NAME

Bitmap – Bitmap header format for *mgr* bitmaps.

SYNOPSIS

#include "dump.h"

DESCRIPTION

There are two styles of bitmaps recognized by **MGR ,** The old machine dependent format, and the new portable format.

Old bitmap files are prepended with a 6 byte ascii header which contains: 1) a two byte magic number, 2) a 2 byte bitmap width, and 3) a two byte bitmap height. The bitmap data follows the header in raster scan order, with each row padded out to a 16 bit boundary.

The new, portable bitmap format consists of an 8 byte ascii header containing: 1) a two byte magic number, 2) a 2 byte bitmap width, 3) a two byte bitmap height, a single byte bitmap depth, and 1 reserved byte. The bitmap data follows the header in raster scan order, with each row padded to a byte boundary.

For different types of displays, MGR represents bitmaps internally with each row padded to 8, 16, or 32 bit boundaries, which ever makes the most sense for the particular hardware. The internal format in use by MGR is avaliable to application programs via a GET_INFO call. MGR will accept any of the above row alignments, differentiated by the magic number in the bitmap header. However, if the external format matches the internal one, bitmaps may be processed more efficiently by the server.

The following macros, defined in *dump.h* may be useful for dealing with bitmap headers:

B_HSIZE
: The bitmap header size in bytes.

B_GETHDR(header,width,height)
: extracts the *width* and *height* from the B_HSIZE buffer *header*

B_PUTHDR(header,width,height)
: produces a bitmap header for a bitmap *width* bits wide and *height* bits high.

B_ISHDR(header)
: returns true if *header* is a valid bitmap header

B_SIZE(width,height)
: returns the size in bytes (not including the header) of a bitmap *width* bits wide and *height* bits high.

B_MAGIC
: is a pointer to a character string whose first 2 bytes are the bitmap header magic number.

BUGS

* The existence of two different bitmap formats is unfortunate. The old format should go away when the programs that use it are rewritten.
* The file "dump.h" gives a more accurate description of multitude of bitmap formats.

SEE ALSO

mgr(1L)

NAME

bounce – A standard graphics demo

SYNOPSIS

bounce [**–s**]

DESCRIPTION

Bounce bounces 10 lines around the window forever. The **–s** flag bounces the lines slower. Bounce stops if its window is obscured.

SEE ALSO

mgr(1L)

AUTHOR

D. Nachbar

NAME

browse – An icon browser for MGR

SYNOPSIS

browse filename...

DESCRIPTION

Browse displays the icon files specified on the command line in the current window. If all of the icons specified won't fit in the window, the pop-up menu accessed from the middle mouse button permits paging back and forth among the icons.

The Right or *pointing* button of the mouse, when clicked over an *icon*, highlights the icon and prints its filename.

BUGS

* The icon files are read from the same host *MGR* is executing on, not the host *browse is running on.*

SEE ALSO

mgr(1L) bitmap(5L) zoom(1L)

AUTHOR

S. A. Uhler

NAME

bury – Bury a *mgr* window.

SYNOPSIS

bury

DESCRIPTION

Bury pushes the window to the *bottom* of the screen.

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

C_MENU – Turn **C** error messages into **vi** menus.

SYNOPSIS

make -k | c_menu; vi

DESCRIPTION

C_menu reads its standard input and looks for lines of the form:

"foo.c", line 19: word undefined.

All such lines are gathered into a set of menus, one for each **C** file, that are useful for locating the errors in the source files using the *vi* edittor.

The main menu contains the names of the **C** files with errors, selecting a file causes that file to be editted. Sliding off to the right of the *file name* menu pops up a list of error messages. Selecting an error message while in *vi* causes *vi* to move its cursor to the line containing the error.

The most common way to use *c_menu* is with the **MGR** *cut* and *paste* facility. After running a *make* or *cc* that produces **C** error messages, simply *cut* the error messages, type **c_menu;vi ,** *paste* the errors into *c_menu ,* then key *CNTL d.*

SEE ALSO

mgr(1L) menu(1L)

BUGS

After adding or deleting lines from the file, *c_menu 's* notion of which line contains the error is incorrect.

AUTHOR

S. A. Uhler

NAME

Chess – A graphical interface to /usr/games/chess

SYNOPSIS

chess [-f]

DESCRIPTION

chess is a simple minded graphical user interface to /usr/games/chess. Pieces are moved by clicking the middle mouse button, first on the piece to be moved, then to the desired position. If the *–f* flag is specified, the computer moves first. If an invalid move is requested, is printed. While /usr/games/chess is figuring out its next move, *(working)* is displayed.

FILES

/usr/games/chess

BUGS

chess knows nothing about chess. It asks /usr/games/chess for the board layout at each move. Consequently, the user's move is not updated until the computer responds with the counter move. In addition, the data exchange format with /usr/games/chess is hardwired in.

SEE ALSO

mgr(1L)

AUTHORS

S. D. Hawley S. A. Uhler

NAME

clock – Digital display of time of day on a *mgr* terminal.

SYNOPSIS

clock [**–b**] [**–f**<font>] [**–s**]

DESCRIPTION

Clock displays a time of day clock on a *mgr* window. The window shrinks to just enclose the display. The **-f** flag is used to specify the font to use for the display. **clock -f5** is typical. **-b** pushes the clock to the bottom of the display, and **-s** doesn't reshape the window.

SEE ALSO

clock2(1L) mgr(1L)

AUTHOR

S. A. Uhler

NAME

clock2 – Analog display of time of day on a *mgr* terminal.

SYNOPSIS

clock2

DESCRIPTION

Clock2 draws an analog clock face that fills the current window, on which it shows the current time of day. Square-ish windows produce the best results.

SEE ALSO

clock(1L) mgr(1L)

AUTHOR

S. A. Uhler

NAME

close – Close a *mgr* window.

SYNOPSIS

close [*message* [*font-number*]]

DESCRIPTION

Close Makes the current window very small, displays *message* in it, and moves it to an unoccupied spot on the screen. If the message includes "**%d** then the window set ID will replace it when the message is displayed. Upon reactivation, the window returns to its former size and position. If the *closed* window is ever covered, it attempts to find and move itself to some other unoccupied spot.

If no *message* is given or the message is of zero length, the current *hostname* along with the parenthesized *window ID* is displayed. The window ID can be used to activate the window from the keyboard. **Left**-*windowID* or **Right**-*windowID* will activate the window if the window ID is a single digit. **Left-w**-*windowID* or **Right-w**-*windowID* will activate the window for any window ID. Window ID **0** is an alias for window ID **10**.

An optional second argument, *fontnumber*, may be specified to indicate the font in which *message* is displayed.

EXAMPLES

close
close 'source directory'
close '' 0
close 'source directory' 7

BUGS

* Not all windows on the screen may be closed at once.

SEE ALSO

mgr(1L)

AUTHORS

S. A. Uhler
M. H. Bianchi

NAME

Color – set the forground and background color for text in an **Mgr** window. **MGR .**

SYNOPSIS

color [dark|light] *color* on [dark|light] *color*

DESCRIPTION

color sets the current foreground and background text color for an **Mgr** window. *Color* is one of black, white, red, green, blue, yellow, cyan or magenta. Alternately, *color* may be specified as an index in the color lookup table. **Color** calls *set_colormap(1L)* to initialize the color map.

SEE ALSO

mgr(1L) set_colormap(1L)

BUGS

Color only works with *color Mgr* .

AUTHOR

S. A. Uhler

NAME

cut – cut text from a *MGR* window and send it to a program.

SYNOPSIS

cut [**-s**] [**command**]

DESCRIPTION

Cut watches the global *MGR* cut buffer and when activated, reads the buffer and starts **<command>** with the contents of the buffer as **<command>'s** standard input.

Initially, *cut* prompts the user for a spot on the display, moves there and becomes a closed file cabinet icon. Any time text is cut to to *MGR 's* global buffer, *cut* highlights the file cabinet to indicate it has seen the cut text. *Cut* prepends the current date, time and message size to the text sent to **<command>.**

If **–s** is specified, **cut** does not prompt the user for a spot, but uses the window as is. This is useful for starting cut from the *MGR* startup file. If no **command** is given, **cut** looks for the command in the environment variable **CUT .**

BUGS

SEE ALSO

mgr(1L) snap(1L)

AUTHOR

S. A. Uhler

NAME

cycle – Display a sequence of icons on an *mgr* terminal.

SYNOPSIS

cycle [**–sspeed**] [**–r**] icon1 icon2 [... **<icon***n*> *]*

DESCRIPTION

cycle will display the list of specified icons in sequence in an **mgr** window.

The flag **–s speed** sets the delay between frames in micro-seconds

The flag **–r** causes the frames to be run forward and then in reverse and then repeat, rather than just running them forward repeatedly.

SEE ALSO

mgr(1L)

AUTHOR

S. D. Hawley

NAME

Dmgr – A rudimentary troff previewer for mgr

SYNOPSIS

ditroff [<*ditroff args*>] ... | **Dmgr**

DESCRIPTION

Dmgr is a simple *troff* previewer for *MGR* . It reads *ditroff* output and places characters on an *MGR* window in their proper relative location on the page, using whatever *MGR* character font happens to be current. Bold face is indicated by overstiking, italics by underlining. **Dmgr** pauses at the end of every page and rings the bell. A *RETURN* causes **dmgr** to continue with the next page.

BUGS

- **Dmgr** uses the current *Mgr* font for output, which is probably never the appropriate font to use. As such **Dmgr** is useful for previewing the page layout; not for actually reading the document.
- **Dmgr** doesn't know about special characters or ligatures, which are printed as dashes "-".
- **Dmgr** invents a page size, suitable for 8-½ by 11 inch printers instead of extracting it from *ditroff* output.

SEE ALSO

mgr(1L) ditroff(1)

AUTHOR

S. A. Uhler

NAME

ether – Display a strip chart of network traffic.

SYNOPSIS

ether [**–c<color_map_indexes>**] [**–f<freq>**] [**–m<max>**]

DESCRIPTION

Ether is a graphical version of *netstat* that runs on *mgr* terminals. **Ether,** displays the number of input packet, output packets, and collisions on the first network interface reported by *netstat.*

The following options are recognized on the command line:

–f<freq>

The display is updated every *freq* seconds, instead of the default 3 seconds.

-m<max>

specifies the maximum number of packets counted per update. The default is 15.

–c If ether is run on a color version of MGR, various parts of the display are shown in different colors. This option alters the colormap index values used. The **–c** is immediately followed by a string of characters in the range of [0-9a-z] where each color represents a colormap index from 0 to 36 respectively. The position in the string determines what portion of the display is affected as follows:

#	use
0	unused
1	background
2	title
3	axis labels
4	grid lines
5	values that exeed the maximum
6	color for plots

mgr(1L) netstat(1)

DIAGNOSTICS

Window is not wide enough

Make the window wider and the graph will continue.

Window is not high enough

Make the window taller and the graph will continue.

BUGS

If the window is reshaped, *ether* requires up to **freq** (usually 3) seconds to learn about the new window size.

Ether calls *netstat(1) and assumes a particular output* (i.e. 4.2 BSD).

AUTHOR

S. A. Uhler

NAME

font – change to a new font in a *mgr* window.

SYNOPSIS

font [**<font number>**]

DESCRIPTION

font Changes the current font to the number indicated, or to the default font if *font number* is omitted.

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

Font – font file format for *mgr* bitmaps.

SYNOPSIS

#include "font.h"

DESCRIPTION

MGR font files consist of a font header followed by the bitmap data for each character. The header format is:

```
struct font_header {
        unsigned char type;             /* font type (magic number) '^v' */
        unsigned char wide;             /* character width (pixels) */
        unsigned char high;             /* char height (pixels) */
        unsigned char baseline;     /* pixels from bottom of glyph */
        unsigned char count;        /* number of chars in font */
        char start;                     /* starting char in font */
  };
```

All of the characters in the font are are stored in a single bitmap, *wide* x *count* pixels wide, and *high* pixels high.

BUGS

Only fixes width fonts are currently supported.

SEE ALSO

mgr(1L) bitmap(5L)

NAME

iconmail - Notification of mail arrival

SYNOPSIS

iconmail [**−s**] [**−x**<pos>] [**−y**<pos>] [**−f**<font>] [**−p**<poll>] [**−M**<mailer>] [**−m**<mail file>] [**−t**<label>]

DESCRIPTION

Iconmail looks for, and announces the arrival of new mail. When initially invoked, *iconmail* shrinks its window into a mailbox icon. When new mail arrives, *iconmail* rings the bell, displays a mailbox with mail in it. If the *iconmail* window is activated, usually by clicking on it with the mouse, It either creates a larger window with *readmail* running in it - if you have mail, or indicates you have no mail.

While in the *readmail* window, the middle mouse button activates a menu of commonly used mail commands.

iconmail recognizes the following command line flags:

-s Don't reshape the window upon **iconmail** invocation.

-x<pos>
Starting x-coordinate of *readmail* window.

-y<pos>
Starting y-coordinate of *readmail* window.

-f<font_number>
Font to use for *readmail* window.

-p<poll interval>
Look for new mail every *poll interval* seconds (defaults to 60).

-M<mailer>
Use *mailer* to read mail, instead of *mail.*

-t<label>
labels the icon with this text

-m<mail file>
Specifies an alternate mail file, to pee passed to the mail program with the *-f* flag.

BUGS

The *readmail* window sleeps for a second at its termination to permit *mail* to indicate new mail arrival while reading mail.

Destroying the *mail* subwindow is a bad thing to do.

FILES

/usr/spool/mail/$USER

SEE ALSO

mail(1) mgr(1L) mgrmail(1L)

AUTHOR

S. A. Uhler

NAME

iconmsgs – message arrival notification

SYNOPSIS

iconmsgs [**–s**] [**–x**<pos>] [**–y**<pos>] [**–f**<font>] [**–p**<poll>]

DESCRIPTION

iconmsgs looks for, and announces the arrival of new **msgs.** When initially invoked, *iconmsgs* shrinks its window to a bulletin board icon, displaying the number of pending messages as notes tacked to the board. When new messages arrive, **iconmsgs** rings the bell, and tacks a new message to the bulletin board. If the *iconmsgs* window is activated when messages are pending, usually by clicking on it with the mouse, iconmsgs creates a larger window with *msgs* running in it.

While in the *msgs* window, the middle mouse button activates a menu of commonly used msgs commands.

iconmsgs recognizes the following command line flags:

-s Don’t reshape the window upon **iconmsgs** invocation.

-x<pos>
Starting x-coordinate of *msgs* window.

-y<pos>
Starting y-coordinate of *msgs* window.

-f<font_number>
Font to use for *msgs* window.

-p<poll interval>
Look for new messages every *poll interval* seconds (defaults to 60).

BUGS

Destroying the *msgs* window while *msgs* is running is a bad thing to do.

FILES

/usr/spool/msgs/bounds

$HOME/.msgsrc

SEE ALSO

msgs(1) mgr(1L) mgrmsgs(1L)

AUTHOR

S. A. Uhler

NAME

ify – Iconify MGR windows

SYNOPSIS

ify [**–s**]

DESCRIPTION

Ify lets the user iconify **MGR** windows. When run, **ify** shrinks its window and displays its icon. If started with the **–s** option, **ify** does not reshape its window to fit the icon.

While **ify**'s window is active, the user can iconify or de-iconify windows by clicking on them with the right mouse button. To iconify a window, **ify** shrinks it and buries it in an unoccupied portion of the screen. To de-iconify a window, **ify** restores the window's size, position, contents and settings. Iconified windows cannot handle input or output; programs doing output to iconified windows will block. If an application has opened several subwindows on the screen, the user may click on any of the windows to iconify the whole group; clicking on any of the iconified windows de-iconfies the whole group.

The middle mouse button displays a pop-up menu listing the currently closed windows; selecting a window from the menu opens it. The menu also contains a *quit* item, which exits **ify**, and a *remake menu* item, which causes **ify** to re-examine the display for closed windows and update its list.

Ify titles an iconified window with its **NOTIFY** string, possibly followed by its window id in parenthesis, if it is a subwindow. The **NOTIFY** string can be set from within C programs using a statement of the form

m_setevent(NOTIFY, "*title*");

or from the shell using the **setname** program, possibly as part of the .mgrc file. If a window has no **NOTIFY** event set, **ify** titles it with the name of its tty.

To find its icon, **ify** searches **MGR**'s current directory (usually the user's home directory) for a file called **.ify.icon**, and then searches the standard **MGR** icon directory for a file called **ify.icon**. If neither of those files exists, it despairs and downloads its own icon.

If there are no free spots on the screen, **ify** starts stacking windows going down the left side of the screen.

NOTES

Ify runs more smoothly when **MGR** has its setuid bit turned on. If **MGR** is not running with the setuid bit on, the pseudo-ttys for the windows end up being owned by root, not by the user. In this situation, since **ify** can't turn off general write permission to its tty, its internal operations become a little less robust; it may start behaving strangely when the system becomes heavily loaded, or if an application has opened many windows.

SEE ALSO

mgr(1L), close(1), setname(1L)

DIAGNOSTICS

If **ify** encounters an unexpected error from a system call, it prints the error message and exits. These messages are usually not very helpful to the end user, and probably indicate bugs in **ify** itself.

BUGS

If a program has several windows open, de-iconifying the group may leave the wrong window selected.

Some programs push their event strings without establishing a new **NOTIFY** string, so iconification leaves them untitled.

Ify stores information in a window's **NOTIFY** string while the window is iconified; if this information is corrupted, **ify** might not be able to reopen the window. To get around this, select the iconified window and press control-Q.

AUTHOR

Jim Blandy

NAME

invert_colormap – inverts the colormap on a SUN color display **MGR .**

SYNOPSIS

invert_colormap

DESCRIPTION

Set_colormap inverts the current color map in the sense of a photographic negative.

SEE ALSO

set_colormap(1L) overlay(1L)

AUTHOR

S. A. Uhler

NAME

lock – lock the sun console

SYNOPSIS

lock

DESCRIPTION

lock displays a screen-phosphor saving pattern on the sun console. When you login password is typed on standard input, the screen is restored. If you mistype your password, the pattern direction reverses, and you may try again.

FILES

/dev/bwtwo0	to find the screen
/etc/passwd	to check the login password

SEE ALSO

lockscreen(1)

BUGS

Error checking is poor

AUTHOR

S. A. Uhler

NAME

Maze – A graphical game of solitare

SYNOPSIS

maze

DESCRIPTION

Maze draws a maze and permits you to navigate around it while displaying both a top and perspective view. The **f** (or **space**), **r, l,** and **b** keys move you forward, right, left, and backwards respectively. You can sometimes see others in the maze if they are playing **maze** somewhere else on the network.

BUGS

* This is truly a mindless endeavor.
* When other maze players die, they leave ghosts in the maze.

SEE ALSO

mgr(1L)

ACKNOWLEDGEMENTS

This program was written by *J. Gosling* for *Andrew* and ported to **mgr.**

NAME

menu – create or select an mgr pop-up menu

SYNOPSIS

menu [**–options**] [menufile(s)]

DESCRIPTION

Menu downloads or selects a pop-up menu for the mgr window manager for a Sun workstations.

OPTIONS

-sn selects the menu previously loaded into position n; that is, it binds the menu in position n to the middle mouse button. No downloading takes place.

-s by itself allows downloading of a new menu, without it being selected.

-ln causes loading of a new menu into position n. By default, n is 1. The loaded menu is also selected (unless -s was specified).

-n"Name of Menu" specifies a name for the menu. Alternatively, a name may be specified by one or more lines of the form name=anything in the menu file. By default, the name of the menu is taken to be the name of the input file converted to upper-case. The menu's name is displayed at the top of the menu. -n by itself suppresses the name altogether.

-fn specifies that the font loaded into font postion n will used for the menu. Default is 6.

-d<char> indicates that <char> will be used instead of the default, TAB, to delimit items from actions in the menu file.

The named input files (or standard input by default) should contain one item-action pair per line, delimited by by one or more delimiter characters (TABs by default). Items and actions are arbitrary character strings not containing the delimiter character. The items are displayed in the pop-up menu, and the user is allowed to select an item using the mouse and middle button. The corresponding action string is then written to the standard input of the process running in the window. If an action ends with \c, the newline character following the action is suppressed.

Optionally, the menu file may begin with a delim=<char> line to specify an alternate delimiter, followed by any number of name=anything lines to specify one or more lines of menu name, and a font=n line to select a font. The total number of characters in all actions and items may not exceed approximately 1000.

EXAMPLES

```
menu <<!
delim=:
name=    FRUITS
name=--------------
apples:echo apples
oranges:echo oranges
pears:echo pears
passion fruit:echo passion fruit
three cherries:echo Jackpot!
!
```

AUTHOR

Paul A. Tukey (bellcore!paul)

NAME

movie_mgr – manage windows on a **SUN** Workstation

SYNOPSIS

mgr [**–f***font_dir*] [**–i***icon_dir*] [**–s***startup_file*] [**–z***"shell_command"*] [**–n**] [**–x**] [**–v**] [**–V**] [**–F***default_font*]

{ [**–d***list*] [**–m** *mouse_device*] [**–B***window_buff*] [**–b***shell_buff*] [**–P***poll_interval*] [**–S***screen*] }

DESCRIPTION

Mgr is a window manager for the **SUN** workstation. It permits the creation and manipulation of overlapping windows, with different processes running in each window. The user controls the function and layout of the display with a mouse. Windows are updated asynchronously even if they are partially (or completely) obscured by other windows, although obscured windows may arrange to have their output suspended until the window is uncovered.

Each window runs a terminal emulator which, in addition to the functions normally required to run screen oriented programs, such as *vi*, provides primitives for drawing lines, doing *bit-blts*, and performing administrative functions such as *reshaping* the window, changing *fonts*, or starting a new window. Details of of the terminal emulator operation are described in the Movie_**MGR** - C Language Application Interface.

The useful command line options are:

–f*font_dir*
Use *font_dir* as the directory to find the fonts, instead of **/usr/mgr/font.**

–i*icon_dir*
Use *icon_dir* as the directory to find the icons, instead of **/usr/mgr/icon.**

–s*startup_file*
Use *startup file* instead of **$HOME/.mgrc** to obtain initial configuration information. See the description of startup commands below.

–n Bitmap files are created using the new, portable bitmap format. The portable format has an 8 byte header, and each row is padded to a byte boundary. Ordinarily the old (6 byte) bitmap header is produced, followed by the bitmap data with each line padded to an 16 bit boundary. Eventually, the sense of **–n** will change, when all of the programs that were written in the old format are changed.

–x Don't use a startup file upon execution.

–v Don't run Movie_**MGR** at all. Print the current version number and creation date instead.

–V Just like **–v** above only prints the compile flags used to make Movie_**MGR** and its home directory.

–F*default_font*
Use *default_font* as the pathname of a Movie_**MGR** font to be used in place of Movie_**MGR**'s builtin default font.

–z*shell_command*
When scripting is started, *shell_command* is started and receives the MGR scripting information on its standard input. When scripting is terminated, the command is killed. Scripting starts either by keying *meta-shift-s* on the keyboard or by sending the MGR process a SIGUSR1 signal. Scripting stops by keying a *meta-s* on the keyboard, or by sending MGR a SIGUSR2 signal. A single dash (-) in lieu of a command instructs Movie_**MGR** to emit scripting information on its standard output. If -Z is used instead of -z, Movie_**MGR** starts with scripting turned on. Some example commands are:

mgr -z "zcat >file"
Which compresses the script output and places it in *file.*

mgr -z "rsh foo do_data -"
Which sends the script to machine *foo* for real-time playback on *foo 's* console.

The rest of the options are:

−d*list* Print debugging information on *stderr*. *list* is one or more of the characters: **∗ABCEFLMNPSUbcdefilmnopsuwxy** each of which turns on debugging output for some aspect of Movie_**MGR**.

−m*mouse_device*
Use *mouse_device* instead of **/dev/mouse** to obtain mouse coordinates.

−S*screen*
Use *screen* instead of **/dev/bwtwo0** as the display device.

−B*win_buff*
Process characters to a window in up to *win_buff* byte chunks (the default is 40).

−b*shell_buff*
Buffer up to *shell_buff* bytes of output from a program before writing it on a window (the default is 256).

−P*poll_interval*
When output is pending in a window, wait *poll_interval* micro-seconds on every polling loop to give more process time to the processes running in the windows. The default is zero.

Startup File Format

Upon invocation Movie_**MGR** reads in the default font information from **.mgrc** located in the font directory, then reads commands from the "startup file", **$HOME/.mgrc** (see **−s** flag above) to initialize the display. Commands are placed one per line with the command arguments separated by spaces or tabs. The following commands are supported:

initcmd *command* [*args...*]
This command line is handed to the shell and executed at the time the startup file is read.

suspendcmd *command* [*args...*]
This command line is handed to the shell and executed each time Movie_**MGR** suspends it self, either due to a main menu selection or the Left-z key.

resumecmd *command* [*args...*]
This command line is handed to the shell and executed each time Movie_**MGR** resumes after a suspension.

quitcmd *command* [*args...*]
This command line is handed to the shell and executed just before Movie_**MGR** quits, either due to a main menu selection or the Left-Q key.

map *n0 n1 n2 n3 n4 n5 n6 n7*
This changes the meaning of the mouse buttons. Each *n[0-7]* represents one of the 8 states of the three *mouse buttons*. The default mapping is: 0 1 2 3 4 5 6 7. To change the meaning of the *left* and *right* buttons, 0 2 4 6 1 3 5 7 would be used. It is possible to map a button out of existence, which may have grave consequences.

font *font_number font_name*
The default font may be overridden by specifying the font *font name* which is to be substituted for the font at position *font_number*. Font numbers are small integers, in the range of 0-99. The *font_name*s are found in the font directory, by default **/usr/mgr/font**. See the **−f** flag above.

window *x y wide high* [*font_number*]
A window is created whose corner is at the coordinates *(x, y)* and whose size is *(wide, high)*. Units are in pixels, with *x* and *y* increasing to the *right* and *down* respectively. *Wide* and *high* can be set in terms of characters in the current font by appending the letter "c" to the value. If *x* and *y* are **−1**, then they are replaced by values that causes new windows to "tile" across the screen. Setting *wide* and *high* to **−1** is identical to setting them to "80c" and "24c" respectively. The scope of the **window** command continues until either another **window** command or **done** is reached. The rest

of the options, **shell, start, init, flags,** and **newwindow** apply only to the current **window** command.

shell *command* [*args...*]
Command is the name of the command or shell to be started in the window. If *command* is not specified, then the environment variable **$SHELL,** or **/bin/csh** is used.

start *command*
The *command* is sent to the *shell* upon startup, as if it had been typed at the keyboard.

init *initial_string*
The *initial string* is sent to the *window* upon startup. The string is terminated by white space, the remainder of the line may be used as a comment. The codes: **\\, \b, \f \e, \n, \r,** or **\s** may be used to represent \, *backspace formfeed, escape, newline, return,* or *space* respectively.

flags *flag...*
Normally a window self destructs when the original process running in it dies. if **nokill,** currently the only flag, is specified, the window hangs around until specifically snuffed by the user.

newwindow
The current window specification is not to be used to initialize the display, but instead will be used when **Left n** or **Right n** to create a new window.

done **done** must be the last line in the startup file if any **window** commands are specified, or the last **window** command will not take affect.

Using The Mouse

User interaction with Movie_**MGR** is with the mouse. Moving the mouse causes a corresponding movement of the *mouse cursor,* usually an arrow pointing to the upper left. The *left* or *command button* of mouse activates a *menu* whose options depend upon the current mouse position. An option is chosen by moving the mouse vertically while the *command* button is depressed, releasing the button when the appropriate selection is highlighted.

When the mouse is over the background pattern, or at the extreme left edge of the screen, the *command menu* is activated by the mouse. The *command* menu options are:

new window A new window is created by moving the the mouse cursor (now a box) to the upper left corner of the window, depressing the *command button,* sweeping out the window, then releasing the *command button.* The new window, if it is big enough, is started with a shell running in it.

redraw The background and windows are redrawn. This is useful if a process unknown to Movie_**MGR** scribbles on the display. It is left to the processing running in a window to fix the contents of its window.

quit Movie_**MGR** is terminated, after the *quit* is confirmed. Alternately, Movie_**MGR** may be suspended (ala **^Z** in **csh**).

When the mouse is over the *active* window, the fat bordered window the keyboard is connected to, the *window menu* is activated by depressing the *command button.* The *window* menu options are:

reshape *Reshape* reshapes the *active* window, using a procedure similar to *new window* above.

move An outline of the current window is moved along with the mouse until the *command button* is depressed and released. The current window is then moved to the new location.

bury The current window is made inactive. Another window (if any) becomes the *active* window.

cut The mouse may be used to sweep out and save text from the current window into a global buffer. A small scissors appears as the mouse cursor. Position the upper left corner of the

scissors with the upper left corner of the first character to be saved, then push one of the mouse buttons, moving the mouse to sweep out the desired text. Releasing the button causes the outlined text to be saved. Using the *command button* with *cut* causes the current contents of the global buffer (if any) to be replaced by the indicated text. Either of the other two buttons causes the indicated text to be appended to the global buffer.

The *cut* facility currently works only for windows containing a single font, aligned on the default character boundaries. Applications which use only the terminal emulator sub-set of *MGR* capabilities, such as the *shell*, *mail*, and *editors* automatically meet this restriction. Cuttability may be restored by issuing a *clear* (i.e. form feed) to the window. The window flashes and beeps if the *cut* operation could not be completed, usually the result of corrupted data in the window. In such cases, no text is saved. See Movie_**MGR** - C Language Application Interface for a detailed description of the various *cut* option settings.

paste The contents of the global buffer (if any) are inserted into the input stream of the current window. The global buffer is filled using **cut** above, or under program control.

destroy All processes associated with the current window are sent a *hangup* signal, and the window is destroyed.

When the mouse is clicked on any window except the *active* window, that window moves to the *front* and becomes the *active* window.

Using The Left and Right Keys

When Movie_**MGR** is invoked from the console keyboard, many of the system menu functions have keyboard equivalents. Some of the more interesting ones are activated by holding down the **Left** or **Right** keys, and then pressing:

space bar
: to activate the previous window

Back Space
: to activate the bottom window

c to initiate a *cut-text* operation

p to initiate a *paste* operation

h hide the top window on the bottom

l to clear the active window

m initiate a *cut-text* operation which will automatically cause a *paste* operation when completed

n to start a new window, 80 x 24 characters (if it will fit), placed in the "tile" position of its window-set ID

N start a new window by sweeping with the mouse

Q to exit Movie_**MGR** quickly

1–9 to activate the window with window-set ID 1 through 9

0 activates the window with window-set ID 10, a synonym for w10<Return>

w*number*<Return>
: activate the window with window-set ID *number*

r to redraw the windows

R to redraw the windows

The environment variable **DEFAULT_FONT** may be assigned the full path name of a Movie_**MGR** font, which will then replace Movie_**MGR**'s built in default font.

FILES

/dev/mouse	place to obtain mouse coordinates.
/dev/bwtwo0	name of the display.
/usr/mgr/icon	place to find Movie_**MGR** icons.
/usr/mgr/font	place to find Movie_**MGR** fonts.
/usr/mgr/font/.mgrc	the global default startup file; delivered with 15 fonts specified.
$HOME/.mgrc	place to find startup commands.
/dev/bell	For ringing the bell.
/dev/[pt]ty[pq]?	Name of the pseudo-tty's.

SEE ALSO

Movie_**MGR** - C Language Application Interface
bounce(1L) browse(1L) bury(1L) clock(1L) clock2(1L) close(1L) dmgr(1L) ether(1L) font(1L) iconmail(1L) iconmsgs(1L) loadfont(1L) maze(1L) menu(1L) mgr(1L) mgrmail(1L) mgrmsgs(1L) oclose(1L) omgrmail(1L) rotate(1L) set_console(1L) set_termcap(1L) shape(1L) show(1L) showfont(1L) snap(1L) startup(1L) stat(1L) stringart(1L) tjfilter(1L) window_print(1L) zoom(1L) bitmap(5L) font(5L)

DIAGNOSTICS

Can't find a frame buffer
: No display device available. Make sure */dev/bwtwo0* exists in */dev*.

Can't find a mouse, or it is already in use
: Movie_**MGR** must have exclusive control of the mouse.

Internal Movie_**MGR** error
: everything else.

BUGS

* A separate application program, set_console(1L) is required to prevent others from scribbling on **/dev/console** and messing up the display.
* As Movie_**MGR** requires exclusive control of the mouse, it may not be invoked from within itself.
* Only fixed-width fonts are supported.

AUTHOR

Stephen A. Uhler

NAME

mgrmail - Notification of mail arrival

SYNOPSIS

mgrmail [**–s**] [**–x**<pos>] [**–y**<pos>] [**–f**<font>] [**–p**<poll>] [**–M**<mailer>]

DESCRIPTION

Mgrmail looks for, and announces the arrival of new mail. When initially invoked, *mgrmail* shrinks its window to the single line **Looking for new mail.** When new mail arrives, *mgrmail* rings the bell, and states *You have new mail.* If the *mgrmail* window is activated, usually by clicking on it with the mouse, It creates a larger window with *readmail* running in it.

While in the *readmail* window, the middle mouse button activates a menu of commonly used mail commands.

Mgrmail recognizes the following command line flags:

-s Don't reshape the window upon **mgrmail** invocation.

-x<pos>
Starting x-coordinate of *readmail* window.

-y<pos>
Starting y-coordinate of *readmail* window.

-f<font_number>
Font to use for *readmail* window.

-p<poll interval>
Look for new mail every *poll interval* seconds (defaults to 60).

-M<mailer>
Use *mailer* to read mail, instead of *mail.*

BUGS

The *readmail* window sleeps for a second at its termination to permit *mail* to indicate new mail arrival while reading mail.

Destroying the *mail* subwindow is a bad thing to do.

FILES

/usr/spool/mail/$USER

SEE ALSO

mail(1) mgr(1L)

AUTHOR

S. A. Uhler

NAME

mgrmsgs – message arrival notification

SYNOPSIS

mgrmsgs [**–s**] [**–x**<pos>] [**–y**<pos>] [**–f**<font>] [**–p**<poll>]

DESCRIPTION

Mgrmsgs looks for, and announces the arrival of new **msgs.** When initially invoked, *mgrmsgs* shrinks its window to the single line displaying the number of pending messages. When new messages arrive, **mgrmsgs** rings the bell, and updates the current message count. If the *mgrmsgs* window is activated when messages are pending, usually by clicking on it with the mouse, It changes to a larger window with *msgs* running in it.

While in the *msgs* window, the middle mouse button activates a menu of commonly used msgs commands.

Mgrmsgs recognizes the following command line flags:

-s Don’t reshape the window upon **mgrmsgs** invocation.

-x<pos>
Starting x-coordinate of *msgs* window.

-y<pos>
Starting y-coordinate of *msgs* window.

-f<font_number>
Font to use for *msgs* window.

-p<poll interval>
Look for new messages every *poll interval* seconds (defaults to 60).

FILES

/usr/spool/msgs/bounds

$HOME/.msgsrc

SEE ALSO

msgs(1) mgr(1L)

AUTHOR

S. A. Uhler

NAME

move – move an *mgr* window.

SYNOPSIS

move { [-|dD] <n> up|left|right|down} ...

DESCRIPTION

move moves the current window to somewhere else on the display

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

oclose – Close a *mgr* window.

SYNOPSIS

oclose [**<message>**] [**-F**n]

DESCRIPTION

Oclose Makes the current window very small, displays *message* in it, and moves it to the bottom of the screen. Upon reactivating the window, it returns to its former size and position. If no *message* is given, the current *hostname* is displayed instead. An optional font number may be specified to indicate the font in which *message* is displayed.

BUGS

* *oclose* does a poor job of placing the icon.
* Not all windows on the screen may be closed at once.

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

omgrmail - Notification of mail arrival

SYNOPSIS

omgrmail [**–s**] [**–x**<pos>] [**–y**<pos>] [**–f**<font>] [**–p**<poll>] [**–M**<mailer>]

DESCRIPTION

Omgrmail looks for, and announces the arrival of new mail. When initially invoked, *omgrmail* shrinks its window to the single line **Looking for new mail.** When new mail arrives, *omgrmail* rings the bell, and states *You have new mail.* If the *omgrmail* window is activated, usually by clicking on it with the mouse, It changes to a larger window with *readmail* running in it.

While in the *readmail* window, the middle mouse button activates a menu of commonly used mail commands. The **omgrmail** window may be relocated on the screen by activating the *readmail* subwindow, and moving its upper left corner to the desired **omgrmail** window location.

Omgrmail recognizes the following command line flags:

-s Don't reshape the window upon **omgrmail** invocation.

-x<pos>
Starting x-coordinate of *readmail* window.

-y<pos>
Starting y-coordinate of *readmail* window.

-f<font_number>
Font to use for *readmail* window.

-p<poll interval>
Look for new mail every *poll interval* seconds (defaults to 60).

-M<mailer>
Use *mailer* to read mail, instead of *mail.*

BUGS

The *readmail* window sleeps for 2 seconds at its termination to permit *mail* to indicate new mail arrival while reading mail.

FILES

/usr/spool/mail/$USER

SEE ALSO

mail(1) mgr(1L)

AUTHOR

S. A. Uhler

NAME

overlay – Enable or disable the overlay plane on a Sun 110. **MGR .**

SYNOPSIS

overlay on|off

DESCRIPTION

Overlay enables or disables the overlay plane on Sun's that have them. Setting the overlay plane *on* causes the monochrome plane to obscure the color planes. Setting the overlay plane *off* turns off the monochrome frame buffer, permitting the color frame buffer to be visible.

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

pgs – Preview postscript files in an MGR window

SYNOPSIS

Pgs [**–a**<interval>] [**–d**] [**–i**] [**–r**<dots_per_inch>] <postscript_file>

DESCRIPTION

Pgs is a Postscript interpreter for MGR. It calls the Ghostscript Postscript interpreter to render a Postscript file into a set of bitmapped images, one per page. The User interface permits the user to peruse through the pages interactively.

Pgs displays the first page as soon as it is finished being rendered. The remaining pages are rendered in the background. The status line at the top of the display prints the current page number, and the number of pages rendered so far. If additional pages are being processed, that is indicated on the status line as well. If the rendered page is larger than the window, the page can be panned around using the right most mouse button. The panning rate and direction is controlled by the mouse cursor position relative to the center of the window. If the button is held down, it goes into auto-repeat mode until it is released.

The middle mouse button activates a pop-up menu that lets the user select the next page, previous page, or goto a specific page, quit or suspend. In addition the characters For "vi" fans, the keys ’h’, ’j’, ’k’ and ’l’ cause the page to move left, down, up, and right respectively. If "m" (or the *mark* item on the menu) is keyed, then new pages are displayed in the same relative location in the window as the current page. The character ’g’ will cause a prompt in the status line for a page number. The characters ’q’ and ’^Z’ cause *pgs* to quit or suspend respectively.

If the *postscript* file is specified as ’-’, standard input is used for the postscript input. The command line options are:

-r<res> Render the images at *res* dots/inch resolution. The default is 75.

-i Render the pages in inverse video

-a<interval>
adjust the auto-repeat interval for the mouse while panning. *Interval* is the repeat interval in 100ths of seconds. The default is 5. A value of zero turns off the auto-repeat.

-d print lots of diagnostic messages on stderr, which should be re-directed to a different window. Setting the environment variable DEBUG has the same effect.

The environment variable TMPDIR can be used to override the default temporary directory, which is "/tmp".

BUGS

* Only the first 20 pages are accessible from the menu button.
* **Pgs** can only be run on the local host.
* Ghostscript must be compiled with the MGR driver

FILES

gs The Ghostscript interpreter.

SEE ALSO

gs(1) mgr(1L) pilot(1)

AUTHOR

S. A. Uhler.

ACKNOWLEDGEMENTS

pgs is based on **pilot** by S. D. Hawley.

NAME

pilot – A bitmap browser for MGR

SYNOPSIS

pilot [-r] **filename**

DESCRIPTION

Pilot displays the MGR bitmap in *filename* on the current window, and permits the user to pan the image around the window. The characters h,j,k and l move the image left, down, up, and right respectively. H,J,K, and L move the image as before, only twice as far. The remaining keys, +, -, R, ?, W and q change the movement increment (up or down), redraw the image, display help, write the current window image in *bitmap.out* and quit, respectively.

The -r flag causes the image to be displayed inverted.

BUGS

* The image file is read from the same host *MGR* is executing on, not the host *pilot is running on.*

SEE ALSO

mgr(1L) bitmap(5L) browse(1L)

AUTHOR

S. D. Hawley

NAME

rotate – Rotate a bitmap 90 degrees.

SYNOPSIS

Rotate [**–w wide –h high**] [**–x**] [**–v**]

DESCRIPTION

Rotate is a filter that rotates a 1 bit deep bitmap clockwise by 90 degrees. Normally **rotate** expects a bitmap in *mgr* format which uses a bitmap header to specify the width and height. Alternately, the **-w** and **-h** flags may be used to indicate the bitmap *width* and *height* if no bitmap header is present. If **-x** is specified, no bitmap header will be produced for the resultant bitmap. The **-v** flags prints **<'s** and **>'s** to help stave off boredom while **rotate** is running.

BUGS

Rotate can't currently rotate bitmaps with more than 1200 rows.

SEE ALSO

bitmap(5L)

AUTHOR

S. A. Uhler

NAME

set_colormap – initialize colormap entries suitable for **MGR .**

SYNOPSIS

set_colormap

DESCRIPTION

Set_colormap initializes the first and last 24 colormap entries suitably for color **MGR .** The first 8 colors are set to white, black, red, green, blue, yellow, cyan, and magenta. The second 8 colors are *dark* versions of the first 8 colors, whereas the third 8 colors are *bright* versions of the first 8 colors. The last 24 colors are set to the inverse of the first 24 colors.

SEE ALSO

mgr(1L) overlay(1L)

BUGS

Set_colormap is Sun specific, and no provision is made to set the remainder of the colormap.

AUTHOR

S. A. Uhler

NAME

set_console – redirect console messages to a *MGR* window.

SYNOPSIS

set_console

DESCRIPTION

Set_console , when run in a *MGR* window, redirects console messages to that window.

SEE ALSO

mgr(1L)

BUGS

* Redirecting console messages raises havoc if the keyboard is not in direct mode. Set_console prints a warning message and fails if the keyboard is not in direct mode.
* Console messages automatically get reset to the console when *MGR* is suspended. Set_console should be reissued after resuming a suspended *MGR* .

AUTHOR

S. A. Uhler

NAME

set_termcap, set_emacs – set an appropriate **TERMCAP** entry for **MGR.**

SYNOPSIS

eval ‘**set_termcap** [**-b**]‘

DESCRIPTION

Set_termcap Prints on *stdout* the *shell* commands required to set the **TERMCAP** environment variable appropriately for the current window size on a *mgr* terminal. *Set_termcap* looks at the **SHELL** environment variable to decide what shell commands are appropriate. the command **eval ‘set_termcap‘** sets the TERM and TERMCAP environment variables appropriately.

Set_emacs optimizes the termcap entry for **GNU-emacs,** which knows how to deal with scrolling regions and multiple line inserts and deletes, whereas *set_termcap* keeps **vi** happy.

BUGS

csh usrs need to use: **set noglob; eval ‘set_termcap‘** to keep the shell from getting confused.

SEE ALSO

csh(1) mgr(1L) sh(1)

AUTHOR

S. A. Uhler

NAME

setname – name an MGR window

SYNOPSIS

setname *window name*

DESCRIPTION

setname sets the **NOTIFY** string of its window. Programs like **ify(1L)** use the **NOTIFY** string as the 'name' of the window.

Window name may contain % escapes, in the manner of **printf(3S)**. **Setname** performs the following substitutions on *window name*:

%m replaced by the name of the machine **setname** is running on

%w replaced by the width of the window, in columns of text

%c replaced by the height of the window, in rows of text

%p replaced by a vague verbal description of the window's position

%% replaced by a single **%**

SEE ALSO

mgr(1L), ify(1L)

DIAGNOSTICS

If **setname** is invoked with no arguments, it prints a short explanatory message.

BUGS

Creeping featurism at its worst. Real hackers use escape codes.

AUTHOR

Jim Blandy

NAME

shape – Reshape *mgr* window.

SYNOPSIS

shape [<**columns**>] [<**rows**>]

DESCRIPTION

Shape Reshapes the window to the specified number of *columns* and *rows.* With no arguments *shape* makes an *80 column* by *24 row* window. With one argument, *shape* changes the number of lines to the number given, leaving the number of columns unchanged.

BUGS

Given unreasonable arguments, *shape* doesn't guarantee reasonable results

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

show – displays a bit-mapped image on a *mgr* window.

SYNOPSIS

show [**–r**] [**<x coord>**] [**<y coord>**] [**<bits wide>**]

DESCRIPTION

Show copies its standard input, which is assumed to be in *MGR bitmap format* to the window as a bit mapped image. The starting position of the bit map relative to the top left corner of the window may be given as *x coord* and *y coord* respectively. If *bits wide* is specified on the command line, **show** assumes no bitmap header is present. Specifying **–r** changes the sense of black and white.

Bit maps too big to fit on the window are clipped. The incoming data for each row should be rounded up to an even number of bytes. The bits are displayed left to right, then top to bottom.

BUGS

Large bit maps take too long to display.

SEE ALSO

mgr(1L) bitmap(5L)

NAME

snap – capture a portion of the display as a bitmap image

SYNOPSIS

snap [**–n**] <**file**>

DESCRIPTION

Snap lets a user capture a bit image of an arbitrary rectangular portion of the display. This image may be saved in a file, send to a printer, or copied back to the display.

When **snap** is active, the user may sweep out an image with the 3rd mouse button. Upon releasing the button, this image is captured and remembered by **Snap.** The middle button pops-up a menu with the following options:

Print

The image snapped is sent to the printer (via *lpr*) in *MGR bitmap format* with the **–v** flag set. If the **PRINTER** environment variable is set, it is used to specify the printer name, otherwise the image is sent to **lp .**

File

The last image snapped is saved in the file specified on the command line when **snap** was invoked, in *MGR bitmap format.* Successive invocations of **file** overwrite any previously stored images.

Review

Once **Review** is selected, The user may sweep out (using the 3rd mouse button, as before) an area on which to display the snapped image. If nothing is swept within 10 seconds, **snap** flashes, rings the bell, and reverts to capture mode. While reviewing in enabled, the *snap* icon remains inverse video. The displayed image is clipped to fit within the region swept out by the user. If the user simply clicks the 3rd mouse button twice without sweeping, **snap** copies the entire saved image to the display, starting at the mouse location.

Quit

Snap de-iconifies itself and quits.

When **–n** is specified on the command line, the new (8 byte header) style bitmap format is produced, instead of the old (6 byte header) format.

FILES

/dev/bwtwo0 to find the display image.

SEE ALSO

lpr(1) window_print(1L)

DIAGNOSTICS

* Can’t find screen *The frame buffer won’t open.*
* Can’t open file *The file can’t be opened for writing.*

BUGS

* The user interface is overly simplistic.
* **snap** only works on the *mgr* host.
* The **review** function can write on the display arbitrarily, destroying its integrity.

AUTHOR

S. A. Uhler

NAME

spot – Display information about the current mouse position

SYNOPSIS

spot

DESCRIPTION

Spot watches the mouse and reports it current state. The x and y coordinates of the mouse (in pixels) are always displayed, along with the current button pushed (if any). When the mouse cursor is over a window, **spot** also displays the contolling terminal for the window, its window and sub window id numbers.

If the mouse state has not changed for a while, **spot** goes to sleep (and displays a *(Z)*). When sleeping, it takes up to a second for **spot** to wake up and accurately track the mouse.

BUGS

Spot should allow the user to specify the desired output format, and automatically reshape its window to fit its display. Currently **spot** requires one row of 36 characters for its display

SEE ALSO

mgr(1L)

AUTHOR

S. A. Uhler

NAME

squeeze, unsqueeze – compress (or uncompress) MGR bitmaps.

SYNOPSIS

Squeeze reads an MGR bitmap from its standard input, squeezes it, and writes the result on its standard output. *Unsqueeze* does the inverse, reading a squeezed bitmap from standard input, unsqueezing it, and producing the normal bitmap format result on standard output.

Squeeze uses a simple run-length encoding strategy. The image is divided into packets, each with a single byte header. If the header byte value X, is greater than 128, then the next byte is repeated X-127 times. Otherwisse the next X+1 bytes of the image follow unchanged.

BUGS

Error checking is poor.

SEE ALSO

mgr(1L) bitmap(5L)

AUTHOR

M. E. Lesk

NAME

startup – produce a **startup** file reflecting the current *mgr* screen layout.

SYNOPSIS

startup

DESCRIPTION

Startup produces the current *mgr* window layout on its standard output, in a form suitable for the *mgr* startup file, **$HOME/.mgrc.**

BUGS

startup produces only the windows positions, neglecting the fonts and commands currently running in the windows.

SEE ALSO

mgr(1L)

NAME

stat – Display a strip chart of one or more current machine statistics.

SYNOPSIS

stat [**–c<color_map_indexes>**] [**–bsf<freq>** [[**–<max>**] **<parameter>**]] ...

DESCRIPTION

Stat is a graphical version of *vmstat* that runs on *mgr* terminals. **Stat,** with no options, displays the list of parameters it will chart.

The following options are recognized on the command line:

–b Do not update the display if the window is obscured. When the window is uncovered, the display rushes to catch up, instead of reflecting reality immediately.

–s Traces are drawn as thin lines, instead of solid filled.

–c If stat is run on a color version of MGR, various parts of the display are shown in different colors. This option alters the colormap index values used. The **–c** is immediately followed by a string of characters in the range of [0-9a-z] where each color represents a colormap index from 0 to 36 respectively. The position in the string determines what portion of the display is affected as follows:

#	use
0	unused
1	background
2	title
3	axis labels
4	grid lines
5	values that exeed the maximum
6	color for plot 1
7	color for plot 2 etc.

–f<freq>

The time interval between display updates is
freq
seconds. The default is 5 seconds.

-<max>

specifies the maximum value of the following parameter, in units appropriate for that parameter.

<parameter>

is a code that represents a particular statistic to plot.
The available parameters are:

r	jobs in run q
b	jobs blocked
w	jobs waiting
fre	free memory
fr	freed pages
d1	disk 1 accesses
d2	disk 2 accesses
d3	disk 3 accesses
d4	disk 4 accesses
in	interrupts
sy	system calls
cs	context switches
us	% user time

kn % system time
id % idle time

SEE ALSO

mgr(1L) vmstat(1)

DIAGNOSTICS

Window is not wide enough
Make the window wider and the graph will continue.

Window is not high enough
Make the window taller and the graph will continue.

BUGS

If the window is reshaped, *stat* requires up to **freq** (usually 3) seconds to learn about the new window size.

Stat calls *vmstat(1),* and assumes a particular (i.e. BSD 4.2) output format from *vmstat(1).*

NAME

stringart – A standard graphics demo

SYNOPSIS

stringart

DESCRIPTION

Stringart draws bunches of lines in a geometric patterns, erases them, and starts again with a different pattern.

SEE ALSO

mgr(1L)

NAME

Tjfilter – Bitmap *lpr* filter for the **HP ThinkJet** printer.

SYNOPSIS

tjfilter [**-r**]

DESCRIPTION

Tjfilter reads its *standard input* which is expected to be a one bit deep *bitmap* image in **mgr** format and transforms it for printing on an **HP ThinkJet** printer. If **-r** is specified, the bitmap is reversed (ala a photographic negative). For bitmaps which are wider than 640 dots, but less than 640 dots high, **tjfilter** calls **rotate(1L)** in order to fit the bitmap on the printer.

Tjfilter is normally used as the **vf** filter in the **HP ThinkJet** printcap entry: ...:vf=/usr/local/bin/tjfilter:...

BUGS

The **HP Thinkjet** printer can print a maximum of 640 dots per line. Bitmaps that don’t fit get truncated.

SEE ALSO

bitmap(5L) rotate(1L) lpr(1L)

NAME

vi – screen oriented (visual) display editor based on ex, with enhancements for MGR

SYNOPSIS

vi [**−t** tag] [**−r**] [+*command*] [**−l**] [**−w***n*] name ...

DESCRIPTION

Vi (visual) is a display oriented text editor based on *ex*(1). *Ex* and *vi* run the same code; it is possible to get to the command mode of *ex* from within *vi* and vice-versa.

The *Vi Quick Reference* card and the *Introduction to Display Editing with Vi* provide full details on using *vi.*

When the terminal type is **MGR**, *vi* prompts for the current window size. The command **^A** (CNTL-A) causes *vi* to re-request the window size information. If **^A** is included as part of the **MGR** reshape event, window sizes in *vi* are handled automatically.

A new option **font** has been added for **MGR** terminals that permit font changes on the fly. The command **set font=11** causes *vi* to begin editting using *font 11* .

FILES

See *ex*(1).

SEE ALSO

ex (1), edit (1), ‘‘Vi Quick Reference’’ card, ‘‘An Introduction to Display Editing with Vi’’.

AUTHOR

William Joy
Mark Horton added macros to *visual* mode and is maintaining version 3

BUGS

Software tabs using **^T** work only immediately after the *autoindent.*

Left and right shifts on intelligent terminals don’t make use of insert and delete character operations in the terminal.

The *wrapmargin* option can be fooled since it looks at output columns when blanks are typed. If a long word passes through the margin and onto the next line without a break, then the line won’t be broken.

Insert/delete within a line can be slow if tabs are present on intelligent terminals, since the terminals need help in doing this correctly.

Saving text on deletes in the named buffers is somewhat inefficient.

The *source* command does not work when executed as **:source**; there is no way to use the **:append**, **:change**, and **:insert** commands, since it is not possible to give more than one line of input to a **:** escape. To use these on a **:global** you must **Q** to *ex* command mode, execute them, and then reenter the screen editor with *vi* or *open.*

NAME

window_print – print an image of an **MGR** window on a printer.

SYNOPSIS

window_print [**-f** filter] [**-j** name] [**-m** message] [**-p** printer] [**-o** option] [**-x** file]

DESCRIPTION

Window print prints the images of windows on a printer. When first invoked, **Window print** iconifies itself with the message *Window dump*. A hard copy of a window is made by activating the window print window and clicking the third mouse button over the desired window. **Window_print** copies the image of the window onto a file, then invokes lpr to print it.

command options

-f filter
The name of an (optional) unix filter for converting the mgr bitmap format file into a form suitable for **lpr.**

-[jJ] name
The name printed on the burst page of the printer, normally **window.**

-m message
The string displayed in the iconified window.

-[pP] printer
The name of the printer, as in **lpr.**

-v option
Normally lpr is invoked with the **-v** flag. If **option** is specified, it is used instead.

-x file
The image files is copied to **file** instead of being sent to **lpr**

FILES

/tmp/pr* temporary bit image file

SEE ALSO

lpr(1)

DIAGNOSTICS

debugging output may be obtained by setting the environment variable **DEBUG.**

BUGS

The temporary file is created on the machine which is running mgr, not the machine running **window_print.**

NAME

zoom – an icon editor for *mgr*

SYNOPSIS

zoom <**icon file**> ...

DESCRIPTION

zoom is a mouse driven icon editter for *mgr*. **Zoom** divides the window into three regions, a banner line at the top containing four status fields, a message line at the bottom, and the remainder of the window for an enlarged, or *zoomed* representation of <*icon*> being editted. If the first file given on the command line is not an icon, zoom prompts for its width and height.

The current state of zoom is indicated by the four status fields in the banner line.

* The first, or *raster function* field displays the current raster-op function to be applied to the next edit operation. This function may be changed with the pop-up menu, activated by pressing the middle mouse button while the mouse track is in the *raster function* field. Normally the choices are **set, clear, toggle** and **grid.** The first three are raster-op functions; the *grid* option toggles the bitmap alignment grid. If the **put** command is pending (see below), the raster-op choices become **copy, paint mask,** and **exclusive-or.**

* The second, or *edit* field displays one of the six possible edit functions: **Yank, Put, Shrink, Grow, Fix,** and **Undo.** Fix and Undo are performed when selected. **Fix** changes the window size to give square pixels. **Undo** un-does the previous edit operation. If any of the other functions is selected, it becomes the pending function, and is highlighted. When a function is pending, the next sweep operation performs that function on the group of pixels enclosed by the sweeping rectangle (the *selected* pixels).

  Yank copies the *selected* pixels into the yank buffer.

  Put combines the yank buffer with the *selected* pixels in a manner determined by the current *raster function* field.

  Shrink makes the icon smaller by scaling the selected pixels to fill the entire window.

  Grow makes the icon bigger by scaling the entire icon to fit into the selected pixel region.

* The Third or *size* field displays the current *width* and *height* of the icon, in pixels. The size of the icon may be changed by selecting the pop-up menu when the mouse track is in the *size* field and responding to the prompt. While in the prompt window, the menu permits the selection of several standard icon sizes.

* The fourth, and final field is the *file* field. The *file* field displays the current file name of the icon. The filing options **Save, Get, Yank,** and Quit are, as usual, accessed by a pop-up menu when the mouse track is in the *file* field. The *file* options prompt for a file name. A list of all of the files specified on the command line is available via the pop-up menu within the prompt window.

  Save saves the icon by the specified name.

  Get edits a new icon , tossing the current icon into the bit-bucket.

  Yank copies the specified icon into the *yank* buffer for use with the **put** command.

  Quit quits Zoom. Quit does **NOT** save the icon. A save must be explicitly issued first. Zoom may also be terminated by typing "Q\r" to the window, or hitting your favorite interrupt key.

For those who are not particularly font of rodents, all of the **zoom** commands may be accessed via 1 or 2 letter keyboard commands (followed by a \r), some of which are:

R	Repaint window
x	toggle alignment grid
w	FIX window aspect ratio

u	UNDO
s1	select SET mode
s2	select CLEAR mode
s3	select TOGGLE mode
f	SAVE file
g	GET a new file
y	YANK a file
Q	QUIT
F1	select YANK function
F2	select PUT function
F3	SHRINK icon
F4	GROW icon
P0	set COPY mode
P1	set PAINT mode
P2	set MASK mode
P3	set XOR mode

Okay, now to edit the icon.

* Pressing the middle button and moving it either sets or clears the pixels it passes over. If the first pixel it touches is clear, the pixels will be set; if it is set, all touched pixels will be cleared.
* Holding, moving, then releasing the right mouse button sweeps out a rectangular region of *selected pixels.* If no function is currently highlighted in the *edit* field, the current raster-op function is performed on the selected pixels. Otherwise, the highlighted function is performed.

BUGS

* Zoom works best on small icons, running on the local machine.
* You can't view the actual size of the icon being editted.
* Icon coordinates must be typed in exactly in the form of **x** , **y** with no spaces or tabs.

SEE ALSO

browse(1L) dump(5L) mgr(1L)