Le système MGR Window

Vincent Broman — Edition 30 Mai 1996

Contents

1 Cet HOWTO

1.1 Archivage

Cet HOWTO est archivé sur `ftp://archimedes.nosc.mil/pub/Mgr/MGR-HOWTO.sgml`, et également sur `ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/MGR-HOWTO`. Dans les répertoires adjacents peuvent apparaître des fichiers de formats différents tels que `MGR-HOWTO.txt`.

1.2 Authentification

Les copies de la distribution MGR dues à Broman doivent être accompagnées des fichiers de signature PGP, signées "Vincent Broman <broman@nosc.mil>".

1.3 Crédits pour cet HOWTO

Bien que Vincent Broman ait été l'initiateur de cet HOWTO, la plupart des informations et des textes furent obtenus à partir de FAQs, READMEs, etc. écrits par Stephen Uhler, Michael Haardt, et d'autres personnes ayant "l'esprit Réseau". Corrections et suggestions par Email à `broman@nosc.mil`.

Uhler fut l'architecte principal de **MGR** – voir Remerciements plus loin.

2 Qu'est-ce que MGR ?

2.1 Fonction

MGR(ManaGeR) est un système de fenêtres graphiques. **MGR** fournit un gestionnaire intégré de fenêtres et un émulateur de terminal graphique sur des systèmes bitmaps couleurs et monochromes. **MGR** est contrôlable par des menus pop-up activés par souris, par action du clavier, et par des séquences d'échappement écrites par un client sur des pseudo-terminaux.

MGR fournit à chaque client une fenêtre avec : fonctions de contrôle terminal en style termcap, primitives graphiques telles que dessins de lignes et de cercles ; aménagements pour manipuler les bitmaps, fontes, icônes et menus pop-up ; commandes pour redessiner et repositionner les fenêtres ; et un système de passage de messages permettant aux programmes-client d'établir des rendez-vous et d'échanger des messages. Les programmes client peuvent demander à être informés lorsqu'un changement dans le système intervient, tel qu'une fenêtre redessinée, un clic de souris, ou un message envoyé à partir d'un autre programme client. Ces changements sont appelés événements. **MGR** notifie un événement à un programme client en envoyant une chaîne ASCII dans un format spécifié par le programme client. Des applications existantes peuvent être intégrées dans l'environnement sans être modifiées : **MGR** imite des actions sur le clavier en réponse à des sélections du menu utilisateur ou d'autres événements.

2.2 Conditions requises

MGR tourne couramment sur Linux, FreeBSD, stations Sun 3/4 avec SunOS, et Coherent. Certaines versions anciennes de **MGR** tournent sur Macintosh, Atari ST MiNT, Xenix, 386-Minix, DEC 3100, et 3b1 Unix-pc. De nombreux petits systèmes industriels en temps réel sous OS9 et Lynx utilisent en Europe (une autre variante de) Mgr comme interface. L'interface de programmation est implémentée en C et Elisp, cependant des supports clients dans d'autres langages ne posent pas de difficultés.

MGR requiert beaucoup moins de ressources que X, ou même gcc. Bien sûr il n'a pas le répertoire de programmes, les bibliothèques de haut niveau de X ou MS-Windows, mais il est élégant et abordable.

On a dit que **MGR** est à X ce que Unix est à Multics.

2.3 Quelles sont les différences entre MGR, X11 et 8.5 ?

MGR consiste en un serveur avec un gestionnaire de fenêtres et un émulateur de terminal, et des clients qui tournent dans cet émulateur en l'utilisant pour communiquer avec le serveur. Il n'y a pas de multiplexage de ressources.

X11 est constitué d'un serveur et de clients, qui sont habituellement connectés au serveur via un socket. Ce qui est visible à l'utilisateur tels qu'émulateurs de terminal, gestionnaires de fenêtres, etc., est réalisé en utilisant les clients.

8.5, le système de fenêtre Plan 9, est un multiplexeur de ressources ; tout processus tournant dans une fenêtre peut accéder à `/dev/bitblt`, `/dev/mouse` et `/dev/kbd` sous son propre nom. Ceux-ci sont multiplexés à `/dev/bitblt`, `/dev/mouse` et `/dev/kbd` de 8.5. Cette approche permet de faire fonctionner 8.5 dans une fenêtre 8.5, de conception très soignée. De plus 8.5 dispose d'un gestionnaire de fenêtres intégré et un émulateur de terminal.

3 Installer MGR

On peut charger la dernière version sur `archimedesbugs.nosc.mil/pub/Mgr/69` et sous Mosaic depuis `http://archimedes.nosc.mil/Mgr/69`. On peut le trouver aussi sur `ftp://sunsite.unc.edu/pub/Linux/apps/MGR` et ses miroirs. Les versions anciennes de Haardt peuvent se trouver sur `tsx-11.mit.edu` et peut-être ailleurs. Des versions Pre-Linux de **MGR** de Uhler et d'autres ont été trouvées sur `bellcore.com/pub/mgr`, mais je pense qu'elles n'y sont plus. J'ai conservé une copie de tout ce que j'ai vu concernant MGR sur l'Internet, et je n'ai pas connaissance qu'il n'y ait quoi que ce soit d'important qui puisse manquer dans cette distribution Linux/Sun. Il y a eu un tas de versions et de mises en circulation de **MGR**, mais la version *Linux* actuelle est 0.69. Cette version devrait passer à 1.0 quand un code VGA-256 stable pour Linux apparaîtra (pour plusieurs types de cartes vidéo). Les numéros de version RCS sont passés de Bellcore 4.3 jusqu'à 4.13 à ce jour.

Pour construire cette distribution de **MGR** il faut m4 (GNU ou tout autre admettant l'option -D), make (GNU, ou tout autre admettant include) et *roff pour les docs. Et enfin sh, awk et install POSIX. Les distributions binaires n'ont pas toujours assemblées aussi faut-il un compilateur C ANSI, du style gcc.

Une installation sous Linux requiert au moins une version 0.99.10 ou ultérieure (1.2.13 est ce que j'utilise à l'heure actuelle) , une carte graphique HGC, EGA, VGA ou SVGA et une souris de type : serial Microsoft, serial MouseSystems 3 et 5 bytes, serial MMSeries, serial Logitech, PS/2, ou une souris bus. Avec Buckey (Meta) hot keys en service, même un système ne possédant pas de souris peut effectuer pas mal de travail utile sous MGR. Le mode graphique monochrome VGA 640x480 est supporté, tout comme le 640x350 et le 640x200. Pour tourner en 800x600, ou d'autres modes que votre BIOS peut initialiser et qui n'ont pas besoin de bank-switching, vous aurez besoin d'un petit programme (`src/vgamisc/regs.exe`) sous DOS pour lire les registres VGA quand ce mode est installé puis d'écrire un fichier header que vous placerez dans le répertoire `src/libbitblit/linux`, de telle sorte qu'il puisse être appelé par le fichier vga.c. Des exemples sont fournis, mais créez quand même le vôtre. Quelques cartes VGA peuvent utiliser des fenêtres de 128k ; celles-ci peuvent tourner avec de plus hautes résolutions.

Le code Linux-colorport peut aussi tourner dans le mode couleur standard VGA 320*200*256 sans difficultés, car il n'y a pas de bank-switching nécessaire. Si vous réfléchissez au peu que représentent 64000 pixels, vous comprendrez que ce mode couleur est tout à fait limité. Un code lent, mais simple, a été ajouté dans la version 0.65, et il travaille avec une carte Tseng ET4000 dans les modes 640x480x256 et 800x600x256. Le code S3 ne marche pas encore dans les résolutions super-VGA. Pour utiliser des nouvelles cartes super-VGA il faut écrire une fonction pour changer de bloc de mémoire et être sûr que le mode écran souhaité peut être initialisé à partir d'un vidage de registre, éventuellement en le peaufinant à la main. Les serveurs couleur sous Linux déforment généralement les fontes écrans, d'où la nécessité d'utiliser restorefont comme dans runx. Si quelqu'un pouvait récupérer le code d'initialisation VGA de X, cela ferait de MGR un système possédant plus de couleurs.

Suns avec SunOS 4.1.2+ et les tampons de trame `bwtwo`, `cgthree`, ou `cgsix` sont acceptés. Leurs vitesses de manipulation des couleurs sont correctes. Les systèmes Coherent doivent se référer au fichier `README.Coh` de la distribution source. Porter le tout dernier **MGR** sur un autre système analogue à POSIX qui possède `select`, des pty et un accès direct à un tampon de trame bitmap devrait être immédiat, en implémentant simplement la bibliothèque `libbitblit` basée sur le code `sunmono` ou `colorport`.

Si vous voulez tout installer, vous aurez besoin de 7 MB d'espace disque pour les binaires, les fonts, les explications, etc. Les sources font environ 4,5 MB plus les fichiers objets durant la compilation.

Normalement, `/usr/mgr` doit être le répertoire ou bien il doit être linké au répertoire où vous installerez les accessoires **MGR**. Tapez

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

et éventuellement

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

pour décompacter. Les sources peuvent être mises n'importe où, par exemple tapez

```
cd /usr/src/local/mgr; tar xvfz la_ou_vous_voulez/mgrsrc-0.69.tgz
```

pour décompacter les sources à partir de `archimedes.nosc.mil`.

L'arborescence source peut être compilée à partir d'un Makefile principal qui fait appel lui-même à des Makefile secondaires, chacun “incluant” un `"Configfile"` au niveau supérieur. Le `Configfile` est créé à l'aide d'un script interactif nommé Configure, qui vous pose un certain nombre de questions, puis utilise m4 avec un fichier `Configfile.m4`. Ensuite vous tapez quelque chose comme ceci :

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

Il est prudent, avant de taper make, de jeter un coup d'oeil dans le fichier `Configfile` généré par le script `Configure`, pour vérifier s'il n'y a pas d'anomalie. Au pire m4 s'interrompt, créant un fichier Configfile tout petit. Si cela arrive, essayez une copie de `Configfile.sun` ou `Configfile.lx`. Tout peut être effectué dans n'importe quel répertoire avec un Makefile à partir du moment où les bibliothèques ont été compilées et installées. Le serveur, les bibliothèques et quelques clients sont implémentés avec `lint`, mais plusieurs clients utilisent le code C K&R ce qui engendre beaucoup d'avertissements pendant la compilation. Plusieurs flags dans MGRFLAGS peuvent être ajoutés ou retranchés pour changer quelques options du serveur, à savoir :

-DWHO

fichier utmp poubelle pour que "who" puisse travailler

-DVI

code pour faire bouger le curseur sous vi avec la souris

-DDEBUG

permet la mise au point en sélectionnant l'option -d

-DFASTMOUSE

XOR le temps d'accès de la souris

-DBUCKEY

permet les commandes serveur par le clavier sans souris

-DPRIORITY

pour planifier la priorité des fenêtres au lieu de fonctionner par requêtes circulaires ; la fenêtre active obtient une plus grande priorité.

-DCUT

pour faire du couper/coller entre fenêtres

-DMGR_ALIGN

force l'alignement des fenêtres pour un scrolling rapide(monochrome)

-DKILL

supprime les fenêtres en cas d'erreurs i/o sur un tty

-DSHRINK

pour utiliser seulement une partie de l'écran ($MGRSIZE dans les variables d'environnement)

-DNOSTACK

interdit l'empilement d'événements

-DBELL

actionne le bip

-DKBD

lit les données `mgr` à partir du clavier sun, au lieu de stdin. Cela autorise la redirection des messages console vers une fenêtre.

-DFRACCHAR

mouvement de caractère fractionnel pour fontes proportionnelles

-DXMENU

menu étendu (expérimental)

-DMOVIE

extension pour faire un film qui enregistre toutes les opérations dans un fichier en vue de les rejouer plus tard (ne marche pas bien sous Linux).

-DEMUMIDMSBUT

Pour simuler le 3ème bouton d'une souris

Méfiez-vous : certaines combinaisons n'ont pas été essayées sur tous les systèmes. La macro BITBLITFLAGS doit contenir `-DBANKED` si vous voulez essayer le super VGA couleur.

Le code C pour les variables statiques du serveur contenant les icônes et les fontes est généré à l'aide d'un traducteur à partir des fichiers icônes et fontes.

Tous les clients ne sont pas compilés et installés par les fichiers Makefile. Les fichiers clients trouvés dans `src/clients` qui ont des noms en majuscule ou bien non compilés par les fichiers Makefile fournis, peuvent donner des problèmes de compilation et/ou d'utilisation ; il peut être intéressant de les examiner.

Plusieurs pilotes d'écran du répertoire `libbitblit` ont plutôt un intérêt historique. Mieux vaut les faire disparaître.

Vérifiez que votre fichier /`etc/termcap` et/ou `terminfo` contient des entrées pour les terminaux **MGR**, comme ceux que l'on trouve dans le répertoire `misc`. Si vos logiciels prennent en compte $TERMCAP dans l'environnement, ceci n'est pas nécessaire, tant que vous utilisez `set_termcap` dans chaque fenêtre.

Il est préférable de travailler avec **MGR** avec setuid root, car il utilise ptys et write dans le fichier utmp. Cela permet au client gestionnaire d'icônes de mieux travailler et d'avoir une plus grande sûreté quant à l'enchaînement des événements. Sur Linux, les permissions root sont nécessaires pour faire fonctionner les systèmes écran. Autrement vous décidez de lui faire confiance.

Avec les versions aux alentours de 0.62 il y a des problèmes avec Sun quand on utilise csh comme shell par défaut : les programmes semblent tourner sur un processus différent que le processus de premier plan du pty de la fenêtre, en contradiction avec les pages de manuel et les spécifications posix. Rien de tel avec bash, sh, ou rc. Vous avez une idée ?

4 Utiliser MGR

Le seul fichier *indispensable* sous **MGR** est le serveur lui-même. Il vous donnera les fenêtres émulateur de terminal avec les shells correspondants, mais pas de jolies horloges, de fontes superbes, de graphiques marrants, etc... Suivant les options, le serveur monochrome a besoin de 200K de RAM plus un espace dynamique pour les fenêtres, bitmaps, etc...

Si `/usr/mgr/bin` est dans votre PATH, tapez seulement "`mgr`" pour démarrer. Après avoir profité de l'écran de démarrage animé, frappez une touche quelconque. Quand le fond hachuré apparaît avec un pointeur de souris, appuyez sur le bouton gauche de la souris, allez sur "new window" dans le menu puis relâchez. Faites promener la souris pour sélectionner l'endroit où vous voulez qu'une fenêtre apparaisse. Celle-ci aura votre shell par défaut. Maintenez le bouton gauche de la souris enfoncé dans une fenêtre existante pour voir un autre menu qui vous permettra de réaliser des choses dans cette fenêtre. Cliquer avec le bouton gauche sur une fenêtre obscurcie l'amènera au premier plan. Le menu que vous aviez vu sur l'arrière plan inclut la commande 'quit'. Pour ceux qui ont une souris avec deux boutons, il suffit d'appuyer sur les deux boutons simultanément pour simuler le troisième. Le sous-menu quit comprend l'option "really quit", une option d'attente qui doit être utilisée seulement si vous utilisez un shell offrant l'édition de la ligne de commande, et un économiseur d'écran avec un verrouillage qui attend que vous tapiez un mot de passe lorsque vous revenez sur votre machine.

En essayant **MGR**, si vous :

ne pouvez trouver l'écran

soyez sûr d'avoir une entrée /dev pour votre vidéo, par ex. sur Sun `/dev/bwtwo0`. Si ce n'est pas le cas, en tant que root allez dans `/dev`, et tapez "MAKEDEV bwttwo0". Sinon, vous devez faire `-S/dev/bwtwo0` ou (sous Linux) `-S640x480` comme option de commande en démarrant `mgr`. Sous Linux, soyez également sûrs que /usr/mgr/bin/mgr a été installé suid root.

ne trouvez pas la souris

assurez-vous que `/dev/mouse` existe, habituellement lié symboliquement au nom réel de votre souris. Si vous n'avez pas la permission d'écrire dans `/dev`, quelque chose comme `-m/dev/cua0` peut être donné comme option en démarrant `mgr`. Soyez également sûrs d'avoir mis le bon protocole souris en configurant `mgr`. La souris peut s'appeler Microsoft, même si ce n'est pas son vrai nom.

ne pouvez obtenir un pty

assurez-vous que tous les `/dev/[tp]ty[pq]?` sont propriété de root, mode 666, que tous les programmes référencés avec l'option "shell" dans le fichier `.mgrc` (si il y en a) existent et sont exécutables.

n'avez rien d'autre que la fonte par défaut

assurez-vous que **MGR** cherche bien au bon endroit pour les fontes. Vérifiez le fichier `Configfile` dans les sources, ou bien regardez si une option comme `-f/usr/mgr/font` résoud le problème.

êtes complètement bloqué (même le pointeur souris ne bouge pas)

logez vous sur votre machine à partir d'un autre terminal et tuez le processus `mgr`. Un ctrl-Q fera quitter **MGR** si le clavier fonctionne encore.

4.1 Applications non liées à MGR

Toute application orientée tty peut tourner sous une fenêtre MGR sans problèmes. Les applications orientées écran utilisant termcap ou curses peuvent obtenir le nombre exact de lignes et de colonnes en utilisant `shape(1)` pour redimensionner la fenêtre ou en utilisant `set_termcap(1)` pour obtenir le termcap adéquat.

4.2 Applications (clients) MGR distribuées avec le serveur

bdftomgr

convertit des fontes BDF en fontes MGR

browse

un browser d'icônes

bury

enterre la fenêtre

c_menu

pour regarder les erreurs de compilation sous C à l'aide de vi

clock

horloge digitale

clock2

horloge analogique

close

ferme la fenêtre et l'iconifie

color

pour les couleurs d'arrière et d'avant-plan du texte dans la fenêtre

cursor

change l'aspect du curseur texte

cut

couper/coller du texte de la fenêtre vers une mémoire tampon

cycle

affiche une séquence d'icônes

dmgr

prévisualisateur ditroff à l'état brut

fade

fait passer d'une scène à une autre dans un film

font

passe à une autre fonte dans la fenêtre

gropbm

un pilote groff PBM utilisant les fontes Hershey

hpmgr

émulateur de terminal hp 2621

ico

anime un isocaèdre ou un autre polyèdre

iconmail

annonce l'arrivée de courrier

iconmsgs

annonce l'arrivée d'un message

ify

iconifie et désiconifie les fenêtres

loadfont

charge une fonte à partir du système de fichiers

maze

jeu de labyrinthe

mclock

horloge comique

menu

crée ou choisit un menu pop-up

mgr

Gestionnaire de fenêtres et serveur Bellcore

mgrbd

jeu boulder-dash

mgrbiff

surveille la boîte aux lettres et annonce le courrier

mgrload

graphique indiquant la charge du système

mgrlock

verrouille la console

mgrlogin

contrôleur graphique de login

mgrmag

loupe sur une partie de l'écran, avec option de sauvegarde

mgrmail

annonce l'arrivée de courrier

mgrmode

ajuste ou annule les modes de fenêtre

mgrmsgs

annonce l'arrivée de messages

mgrplot

filtre graphique Unix "plot"

mgrsclock

sablier

mgrshowfont

parcourt les fontes mgr

mgrsketch

programme de schémas et dessins

mgrview

visualise des images bitmap

mless

démarre less/more dans une fenêtre séparée, avec un menu pour less wtag/mnew/démarrer n'importe quel programme dans une fenêtre séparée indépendante.

mvi

démarre vi dans une fenêtre séparée, avec souris

oclose

ferme une fenêtre (ancien)

omgrmail

annonce l'arrivée de courrier (ancien)

pbmrawtomgr,pgmrawtomgr,ppmrawtomgr

convertit des bitmats PBM,PGM,PPM brutes en format bitmap mgr

pbmstream

fractionne une séquence de bitmaps

pbmtoprt

impression à partir de PBM

pgs

un patch ghostscript et interface, un visualisateur PS

pilot

balaye des bitmaps et visualise des images

resetwin

fait le ménage dans une fenêtre si le client se plante

rotate

rotation d'un bitmap de 90 degrés.

screendump

sauvegarde un écran graphique dans un fichier bitmap

set_console

redirige les messages console vers une fenêtre

set_termcap

ajuste une valeur TERMCAP appropriée

setname

nomme une fenêtre, pour les messages et pour l'iconifier

shape

redimensionne une fenêtre

square

transforme une fenêtre en carré

squeeze

compresse un bitmap mgr

startup

fournit un fichier de démarrage pour la disposition de la fenêtre courante

texmgr

pour prévisualiser un fichier TeX dvi

text2font, font2text

conversion entre formats fonte mgr et un texte dump

unsqueeze

pour décompresser un bitmap mgr

vgafont2mgr,mgrfont2vga

conversion de format de fontes mgr et VGA

window_print

imprime l'image d'une fenêtre

zoom

éditeur d'icônes

bounce, grav, hilbert, mgreyes, stringart, walk

démos graphiques

4.3 Applications MGR distribuées séparément, cf fichier "SUPPORT"

calctool

calculatrice

chess

interface pour `/usr/games/chess`

gnu emacs

éditeur avec souris `lisp/term/mgr.el` et aide menu

gnuplot

traceur de données scientifiques universel

metafont

création et conception de fontes

origami

éditeur de dossier

pbmplus

conversions et manipulations de format portable bitmap

plplot

traceur de données scientifiques superbe

Le support Emacs dans misc/mgr.el et misc/mailcap comprend un support MIME très utile, via Rmail et metamail. Un afficheur d'image de différents types pourrait être fabriqué à partir d'un pilote et de filtres netPBM, mais je n'ai pas pris le temps de le faire.

5 Programmation pour MGR

Le manuel des programmeurs **MGR**, l'interface des applications langage C, se trouvent dans le répertoire doc sous forme de fichier exploitable par troff/nroff. Il traite de concepts généraux, des appels fonction/macro contrôlant le serveur, d'un exemple d'application, avec index et glossaire. Porter le code client utilisé avec les anciennes versions de MGR demande le remplacement de

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

par

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

et des clients utilisant les vieux B_XOR, B_CLEAR, etc. au lieu de BIT_XOR, BIT_CLR et autres peuvent être adaptés en écrivant :

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

Compiler le code client demande en général des options telles que :

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

Vous pouvez obtenir un aperçu de l'interactivité des fonctions serveur **MGR** en lisant et essayant le pilote de terminal `mgr.el` pour GNU Emacs, qui met en oeuvre la bibliothèque interface **MGR** en Elisp. L'habitude qui consiste à s'enquérir de l'état du serveur a pour risque de se casser la figure si le client attend en même temps un grand volume de notification d'événements. Ce problème arrive lorsque une notification d'événement (asynchrone) survient quand une réponse à une demande (synchrone) était attendue. Si cela arrive dans la pratique (non habituel) alors les fonctions de demande d'état **MGR** doivent être intégrées avec votre boucle de manipulation d'événement.

La seule fonction manquante pour les dessins dans le protocole **MGR** est celle de remplissage de surfaces autres que les rectangles. Il y a un nouveau code pour manipuler la carte des couleurs globale, et aussi pour l'allocation et la délivrance des indices de couleur appartenant à chaque fenêtre. Si vous voulez fouiller dans les progammes serveurs, vous trouverez le pilote de souris dans `mouse.*` et `mouse_get`, les abominables choses concernant l'interface clavier dans `kbd.c`, et l'interface vidéo dans les répertoires `src/libbitblit/*`. La procédure principale, plutôt l'initialisation, et la boucle d'entrée de niveau supérieur sont dans `mgr.c`, enfin l'interprétation des séquences d'échappement dans `put_window.c`.

6 Documentation supplémentaire

Le manuel du programmeur est indispensable pour les idées générales. La plupart des clients sont fournis avec une documentation man installée dans `/usr/mgr/man/man1` ou `/man6`.

D'autres documentations utiles sont `bitblit.3`, `font.5`, et `bitmap.5`. Il y a une ambiguïté dans les documentations pour distinguer le format des bitmaps internes trouvés dans votre tampon de trame et le format des bitmaps externes trouvés dans des fichiers, par exemple les icônes.

La documentation `mgr.1` traite des options de commande en ligne, des commandes dans le fichier `.mgrc`, de la souris et des interactions du menu avec le serveur, et enfin des raccourcis clavier disponibles sur sur les systèmes qui en possèdent. Beaucoup de fontes de `/usr/mgr/font/*` sont décrites dans les fichiers `/usr/mgr/font/*.txt`, par exemple `/usr/mgr/font/FONTDIR.txt` donne la description des fontes de style X pour celles obtenues dans le format .bdf. La fin des noms de fontes (`WxH`) s'interprète comme suit : `W` est la largeur du caractère en décimal `H` est la hauteur en pixels.

7 Remerciements

Stephen Uhler, avec d'autres travaillant chez Bellcore, fut le concepteur initial de **MGR**, dès lors Bellcore a des droits d'auteur sur la plupart des codes et des documentations de **MGR**, dans les conditions suivantes :

```
Vous avez la permission de copier et d'utiliser ce programme,
MAIS
* vous ne pouvez le vendre
* cette note concernant les droits de copie doit accompagner les copies
* mentionner Bellcore quand c'est necessaire
```

Une vue de la notice de copyright apparaît dans le titre de démarrage.

Autres remerciements :

- Stephen Hawley pour ses merveilleuses icônes.
- Tommy Frandsen pour la bibliothèque VGA linux.

- Tom Heller pour sa bibliothèque.
- Andrew Haylett pour le code du pilote de souris.
- Dan McCrackin pour ses patches gasblit -> linux.
- Dave Gymer, dgymer@gdcarc.co.uk, pour la correction de l'effet Startrek.
- Alex Liu pour la première version de Linux de **MGR**.
- Lars Aronsson (aronsson@lysator.liu.se) pour les fontes tex2 et ISO8859.
- Harry Pulley (hcpiv@grumpy.cis.uoguelph.ca, hcpiv@snowhite.cis.uoguelph.ca) pour l'accès Coherent.
- Vance Petree & Grant Edwards & Udo Munk pour leur travail sur Hercules.
- Udo Munk pour son travail sur l'initialisation et la sélection de la souris série.
- Norman Bartek, Hal Snyder de Mark Williams Co. pour leur aide sur quelques bugs et les pilotes de périphérique Coherent.
- Grand merci à Zeyd Ben Halim pour des tas de patches utiles, spécialement pour adapter les systèmes de sélection.
- Bradley Bosch (brad@lachman.com) pour des tas de patches depuis le port accès 3b1, qui corrige des bugs et permet d'installer des accessoires nouveaux et attractifs.
- Andrew Morton (applix@runxtsa.runx.oz.au) qui écrivit le premier le code cut-word.
- Kapil Paranjape (kapil@motive.math.tifr.res.in) pour le support EGA.
- Michael Haardt pour les corrections du support MOVIE, les corrections de bugs, la séparation du code libbitblit en pilotes de sortie, l'extension de libmgr, la compression origami du code.
- Yossi Gil pour de nombreuses fontes.
- Carsten Emde, carsten@thlmak.pr.net.ch, pour mphoon.
- Vincent Broman pour l'émulation du 3ème bouton de la souris, le support cgsix Sun, l'accès à la table des couleurs VGA, et l'intégration du code sunport de l'arrangement en couches Haardt, le rassemblement des fontes, l'économiseur d'écran, et la maintenance continue.
- Kenneth Almquist, ka@socrates.hr.att.com, pour les salutaires reports de bogues.
- Tim Pierce, twpierce@midway.uchicago.edu, pour le portage vers FreeBSD 2.0R avec la carte VGA Trident.

Toutes les fontes bitmap de toutes les sources sont du domaine public aux USA. Les fontes 583 à largeur fixe fournies avec **MGR** furent obtenues par Uhler, dans la distribution X, Yossi Gil, et autre part. Les fontes vectorielles Hershey et le code nécessaire sont probablement librement redistribuables.