x C-e`, il quale chiama la funzione `eval-last-sexp` nel caso non siate sicuri. Quando fate questo, notate che `20` (o qualsiasi valore avete usato) vi viene mostrato nel mini-buffer alla base dello schermo. Questo è giusto il valore ritornato dall'espressione che avete valutato.

Giusto per provare che funziona, digitate una frase o due. Se avete `auto-fill-mode` abilitato (probabilmente non lo avete), noterete il testo allineato alla colonna marcata 20. Altrimenti, dopo che avrete digitato qualcosa, digitate `M-q` che chiama la funzione `fill-paragraph`. Questo eseguirà l'allineamento del testo.

4.1.2 Associazioni di file

Potete configurare Emacs per fare automaticamente qualcosa quando aprite un file di un tipo particolare (come certe GUI che lanciano automaticamente una specifica applicazione se cliccate sull'icona di un determinato file). Per esempio, io potrei volere che automaticamente Emacs entrasse in `text-mode` ogni volta che io apro un file con estensione `.txt`. Beh!, questo già succede. `:-)` Allora diciamo a Emacs di entrare sempre in `text-mode` quando si apre un file chiamato "README".

```
(setq auto-mode-alist (cons '("README" . text-mode) auto-mode-alist))
```

Eh?

Senza immergersi dentro tanti comandi Lisp che in effetti non avete bisogno di conoscere (ma che non vi farebbe male imparare), diciamo solo che la variabile `auto-mode-alist` contiene una lista di coppie. Ciascuna coppia contiene un'espressione regolare ed un nome di modalità di Emacs. Se un file che aprite coincide con l'espressione regolare (in questo caso, la stringa `README`) Emacs avvia la modalità che avete specificato.

La strana sintassi riportata sopra è perché noi stiamo di fatto aggiungendo un altra coppia a quella lista delle modalità. E voi non vorreste giusto assegnare qualcosa a `auto-mode-alist` senza assicurarvi che i valori che già contiene non vengano persi.

E se io voglio che Emacs entri automaticamente in `html-helper-mode` ogni volta che apro un file che finisce con `.html` o `.htm`, lo aggiungerò al mio file .emacs:

```
(setq auto-mode-alist (cons '("\\.html$" . html-helper-mode) auto-mode-alist))
(setq auto-mode-alist (cons '("\\.htm$" . html-helper-mode) auto-mode-alist))
```

Le possibilità sono veramente infinite.

4.2 Usare un file `.emacs`

Dopo che avrete speso un po' di tempo con Emacs e avrete un'idea basilare di che cosa la personalizzazione possa fare per voi, vorrete probabilmente personalizzare un po' di cose in maniera permanente (o almeno finché non cambiate idea). Se vi ritrovate ad usare Emacs quotidianamente, noterete anche che il vostro file `.emacs` diventa sempre più grande con il passare del tempo. Questa è una *Buona Cosa* perché significa che avete capito come fare in modo che Emacs lavori nel modo in cui **voi** volete che lavori. È una vergogna che tanti prodotti software non vi diano la possibilità di fare questo.

Nel caso non lo abbiate ancora indovinato, ogni volta che avviate Emacs, questi cerca un file chiamato `.emacs` nella vostra directory home. Il vostro file `.emacs` si trova dove dovreste mettere qualsiasi codice Lisp che volete lanciare automaticamente e che include qui il tipo di personalizzazione che abbiamo gestito.

Un altro esempio dal mio file `.emacs`:

```
(setq inhibit-startup-message t)
```

La variabile `inhibit-startup-message` controlla se Emacs mostra o no il messaggio di benvenuto quando si avvia. Dopo un po', mi sono stufato di vederlo (perché sapevo già come trovare o non trovare l'help), quindi ho cercato un modo per disattivarlo.

Come esercizio, provate a creare un file `.emacs` da soli e ad aggiungergli la riga riportata sopra. Poi uscite e riavviate Emacs di nuovo. Non dovreste più vedere il messaggio di benvenuto.

Spesso quando leggerete riguardo una modalità di Emacs (o un package), la documentazione suggerirà di aggiungere del codice al vostro file `.emacs` in modo da far lavorare la modalità o il package in una maniera particolare.

Le FAQ di GNU Emacs (`C-h F`) contengono delle informazioni relative ai file `.emacs` che potreste trovare utili.

4.3 Il package Customize

A mano a mano che Emacs è cresciuto in popolarità e ha continuato ad evolversi, qualcuno evidentemente deve essersi detto "ci deve essere un modo migliore per i nuovi utenti di personalizzare il loro Emacs." E nacque `customize`.

Customize fornisce un modo più intuitivo di personalizzare parti di Emacs. Per provarlo, visitate il sottomenù `Customize` nel vostro menù `Help`, o digitate `M-x customize`.

Customize raggruppa le personalizzazioni in gruppi logici come "Editing", "Programming", "Files" e così via. Alcuni gruppi contengono dei sottogruppi.

Se fate dei cambiamenti usando l'interfaccia di customize, Emacs salverà i cambiamenti nel vostro file `.emacs`. Questo è piuttosto pratico, perché potete facilmente controllare (e cambiare) i cambiamenti che customize fa per voi.

Io non uso l'interfaccia Customize, quindi non posso dirvi molto altro al riguardo..

4.4 Visualizzazione in X Windows

Come qualsiasi applicazione X ben fatta, Emacs rispetta le vostre risorse X. Questo significa che potete controllare i colori iniziali, la geometria e altre cose specifiche di X, come potete farlo di solito con un `xterm`, `nxterm`, o altro.

Ecco qui la parte più importante del mio file `~/.Xdefaults`:

```
emacs*Background: DarkSlateGray
emacs*Foreground: Wheat
emacs*pointerColor: Orchid
emacs*cursorColor: Orchid
emacs*bitmapIcon: on
```

```
emacs*font: fixed
emacs.geometry: 80x25
```

Consultate la vostra pagina man su `X` per maggiori dettagli riguardo le risorse X.

Chris Gray (*cgray4@po-box.mcgill.ca*) inoltre nota:

> In Debian, il file `~/.Xdefaults` non sembra essere usato. Comunque, gli utenti Debian possono mettere quello che hanno stabilito in `/etc/X11/Xresources/emacs` e potranno avere i simpatici colori che avevano quando usavano RedHat.

5 Package diffusi

In aggiunta alle molte differenti modalità disponibili per Emacs, ci sono anche molti **package** add-on. Li ho chiamati package perché sono qualcosa di più che solo nuove modalità. Questi spesso includono delle utility extra o sono così grandi che chiamarli modalità non rende loro giustizia. In altri casi ancora, questi sono software che estendono o integrano altre modalità e package di Emacs. La distinzione non è pienamente chiara, ma va bene lo stesso.

5.1 VM (Mail)

Per citare la FAQ di VM:

> VM (View Mail) è un sottosistema Emacs che permette di leggere e disporre della posta all'interno di Emacs. I comandi esistenti permettono di fare le normali cose che ci si aspetta da un agente client di posta, tipo generare reply, salvare messaggi in cartelle, cancellare messaggi e così via. Ci sono altri comandi avanzati per scopi come far sgorgare e creare riassunti, inoltro di messaggi e organizzazione della presentazione di messaggi in accordo con vari criteri.

Quando ho cominciato ad usare Emacs, ho provato VM per un po' di tempo. L'ho trovato essere un ottimo rimpiazzo di Pine, Elm o per la maggior parte di altri programmi di posta. Ma non ho voluto usare programmi diversi per leggere posta e news. VM è ad oggi attivamente sviluppato e ben supportato.

È disponibile qui: *http://www.wonderworks.com/vm/* .

5.2 Gnus (Mail e News)

Per citare il manuale di GNUS:

> Gnus è un laboratorio di lettura di messaggi. Vi dà la possibilità di accedere a quasi tutto come se fosse un newsgroup. Potete leggere la posta, scorrere fra le directory, fare ftp (potete perfino leggere le news con esso!)
>
> Gnus cerca di dare pieni poteri alle persone che leggono le news nello stesso modo in cui Emacs dà pieni poteri alle persone che editano testo. Gnus non crea limiti a quello che l'utente può essere in grado di fare. Gli utenti sono incoraggiati ad estendere Gnus a comportarsi come loro vogliono che si comporti. Un programma non deve controllare le persone; la gente deve essere autorizzata a fare quello che vuole usando (o abusando) del programma.

GNUS è quello che attualmente uso per mail e news (come alludevo sopra). GNUS è anche attivamente sviluppato e ben supportato ad oggi.

È disponibile qui: *http://www.gnus.org/* .

5.3 BBDB (Un rollodex)

BBDB è un insidioso database da Grande Fratello, un programma tipo-rollodex per Emacs che funziona con la maggior parte dei package più popolari di posta per Emacs Mail (VM e GNUS inclusi).

È disponibile qui: *http://pweb.netcom.com/˜simmonmt/bbdb/index.html* .

5.4 AucTeX (un altra modalità TeX)

AucTeX è un'altra modalità per editare file TeX.

Per citare il sito web di AucTeX:

> AUC TeX è un package ampliabile che supporta scrittura e formattazione di file TeX per la maggior parte delle varianti di GNU Emacs. Sono supportate la maggior parte dei package macro, incluso AMS TeX, LaTeX e TeXinfo.

È disponibile qui: *http://sunsite.auc.dk/auctex/* .

6 Altre risorse

Questa sezione riguarda libri, siti web, newsgroup, mailing list e altri posti in cui potete trovare ulteriori informazioni riguardo Emacs.

6.1 Libri

Ci sono alcuni libri veramente buoni per imparare Emacs. In aggiunta a questi, troverete che anche molti libri su Linux e Unix contengono un capitolo o due riguardo Emacs (e `vi`).

6.1.1 Learning GNU Emacs

(Imparare GNU Emacs)

Autori: Debra Cameron, Bill Rosenblatt, Eric S. Raymond

Editore: O'Reilly & Associates - *http://www.ora.com/*

Commento: Questo è probabilmente il miglior libro da cui iniziare. Dopo aver letto l'HOWTO e scorso le FAQ, questo libro serve come una guida esauriente e molto accessibile.

6.1.2 Writing GNU Emacs Extensions

(Scrivere estensioni per GNU Emacs)

Autore: Bob Glickstein

Editore: O'Reilly & Associates - *http://www.ora.com/*

Commento: Dopo che avrete usato Emacs per un po' e avrete deciso che vi piacerebbe una modalità vostra o forse provare qualche personalizzazione avanzata, questo sarà il libro per voi. Nonostante non provi ad insegnare il Lisp, contiene comunque una breve introduzione al linguaggio.

6.1.3 Programming in Emacs Lisp: An Introduction

(Programmare in Emacs Lisp: Un'introduzione)

Autore: Robert J. Chassell

Dal file README:

> Questa è una introduzione elementare alla programmazione in Emacs Lisp per persone che non sono programmatori e per chi non è necessariamente interessato alla programmazione, ma per chi vuole personalizzare o estendere il proprio ambiente di lavoro.

Potete scaricare il manuale nella sua interezza via FTP anonimo dal server GNU FTP: *ftp://prep.ai.mit.edu/gnu/emacs/* .

Commento: Questo è un buon manuale introduttivo per Emacs Lisp, se non siete un programmatore con grandi pretese.

6.1.4 The GNU Emacs Lisp Reference Manual

(Il manuale di riferimento di GNU Emacs)

Autore: Richard Stallman

Editore: The Free Software Foundation - *http://www.fsf.org/*

Potete scaricare il manuale nella sua interezza via FTP anonimo dal server GNU FTP: *ftp://prep.ai.mit.edu/gnu/emacs/* .

Commento: Questa è la guida più esauriente al linguaggio di programmazione Emacs Lisp.

6.2 Siti Web

6.2.1 EMACSulation

EMACSulation è una rubrica scritta da Eric Marsden che appare nella rivista on-line Linux Gazette che si trova a *http://www.linuxgazette.com/* . La rubrica più recente al momento in cui scrivo si trova a *http://www.linuxgazette.com/issue39/marsden.html* . Cercare alla fine dell'articolo per i collegamenti a quelle precedenti.

6.3 Newsgroup

Cercate dal vostro fornitore locale di news dei newsgroup che contengano la stringa "emacs" e ne troverete probabilmente molti. Quelli che il mio server pubblica sono

- comp.emacs
- comp.emacs.sources
- gnu.emacs

- gnu.emacs.bug
- gnu.emacs.help
- gnu.emacs.sources

6.4 Mailing List

C'è una mailing list dedicata a GNU Emacs che è ospitata da Free Software Foundation. Vedere il sito *http://mail.gnu.org/mailman/listinfo/help-gnu-emacs* per maggiori informazioni.

La sola mailing list dedicata a Emacs che io conosco per ora è la NT-Emacs list. È una lista per coloro che usano la versione per Micro$oft Windows di Emacs. Vedere le FAQ di NT-Emacs *http://www.cs.washington.edu/homes/voelker/ntemacs.html* per maggiori informazioni.

6.5 L'Emacs Lisp Archive

Dal README dell'Emacs Lisp Archive:

> Gli archivi Emacs Lisp su ftp.cis.ohio-state.edu contengono vari pezzi e pacchetti di codice Emacs Lisp. Emacs Lisp è il linguaggio usato per estendere l'editor GNU Emacs pubblicato dalla Free Software Foundation. Nonostante molto codice Emacs Lisp sia incluso nella distribuzione GNU Emacs, molte persone hanno scritto dei pacchetti per interfacciarsi con altri sistemi, per supportare meglio l'editing del linguaggio di programmazione che loro usano, per aggiungere nuove funzionalità, o per cambiare l'ambiente di default di Emacs. La maggior parte del contenuto di questo archivio è stato scritto da singole persone e pubblicamente distribuito su Internet attraverso le mailing list info-emacs o info-gnu-emacs o i newsgroup comp.emacs, gnu.emacs, o gnu.emacs.sources.

Gli archivi sono disponibili tramite FTP anonimo da *ftp://ftp.cis.ohio-state.edu/pub/emacs-lisp/* .

NOTA: Per quello che posso dire, l'Emacs Lisp Archive sta lentamente diventando un po' datato. Vedo che vi appaiono molto pochi package nuovi (o aggiornamenti), sebbene sappia che ne esistono. Questi vengono *in realtà* inviati al newsgroup `comp.emacs.sources` (correggetemi pure se questo è sbagliato).

7 Crediti

Le seguenti persone hanno contribuito al successo di questo documento.

- Craig Lyons *Craig.Lyons@compaq.com*
- Robert Vollmert *rvollmer@gmx.net*
- Larry Brasfield *larrybr@seanet.com*
- Etienne Grossmann *etienne@anonimo.isr.ist.utl.pt*
- Thomas Weinell *kf6mli@amsat.org*
- Adam C. Finnefrock *adam@bigbro.biophys.cornell.edu*
- Chris Gray *cgray4@po-box.mcgill.ca*
- Robert J. Chassell *bob@rattlesnake.com*

- Isaac To *kkto@csis.hku.hk*
- Matteo Valsasna *valsasna@elet.polimi.it*
- Tijs van Bakel *smoke@casema.net*