The MGR Window System HOWTO

Vincent Broman Draft 8 April 1994

Contents

1 This HOWTO

1.1 Archiving

This HOWTO is temporarily archived for anonymous FTP on `bugs.nosc.mil` in `pub/Mgr/MGR-HOWTO`, and more permanently on `sunsite.unc.edu` in `pub/Linux/docs/HOWTO/MGR-HOWTO`.

1.2 Credit for the HOWTO

While Vincent Broman first put together this HOWTO, much of the information and text was obtained from FAQs, READMEs, etc. written by Stephen Uhler, Michael Haardt, and other public-spirited net-persons. Email corrections and suggested changes to `broman@nosc.mil`.

Uhler was the main architect of **MGR** – see the Credit section below.

2 What is the MGR window system?

2.1 Function

MGR (ManaGeR) is a graphical window system. The **MGR** server provides a builtin window manager and windowed graphics terminal emulation on color and monochrome bitmap displays. **MGR** is controlled by mousing pop-up menus, by keyboard interaction, and by escape sequences written on pseudo-terminals by client software.

MGR provides each client window with: termcap-style terminal control functions, graphics primitives such as line and circle drawing; facilities for manipulating bitmaps, fonts, icons, and pop-up menus; commands to reshape and position windows; and a message passing facility enabling client programs to rendezvous and exchange messages. Client programs may ask to be informed when a change in the window system occurs, such as a reshaped window, a pushed mouse button, or a message sent from another client program. These changes are called events. **MGR** notifies a client program of an event by sending it an ASCII character string in a format specified by the client program. Existing applications can be integrated into the windowing environment without modification by having **MGR** imitate keystrokes in response to user defined menu selections or other events.

2.2 Requirements

MGR currently runs on Linux, Sun 3/4 workstations with SunOS, and Coherent. Various older versions of **MGR** run on the Macintosh, Atari ST MiNT, Xenix, 386-Minix, DEC 3100, and the 3b1 Unix-pc. The programming interface is implemented in C and in ELisp, although supporting clients written in other languages is quite easy.

Running **MGR** requires much less in resources than X, or even gcc. It does not have the user-base, software repertory, or high-level libraries of X or MS-Windows, say, but it is quite elegant and approachable.

It has been said that **MGR** is to X as Unix was to Multics.

2.3 How do MGR, X11 and 8.5 compare?

MGR consists of a server with builtin window manager and terminal emulator, and clients which run in this terminal emulator and use it to communicate with the server. No resource multiplexing is done.

X11 consists of a server and clients, which usually connect to the server using a socket. All user visible things like terminal emulators, window managers etc are done using clients. No resource multiplexing is done.

8.5, the Plan 9 window system, is a resource multiplexer, as each process running in a window can access `/dev/bitblt`, `/dev/mouse` and `/dev/kbd` in its own namespace. These are multiplexed to the `/dev/bitblit`, `/dev/mouse` and `/dev/kbd` in the namespace of 8.5. This approach allows one to run 8.5 in an 8.5 window, a very clean design. 8.5 further has an integrated window manager and terminal emulator.

3 Installing MGR

The latest version can be FTPed from `bugs.nosc.mil:pub/Mgr/62`. You can get older **MGR** sources from `ftp.thp.uni-koeln.de` 134.95.64.1 in `pub/linux/mgr`, or on 134.95.80.1 in `pub/thp/linux/mgr`, if you can't reach the first site. Even older versions of this distribution from Haardt can be found on `tsx-11.mit.edu` and elsewhere. Pre-Linux versions of **MGR** from Uhler and others (prior to Haardt) can be found at `bellcore.com:pub/mgr`, although no one seems to maintain things there. **MGR** has been through a lot of versions and releases, but the current *Linux* version number is 0.62. This version number ought to arrive at 1.0 when 256-color VGA code for Linux appears.

Required tools to build this distribution of **MGR** are m4 (GNU, or perhaps another supporting the -D option), make (GNU, or perhaps another supporting include) and *roff for the docs. Also sh, awk, and POSIX install. Binary distributions have not been assembled yet, so you need an ANSI C compiler environment, e.g. gcc + gas.

A Linux installation requires Linux 0.99.10 or better, an HGC, EGA, VGA, or SVGA graphics card, and a mouse. Mouses supported are: serial Microsoft mouse, serial MouseSystems 3 and 5 byte mouse, serial MMSeries mouse, serial Logitech mouse, PS/2 mouse, or a bus mouse. The VGA 640x480 monochrome graphics mode is supported out of the box, as is 640x350 and 640x200. To run 800x600, or other modes that your BIOS can initialize and which do not require bank-switching, you need to run a small program (supplied as `src/vgamisc/regs.exe`) under DOS to read the VGA registers while that mode is set and write a header file which you place in the directory `src/libbitblit/linux`. Some VGA cards can use 128k windows, and these can run higher resolutions. The Linux code does not yet support more than two colors.

Suns with SunOS 4.1.2 and `bwtwo`, `cgthree`, or `cgsix` frame buffers are supported. Coherent installations should refer to the `README.Coh` file in the source distribution. Porting the latest-and-greatest **MGR** to another POSIX-like system which provides `select()` and pty's and direct access to a bitmapped frame-buffer ought to be straightforward, just implementing the `libbitblit` library based on the `sunmono` or `colorport` code, say.

If you want to install everything, you need 4.8 MB disk space for binaries, fonts, manual pages etc. The sources are about 2 MB, plus object files during compilation.

Normally, `/usr/mgr` should be either the directory or a link to the directory where you install **MGR** stuff for runtime use. Typing

```
chdir /usr/mgr; gunzip < whereveryouputit/mgrusr.tgz | tar xvf -
```

and optionally

```
chdir /usr/mgr; gunzip < wherever/morefonts.tgz | tar xvf -
```

will unpack these. The source can be put anywhere, e.g. typing

```
chdir /usr/src/local/mgr; gunzip < wherever/mgrsrc.tgz | tar xvf -
```

to unpack the sources from `bugs.nosc.mil`.

The source tree can be compiled from one top-level Makefile which invokes lower-level Makefiles, all of which "include" a `"Configfile"` at the top level. The `Configfile` is created by an interactive sh script named `Configure`, which runs m4 on a `Configfile.m4`, so you do something like this:

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

It might be wise, before running make, to eyeball the `Configfile` generated by the `Configure` script, checking that it looks reasonable. (At least one m4 poops out, creating a very short `Configfile`. If this happens, try editing a copy of `Configfile.sun` or `Configfile.lx`) Several flags in MGRFLAGS can be added/omitted to change some optional features in the server, viz:

-DWHO

muck utmp file so "who" works

-DVI

code for clicking the mouse in vi moving the cursor

-DDEBUG

enable debugging output selectable with -d options.

-DFASTMOUSE

XOR the mouse track

-DBUCKEY

for hot-key server commands without mousing

-DPRIORITY

for priority window scheduling instead of round-robin; the active window gets higher priority

-DCUT

for cut/paste between windows and a global snarf buffer

-DALIGN

forces window alignment for fast scrolling (monochr)

-DKILL

kills windows upon tty i/o errors

-DSHRINK

use only some of the screen ($MGRSIZE in environment)

-DNOSTACK

don't permit event stacking

-DBELL

really ring the bell

-DKBD

read `mgr` input from the sun kbd, instead of stdin. This permits redirection of console msgs to a window.

-DFRACCHAR

fractional character movement for proportional fonts

-DXMENU

extended menu stuff (experimental)

-DMOVIE

movie making extension which logs all operations to a file for later replay – not quite working under Linux

-DEMUMIDMSBUT

Emulate a missing middle mouse button by chording

Not all combinations of these options work on all systems.

If a make complains about the lack of a `default_font.h` or an `icon_server.h` in the directory `src/mgr`, then just do a

```
make default_font.h icon_server.h
```

in that directory. C code for the static variables containing icons and fonts is generated by a translator from icon and font files.

Not all the clients are compiled and installed by the Makefiles. Clients found under `src/clients` having capitalized names or not compiled by the supplied Makefiles may have problems compiling and/or running, but they may be interesting to hack on. Several screen drivers found under the `libbitblit` directory are of mainly archeological interest. Grave robbing can be profitable.

At some point check that your `/etc/termcap` and/or `terminfo` file contain entries for **MGR** terminals such as found in the `misc` directory. If all your software checks $TERMCAP in the environment, this is not needed, as long as you run `set_termcap` in each window.

MGR works better if run setuid root, because it wants to chown ptys and write in the utmp file. This helps the ify iconifier client work better and the event passing mechanism be more secure. On Linux, root permissions are required in order to do in/out on the screen device. Otherwise, you decide whether to trust it.

In versions around 0.62 there are troubles on the Sun with using the csh as the default shell. Programs seem to run in a different process group than the foreground process group of the window's pty. There is no trouble with bash, sh, or rc. Ideas why?

4 Running MGR

The only file *required* in an **MGR** installation is the server itself. That would give you terminal emulator windows with shells running in them, but no nice clocks, extra fonts, fancy graphics, etc. Depending on options, the server needs about 200K of RAM plus dynamic space for windows, bitmaps, etc.

If `/usr/mgr/bin` is in your PATH, then just type `"mgr"` to start up. When the hatched background and mouse pointer appear, hold down the left mouse button, highlight the "new window" menu item, and release the button. Then drag the mouse from corner to corner where you want a window to appear. The window will have your default shell running in it. Hold down the left mouse button over an existing window to see another menu for doing things to that window. The menu you saw that pops-up over the empty background includes the quit command. For people with a two button mouse: press both buttons together to emulate the missing middle button.

When trying to run **MGR**, if you get:

can't find the screen

make sure you have a `/dev` entry for your display device, e.g. on a Sun `/dev/bwtwo0`. If not, as root cd to `/dev`, and type "MAKEDEV bwtwo0". Otherwise, you might need the `-S/dev/bwtwo0` or (on Linux) the `-S640x480` command line option when starting `mgr`.

can't find the mouse

make sure `/dev/mouse` exists, usually as a symbolic link to the real device name for your mouse. If you haven't permission to write in `/dev`, then something like a `-m/dev/cua0` option can be given when starting `mgr`. Also, make sure you've supplied the right mouse protocol choice when you configured `mgr`. The mouse may speak Microsoft, even if that is not the brand name.

can't get a pty

make sure all of `/dev/[tp]ty[pq]?` are owned by root, mode 666, and all programs referenced with the "shell" option in your `.mgrc` startup file (if any) exist and are executable.

none but the default font

make sure **MGR** is looking in the right place for its fonts. Check the `Configfile` in the source or see whether a `-f/usr/mgr/font` option to `mgr` fixes the problem.

completely hung (not even the mouse track moves)

login to your machine from another terminal (or rlogin) and kill the `mgr` process. A buckey-Q key will quit **MGR** if the keyboard still works. Sometimes you can switch to another virtual terminal with an alt-F key and kill `mgr` from there.

4.1 Applications not aware of MGR

Any tty-oriented application can be run in an **MGR** window without further ado. Screen-oriented applications using termcap or curses can get the correct number of lines and columns by your using `shape(1)` to reshape the window or using `set_termcap(1)` or `set_emacs(1)` to obtain the correct termcap.

4.2 MGR Applications (clients) distributed with the server

browse

an icon browser

bury

bury this window

clock

digital display of time of day

clock2

analog display of time of day

close

close this window, iconify

cmenu

vi menus from C compiler errors

color

set the foreground and background color for text in this window

cursor

change appearance of the character cursor

cut

cut text from this window into the cut buffer

cycle
: display a sequence of icons

dmgr
: crude ditroff previewer

fade
: fade a home movie script from one scene to another

font
: change to a new font in this window

hpmgr
: hp 2621 terminal emulator

ico
: animate an icosahedron or other polyhedron

iconmail
: notification of mail arrival

iconmsgs
: message arrival notification

ify
: iconify and deiconify windows

loadfont
: load a font from the file system

maze
: a maze game

mclock
: micky mouse clock

menu
: create or select a pop-up menu

mgr
: bellcore window manager and server

mgrbd
: boulder-dash game

mgrbiff
: watch mailbox for mail and notify

mgrload

graph of system load average

mgrlock

lock the console

mgrlogin

graphical login controller

mgrmag

magnify a part of the screen, optionally dump to file

mgrmail

notification of mail arrival

mgrmode

set or clear window modes

mgrmsgs

message arrival notification

mgrplot

Unix "plot" graphics filter

mgrsclock

sandclock

mgrshowfont

browse through mgr fonts

mgrsketch

a sketching/drawing program

mgrview

view mgr bitmap images

mless

start up less/more in separate window, menu added for less

mvi

start up vi in a separate window, mouse pointing

oclose

(old) close a window

omgrmail

(old) notification of mail arrival

pbmrawtomgr

convert pbm raw bitmap to mgr bitmap format

pbmstream

split out a stream of bitmaps

pbmtoprt

printer output from PBM

pilot

a bitmap browser

resetwin

cleanup window state after client crashes messily

rotate

rotate a bitmap 90 degrees.

screendump

write graphics screen dump to a bitmap file

set_console

redirect console messages to this window

set_termcap, set_emacs

output an appropriate TERMCAP value

setname

name a window, for messages and iconifying

shape

reshape this window

square

square this window

squeeze

compress mgr bitmap using run-length encoding

startup

produce a skeleton startup file for current window layout

texmgr

tex dvi file previewer

text2font, font2text

convert between mgr font format and text dump

unsqueeze

uncompress mgr bitmap using run length encoding

window_print

print an image of a window

zoom

an icon editor

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

graphics demos

4.3 MGR-aware clients distributed separately, see "SUPPORT" file

chess

frontend to `/usr/games/chess`

gnu emacs

editor with `lisp/term/mgr.el` mouse & menu support

gnuplot

universal scientific data plotting

metafont

font design and creation

origami

folding editor

pbmplus

portable bitmap format conversions, manipulations

pgs

ghostscript patch and front end

plplot

slick scientific data plotting

?

a groff PBM driver using Hershey fonts

5 Programming for MGR

The **MGR** programmers manual, the C language applications interface, is found in the doc directory in troff/nroff form. It covers general concepts, the function/macro calls controlling the server, a sample application, with an index and glossary.

Porting client code used with older versions of **MGR** sometimes requires the substitution of

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

for

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

and substituting or redefining BIT_XOR, BIT_AND, et al for B_XOR, B_AND, et al in the bitblt operations.

Compiling client code generally requires compiler options like

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

as well as an occasional -DOLDMGR or -DOLDLIBMGR for some dusty decks.

One can get some interactive feel for the **MGR** server functions by reading and experimenting with the `mgr.el` terminal driver for GNU Emacs which implements the **MGR** interface library in ELisp.

The usual method of inquiring state from the server has the potential of stumbling on a race condition if the client also expects a large volume of event notifications. The problem arises if an (asynchronous) event notification arrives when a (synchronous) inquiry response was expected. If this arises in practice (unusual) then the **MGR** state inquiry functions would have to be integrated with your event handling loop.

The only major drawing function missing from the **MGR** protocol, it seems, is an area fill for areas other than upright rectangles. At present, the color palette itself is manipulated by programs external to **MGR**, but version 0.62 has experimental code for clients to get **MGR** to manipulate the color palette.

If you are thinking of hacking on the server, you can find the mouse driver in `mouse.*` and `mouse_get.*`, the grotty parts of the keyboard interface in `kbd.c`, and the interface to the display in the `src/libbitblit/linux` directory. The main procedure, much initialization, and the top level input loop are in `mgr.c`, and the interpretation of escape sequences is in `put_window.c`.

6 More documentation

The programmer's manual is essential for concepts.

Nearly all the clients supplied come with a man page which is installed into `/usr/mgr/man/man1` or `man6`. Other useful man pages are `bitblit.3`, `font.5`, and `bitmap.5`. There is some ambiguity in the docs in distinguishing the internal bitmap format found in your frame-buffer and the external bitmap format found in files, e.g. icons.

The `mgr.1` man page covers command line options, commands in the `~/.mgrc` startup file, mouse and menu interaction with the server, and hot-key shortcuts available on systems with such hot-keys.

Many of the fonts in `/usr/mgr/font/*` are described to some extent in `/usr/mgr/font/*.txt`, e.g. `/usr/mgr/font/FONTDIR.txt` gives X-style font descriptions for the fonts obtained in .bdf format. Font names end in `WxH`, where `W` and `H` are the decimal width and height in pixels of each character box.

7 Credit for MGR

Stephen Uhler, with others working at Bellcore, was the original designer and implementer of **MGR**, so Bellcore has copyrighted much of the code and documentation for **MGR** under the following conditions.

```
* 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.
```

One required showing of the copyright notice is the startup title screen.

Other credits to:

- Stephen Hawley for his wonderful icons.
- Tommy Frandsen for the VGA linux library.
- Tom Heller for his Gasblit library.
- Andrew Haylett for the Mouse driver code.
- Dan McCrackin for his gasblit-¿linux patches.
- Dave Gymer, dgp@cs.nott.ac.uk, for the Startrek effect fix.
- Alex Liu for first releasing a working Linux version of **MGR**.
- Lars Aronsson (aronsson@lysator.liu.se) for text2font and an ISO8859-1 8-bit font.
- Harry Pulley (hcpiv@grumpy.cis.uoguelph.ca, hcpiv@snowhite.cis.uoguelph.ca) for the Coherent port.
- Vance Petree & Grant Edwards & Udo Munk for their work on Hercules.
- Udo Munk for his work on serial mouse initialization & select.
- Norman Bartek & Hal Snyder at Mark Williams Co. for their help with some bugs & with Coherent device drivers.
- Extra thanks to Zeyd Ben Halim for lots of helpful patches, especially the adaptation of selection.
- Bradley Bosch, brad@lachman.com, for lots of patches from his 3b1 port, which fix bugs and implement new and desirable features.
- Andrew Morton, applix@runxtsa.runx.oz.au, who first wrote the cut-word code.

- Kapil Paranjape, kapil@motive.math.tifr.res.in, for the EGA support.
- Michael Haardt for MOVIE support fixes, bug fixes, separation of the libbitblit code into output drivers, origami folding of the code.
- Yossi Gil for many fonts.
- Vincent Broman for middle mouse-button emulation, linting, Sun cgsix support, VGA colormap acess, and integration of the sunport code into Haardt's layering scheme.

All bitmap fonts from any source are strictly public domain in the USA. The 500 fixed-width fonts supplied with **MGR** were obtained from Uhler, the X distribution, Yossi Gil, and elsewhere. The Hershey vector fonts and the code for rendering them are probably freely redistributable.