L'HOWTO sul sistema a finestre MGR

Vincent Broman

Bozza, 30 Maggio 1996

Contents

1 Questo HOWTO

```
Per motivi di origine legale, io Francesco Tapparo (cesco@goldnet.it),
traduttore di questo HOWTO preferisco lasciare non tradotto il copyright
originale dell'autore:
```

1.1 Dove trovare questo HOWTO

Questo HOWTO si può reperire in `ftp://archimedes.nosc.mil/pub/Mgr/MGR-HOWTO.sgml`, oltre a trovarsi in `ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/MGR-HOWTO`. In directory vicine potrebbero comparire gli stessi documenti in formati alternativi, come `MGR-HOWTO.txt`.

1.2 Autenticazione

Le copie della distribuzione di MGR dovuta a Broman dovrebbero essere accompagnate da file di firma PGP, firmati da "Vincent Broman <broman@nosc.mil>".

1.3 Ringraziamenti per l'HOWTO

Nonostante Vincent Broman abbia per primo composto questo HOWTO, molte delle informazioni e del testo è stato ottenuto da FAQ, README, etc. scritti da Stephen Uhler, Michael Haardt, e altri generosi utenti della Rete. Correzioni Email e suggerimenti di modifiche devono essere spedite a `broman@nosc.mil`.

Uhler è stato il principale artefice di **MGR** – vedi la sezione Ringraziamenti più avanti.

2 Cos'e l'MGR Window System?

2.1 Funzionalità

MGR (ManaGeR) è un sistema grafico a finestre. Il server **MGR** fornisce nativamente un window manager e l'emulazione di terminali grafici a finestra su schermi a pixel, siano essi a colori o monocromatici. **MGR** è controllato selezionando dei menu a scomparsa con il mouse, interagendo con la tastiera, e attraverso sequenze di escape scritte su pseudo-terminali dal software client.

MGR fornisce ogni client di: funzioni di controllo terminale alla termcap; primitive grafiche come disegno di linee e cerchi; utility per manipolare bitmap, font, icone, e menu a scomparsa; comandi per il ridisegno e il posizionamento delle finestre; e una facility di inoltro messaggi che abilita i client a ritornare e scambiare messaggi. I programmi possono chiedere di venire informati quando avviene un cambiamento nel window system, come la modifica delle dimensioni di una finestra, la pressione di un tasto del mouse, o l'arrivo di un messaggio da parte di un altro client. Questi cambiamenti vengono detti eventi. **MGR** notifica un evento a un client inviandogli una stringa di caratteri ASCII in un formato da questi specificato. Le applicazioni preesistenti si possono facilmente integrare nell'ambiente window senza modifiche poiché **MGR** imita la pressione di tasti in risposta a selezioni da menu definiti dall'utente o altri eventi.

2.2 Requisiti

Allo stato attuale, **MGR** gira su Linux, FreeBSD, sulle workstation Sun 3/4 con SunOS, e su Coherent. Varie versioni precedenti di **MGR** girano su Macintosh, Atari ST MiNT, Xenix, 386-Minix, DEC 3100, e 3b1 Unix-pc. Molti piccoli sistemi industriali real-time sotto OS9 o Lynx in Europa usano (un altra variante di) MGR per la loro interfaccia utente. L'interfaccia di programmazione è scritta in C e in ELisp, quantunque sia abbastanza facile supportare client scritti in altri linguaggi. **MGR** richiede molte meno risorse di X, o addirittura di gcc. Non ha la base utenti, il repertorio software o le librerie di alto livello di X o MS-Windows, ad esempio, ma è abbastanza elegante ed accessibile.

È stato detto che **MGR** sta ad X come Unix sta a Multics.

2.3 Come comparare MGR, X11 e 8.5?

MGR consiste di un server con window manager ed emulatore di terminale nativi, client che girano in questo emulatore usandolo per comunicare con il server. Non viene eseguito alcun multiplexing delle risorse.

X11 consiste di un server e di client, che usualmente si connettono al server usando un socket. Tutti i servizi visibili all'utente, come emulatori di terminale, window manager etc. sono forniti da client. Non viene eseguito alcun multiplexing delle risorse.

8.5, il sistema a finestre Plan 9, è un multiplexer di risorse, in quanto ogni processo in una finestra può accedere a `/dev/bitblt`, `/dev/mouse` e `/dev/kbd` nel proprio namespace. Questi sono multiplexati ai `/dev/bitblit`, `/dev/mouse` e `/dev/kbd` nel namespace di 8.5. Questo approccio permette di girare 8.5 in una finestra di 8.5, un design molto pulito. 8.5 inoltre ha un window manager e un emulatore di terminale integrati.

3 Installazione di MGR

L'ultima distribuzione dei sorgenti può essere prelevata tramite FTP da `ftp://bugs.nosc.mil/pub/Mgr/69` o tramite Mosaic da `http://archimedes.nosc.mil/Mgr/69`. Essa si può trovare anche presso `ftp://sunsite.unc.edu/pub/Linux/apps/MGR` ed i suoi mirrors. Versioni precedenti di questa distribuzione, curate da Haardt, si possono trovare in `tsx-11.mit.edu`, e probabilmente altrove. Versioni di **MGR** precedenti al port per Linux, di Uhler ed altri si trovavano in `ftp://bellcore.com/pub/mgr`, ma penso che ora non vi siano più. Io ho salvato una copia di ogni cosa riguardante **MGR** vista su Internet, ma non sono a conoscenza di nulla di una certa importanza che manchi da questa distribuzione per Linux e Sun. **MGR** è passato attraverso un bel po' di versioni e releases, ma la corrente versione *per Linux* è la 0.69. Essa passerà a 1.0 quando vi sarà del codice stabile per supportare la VGA a 256 colori sotto Linux (per più di una scheda grafica). I numeri di versione RCS sono saliti dal 4.3 della Bellcore al 4.13 attuale.

Per costruire questa distribuzione di **MGR** sono necessari i seguenti tools: m4 (GNU, o forse anche altri dotati dell'opzione -D), make (GNU, o forse anche altri che supportino gli include) e *roff per la documentazione. Ed anche sh e POSIX install. Le distribuzioni binarie spesso non vengono assemblate, cosicché è necessario un compilatore ANSI C, come per esempio gcc.

Una installazione Linux richiede Linux 0.99.10 o meglio (quello che io uso attualmente per il testing è la versione 1.2.13), una scheda grafica HGC, EGA, VGA o SVGA, e un mouse. Sono supportati i seguenti mouse: mouse seriale Microsoft, mouse seriale MouseSystems a 3 e 5 byte mouse seriale MMSeries, mouse seriale Logitech, mouse PS/2, oppure un bus mouse. Con le combinazioni Buckey (Meta) da tastiera, anche un sistema senza mouse potrebbe fare un po' di utile lavoro sotto **MGR**. Il modo grafico monocromatico VGA 640x480 è supportato senza bisogno di alcun intervento particolare, ed egualmente quello 640x350 e 640x200. Per supportare la modalità 800x600, o altre che il tuo BIOS può inizializzare e che non richiedono bank-switching, è necessario far girare sotto DOS, o un emulatore, un piccolo programma (`src/vgamisc/regs.exe`) per leggere i registri VGA e scrivere un file header da piazzarsi nella directory `src/libbitblit/linux`, cosicché possa venire `#incluso` dal file lì presente `vga.c`. Sono forniti anche degli esempi di tali file, ma per favore creati quelli specifici della tua particolare scheda. Alcune schede VGA possono usare modalità grafiche che richiedono 128k di RAM, e queste potrebbero supportare risoluzioni monocromatiche più elevate.

Il codice per la modalità a colori in Linux funziona senza problemi anche nella modalità policromatica VGA 320x200x256, poiché essa non richiede alcun bank-switching. Se pensi quanto pochi siano 64000 pixel, capirai che questa modalità è assai limitata. Nella versione 0.65 è stato aggiunto del codice per il bank-switching, non molto veloce, ma semplice: funziona con una scheda grafica Tseng ET4000 nelle modalità 640x480x256 e 800x600x256. Il codice per la S3 non funziona ancora nelle risoluzioni super VGA. Supportare nuove schede super VGA richiede la scrittura di una funzione per switchare i banchi, e quindi assicurare che la modalità desiderata possa venire inizializzata da un register dump, magari lavorandoci un po' sopra a mano. I servers a colori per Linux generalmente corrompono i font dello schermo, rendendo necessario l'uso di restorefont come in runx.

Sono supportati anche i Sun con SunOS 4.1.2+ e i frame buffer `bwtwo`, `cgthree`, o `cgsix`. La loro velocità lavorando nelle modalità a colori è buona. Le installazioni Coherent dovrebbero riferirsi al file `README.Coh` nella distribuzione dei sorgenti. Il porting della versione più recente di `MGR` ad un altro sistema POSIX-like, che supporti `select`, i pty e l'accesso diretto ad un frame-buffer a mappa di bit dovrebbe essere immediato, richiedendo solo l'implementazione della libreria `libbitblt` basandosi, per esempio, sul codice `sunmono` o `colorport`.

Per una installazione completa, occorrono 7 Mb di spazio su disco per i file binari, i font, le man-page etc. I file sorgente occupano circa 4.5 Mb, cui vanno aggiunti i file oggetto durante la compilazione.

Normalmente, la directory dove installare i file che **MGR** usa a runtime dovrebbe essere `/usr/mgr`, o almeno dovrebbe venire linkata ad essa. Scrivendo:

```
cd /usr/mgr; tar xvfz dovelohaimesso/mgrusr-0.69.tgz
```

e facoltativamente

```
cd /usr/mgr; tar xvfz dovelohaimesso/morefonts-0.69.tgz
```

si scomprimeranno questi file. I sorgenti possono venire installati ovunque, ad esempio scrivendo

```
cd /usr/src/local/mgr; tar xvfz dovelohaimesso/mgrsrc-06.9.tgz
```

per scomprimere i sorgenti prelevati da `archimedes.nosc.mil`.

L'albero dei sorgenti può venire compilato da un unico Makefile nella directory principale che chiama quelli nelle secondarie, ognuno dei quali "include" il file "`Configfile`" nella directory base. `Configfile` è creato dallo shell script interattivo `Configure`, che, tenendo conto delle tue risposte ad alcune sue domande, fa processare `Configfile.m4` ad m4. Quindi si deve scrivere qualcosa di simile a questo:

```
chdir /usr/src/local/mgr
sh ./Configure
make first
make depend
make install
make clean
```

Potrebbe essere saggio, prima di far partire make, dare uno sguardo al `Configfile` generato dallo script `Configure`, controllando che sembri ragionevole. (Come minimo un m4 (il `/usr/bin/m4` dei Sun) si ferma subito creando un `Configfile` cortissimo. Se accade questo, bisogna editare a mano una copia di `Configfile.sun` o `Configfile.lx`) Si può anche editare `make all` in ogni directory in cui sia stato creato un Makefile, non appena siano state compilate ed installate le librerie. Il server, le librerie, e alcuni client sono stati lintati, ma molti client sono scritti in K&R C, e fanno generare molti warning al compilatore.

Si possono aggiungere od omettere molti flag nella variabile MGRFGLAGS in Configfile per modificare alcune caratteristiche opzionali del server, ossia:

-DWHO

modifica il file utmp così da permettere il funzionamento di "who"

-DVI

inserisce del codice che permette di muovere il cursore con il mouse in vi

-DDEBUG

abilita la possibilità di selezionare con l'opzione -d un output di debug

-DFASTMOUSE

utilizza l'operazione XOR nel tracciamento del mouse

-DBUCKEY

abilita i comandi da tastiera, senza l'utilizzo del mouse

-DPRIORITY

usa uno scheduling diverso dal round-robin standard, dando una priorità più alta alla finestra attiva

-DCUT

abilita il supporto per il copia/incolla fra le finestre.

-DMGR_ALIGN

forza l'allineamento della finestra sul confine del byte, per una maggiore velocità di scrolling (solo nelle modalità monocromatiche)

-DKILL

uccide le finestre nel caso che la tty riporti un errore di i/o

-DSHRINK

usa solo parte dello schermo (pari al valore della variabile di ambiente $MGRSIZE)

-DNOSTACK

non permette lo stacking degli eventi

-DBELL

attiva il beep

-DKBD

nei Sun legge l'input di `MGR` dalla tastiera, invece che da stdin. Questo permette la redirezione a una finestra dei messaggi della console.

-DFRACCHAR

abilita il supporto per il movimento frazionale dei caratteri permettendo così l'uso di font proporzionali

-DXMENU

supporto esteso per i menu (sperimentale)

-DMOVIE

estensione per fare moviole: tutte le operazioni vengono salvate su di un file per un successivo replay – non del tutto funzionante su Linux

-DEMUMIDMSBUT

nei mouse a due tasti, emula il bottone di mezzo premendo insieme gli altri 2.

Non tutte le combinazioni di queste opzioni sono state testate su tutti i sistemi.

La macro BITBLITFLAGS deve contenere `-DBANKED` se si sta provando il supporto per la SVGA a colori.

Un convertitore genera il codice C per le variabili statiche contenenti le icone e i font dai file corrispondenti.

Non tutti i client vengono compilati ed installati dai Makefile. I client che si trovano sotto `src/clients` e che hanno i nomi capitalizzati, o che non vengono compilati dai Makefile forniti possono presentare problemi durante la compilazione o l'esecuzione, ma può essere interessante farci un po' di hacking. Molti dei driver video nella directory `libbitblit` sono di interesse soprattutto archeologico. Qualche volta rovistando nelle tombe può saltare fuori qualcosa di interessante.

A un certo punto controlla che i tuoi file `/etc/termcap` e/o `terminfo` contengano le entrate per i terminali **MGR** come quelli che si trovano nella directory `misc`. Se tutto il tuo software controlla $TERMCAP nell'ambiente , questo non è più necessario, finché fai partire `eval 'set_termcap'` in ogni finestra.

MGR funziona meglio se gira setuid root, poiché cerca di cambiare le proprietà dei pty e scrivere nel file utmp. Questo aiuta il gestore di icone ify a lavorare meglio e il meccanismo di trasmissione degli eventi ad essere più sicuro. Su Linux sono *necessarie* le permissioni del root per l'input/output sullo schermo. D'altra parte, tu decidi se fidarti o no.

Nelle versioni intorno alla 0.62 c'erano problemi sul Sun usando la csh come shell di default. I programmi sembravano girare in un gruppo di processi diverso da quello del gruppo di processi di primo piano del pty della finestra, in contrasto alle man-page e alle specifiche POSIX. Non c'era nessun problema con bash, sh o rc. Qualcuno ha un idea del perché ?

4 Usare MGR

Il solo file *richiesto* in una installazione **MGR** è il server stesso. Questo ti darà finestre ad emulatore di terminale con shell giranti in esse e la possibilità di effettuare il cut/paste con il mouse ma, nessun bell'orologio, nessun font extra, nessun capriccio grafico, etc. A seconda delle opzioni, un server monocromatico necessita di circa 200K di RAM, oltre allo spazio allocato dinamicamente per window, bitmap, etc.

Se `/usr/mgr/bin` è nel tuo PATH, digita semplicemente "`mgr`" per partire. Dopo esserti goduto l'animazione di avvio, premi un tasto. Quando compaiono lo sfondo tratteggiato e il puntatore del mouse, mantieni premuto il pulsante sinistro del mouse, seleziona la voce "new window" dal menu, e rilascia il pulsante. Quindi definisci la finestra trascinando i suoi angoli con il mouse. Nella finestra girerà la tua shell di default. Tieni premuto il pulsante sinistro del mouse sopra un finestra per vedere un altro menu con il quale eseguire azioni su quella finestra. Premendolo sopra una finestra parzialmente coperta, essa viene riportata in primo piano. Il menu che hai visto generarsi sullo sfondo vuoto comprende il comando quit. Per le persone con un mouse a due pulsanti: premete entrambi i pulsanti insieme per emulare il pulsante di mezzo mancante, usato da alcuni programmi.

Il sottomenu quit comprende l'opzione "really quit", un'opzione di sospensione che sarà usata solo se stai facendo girare una shell supportante il job-control, ed una opzione per uno screen-saver/locker che aspetta che tu batta la password quando ritorni alla tua macchina.

Se ,provando a far partire **MGR**, ottieni:

can't find the screen

accertati di avere un file `/dev` per il tuo display, per esempio `/dev/bwtwo0` su un Sun. Se non lo hai, va come root nella directory `/dev`, e digita "MAKEDEV bwtwo0". Altrimenti, potresti aver bisogno nella riga di comando dell'opzione `-S/dev/bwtwo0` o (su Linux) della opzione `-S640x480` quando fai partire `MGR`. Su Linux dovresti anche assicurarti che `/usr/mgr/bin/mgr` sia stato installato setuid root.

can't find the mouse

accertati che esista `/dev/mouse` , usualmente un link simbolico al vero file di periferica del tuo mouse. Se non hai le permissioni per scrivere in `/dev`, puoi dare un'opzione come `-m/dev/cua0` quando fai partire `MGR`. Inoltre sii sicuro di aver fornito il giusto protocollo del mouse quando hai configurato `MGR`. Il mouse può può utilizzare quello Microsoft, anche se questo non è il nome della marca.

can't get a pty

accertati che tutti i `/dev/[tp]ty[pq]?` siano posseduti da root, con modo 666, e che tutti i programmi indicati con l'opzione "shell" nel tuo file di avvio `.mgrc` (se c'è) esistano e siano eseguibili.

none but the default font

accertati che **MGR** stia cercando i font nella directory giusta. Controlla `Configfile` nei sorgenti o vedi se dare una opzione `-f/usr/mgr/font` a `mgr` risolve il problema.

completamente bloccato (non si muove nemmeno il cursore del mouse)

loginati alla tua macchina da un altro terminale (o usa rlogin) e uccidi `MGR`. Se la tastiera funziona ancora, un tasto Buckey-Q può farti uscire da **MGR**.

4.1 Applicazioni che supportano MGR in modo trasparente

Ogni applicazione che giri in modalità tty può essere fatta partire in una finestra **MGR** senza ulteriore pena. Le applicazioni che girano in modalità testo usando termcap o curses possono ottenere da sole il numero corretto di linee e colonne della finestra usando `shape(1)` per ridisegnare la finestra o `set_termcap(1)` per ottenere l'entrata termcap corretta.

4.2 Applicazioni MGR (client) distribuite con il server

bdftomgr

converte dei font .bdf in font MGR

browse

un browser di icone

bury

nasconde questa finestra

c_menu

costruisce menu dagli errori del compilatore C

clock

orologio digitale

clock2

orologio analogico

close

chiude la finestra e la iconifica

color

setta i colori di primo piano e di sfondo del testo nella finestra.

colormap

legge o modifica la palette

cursor

cambia l'aspetto del cursore

cut

ritaglia del testo dalla finestra e lo ricopia nel cut buffer

cycle

visualizza una serie di icone

dmgr

primitivo previewer di file ditroff

fade

fa sfumare uno script home movie da una scena all'altra

font

passa ad usare un nuovo font nella finestra

gropbm

un driver che traduce dal formato roff al formato PBM, usando i font Hershey.

hpmgr

emulatore di terminale hp 2621

ico

disegna un icosaedro, o altro poliedro, animato

iconmail

notifica l'arrivo della posta

iconmsgs

notifica l'arrivo dei messaggi

ify

iconifica e deiconifica le finestre

loadfont

carica un font dal disco

maze

un gioco di labirinto

mclock

orologio micky mouse

menu

crea o seleziona un menu a scomparsa

mgr

server e window manager del window system della Bellcore

mgrbd

il gioco boulder-dash

mgrbiff

guarda se è arrivata posta nella mailbox e la notifica

mgrload

disegna un grafico del load average del sistema

mgrlock

blocca la console

mgrlogin

manager grafico di login

mgrmag

ingrandisce una parte dello schermo, salvandolo su file se si vuole

mgrmail

notifica l'arrivo della posta

mgrmode

attiva o disattiva i modi delle finestre

mgrmsgs

notifica l'arrivo dei messaggi

mgrplot

filtro grafico per Unix "plot"

mgrsclock

clessidra

mgrshowfont

esamina i font MGR

mgrsketch

un programma per fare schizzi/disegni

mgrview

visore per i bitmap MGR

mless

fa partire less o more in una finestra separata, con un menu aggiunto nel caso di less.

mnew

fa partire un qualunque programma in un finestra separata e indipendente.

mphoon

visualizza la corrente fase lunare

mvi

fa partire vi in una finestra separata, con puntamento del mouse

oclose

(obsoleto) chiude una finestra

omgrmail

(obsoleto) notifica l'arrivo della posta

pbmrawtomgr, pgmrawtomgr, ppmrawtomgr

convertono i file di immagini in formato raw PBM, PGM, PPM.

pbmstream

legge un flusso di bitmap e fa partire un programma per ognuno di essi

pbmtoprt

produce output per stampante da un PBM

pgs

front-end per ghostscript (un visore postscript), con la patch necessaria per poterlo utilizzare.

pilot

un bitmap browser o visore di immagini.

resetwin

risistema lo stato di una finestra dopo che il crash di un client lo ha rovinato

rotate

ruota un bitmap di 90 gradi

screendump

scrive una copia dello schermo grafico ad un file bitmap

set_console

ridirige alla finestra i messaggi destinati alla console

set_termcap

visualizza un appropriato settaggio di TERM e TERMCAP per la finestra attuale

setname

da un nome ad una finestra, cosa utile per i messaggi e l'iconificazione

shape

cambia le dimensioni di una finestra

square

quadra questa finestra

squeeze

comprime un bitmap MGR usando l'algoritmo di compressione run-length encoding.

startup

produce lo scheletro di un file di startup che produca il corrente layout di finestra

texmgr

visore di file TeX dvi

text2font, font2text

per passare dal formato dei font per MGR al formato testo e viceversa

unsqueeze

decomprime un bitmap MGR compresso con l'algoritmo run-length encoding

vgafont2mgr, mgrfont2vga

convertono i font dal formato VGA al formato MGR, e viceversa.

window_print

stampa un'immagine di una finestra

zoom

un editor di icone

bounce, grav, grid, hilbert, mgreyes, stringart, walk

demo grafici

4.3 client predisposti per MGR distribuiti a parte, vedi il file "SUPPORT"

calctool

calcolatrice su schermo

chess

frontend a `/usr/games/chess`

gnu emacs

editor con `lisp/term/mgr.el` e supporto per mouse & menu

gnuplot

programma universale per la rappresentazione di dati scientifici

metafont

programma per il progetto e la creazione di font

origami

l'editor pieghevole

pbmplus

conversioni e manipolazioni per il formato pbm

plplot

programma per la veloce rappresentazione di dati scientifici

Il supporto Emacs in `misc/mgr.el` e `misc/mailcap` include un supporto MIME molto funzionale, attraverso Rmail e metamail.

Utilizzando `pilot` e i filtri netPBM, si potrebbe costruire un visore di immagini che sappia interpretare molti formati diversi, ma non ho avuto il tempo per farlo.

5 Programmazione in MGR

Il manuale del programmatore di **MGR**, con l'interfaccia in linguaggio C alle applicazioni, si trova nella directory doc in formato troof/nroff. Esso copre i concetti generali, le chiamate alle funzioni/macro che controllano il server, una semplice applicazione, un indice e un glossario.

Il porting di programma client usato con versioni precedenti di **MGR** richiede qualche volta la sostituzione di

```
#include <mgr/mgr.h>
```

con

```
#include <term.h>
#include <dump.h>
```

e i client che usano le vecchio stile b_XOR, B_CLEAR, e simili invece di BIT_XOR, BIT_CLR e simili possono venire sistemati scrivendo

```
#define OLDMGRBITOPS
#include <mgr/mgr.h>
```

Compilare codice per client richiede in genere opzioni di compilazione di questo tipo:

```
-I/usr/mgr/include   -L/usr/mgr/lib -lmgr
```

Si può fare un po' di esperienza con le funzioni del server **MGR** in modo interattivo leggendo e sperimentando con il driver di terminale per GNU Emacs `mgr.el` che implementa la libreria di interfaccia a **MGR** in ELisp.

Il metodo usuale di procurarsi lo stato dal server può portare a dei problemi se il client si aspetta anche un grande volume di notifiche di eventi. Il problema sorge se una notifica (asincrona) arriva quando si aspetta la risposta ad una richiesta (sincrona). Se questo accade effettivamente (cosa inusuale) le funzioni di **MGR** per procurarsi lo stato dovranno essere integrate con il tuo loop per la gestione degli eventi.

Il solo servizio fondamentale di disegno mancante nel protocollo **MGR** sembra essere la colorazione di un area per aree diverse da rettangoli orizzontali. C'è del nuovo codice per la manipolazione della mappa globale dei colori, come per l'allocazione (suggerita) e deallocazione degli indici dei colori posseduti dalle finestre.

Se pensi di effettuare dell'hacking su server, puoi trovare il driver del mouse in `mouse.*` e `mouse_get.*`, le parti grotty dell'interfaccia alla tastiera in `kbd.c`, e l'interfaccia al display nelle directory `src/libbitblit/*`. La procedura main, la maggior parte delle inizializzazioni, e la parte di alto livello del loop di input si trovano in `mgr.c`, mentre l'interpretazione delle sequenze di escape è in `put_window.c`.

6 Documentazione ulteriore

Il manuale del programmatore è indispensabile per i concetti generali.

Quasi tutti i client forniti sono dotati di una man page installata in `/usr/mgr/man/man1` o `man6`. Altre utili man page sono `bitblit.3`, `font.5`, e `bitmap.5`. C'è un po' di ambiguità nella documentazione nel distinguere il formato bitmap interno nel tuo frame-buffer e il formato bitmap nei file, come per esempio le icone.

La man page `mgr.1` copre le opzioni nella riga di comando, i comandi nel file di startup `~/.mgrc`, l'interazione del mouse e dei menu con il server, e i comandi da tastiera disponibili sui sistemi con i tasti necessari.

Molti dei font in `/usr/mgr/font/*` sono descritti almeno parzialmente in `/usr/mgr/font/*.txt`, ad esempio `/usr/mgr/font/FONTDIR.txt` da una descrizione analoga a quella fornita per X-Windows per i font in formato .bdf. I nomi dei font finiscono in `WxH`, dove `W` e `H` sono la larghezza e l'altezza in pixel(espresse in base 10) di ogni carattere.

7 Ringraziamenti per MGR

Stephen Uhler, con altri che lavoravano alla Bellcore, è stato l'originale designer ed implementatore di **MGR**, cosicché la Bellcore ha protetto da copyright molto del codice e della documentazione di **MGR** ponendolo sotto i seguenti termini d'uso.

`( N.d.T.: per motivi legali ho deciso di non tradurre il seguente copyright )`

```
* Permission is granted to copy or use this program, EXCEPT that it
* may not be sold for profit, the copyright notice must be reproduced
* on copies, and credit should be given to Bellcore where it is due.
```

Si richiede di mostrare l'avviso di copyright nello schermo di startup.

Altri ringraziamenti a:

- Stephen Hawley per le sue stupende icone.
- Tommy Frandsen per la libreria VGA per Linux.
- Tom Heller per la sua libreria Gasblit.
- Andrew Haylett per il driver del mouse.
- Dan McCrackin per le sue patch Gasblit->Linux.
- Dave Gymer, dgymer@gdcarc.co.uk, per il fix dell'effetto Startrek.
- Alex Liu per aver rilasciato per primo una versione funzionante di **MGR** per Linux.
- Lars Aronsson (aronsson@lysator.liu.se) per text2font e un font ISO8859-1 a 8 bit.
- Harry Pulley (hcpiv@grumpy.cis.uoguelph.ca, hcpiv@snowhite.cis.uoguelph.ca) per il port a Coherent.
- Vance Petree & Grant Edwards & Udo Munk per il loro lavoro sulla Hercules.
- Udo Munk per il suo lavoro sulla inizializzazione del mouse seriale & select.
- Norman Bartek & Hal Snyder e Mark Williams Co. per il loro aiuto con alcuni bug & con i device driver per Coherent.
- Ringraziamenti ulteriori a Zeyd Ben Halim per le molte utilissime patch, in particolare l'adattamento della selezione.
- a Bradley Bosch, brad@lachman.com, per molte patch del suo port per 3b1, che fissano errori ed implementano nuove e desiderabili caratteristiche.
- Andrew Morton, applix@runxtsa.runx.oz.au, che ha per primo scritto il codice cut-word.
- Kapil Paranjape, kapil@motive.math.tifr.res.in, per il supporto alla EGA.
- Michael Haardt per i fix al supporto MOVIE, le correzioni dei bug, la separazione del codice della libbitblt nei driver di output, l'espansione della libmgr, e la formattazione del codice per la navigazione con origami.
- a Yossi Gil per i molti font.
- a Carsten Emde, carsten@thlmak.pr.net.ch, per mphoon.

- a Vincent Broman per l'emulazione del bottone di mezzo del mouse, l'opera di linting, il supporto per il cgsix Sun, l'accesso alla tavola dei colori VGA, l'integrazione del codice per il port a Sun nella struttura software a strati disegnata da Haardt, la raccolta dei font, il salva schermo, e la manutenzione continua.

Tutti i font bitmap di qualunque provenienza sono rigorosamente di pubblico dominio negli USA. I 583 font a larghezza fissa forniti con **MGR** sono stati resi disponibili grazie a Uhler, la distribuzione X, YossiGil, e altri. I font vettoriali Hershey e il codice per disegnarli sono probabilmente liberamente ridistribuibili.