FIM : Fbi IMproved

dezperado _ presso _ autistici _ punto _ org

28,29,30 Settembre 2007

Un *image viewer* nello spirito di Vim

`Hackit 0x0A` (Hackmeeting 2007), Pisa

svn export `http://code.autistici.org/svn/fim`

Liberatoria

Presentazione e slides **estremamente** noiose e non interessanti per chi non si ritrova almeno in una tra queste categorie :

- utenti tranquilli senza X
- proseliti di `vim/emacs/nvi` (si, **escludo** gli *apologeti* di `joe` e `nano`)
- zeloti di `mutt`
- entusiasti dello *scripting*

Premesse

- filosofia UNIX : offrire *meccanismi*, non *politiche (ingl. policies)*

Premesse

- filosofia UNIX : offrire *meccanismi*, non *politiche (ingl. policies)*
- le possibilità dello shell scripting rendono *esponenziale* il potenziale dei nostri programmi UNIX (almeno quelli *versatili*)

Premesse

- filosofia UNIX : offrire *meccanismi*, non *politiche (ingl. policies)*
- le possibilità dello shell scripting rendono *esponenziale* il potenziale dei nostri programmi UNIX (almeno quelli *versatili*)
- ci sono *power editors* ... perchè no un *power image viewer* per il framebuffer di Linux?

Premesse

- filosofia UNIX : offrire *meccanismi*, non *politiche (ingl. policies)*
- le possibilità dello shell scripting rendono *esponenziale* il potenziale dei nostri programmi UNIX (almeno quelli *versatili*)
- ci sono *power editors* ... perchè no un *power image viewer* per il framebuffer di Linux?
- idea di un viewer *modale*, *scriptabile*, integrabile come utility

Premesse

- filosofia UNIX : offrire *meccanismi*, non *politiche (ingl. policies)*
- le possibilità dello shell scripting rendono *esponenziale* il potenziale dei nostri programmi UNIX (almeno quelli *versatili*)
- ci sono *power editors* ... perchè no un *power image viewer* per il framebuffer di Linux?
- idea di un viewer *modale*, *scriptabile*, integrabile come utility
- il nome ? `fbi` è un programma di Gerd Knorr, e `fim` cerca di migliorarlo

Installazione da tarball

Si prende da web (chi ha Gentoo Linux veda README)

- `wget http://www.autistici.org/dezperado/fim-0.1-alpha.tar.gz`
- `tar xzf fim-0.1-alpha.tar.gz`
- `less README; less INSTALL ; ls doc ; less BUGS`
- `./configure --help`
- `./configure`
- `make`
- `su -c 'make install'`

Installazione da repository svn (Subversion)

è necessario avere automake,autoconf,... installati

- svn export http://code.autistici.org/svn/fim
- less README; less INSTALL ; ls doc ; less BUGS
- ./autogen.sh
- ./configure --help
- ./configure
- make
- su -c 'make install'

Feature base usabili in maniera *intuitiva*

- chi ha usato almeno una volta `fbi`, non può avere problemi con `fim`
- `fim` ha una *command line* ma ...
- ... non c'è necessità di usarla forzatamente!

Feature *intuibili* in modo intuitivo

- le frecce ($\rightarrow, \leftarrow, \uparrow, \downarrow$) fanno cio' che ci si aspetta in un *image viewer*

Feature *intuibili* in modo intuitivo

- le frecce ($\rightarrow, \leftarrow, \uparrow, \downarrow$) fanno cio' che ci si aspetta in un *image viewer*
- idem per i tasti `pageup,pagedown,+,-`

Feature *intuibili* in modo intuitivo

- le frecce ($\rightarrow, \leftarrow, \uparrow, \downarrow$) fanno cio' che ci si aspetta in un *image viewer*
- idem per i tasti `pageup,pagedown,+,-`
- i tasti `h,j,k,l,' '` ciò che ci si aspetterebbe in un `vi`

Feature *intuibili* in modo intuitivo

- le frecce ($\rightarrow, \leftarrow, \uparrow, \downarrow$) fanno cio' che ci si aspetta in un *image viewer*
- idem per i tasti `pageup,pagedown,+,-`
- i tasti `h,j,k,l,' '` ciò che ci si aspetterebbe in un `vi`
- tasto `q` per uscire

Feature *intuibili* in modo intuitivo

- le frecce ($\rightarrow, \leftarrow, \uparrow, \downarrow$) fanno cio' che ci si aspetta in un *image viewer*
- idem per i tasti `pageup,pagedown,+,-`
- i tasti `h,j,k,l,' '` ciò che ci si aspetterebbe in un `vi`
- tasto `q` per uscire
- tasto `r` per ruotare, `m` per specchiare, `f` per capovolgere

Feature *curiose* in modo intuitivo

- ...e le prime curiosità:
- tasti `C+,C-` (`C` sta per `'CTRL'` per influenzare il *fattore* di ingrandimento

Feature *curiose* in modo intuitivo

- ...e le prime curiosità:
- tasti `C+,C-` (`C` sta per `'CTRL'` per influenzare il *fattore* di ingrandimento
- tasto `.` per *ripetere* l'ultima operazione

Feature *curiose* in modo intuitivo

- ...e le prime curiosità:
- tasti `C+,C-` (`C` sta per `'CTRL'` per influenzare il *fattore* di ingrandimento
- tasto `.` per *ripetere* l'ultima operazione
- tasto `Cm` per *segnalare* il filename corrente su `stdout` (*standard output*) a fine esecuzione

Feature *curiose* in modo intuitivo

- ...e le prime curiosità:
- tasti `C+,C-` (`C` sta per `'CTRL'` per influenzare il *fattore* di ingrandimento
- tasto `.` per *ripetere* l'ultima operazione
- tasto `Cm` per *segnalare* il filename corrente su `stdout` (*standard output*) a fine esecuzione
- tasti `N/P` per *avanzare/arretrare* di **diversi** file in lista

Feature *curiose* in modo intuitivo

- ...e le prime curiosità:
- tasti `C+,C-` (`C` sta per `'CTRL'` per influenzare il *fattore* di ingrandimento
- tasto `.` per *ripetere* l'ultima operazione
- tasto `Cm` per *segnalare* il filename corrente su `stdout` (*standard output*) a fine esecuzione
- tasti `N/P` per *avanzare/arretrare* di **diversi** file in lista
- tasti `d,D,X,x` per spostare l'immagine *diagonalmente*..

Uso *command line oriented*

Le azioni di prima possono essere eseguite anche in *command line* (premendo `:` per entrare in modalità comandi)

- `:help` tanto per iniziare (*)
- `:next` porta alla prossima immagine

((*) in `doc/FIM.TXT` la documentazione piena)

Uso *command line oriented*

Le azioni di prima possono essere eseguite anche in *command line* (premendo `:` per entrare in modalità comandi)

- `:help` tanto per iniziare (*)
- `:next` porta alla prossima immagine
- `:2prev` porta indietro di due immagini

((*) in `doc/FIM.TXT` la documentazione piena)

Uso *command line oriented*

Le azioni di prima possono essere eseguite anche in *command line* (premendo `:` per entrare in modalità comandi)

- `:help` tanto per iniziare (*)
- `:next` porta alla prossima immagine
- `:2prev` porta indietro di due immagini
- `:4magnify` esegue '4 volte' l'ingrandimento

((*) in `doc/FIM.TXT` la documentazione piena)

Uso *command line oriented*

Le azioni di prima possono essere eseguite anche in *command line* (premendo `:` per entrare in modalità comandi)

- `:help` tanto per iniziare (*)
- `:next` porta alla prossima immagine
- `:2prev` porta indietro di due immagini
- `:4magnify` esegue '4 volte' l'ingrandimento
- `:next;magnify` apre la prossima immagine e ingrandisce

((*) in `doc/FIM.TXT` la documentazione piena)

comandi vim like

Alcuni 'comandi' scattano con una sintassi *ad hoc*:

- `:2` porta alla seconda immagine
- `:$` porta all'ultima immagine
- `:*2` ingrandisce due volte rispetto ad ora
- `:200%` ingrandisce due volte rispetto all'originale
- `:-20%` diminuisce del 20% la dimensione della immagine corrente

Il comando bind

`:bind ( {tasto} | {sequenza tasti} ) {azione}`

Le *azioni* devono essere istruzioni valide in `fim` (come documentato in `doc/FIM.TXT`)

- `:bind 'n' 'next;'`
- `:bind 'N' '10next;'`
- `:bind 'C-N' ''`
- `:bind 'q' 'quit;'`

Ricerca *regexp-based*

Abbiamo selezionato mille immagini ma ci interessano solo quelle con nomi particolari ?

`/ { espressione regolare GNU }`

- `/[rR]enoir.*jpg`
- `C-n` per la prossima immagine il cui nome *corrisponde* all'espressione
- vedi (`man regex`)

Integrazione con mutt (mailcap)

nel file $HOME/.mailcap:

```
image/*;( [ "$DISPLAY" != "" ] && kuickshow %s ) ||
        ( ( ( tty | grep tty ) && fim %s ) || cacaview %s )
image/gif;( [ "$DISPLAY" != "" ] && kuickshow %s ) || fim  %s
application/pdf;( [ "$DISPLAY" != "" ] && acroread %s ) ||
        fimgs %s
application/ps;( [ "$DISPLAY" != "" ] && gv %s ) || fimgs %s
application/postscript;( [ "$DISPLAY" != "" ] && gv %s ) ||
        fimgs %s
application/x-dvi;( [ "$DISPLAY" != "" ] && xdvi %s ) ||
        fimgs %s
```

Integrazione con mc (midnight commander)

nel file $HOME/.mc/bindings:

```
include/image
        Open=if [ "$DISPLAY" = "" ]; then fim %f;
                 else (kuickshow %f &); fi
        View=%view{ascii} identify %f

type/^PDF
        Open=if [ "$DISPLAY" = "" ]; then fimgs %f;
                 else (acroread %f &); fi
        View=%view{ascii} pdftotext %f -
```

altrettanto sipuò fare per links,elinks...

Uso da shell

- `$ find /mnt/media -name '*jpg' | fim -`
- `$ find /mnt/media -name '*jpg' | grep -i mountain | fim -`

Script nel linguaggio proprio di fim

il file doc/FIM.TXT documenta adeguatamente sintassi e semantica del linguaggio di fim.
quindi è possibile anche scrivere programmini e trattarli come programmi *imperativi*.

- `:source 'script.fim'`

... oppure (vedi dopo) scrivere *file di configurazione* ad hoc

Costrutto if

- `:alias 'bignext' 'mini=100;if( width<mini || height < mini )next;'`
- `:bind "N" "bignext";`

Costrutto `while`

- `:alias "endless_slideshow" "while(1){next;display;sleep '1';};";`

Costrutto while

- `:alias "endless_slideshow" "while(1){next;display;sleep '1';};";`
- `:alias "pornview" "echo 'press any key repeatedly to terminate' ;endless_slideshow;";`
- `:bind "C-p" "pornview;";`

Il file $HOME/.fimrc

- I comandi di configurazione in $HOME/.fimrc verranno eseguiti prima di qualunque azione utente

attenzione: attualmente non è prevista una sandbox!
(a differenza di vim, ad esempio)

No, non quelle di Gates

```
:window ( "split" | "vsplit" | "close" | "upper" |
"lower" | "left" | "right" | "swap" )
```

- come in `vim`, è possibile avere aperte diverse finestre interne con (eventualmente) lo stesso file
- ma differentemente da `vim`, è possibile avere aperte diverse *istanze* dello stesso file (`fim` è un viewer, non un editor)
- è quindi possibile spostarsi tra le finestre, e suddividerle orizzontalmente o verticalmente, a volontà

Il file `$HOME/.inputrc` (`GNU readline` library

)

- Questo file viene usato da `bash`,`octave`,`gdb`,`...` per impostare il comportamento della riga comandi
- ... anche da `fim` !
- è possibile quindi usare nella *command line* di `fim` gli stessi shortcut per copia/incolla/... che in `bash` (`Cw`,`Cy`,`Ck`,`...`)

Autocomandi : Il comando `autocmd`

`:autocmd {evento} {pattern} {azione}`

- ... `vim` like (vedi `:help au` in `vim`!)
- `:autocmd "PostNext" "" "reload;";`
- `:autocmd "PostReload" "" "{redisplay;}";`

Autocomandi : Il cuore di fim

`:autocmd {evento} {pattern} {azione}`

- in realtà gran parte del comportamento di `fim` è *programmato* con gli autocommands.
- in un file di configurazione incorporato durante la compilazione
- vedere il file `src/fimrc` distribuito con `fim` o eseguire `$ fim --dump-default-fimrc`

Lo script fimgs

Uno script bash (a base di fbgs di Gerd Knorr) per *renderizzare* documenti o visualizzare il contenuto di archivi tramite fim:

- .pdf, .ps, .dvi (renderizzati in .png con gs)
- .tar.gz, .tgz, .bz2 (trattati come archivi di immagini, senza ricorsione)
- .rar, .cbr (trattati come archivi .rar di immagini, senza ricorsione)
- .zip, .cbz (trattati come archivi .zip di immagini, senza ricorsione)
- http:// cioe' qualunque dei precedenti, wget'tato dal web

Uso combinato con altri programmi

- raffinare 'ricerche tra immagini' usando `:mark` (`Enter`)
- `$ find /mnt/media -name '*jpg' | fim - | fim - | fim - > preferiti.txt`
- `$ fim ‘fim *‘`
- `$ find ./* -name '*.jpg' | fim - | xargs -I '{}' convert '{}' -resize 320x240 thumb_'{}'`
- `$ find * -name '*.jpg' | fim - | tar czf nicer.tgz --files-from -`

Comandi *runtime*

una sintassi simile a quella di `vim`

- `$ fim -c '{pre_comandi}'`
- per specificare dei comandi da eseguire *prima* dell'interazione

Comandi *runtime*

una sintassi simile a quella di `vim`

- `$ fim -c '{pre_comandi}'`
- per specificare dei comandi da eseguire *prima* dell'interazione
- `$ fim -F '{post_comandi}'`
- per specificare dei comandi da eseguire *dopo* l'interazione (immediatamente prima dell'interazione)
- es.: `$ alias 'fimevil' 'fim -F "autocmd PreNext system rm filename"'` (con gli escape giusti però!)

Uso *avventuroso*

A volte avvertiamo uno strano prurito ...

- I binding alle omonime funzioni C: `:pipe,:system,:return,...`
- interazione con sistema : `:bind 'C-g' "system 'fbgrab fim.png'";`
- `:alias "plisten" 'popen "nc -l -p 9999 "';`

Che programmazione sarebbe senza bugs ?

- `:system` coniugato con filename contenenti $ ...
- MAI premere `Cz` (`CTRL-Z`) in `fim` o `fbi` (vedi `BUGS`)!
- il file `BUGS` riporta i bug correnti man mano che capitano e vengono risolti
- lo script `fimgs` va ancora *sanitizzato* per bene, in quanto è uno script shell che invoca `wget`
- lo stesso `fbi` invoca `convert` (quello di Imagemagick) internamente
- ...

bug reporting e suggerimenti sono ben accetti!

Fim e' una patch applicata a Fbi

È stato programmato in C/C++ con STL, integrando `flex` (`lex`) e `bison` (`yacc`) e la libreria GNU/readline al codice sorgente originale di `fbi`.
Architettura un pò più ordinata che in Fbi, orientata (a grandi linee) verso 'Model-View-Controller'.

`fim` *non* è per tutti!

...ma certamente ha la sua nicchia d'utilizzo!

- gusti estetici personali o pigrizia nell'avvio di X per una misera immagine da vedere
- ergonomia particolare (quando l'uso del mouse è *irritante* o addirittura *dannoso*)
- non disponibilità di X
- automazione di visualizzazione di grandi quantità di immagini in modo *semi supervisionato*

\ Program :	kuickshow	eog	[xz]gv	fbi	FIM	GQview
Features						
status:(a)lpha/(m)ature	m	m	m	a	m	m
environment(f=framebuffer)	X	X	X,svga	f	f	X
linux specific			?	*	*	
key rebindings	*				*	*
external scriptability(bash,..)				~	*	
internal scriptability					*	
internal scriptability minilanguage					*	
printing support	*	*	?			*
slideshow	*	*		*	*	*
caching	*	?			*	*
preview browser	*					*
EXIF tag display	*			*		*
internal windowing					*	

Links utili, sito e documentazione

- `dict vi`
- `dict vim`
- `man readline`
- `man fbi`
- `man fim`
- `svn export http://code.autistici.org/svn/fim`
- http://www.autistici.org/dezperado/fim/FIM.html
- http://code.autistici.org/svn/fim/doc/FIM.TXT
- http://code.autistici.org/svn/fim/README
- http://code.autistici.org/svn/fim/INSTALL