Framebuffer HOWTO

Alex Buell, alex.buell@tahallah.clara.co.uk v1.2, 27 febbraio 2000

Questo documento descrive come utilizzare i dispositivi framebuffer in Linux per una varietà di piattaforme. Inoltre comprende spiegazioni per configurare più uscite (ovvero, una configurazione "multi-headed") per gli schermi. Documentazione tradotta in italiano e mantenuta da Manuele Rampazzo - *manu@linux.it* .

Contents

1 Storia

Cronologia delle versioni

19990607 - Rilascio della 1.0

19990722 - Rilascio della 1.1

20000222 - Rilascio della 1.2

2 Contributori

Ringraziamenti vanno alle persone elencate qui di seguito per l'aiuto nel perfezionamento del Framebuffer HOWTO.

- Jeff Noxon `jeff@planetfall.com`
- Francis Devereux `f.devereux@cs.ucl.ac.uk`
- Andreas Ehliar `ehliar@futurniture.se`
- Martin McCarthy `marty@ehabitat.demon.co.uk`
- Simon Kenyon `simon@koala.ie`
- David Ford `david@kalifornia.com`
- Chris Black `cblack@cmpteam4.unil.ch`
- N Becker `nbecker@fred.net`
- Bob Tracy `rct@gherkin.sa.wlk.com`
- Marius Hjelle `marius.hjelle@roman.uib.no`
- James Cassidy `jcassidy@misc.dyn.ml.org`
- Andreas U. Trottmann `andreas.trottmann@werft22.com`
- Lech Szychowski `lech7@lech.pse.pl`
- Aaron Tiensivu `tiensivu@pilot.msu.edu`

- Jan-Frode Myklebust per le sue informazioni sulle schede permedia`janfrode@ii.uib.no`
- Molti altri troppo numerosi da aggiungere, ma grazie!

Ringraziamenti vanno a Rick Niles `frederick.a.niles@gsfc.nasa.gov` che, molto gentilmente, ha concesso il suo Multi-Head Mini-HOWTO per l'inclusione in questo HOWTO.

Ringraziamenti alle persone elencate qui di seguito per l'aver creato versioni libc5/glibc2 del driver framebuffer X11 XF86_FBdev per l'X11 su piattaforme Intel:

- Brion Vibber `brion@pobox.com`
- Gerd Knorr `kraxel@cs.tu-berlin.de`

e naturalmente agli autori dei dispositivi framebuffer:

- Martin Schaller - autore originario del concetto di framebuffer
- Roman Hodek `Roman.Hodek@informatik.uni-erlangen.de`
- Andreas Schwab `schwab@issan.informatik.uni-dortmund.de`
- Guenther Kelleter
- Geert Uytterhoeven `Geert.Uytterhoeven@cs.kuleuven.ac.be`
- Roman Zippel `roman@sodom.obdg.de`
- Pavel Machek `pavel@atrey.karlin.mff.cuni.cz`
- Gerd Knorr `kraxel@cs.tu-berlin.de`
- Miguel de Icaza `miguel@nuclecu.unam.mx`
- David Carter `carter@compsci.bristol.ac.uk`
- William Rucklidge `wjr@cs.cornell.edu`
- Jes Sorensen `jds@kom.auc.dk`
- Sigurdur Asgeirsson
- Jeffrey Kuskin `jsk@mojave.stanford.edu`
- Michal Rehacek `michal.rehacek@st.mff.cuni.edu`
- Peter Zaitcev `zaitcev@lab.ipmce.su`
- David S. Miller `davem@dm.cobaltmicro.com`
- Dave Redman `djhr@tadpole.co.uk`
- Jay Estabrook
- Martin Mares `mj@ucw.cz`
- Dan Jacobowitz `dan@debian.org`
- Emmanuel Marty `core@ggi-project.org`
- Eddie C. Dost `ecd@skynet.be`
- Jakub Jelinek `jj@ultra.linux.cz`
- Phil Blundell `philb@gnu.org`
- Chiunque altro, si alzi e sarà contato. :o)

3 Cos'è un device framebuffer?

Un device framebuffer è un'astrazione dell'hardware grafico. Rappresenta il buffer dei frame di alcuni hardware video e permette alle applicazioni software di accedere all'hardware grafico attraverso un'interfaccia ben definita, in modo tale che il software non abbia bisogno di conoscere nulla riguardo le faccende dell'interfaccia di basso livello [Tratto dal framebuffer.txt di Geert Uytterhoeven nelle sorgenti del kernel di linux].

4 Che vantaggi hanno i dispositivi framebuffer?

Il logo del Pinguino. :o) Seriamente, il maggior vantaggio dei driver framebuffer è che questi rappresentano una generica interfaccia per tutte le piattaforme. La situazione fino agli ultimi sviluppi del kernel 2.1.x era che le piattaforme Intel avevano driver per console completamente differenti da quelli delle altre piattaforme. Con l'introduzione del 2.1.109 tutto ciò è cambiato in meglio, introducendo una gestione più uniforme delle console per piattaforme Intel, nonché reali console grafiche bitmap portando il logo del Pinguino per la prima volta su Intel, e permettendo al codice d'essere condiviso su differenti piattaforme. Da notare che i kernel 2.0.x non supportano i dispositivi framebuffer, ma è possibile che qualcuno faccia un giorno un port all'indietro del codice dai kernel 2.1.x a quelli 2.0.x. C'è un'eccezione alla regola in quanto il port per piattaforme m68k del kernel v0.9.x ha incluso in sé il supporto per il device framebuffer.

Col rilascio del kernel 2.2.x, il supporto del device framebuffer è molto solido e stabile. Dovresti usare il device framebuffer se la tua scheda grafica lo supporta e se stai usando kernel 2.2.x. I precedenti kernel 2.0.x non supportano i device framebuffer, almeno per le piattaforme Intel.

- 0.9.x (m68k) - introdotto il device framebuffer dell'm68k. Da notare che il 0.9.x dell'm68k è funzionalmente equivalente all'1.0.9 dell'Intel (più gli sviluppi dell'1.2.x)
- 2.1.107 - introdotti i device per framebuffer/nuove console su Intel ed aggiunto un supporto generico, senza il supporto per lo scorrimento all'indietro del buffer.
- 2.1.113 - aggiunto il supporto per lo scorrimento all'indietro del buffer al vgacon.
- 2.1.116 - aggiunto il supporto per lo scorrimento all'indietro del buffer al vesafb.
- 2.2.x - inclusi matroxfb(Matrox) e atyfb(ATI).

Ci sono alcune simpatiche caratteristiche sui device framebuffer, cui puoi attribuire opzioni generiche al kernel al momento dell'avvio, incluse opzioni specifiche per un particolare device framebuffer. Queste sono:

- `video=xxx:off` - disabilita il rivelamento per uno specifico device framebuffer
- `video=map:octal-number` - associa la console virtuale (VCs) al device framebuffer (FB)
  - `video=map:01` Associerà VC0 a FB0, VC1 a FB1, VC2 a FB0, VC3 a FB1...
  - `video=map:0132` Associerà VC0 a FB0, VC1 a FB1, VC2 a FB3, VC4 a FB2, VC5 a FB0

Normalmente i device framebuffer son rivelati nell'ordine specificato nel kernel, ma specificando l'opzione `video=xxx` si può aggiungere lo specifico device framebuffer che si vuole rivelare prima degli altri specificati nel kernel.

5 Utilizzare i device framebuffer sulle piattaforme Intel

5.1 Cos'è vesafb?

Vesafb è un driver framebuffer per l'architettura Intel funzionante con schede grafiche compatibili VESA 2.0, in stretta relazione coi device driver del kernel per il framebuffer.

Vesafb è un driver per schermi che attiva l'utilizzo delle modalità grafiche sulla tua piattaforma Intel per console testuali bitmapped. Può inoltre visualizzare un logo, che è probabilmente la principale ragione per cui vorresti usare il vesafb :o)

Sfortunatamente, non puoi utilizzare con successo il vesafb con schede VESA 1.2. Questo perché queste schede 1.2 non utilizzano un frame buffering *lineare*. Frame buffering lineare significa semplicemente che la CPU del sistema è in grado di accedere ogni bit dello schermo. Storicamente, vecchi adattatori grafici permettevano alla CPU di accedere solo a 64K per volta, di qui le limitazioni delle spaventose modalità grafiche CGA/EGA! Può darsi che qualcuno scriva un device driver vesafb12 per queste schede, ma ciò consumerà preziosa memoria del kernel e comporterà uno smanettamento osceno.

Potenzialmente, esiste comunque una soluzione per aggiungere alla tua scheda "legacy" VESA 1.2 le estensioni VESA 2.0. Dovresti scaricare un programma per caratteri TSR che eseguirai da DOS che, usato assieme a loadlin, può aiutare a configurare la scheda per l'appropriata modalità grafica su console. Da notare che questo non sempre funziona, per esempio alcune schede Cirrus Logic come le serie VLB 54xx sono associate ad un intervallo di indirizzi di memoria (ad esempio, nell'intervallo 15MB-16MB) per il frame buffering che preclude a questo d'essere utilizzato con successo con sistemi con più di 32MB di memoria. C'è un modo per far funzionare il tutto, cioè, se hai un'opzione nel BIOS per lasciare un memory hole nell'intervallo 15MB-16MB, potrebbe funzionare, ma Linux non supporta l'utilizzo dei memory hole. Comunque penso ci siano patch per questa opzione [Chi le ha e dove le si possono trovare?]. Se vuoi sperimentare quest'opzione, son disponibili in abbondanza programmi di stile TSR, un primo esempio è UNIVBE, che può essere trovato su Internet.

Alternativamente, dovresti scaricare patch del kernel per permettere alla tua scheda VESA 1.2 di funzionare con i driver per il framebuffer VESA. Per esempio, ci sono patch per l'uso con vecchie schede S3 (come l'S3 Trio o l'S3 Virge) che supportano il VESA 1.2. Per queste schede, poi prelevare le patch da

`ftp://ccssu.crimea.ua/pub/linux/kernel/v2.2/unofficial/s3new.diff.gz`

5.2 Come attivo i driver vesafb?

Assumendo che tu stia usando menuconfig, avrai bisogno di fare i seguenti passi:

Se il tuo processore (su piattaforme Intel) supporta i MTRR, attivali. Ciò velocizzerà le copie della memoria tra il processore e la scheda grafica, ma non è strettamente necessario. Puoi naturalmente far questo dopo aver fatto funzionare il dispositivo console.

IMPORTANTE: Per i kernel 2.1.x, vai nel menu Code Maturity Level ed attiva il prompt per develpoment andor incomplete drivers. Questo non è più necessario per il kernel 2.2.x.

Vai nel menu Console Drivers, ed attiva i seguenti:

- VGA Text Console
- Video Selection Support
- Support for frame buffer devices (experimental)
- VESA VGA Graphic console

- Advanced Low Level Drivers
- Seleziona Mono, 2bpp, 4bpp, 8bpp, 16bpp, 24bpp and 32bpp packed pixel drivers

VGA Chipset Support (text only) - vgafb - è solitamente parte della lista precedente, ma è stato rimosso in quanto è ora sconsigliato e non più supportato. Verrà presto rimosso. Usa invece VGA Text Console (fbcon). VGA Character/Attributes è usato solo con il VGA Chipset Support e non ha bisogno d'essere selezionato.

Assicurati che Mac variable bpp packed pixel support non sia attivato. Il kernel di Linux release 2.1.111 (e 112) sembra attivare quest'opzione automaticamente se si seleziona Advanced Low Level Drivers per la prima volta. Questo non succede più col 2.1.113.

C'è inoltre l'opzione di compilare i caratteri nella memoria, ma questo non è realmente necessario e puoi sempre usare l'utility setfont di kbd-0.99 (vedere la sezione sui caratteri) per modificare i caratteri caricandone nel device console.

Assicurati che questi non diventino moduli. [Non son sicuro che sia comunque possibile crearli come moduli - per favore correggetemi su questo punto]

Dovrai creare il device framebuffer in /dev. Ne hai bisogno di uno per ogni device ramebuffer, quindi tutto quel che dovrai fare sarà di scrivere mknod /dev/fd0 c 29 0 per il primo. I successivi saranno in multipli di 32, quindi, ad esempio, per creare /dev/fb1 dovrai scrivere mknod /dev/fb1 c 29 32 e così via fino all'ottavo device framebuffer (mknod /dev/fb7 c 29 224).

Quindi ricompila il kernel, modifica /etc/lilo.conf per includere il parametro VGA=ASK, ed esegui lilo, il che è richiesto per permetterti di selezionare la modalità che preferisci usare.

Questo è un esempio di configurazione LILO (presa dalla mia macchina)

```
# File di configurazione del LILO
boot = /dev/hda3
delay = 30
prompt
vga = ASK # Permette all'utente di entrare nelle modalità desiderate
image = /vmlinuz
  root = /dev/hda3
  label = Linux
  read-only # I filesystem Non-UMSDOS filesystems devono essere
            # montati in sola lettura per il controllo
```

Riavvia la macchina e, come primo test, prova ad inserire 0301 al prompt VGA (questo ti darà 640x480 @ 256) e dovresti essere in grado di vedere un piccolo logo molto carino col Pinguino.

Da notare che al prompt VGA ti sarà richiesto di scrivere il numero nel formato di "0" più la cifra di 3 numeri, tralasciando la 'x'. Questo non è necessario se stai usando LILO.

Una volta che hai visto che tutto funziona correttamente, potrai esplorare le varie modalità VESA (vedi oltre) e decidere quale di queste tu preferisci ed includerla nel parametro "VGA=x" in lilo.conf. Quand'hai deciso quale preferisci, controlla l'equivalente numero esadecimale nella tabella sottostante ed usalo (cioè, per 1280x1024 @ 256, userai esattamente "VGA=0x307"), e riesegui lilo. Questo è tutto quel che c'è da fare. Per ulteriori indicazioni, leggere gli HOWTO LoadLin/LILO.

NOTA! il vesafb non attiva lo scorrimento all'indietro nel buffer come predefinito. Dovrai passare al kernel l'opzione per attivarlo. Usa video=vesa:ypan oppure video=vesa:ywrap per farlo. Ambedue fanno la stessa cosa, ma in modi differenti. ywrap è molto più veloce di ypan ma potrebbe non funzionare su schede grafiche VESA 2.0 lievemente scorrette. ypan è più lento di ywrap, ma è molto più compatibile. Questa opzione è presente solo nei kernel 2.1.116 e successivi. I kernel precedenti non erano in grado di permettere lo scorrimento all'indietro nel buffer in vesafb.

5.3 Che modalità VESA sono disponibili?

Questo dipende effettivamente dal tipo di scheda grafica compatibile VESA 2.0 che hai nel tuo sistema e dall'ammontare di memoria video disponibile. Quindi è solo questione di provare quali modalità funzionano meglio con la tua scheda grafica.

La tabella seguente mostra i numeri di modalità che puoi immettere al prompt VGA o da usare col programma LILO. (Effettivamente questi numeri sono addizionati con 0x200 per rendere più semplice il riferirsi alla tabella)

Colori	640x400	640x480	800x600	1024x768	1152x864	1280x1024	1600x1200
4 bit	?	?	0x302	?	?	?	?
8 bit	0x300	0x301	0x303	0x305	0x161	0x307	0x31C
15 bit	?	0x310	0x313	0x316	0x162	0x319	0x31D
16 bit	?	0x311	0x314	0x317	0x163	0x31A	0x31E
24 bit	?	0x312	0x315	0x318	?	0x31B	0x31F
32 bit	?	?	?	?	0x164	?	

Spiegazione: 8 bit = 256 colori, 15 bit = 32.768 colori, 16 bit = 65.536 colori, 24 bit = 16,8 milioni di colori, 32 bit - lo stesso di 24 bit, ma con ulteriori 8 bit che possono essere usati per altre cose, e combacia perfettamente con i bus a 32 bit PCI/VLB/EISA.

Ci sono modalità addizionali a discrezione del fabbricante poichè il documento VESA 2.0 definisce solamente modalità fino a 0x31F. Potrai aver bisogno di giocherellare un po' per scovare queste ulteriori modalità.

5.4 Hai una scheda Matrox?

Se hai una scheda grafica Matrox, non hai realmente bisogno del vesafb, ma invece del driver matroxfb. Questo incrementa notevolmente le possibilità della tua scheda. Matroxfb funzionerà con le Matrox Mystique, Millennium I & II, G100 e G200. Inoltre supporta sistemi "multi-headed" (ossia, se hai due schede Matrox nella tua macchina, puoi usare due schermi sulla stessa macchina!). Per configurare per la Matrox, dovrai fare quanto segue:

Penso che dovresti aggiornare il BIOS della Matrox, puoi scaricare l'aggiornamento del BIOS da

`http://www.matrox.com/mgaweb/drivers/ftp_bios.htm`

. Attenzione che per far questo hai bisogno del DOS.

Vai nel menu Code Maturity Level ed attiva il prompt for development and/or incomplete drivers [da notare che questo potrebbe cambiare per kernel futuri - quando questo accadrà, quest'HOWTO verrà rivisto]

Vai nel menu Console Drivers ed attiva i seguenti:

- VGA Text Console
- Video Selection Support
- Support for frame buffer devices (experimental)

- Matrox Acceleration
- Seleziona quanto segue a seconda della scheda in tuo possesso
  - Millennium I/II support
  - Mystique support
  - G100/G200 support
- Attiva Multihead Support se vuoi usare più di una scheda Matrox
- Advanced Low Level Drivers
- Seleziona Mono, 2bpp, 4bpp, 8bpp, 16bpp, 24bpp and 32bpp packed pixel drivers

Ricompila il tuo kernel, Quindi dovrai modificare il tuo file lilo.conf per attivare il device Matroxfb. Il modo più rapido e semplice è di riutilizzare il mio.

```
# File di configurazione del LILO
boot = /dev/hda3
delay = 30
prompt
vga = 792    # Ne hai bisogno affinché parta in una modalità sicura
# Inizio delle partizioni Linux avviabili
image = /vmlinuz
  append = "video=matrox:vesa:440" # Quindi passa a Matroxfb
  root = /dev/hda3
  label = Linux
  read-only # I filesystem Non-UMSDOS filesystems devono essere
            # montati in sola lettura per il controllo
```

Alla fine, dovrai creare il device framebuffer in /dev. Te ne serve uno per device framebuffer, cioè tutto ciò che dovrai fare sarà di scrivere mknod /dev/fb0 c 29 0 per il primo. I successivi saranno in multipli di 32, cioè per esempio per creare /dev/fb1 dovrai scrivere mknod /dev/fb1 c 29 32 e così via fino all'ottavo device framebuffer (mknod /dev/fb7 c 29 224).

E questo è quanto! [NOTA: Se qualcuno usa questo supporto per più uscite video, per favore mi contatti ASAP - Ho bisogno di parlare con lui a proposito di questo per documentarlo!

5.5 Hai una scheda Permedia?

Le schede Permedia non possono essere utilizzate con il driver vesafb, ma fortunatamente c'è il driver framebuffer per Permedia da utilizzare. Assumendo che tu stia usando menuconfig, fai quanto segue:

Vai nel menu Code Maturity Level ed attiva il prompt for development and/or incomplete drivers [da notare che questo potrebbe cambiare per kernel futuri - quando questo accadrà, questo HOWTO sarà rivisto]

Vai nel menu Console Drivers e seleziona quanto segue:

- VGA Text Console
- Video Selection Support
- Support for frame buffer devices (experimental)
- Permedia2 support (experimental)

- Generic Permedia2 PCI board support
- Advanced Low Level Drivers
- Seleziona Mono, 2bpp, 4bpp, 8bpp, 16bpp, 24bpp and 32bpp packed pixel drivers
- Opzionalmente, seleziona i seguenti, se desideri usare i caratteri inclusi
  - Select compiled-in fonts
  - Seleziona Sparc console 12x22 font

Ricompila il kernel. Quindi dovrai modificare il tuo file lilo.conf per attivare il device pm2fb. Il modo più rapido e semplice è di riutilizzare il seguente

```
# File di configurazione per LILO
boot = /dev/hda3
delay = 30
prompt
vga = 792    # Ne hai bisogno affinché parta in una modalità sicura
# Inizio delle partizioni Linux avviabili
image = /vmlinuz
  append = "video=pm2fb:mode:1024x768-75,font:SUN12x22,ypan" # Quindi passa a pm2fb
  root = /dev/hda3
  label = Linux
  read-only # I filesystem Non-UMSDOS filesystems devono essere
            # montati in sola lettura per il controllo
```

La riga "pm2fb:mode:1024x768-75,font:SUN12x22,ypan" indica che stai selezionando una modalità a 1024x768 e 75Hz, con il carattere SUN12x22 selezionato (se l'avevi selezionato), includendo ypan per il supporto dello scorrimento all'indietro. Puoi selezionare altre modalità se desideri.

Alla fine, dovrai creare il device framebuffer in /dev. Te ne serve uno per device framebuffer, cioè tutto ciò che dovrai fare sarà di scrivere mknod /dev/fb0 c 29 0 per il primo. I successivi saranno in multipli di 32, cioè per esempio per creare /dev/fb1 dovrai scrivere mknod /dev/fb1 c 29 32 e così via fino all'ottavo device framebuffer (mknod /dev/fb7 c 29 224).

Per informazioni aggiuntive sulle altre caratteristiche del driver per framebuffer Permedia, punta il tuo browser su:

```
http://www.cs.unibo.it/~nardinoc/pm2fb/index.html
```

video=pm2fb:[opzione[,opzione[,opzione...]]]

dove l'opzione è una delle seguenti

- off per disabilitare il driver.
- mode:risoluzione per impostare la risoluzione della console. Le modalità son state prese dal file fb.modes.ATI nel pacchetto fbset di Geert. La profondità per tutte le modalità è 8bpp. Questa è la lista delle modalità disponibili:
  - 640x480-(60,72,75,90,100)
  - 800x600-(56,60,70,72,75,90,100)
  - 1024x768-(60,70,72,75,90,100,illo) illo=80KHz 100Hz

  - 1152x864-(60,70,75,80)
  - 1280x1024-(60,70,74,75)
  - 1600x1200-(60,66,76)
- La risoluzione predefinita è 640x480-60.
- font:nome del carattere per definire il carattere della console. Esempio: font:SUN12x22
- ypan imposta la corrente altezza virtuale tanto grande quanto concesso dalle dimensioni della memoria video
- oldmem quest'opzione è solo per gli utilizzatori di CybervisionPPC. Specificala se la tua scheda monta SGRAM Fujitsu (tutte le CVisionePPC prima del 30-dic-1998).
- virtual (temporaneo) specificala se il kernel riassocia le regioni PCI sulla tua piattaforma.

5.6 Hai una scheda ATI?

[Nota: Quest'informazione è, nel migliore dei casi, solo di seconda o terza mano in quanto non possiedo una scheda ATI su cui testarla. Sentiti libero di correggermi se mi son sbagliato o massacrami!] 8)

Le schede ATI possono essere utilizzate con il driver vesafb, ma potresti, come no, avere problemi, a seconda di quanto orrendamente scorretta sia la scheda. Fortunatamente, esiste il driver per framebuffer atyfb disponibile per essere usato. Assumendo che tu stia usando menuconfig, fai quanto segue:

Vai nel menu Code Maturity Level ed attiva il prompt for development and/or incomplete drivers [da notare che questo potrebbe cambiare per kernel futuri - quando questo accadrà, questo HOWTO sarà rivisto]

Via nel menu Console Drivers e seleziona quanto segue:

- VGA Text Console
- Video Selection Support
- Support for frame buffer devices (experimental)
- ATI Mach64 display support
- Advanced Low Level Drivers
- Seleziona Mono, 2bpp, 4bpp, 8bpp, 16bpp, 24bpp and 32bpp packed pixel drivers
- Opzionalmente, seleziona i seguenti, se desideri usare i caratteri inclusi
  - Select compiled-in fonts
  - Seleziona Sparc console 12x22 font

Ricompila il kernel. Quindi dovrai modificare il tuo file lilo.conf per attivare il device atyfb. Il modo più rapido e semplice è di riutilizzare il seguente

```
# File di configurazione del LILO
boot = /dev/hda3
delay = 30
prompt
vga = 792    # Ne hai bisogno affinché parta in una modalità sicura
# Inizio delle partizioni Linux avviabili
```

```
image = /vmlinuz
  append = "video=atyfb:mode:1024x768,font:SUN12x22"
  root = /dev/hda3
  label = Linux
  read-only # I filesystem Non-UMSDOS filesystems devono essere
            # montati in sola lettura per il controllo
```

La riga "atyfb:mode:1024x768,font:SUN12x22" indica che stai usando la modalità a 1024x768.

Alla fine, dovrai creare il device framebuffer in /dev. Te ne serve uno per device framebuffer, cioè tutto ciò che dovrai fare sarà di scrivere mknod /dev/fb0 c 29 0 per il primo. I successivi saranno in multipli di 32, cioè per esempio per creare /dev/fb1 dovrai scrivere mknod /dev/fb1 c 29 32 e così via fino all'ottavo device framebuffer (mknod /dev/fb7 c 29 224).

video=atyfb:[opzione[,opzione[,opzione...]]]

dove l'opzione è una delle seguenti

- font:STRINGA seleziona il carattere incluso (compilato nel kernel)
- noblink Disattiva il lampeggiamento
- noaccel Disattiva l'accelerazione
- vram:ULONG Informa il driver atyfb di quanta memoria sei a disposizione
- pll:ULONG Sconosciuta
- mclk:ULONG Sconosciuta
- vmode:ULONG Sconosciuta
- cmode:ULONG - imposta la profondità - 0, 8, 15, 16, 24 e 32

5.7 Che schede grafiche sono compatibili VESA 2.0?

Questa lista include tutte le schede grafiche di cui sia certo il funzionamento con il device vesafb:

- ATI PCI VideoExpression 2MB (max. 1280x1024 @ 8bit)
- ATI PCI All-in-Wonder
- Matrox Millennium PCI - BIOS v3.0
- Matrox Millennium II PCI - BIOS v1.5
- Matrox Millennium II AGP - BIOS v1.4
- Matrox Millennium G200 AGP - BIOS v1.3
- Matrox Mystique & Mystique 220 PCI - BIOS v1.8
- Matrox Mystique G200 AGP - BIOS v1.3
- Matrox Productiva G100 AGP - BIOS v1.4
- Tutte le schede basate su Riva 128
- Diamond Viper V330 PCI 4MB

- Genoa Phantom 3D/S3 ViRGE/DX
- Hercules Stingray 128/3D con l'uscita TV
- Hercules Stingray 128/3D senza uscita TV - serve aggiornamento del BIOS (gratuito da support@hercules.com)
- SiS 6326 PCI/AGP 4MB
- STB Lightspeed 128 (basata su Nvida Riva 128) PCI
- STB Velocity 128 (basata su Nvida Riva 128) PCI
- Jaton Video-58P ET6000 PCI 2MB-4MB (max. 1600x1200 @ 8bit)
- Voodoo2 2000

Questa lista è composta da chipset integrati sulle schede madri dei sistemi:

- Trident Cyber9397
- SiS 5598

La lista successiva mette nella "lista nera" le schede grafiche che non funzionano col device vesafb:

- TBA

5.8 Posso creare il vesafb come modulo?

Da quel che ne so, il vesafb non può essere modularizzato, a meno che in un qualche momento gli sviluppatori non decidano di modificare le sorgenti per la modularizzazione. Da notare che qualora la modularizzazione fosse possibile, al momento dell'avvio non si sarà in grado di vedere alcunché fin quando il vesafb non venga *modprobato*. Probabilmente è quindi ben più saggio lasciarlo nel kernel per i casi in cui ci fossero problemi all'avvio.

5.9 Come modifico il cursore?

[Preso da VGA-softcursor.txt - grazie Martin Mares!]

Linux ha ora alcune capacità per manipolare l'apparenza del cursore. Normalmente, puoi impostare le dimensioni del cursore hardware (e pure aggirare alcuni bug terrificanti in quelle miserrime schede Trident – vedi #define TRIDENT_GLITCH in drivers/char/vga.c). Qualora tu attivassi "Software generated cursor" nella configurazione del sistema, potrai provare un po' di nuovi trucchi: puoi rendere il tuo cursore somigliante ad un blocco rosso non lampeggiante, renderlo sfondo invertito rispetto al carattere su cui è posizionato oppure di evidenziare tal carattere e pure scegliere quando l'originale cursore hardware debba rimanere visibile o meno. Ci possono essere anche altre possibilità che non ho mai nemmeno immaginato.

Le apparenze del cursore sono controllate da una sequenza d'escape

```
<ESC>[?1;2;3c
```

dove 1, 2 e 3 sono parametri descritti in seguito. Se ne ometti uno qualsiasi, verranno impostati a zero.

Il parametro 1 specifica la dimensione del cursore (0=predefinita, 1=invisibile, 2=sottolineato, ..., 8=blocco pieno) + 16 se vuoi che il cursore software venga applicato + 32 se vuoi cambiare sempre il colore di sfondo

+ 64 se non ti piace avere lo sfondo come il primo piano. Le evidenziazioni sono ignorate per le due ultime flag.

Il secondo parametro seleziona i bit degli attributi del carattere che vuoi cambiare (semplicemente XORrandoli con il valore di questo parametro). Su VGA standard, i quattro bit alti specificano lo sfondo ed i quattro bassi il primo piano. In ambo i gruppi, tre bit bassi specificano il colore (come nei normali codici colore usati dalla console) e quello più significativo attiva l'evidenziazione (oppure, qualche volta, il lampeggiamento – dipende dalla configurazione della tua VGA).

Il terzo parametro consiste nei bit degli attributi dei carattere che vuoi impostare. L'impostazione dei bit ha luogo prima del cambio di stato dei bit, per cui puoi semplicemente svuotare un bit includendolo sia nella toggle mask che in quella delle impostazioni .

Per avere normale sottolineatura lampeggiante: echo -e '\033[?2c' Per avere un blocco lampeggiante: echo -e '\033[?6c' Per avere un blocco rosso non lampeggiante: echo -e '\033[?17;0;64c'

6 Utilizzare i device framebuffer su piattaforme Atari m68k

Questa sezione descrive le opzioni framebuffer sulle piattaforme Atari m68k.

6.1 Che modalità sono disponibili sulle piattaforme Atari m68k?

```
Colori  320x200 320x480 640x200 640x400 640x480 896x608 1280x960
-------+---------------------------------------------------------
 1 bit |                         sthigh   vga2    falh2   tthigh
 2 bit |                 stmid            vga4
 4 bit | stlow                         ttmid/vga16 falh16
 8 bit |        ttlow                    vga256
```

`ttlow, ttmid e tthigh` sono usati solo dalla TT, whilst `vga2, vga4, vga15, vga256, falh3 e falh16` sono usati solo dalla Falcon.

Quando usate con l'opzione del kernel `video=xxx`, senza sub-opzioni, il kernel effettuerà il rivelamento delle modalità nel seguente ordine, finché non troverà una modalità che è possibile con l'hardware stabilito.

- `ttmid`
- `tthigh`
- `vga16`
- `sthigh`
- `stmid`

Puoi specificare la modalità specifica che intendi usare, se non desideri d'effettuare l'autorivelamento per le modalità che preferisci. Per esempio, `video=vga16` ottiene uno schermo 640x480 a 4 bit.

6.2 Sub-opzioni addizionali sulle piattaforme Atari m68k

Ci sono un po' di sub-opzioni disponibili col parametro `video=xxx`:

- `inverse` - inverte la visualizzazione dei colori di sfondo e primo piano sullo schermo in modo che questi siano invertiti. Normalmente lo sfondo è nero, ma con questa sub-opzione viene impostato a bianco.
- `font` - imposta il carattere da utilizzare nelle modalità testuali. Attualmente si può selezionare solo `VGA8x8`, `VGA8x16`, `PEARL8x8`. Predefinito è l'utilizzo del `VGA8x8` solo se la dimensione verticale dello schermo è inferiore ai 400 pixel, altrimenti diventa predefinito il `VGA8x16`.
- `internal` - un'opzione molto interessante. Vedi la prossima sezione per informazioni.
- `external` - come sopra.
- `monitorcap` - descrive le capacità per multifrequenze. NON usare con un monitor a frequenza fissa!

6.3 Utilizzare la sub-opzione "internal" sulle piattaforme Atari m68k

Sintassi: `internal:(xres);(yres)[;(xres_max);(yres_max);(offset)]`

Questa opzione specifica le capacità interne estese di alcuni hardware video, ad esempio le modalità OverScan. `(xres)` e `(yres)` danno le dimensioni estese dello schermo.

Se la tua modalità OverScan necessita di un bordo nero, dovrai scrivere gli ultimi tre argomenti della sub-opzione `internal:`. `(xres_max)` è la lunghezza massima delle righe che l'hardware permette, `(yres_max)` è il massimo numero di righe e `(offset)` è lo spazio in bytes tra la parte visibile della memoria video ed il suo inizio fisico.

Spesso, le capacità interne estese del video hardware devono essere attivate, per far questo si deve utilizzare l'opzione `"switches=*"`. [Nota: l'Autore desidererebbe informazioni aggiuntive su questo punto, per favero. La documentazione sul m68k nel kernel non è abbastanza chiara su questo punto e lui non possiede un Atari! Anche esempi possono essere utili]

6.4 Utilizzare la sub-opzione "external" sulle piattaforme Atari m68k

Sintassi: `external:(xres);(yres);(depth);(org);(scrmem)[;(scrlen)[;(vgabase)[;(colw)[;(coltype)[;(xres_virtual)]]]]]`

Ciò è piuttosto complicato, quindi il presente documento tenterà di spiegarlo nel modo più chiaro possibile, ma l'Autore apprezzerà molto se qualcuno potrà darci una controllata per vedere se s'è dimenticato di fsckare qualcosa! :o)

Questa sub-opzione specifica che hai un hardware video esterno (molto probabilmente una scheda grafica) e come utilizzarlo con Linux. Il kernel è fondamentalmente limitato a ciò che conosce dell'hardware video interno, pertanto gli si dovrà fornire i parametri necessari affinché sia in grado d'utilizzare l'hardware video esterno. Ci son due limitazioni: devi passare a questa modalità prima dell'avvio del sistema e, una volta avviato, non potrai più cambiare modalità.

I primi tre parametri sono evidenti: danno le dimensioni in pixel dello schermo in altezza, larghezza e profondità. La profondità fornita dovrebbe essere il numero di colori definiti come 2^n il numero di piani richiesto. Per esempio, se desideri usare una visualizzazione a 256 colori, dovrai indicare 8 come la profondità. Ciò dipende dall'hardware grafico esterno, per cui sarai limitato da quel che l'hardware può fare.

Conseguentemente a questo, dovrai pure informare il kernel su come la memoria video è organizzata - fornendo una lettera come parametro `(org)`.

- `n` - usa piani normali, ad esempio un intero piano dopo un altro
- `i` - usa piani interlacciati, cioè 16 bit del primo piano, quindi 16 bit del piano successivo e così via. Solamente le modalità video proprie dell'Atari l'utilizzano - e non esistono schede grafiche in grado di supportarla.

- `p` - usa pixel a packetti ("packed"), ad esempio bit consecutivi significano tutti di un pixel. Questo è la modalità più comune per visualizzazioni a 256 colori su schede grafiche.
- `t` - usa colori reali, ad esempio questo è effettivamente "packed" pixel, ma non richiede, ma non richiede una tavola di confronto dei colori ("colour lookup table") come invece le altre modalità a pacchetti di pixel fanno. Queste modalità sono usualmente a 24 bit - che forniscono 16.8 milioni di colori.

Comunque, per modalità monocromatiche, il parametro `(org)` ha un significato differente

- `n` - usa colori normali, cioè 0=bianco, 1=nero
- `i` - usa colori invertiti, cioè 0=nero, 1=bianco

Il prossimo punto importante a proposito dell'hardware video è l'indirizzo di base della memoria video. Questo è dato dal parametro `(scrmem)` come numero esadecimale con un prefisso `0x`. Lo dovrai recuperare dalla documentazione in allegato al tuo hardware video esterno.

Il successivo parametro, `(scrlen)`, informa il kernel delle dimensioni della memoria video. Se manca, viene calcolato dai parametri `(xres)`, `(yres)` e `(depth)`. Non è comunque più utile scrivere un valore per questo parametro. Per lasciarlo vuoto, scrivi due punti e virgola consecutivi se devi indicare il parametro `(vgabase)`, altrimenti ignoralo semplicemente.

Il parametro `(vgabase)` è opzionale. Se non è dato, il kernel non può leggere/scrivere alcun registro dei colori per l'hardware video e di conseguenza dovrai impostare gli appropriati colori prima dell'avvio di Linux. Ma se la tua scheda è compatibile VGA puoi dare al kernel l'indirizzo dove può trovare il registro VGA in modo che possa modificare la tavola di confronto dei colori. Questa informazione può essere rintracciata nella documentazione del tuo hardware video esterno. Per *chiarire*, `(vgabase)` è l'indirizzo di *base*, ad esempio un indirizzo allineato a 4k. Per leggere/scrivere i registri dei colori, il kernel utilizza l'intervallo d'indirizzo compreso tra `(vgabase) + 0x3c7` e `(vgabase) + 0x3c9`. Questo parametro è dato in esadecimale e dev'avere un prefisso `0x`, proprio come `(scrmem)`.

`(colw)` è significativo solo se il parametro `(vgabase)` viene specificato. Avvisa il kernel su quanto grande sia ogni registro dei colori, ad esempio il numero di bit per ogni singolo colore (rosso/verde/blu). Il valore preimpostato è di solito 6 bit, ma è pure comune specificare 8 bit.

`(coltype)` è usato col parametro `(vgabase)` ed informa il kernel sul modello del registro dei colori della tua scheda grafica. Attualmente le tipologie supportate sono `vga` e `mv300`. `vga` è quella preimpostata.

`(xres_virtual)` è richiesto solo per le schede ProMST/ET4000 in cui la lunghezza fisica delle righe è diversa da quella visibile. Con la ProMST dovrai indicare 2048, mentre per la ET4000 dipende dall'inizializzazione della scheda.

7 Utilizzare i device framebuffer sulle piattaforme Amiga m68k

Questa sezione descrive le opzione per gli Amiga, che sono abbastanza simili a quelle per le piattaforme Atari m68k.

7.1 Che modalità sono disponibili per le piattaforme Amiga m68k?

Questo dipende dal chipset usato nell'Amiga. Ce ne sono tre tipi principali; `OCS, ECS e AGA`, i quali usano tutti il device framebuffer a colori.

- Modalità NTSC

  - `ntsc` - 640x200
  - `ntsc-lace` - 640x400
- Modalità PAL
  - `pal` - 640x256
  - `pal-lace` - 640x512
- Modalità ECS - colori a 2b su ECS, ad 8 bit solo su chipset AGA.
  - `multiscan` - 640x480
  - `multiscan-lace` - 640x960
  - `euro36` - 640x200
  - `euro36-lace` - 640x400
  - `euro72` - 640x400
  - `euro72-lace` - 640x800
  - `super72` - 800x300
  - `super72-lace` - 800x600
  - `dblntsc` - 640x200
  - `dblpal` - 640x256
  - `dblntsc-ff` - 640x400
  - `dblntsc-lace` - 640x800
  - `dblpal-ff` - 640x512
  - `dblpal-lace` - 640x1024
- Modalità VGA - colori a 2b su ECS, ad 8 bit solo su chipset AGA.
  - `vga` - 640x480
  - `vga70` - 640x400

7.2 Sub-opzioni aggiuntive per le piattaforme Amiga m68k

Sono simili a quelle per l'Atari m68k. Sono:

- `depth` - specifica la profondità del bit del pixel.
- `inverse` - fa la stessa cosa della sub-opzione dell'Atari.
- `font` - fa la stessa cosa della sub-opzione dell'Atari, benché venga usato il carattere `PEARL8x8` anziché il `VGA8x8` qualora la larghezza dello schermo fosse inferiore ai 400 pixel.
- `monitorcap` - specifica le capacità del monitor multifrequenza. Da non usare con monitor a frequenza fissa.

7.3 Schede grafiche d'espansione per Amiga supportate

- `Phase5 CyberVision 64` (S3 Trio64 chipset)
- `Phase5 CyvverVision 64-3D` (S3 ViRGE chipset)
- `MacroSystems RetinaZ3` (NCR 77C32BLT chipset)
- `Helfrich Piccolo, SD64, GVP ECS Spectrum, Village Tronic Picasso IIII+ e IV/` (Cirrus Logic GD542x/543x)

8 Utilizzare i device framebuffer su piattaforme Macintosh m68k

Attualmente, il device framebuffer implementato supporta solo le modalità selezionate in MacOS prima dell'avvio in Linux, inoltre supporta le modalità a colori da 1, 2, 4 e 8 bit.

Le sub-opzioni del framebuffer sono selezionate utilizzando la seguente sintassi

```
video=macfb:<font>:<inverse>
```

Puoi selezionare caratteri come VGA8x8, VGA8x16 e 6x11 etc. L'opzione "inverse" permette d'usare un video a colori invertiti

9 Utilizzare i dispositivi framebuffer su piattaforme PowerPC

L'autore apprezzerebbe moltissimo ricevere informazioni su come utilizzare i framebuffer su questa piattaforma.

10 Utilizzare i dispositivi framebuffer su piattaforme Alpha

10.1 Che modalità sono disponibili?

Finora, c'è solo la scheda TGA PCI - che consente solo 80x30 con una risoluzione di 640x480 sia a 8 che a 24/32 bit.

10.2 Che schede grafiche possono funzionare col device framebuffer?

In questa lista sono presenti tutte le schede grafiche note per funzionare:

- `DEC TGA PCI (DEC21030)` - 640x480 @ versioni a 8 oppure 24/32 bit

11 Utilizzare i dispositivi framebuffer su piattaforme SPARC

11.1 Quali schede grafiche possono funzionare con il device framebuffer?

In questa lista vi sono tutte le schede grafiche disponibili:

- MG1/MG2 - SBus od integrata su Sun3 - max. 1600x1280 @ mono (BWtwo)
- CGthree - Simile a MG1/MG2 ma con supporto per il colore - risoluzione max ?
- GX - SBus - max. 1152x900 @ 8bit (CGsix)
- TurboGX - SBus - max. 1152x900 @ 8 bit (CGsix)
- SX - solo SS10/SS20 - max. 1280x1024 @ 24 bit - (CGfourteen)
- ZX(TZX) - SBus - schede 3D accelerate a 24bit - risoluzione max ? (Leo)
- TCX - AFX - solo per Sparc 4 - max. 1280x1024 @ 8bit

- TCX(S24) - AFX - solo per Sparc 5 - max. 1152x900 @ 24bit
- Creator - SBus - max. 1280x1024 @ 24bit (FFB)
- Creator3D - SBus - max. 1920x1200 @ 24bit (FFB)
- ATI Mach64 - accelerata a 8/24bit solo per Sparc64 PCI

Esiste l'opzione per utilizzare la PROM per inviare caratteri allo schermo od alla console seriale.

Inoltre, dai un'occhiata alle FAQ sul Sparc Frame Buffer presso

`http://c3-a.snvl1.sfba.home.com/Framebuffer.html`

11.2 Configurare i device framebuffer

Durante make config, devi scegliere se compilare `promcon` e/o `fbcon`. Puoi selezionarle tutte e due, ma se lo fai dovrai impostare le flag del kernel per selezionare il device. `fbcon` ha sempre la precedenza quando non c'è impostazione a proposito. Se `promcon` non è selezionata, all'avvio diventa predefinito `dummycon`. Se `promcon` è selezionato, userà il suo device. Non appena i bus sono avviati, e `fbcon` vi è compilato, il kernel rileva i framebuffer in questione ed userà `fbcon`. Se non esistono device framebuffer, il valore predefinito diventa `promcon`.

Vi sono le opzioni del kernel

```
video=sbus:opzione
        dove opzione è una lista, separata da virgole:
                nomargins       imposta i margini a 0,0
                margins=12x24   imposta i margini a 12,24 (il valore
                                predefinito è calcolato per la
                                risoluzione)
                off             non effettua il rivelamento per alcun
                                framebuffer SBus/UPA
                font=SUN12x22   utilizza un carattere specificato
```

Quindi, per esempio, avviare con

```
video=sbus:nomargins,font=SUN12x22
```

otterrà una piacevolmente veloce console testuale con una risoluzione di 96x40, all'apparenza simile alla console di Solaris, ma con i colori ed i terminali virtuali proprio come sulla piattaforma Intel.

Se vuoi usare il carattere `SUN12x22` dovrai attivarlo durante make config (disabilitando l'opzione `fontwidth != 8`). I framebuffer accelerati possono supportare qualsiasi larghezza dei caratteri compresa tra 1 o 16 pixel, mentre i framebuffer stupidi possono supportare solamente caratteri larghi 4, 8, 12 e 16 pixel.

Si raccomanda di recuperare un pacchetto consoletools recente.

12 Utilizzare i dispositivi framebuffer su piattaforme MIPS

Non c'è bisogno di modificare alcunché su questa piattaforma, è già tutto gestito automaticamente. In particolare, gli Indy han già inclusa una dimensione della console di 160x64. Comunque, ci sono movimenti in corso per riscrivere il codice per la console su questi Indy, quindi tieni sotto controllo questa sezione.

13 Utilizzare i dispositivi framebuffer su piattaforme ARM

13.1 Netwinder

Per i Netwinder (che usano il chip ARM SA110 RISC - un adorabile processore britannico), ci sono due versioni del driver framebuffer Cyber2000 - una per i kernel 2.0.x ed una per quelli 2.2.x. Attivare ed utilizzare questo driver è decisamente una cavolata su ambo i kernel, comunque la vecchia versione è "hardcoded" per profondità e risoluzione (bleah), ma le buone notizie sono che la nuova versione nei kernel 2.2.x è molto più flessibile, ma attualmente è ancora in sviluppo. Per averlo attivo e funzionante, il miglior modo è leggere la documentazione inclusa nel port per ARM delle sorgenti del kernel.

I Netwinder utilizzano un chipset compatibile VGA, ma sfortunatamente nessuno ha ancora fatto un port del vgafb. Ciò accadrà quando qualcuno avrà abbastanza tempo libero da dedicarci. [Lo farei io se qualcuno mi desse un Netwinder con cui giocare]

13.2 Acorn Archimedes

Gli Acorn han sempre avuto supporto per il framebuffer a partire dai giorni di Linux 1.9.x. Comunque il driver Acornfb nel 2.2.x è totalmente nuovo in quanto l'interfaccia generica per framebuffer è cambiata durante lo sviluppo dei kernel 2.1.x (che, ovviamente, son poi divenuti 2.2.x). Come nel caso precedente, è un gioco da ragazzi attivare il driver ed impostare profondità e risoluzione.

13.3 Altri port ARM (SA 7110s et. al)

A sorpresa, c'è un driver framebuffer per il Psion 5 e per il Geofox! Mi han detto che visualizza il Pinguino piuttosto bene. [Qualcuno mi regali un Psion 5!]

14 Utilizzare i framebuffer "multi-headed"

Questa parte del documento è stata donata molto gentilmente da Frederick A. Niles, che conserva tutti i diritti per le informazioni incluse in questa sezione dell'HOWTO.

14.1 Introduzione

L'obiettivo principale di questo documento è di introdurti all'avvio di una configurazione "dual head" [N.d.T.: ossia con due o più schede video] di Linux. Per quanto il processo sia piuttosto semplice, ci son molte cose che uno può sbagliare lungo il percorso.

L'esempio su cui mi concentrerò è l'avvio di un X-server su di un secondo monitor. L'ho trovata cosa simpatica in quanto si possono trovare vecchi, larghi monitor da 19" e 21" a frequenza fissa che la gente da via perché non può più utilizzarli. In questo modo puoi avviare una piccola multifrequenza e quindi usare X su un bel monitor grande.

Per favore tieni conto che il supporto per il dual head è attualmente in sviluppo, quindi queste informazioni cambiano rapidamente. Qualsiasi punto di questo documento può essere datato e semplicemente scorretto nel momento in cui lo stai leggendo.

** ATTENZIONE ** Questo documento è stato scritto prima di qualsiasi rilascio dell'XFree86 4.0. Se lo stai leggendo e l'XFree86 4.0 è già stato rilasciato molte cose possono essere cambiate. Prova ad ottenere una nuova versione di questo documento se disponibile.

14.2 Per contattare l'autore

Per questo documento, sarò molto lieto se sarò contattato. Senza i vostri suggerimenti e proposte, questo documento non esisterebbe. Per questo, inviatemi aggiunte, commenti e critiche a: Frederick.A.Niles@gsfc.nasa.gov. Frederick.A.Niles@gsfc.nasa.gov.

14.3 Contributori

Le seguenti persone han contribuito a questo mini-HOWTO.

* Petr Vandrovec `vandrove@vc.cvut.cz`

* Andreas Ehliar `ehliar@lysator.liu.se` (x2x)

* Marco Bizzarri `m.bizzarri@icube.it` (X server multipli)

14.4 Standard Disclaimer (in lingua originale)

No liability for the contents of this document can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies that could be damaging to your system. Proceed with caution, and although this is highly unlikely, I don't take any responsibility for that.

[** Traduzione italiana **]

Nessuna responsibilità per il contenuto di questo documento sarà accettata. Utilizza i concetti, esempi ed altri contenuti a tuo proprio rischio. In quanto questa è una nuova edizione del documento, vi possono essere errori ed imperfezioni tali da danneggiare il tuo sistema. Procedi con cautela e, sebbene questo sia alquanto inverosimile, non mi prenderò alcuna responsabilità se ciò accadrà.

14.5 Copyright Information (in lingua originale)

restrizioni addizionali alla sua distribuzioni. Eccezioni a queste regole possono essere garantiti sotto certe condizioni; per favore contatta il coordinatore degli HOWTO Linux all'indirizzo indicato più avanti.

* Se hai domande, per favore contatta il coordinatore degli HOWTO Linux presso linux-howto@sunsite.unc.edu

14.6 Che hardware è supportato?

La maggior parte delle schede grafiche assume d'essere l'unica in un sistema e sono permanentemente impostate con l'indirizzamento dell'adattatore primario per lo schermo. Ci sono poche eccezioni.

* Schede Matrox: Queste includono le schede video Matrox Millennium, Matrox Millennium II, Matrox Mystique, Matrox Mystique 220, Matrox Productiva G100, Matrox Mystique G200, Matrox Millennium G200 e Matrox Marvel G200

* MDA: Questa include gli adattatori grafici monocromatici Hercules tra le altre. Questa è solo per un supporto testuale sul seconda uscita.

Nota: è solo il secondo adattatore a dover essere uno dei precedenti.

14.7 Supporto commerciale

Questo mini-HOWTO principalmente riguarda il free software. Comunque, ci sono X server commerciali con incluso il supporto per il "multi-head". Tra questi Metro Link's (www.metrolink.com) Metro-X e Xi Graphics' (www.xig.com) Accelerated-X.

14.8 Ottenere tutto il necessario.

Hai bisogno dei seguenti patch e programmi:

* Il programma "fbset" prova:

```
http://www.cs.kuleuven.ac.be/~geert/bin/
```

(nota: questo programma è incluso nella RedHat 6.0)

* "fbaddon", patch del Kernel Linux per dual head Matrox prova:

```
ftp://platan.vc.cvut.cz/pub/linux/matrox-latest/
```

* Il programma "con2fb" prova:

```
ftp://platan.vc.cvut.cz/pub/linux/matrox-latest/
```

* Il frame buffer server X11 XF86_FBDev. Lo si trova come componente standard di XFree86 3.3.1.

14.9 Iniziamo

La prima cosa che dovrai fara sarà patchare una copia delle sorgenti di Linux con la patch "fbaddon". Quindi dovrai configurare il kernel ed attivare il supporto framebuffer. Se hai schede Matrox attiva il supporto per il driver unificato accelerato Matrox così come il tipo particolare di scheda in tuo possesso. Non attivare il supporto per il frame buffer VESA. Questo potrebbe causare un conflitto. Attiva (ovviamente) il supporto per il multi-head. Crea il kernel e riavvia.

Ora dovrai installare il programma "fbset" e leggere attentamente tutta la documentazione su come regolare le impostazioni. L'utilizzo di un file "/etc/fb.modes" è caldamente consigliato una volta che avrai deciso le tue impostazioni. Il programma fbset include uno script in Perl per convertire il tuo file XF86Config nelle impostazioni fb.modes. Ho incluso il mio script in octave/Bourne shell per convertire il tuo file XF86Config nelle Appendici A & B.

Ti deve prima diventare agevole l'utilizzo del device frame buffer su di un monitor, comprendendo ogni problematica che può spuntar fuori dal tuo impostare che non abbia nulla a che vedere col supporto per il multi-head. Questo può risparmiare un mucchio di grattamenti di testa successivamente.

Mi sto per concentrare nella mia spiegazione su come avviare X sul secondo schermo rendendo molte altre configurazioni semplicemente ed ovviamente presupposti della procedura stessa.

14.9.1 Spostiamo una console...

Compila il programma "con2fb". Se lo lanci senza alcun argomento otterrai il seguente messaggio per l'utilizzo:

"usage: con2fb fbdev console".

Quindi, un comando esemplificativo potrebbe essere "con2fb /dev/fb1 /dev/tty6" per spostare la console virtuale numero sei sul secondo schermo. Usa Ctrl-Alt-F6 per spostarti su tal console e vedere che per davvero spunta sul secondo schermo.

14.9.2 Utilizzare "fbset" per regolare le impostazioni di questo secondo monitor

"fbset" regola le impostazioni solo sullo schermo in cui lo lanci. Per questo, dovrai fare attenzione ad usare la flag "-fb" sul secondo schermo. In particolare, se non altro probabilmente vorrai di impostare almeno la risoluzione virtuale verticale pari alla tua reale risoluzione verticale.

esempio "fbset -fb /dev/fb1 -vyres 600"

Questo rallenterà pesantemente la modalità testuale, ma X sarebbe sgradevole senza di questo.

14.9.3 Impostare X per il supporto Frame Buffer.

Il file framebuffer.txt spiega questo meglio di quanto possa io, ma qui ci son due punti importanti.

Assicurati d'aver impostato il collegamento per "X" in modo che punti a "XF86_FBDev".

Quindi dovrai aggiungere una sezione "monitor" al tuo file XF86Config per il device frame buffer. Qui c'è un esempio:

```
# Il server Frame Buffer

Section "Screen"
    Driver      "fbdev"
    Device      "Millennium"
    Monitor     "NEC MultiSync 5FGp"
    Subsection "Display"
        Depth       8
        Modes       "default"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
```

```
        Depth       16
        Modes       "default"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       24
        Modes       "default"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       32
        Modes       "default"
        ViewPort    0 0
    EndSubsection
EndSection
```

Utilizza le modalità di "default" in quanto non credo che qualsiasi altra possa funzionare con frame buffer per Matrox.

14.9.4 Provare a far partire l'X server sul secondo schermo.

Imposta la variabile FRAMEBUFFER al secondo frame buffer.

"export FRAMEBUFFER=/dev/fb1"

oppure

"setenv FRAMEBUFFER /dev/fb1"

Dovrai avviare l'X server in modo che si combini con la profondità di colore selezionata e che appaia sullo stesso schermo da cui l'hai avviato.

esempio "startx – :0 -bpp 16 vt06"

Questo esempio avvia un X "zero" sulla console virtuale sei con 16 bit di colore. Utilizzare ":1" all'avvio di un altro server X per l'altro frame buffer ti permetterà d'avere due server X avviati.

14.10 Sommario

I passi necessari per avere un X server attivo su un secondo schermo possono essere riassunti come segue:

* Recuperare la patch del kernel, fbset e con2fb.

* Applicare la patch al kernel, configurare, ricompilare e riavviare.

* Aggiungere la sezione XF86_FBDev al file XF86Config ed impostare il collegamento per X.

Quindi, ad ogni riavvio:

* Spostare una console. es. "con2fb /dev/fb1 /dev/tty6"

* Regolare le impostazioni es. "fbset -fb /dev/fb1 1280x1024"

* Impostare la FRAMEBUFFER es. "export FRAMEBUFFER=/dev/fb1"

* Lanciare il server X es. "startx – -bpp 16 vt06"

Puoi automatizzare tutto questo per ogni riavvio grazie ad un alias della shell. Dev'essere un alias e non uno script della shell in quanto deve rilevare il corrente numero della console. Questo è il mio alias per la

C-shell per avviare X in un secondo monitor a frequenza fissa:

```
alias startxfb = "
setenv FRAMEBUFFER /dev/fb\!*;    # Imposta la var env all'arg del cmd.
con2fb $FRAMEBUFFER /dev/$tty;    # Sposta fb sulla corrente tty.
fbset -fb $FRAMEBUFFER 1280x1024@62;  # Favoriti da /etc/fb.modes
startx -- :\!* -bpp 16 vt0`echo $tty | cut -dy f 2`' # X su questa tty.
"
```

Nel mio file .cshrc questo è tutto sulla stessa riga senza commenti, ma è più facile da leggere con gli a capi e con i commenti inseriti. Devo semplicemente dare il numero del frame buffer come un argomento e partirà senza problemi.

Non son sicuro su come far lo stesso alias con la bash. Non so come determinare la corrente tty o passare l'argomento ad un alias con la bash. Se qualcuno me lo farà sapere l'aggiungerò qui. Comunque, puoi usare il comando "tty" per ottenere il nome della corrente VT a quindi semplicemente realizzare due alias separati per ogni server X.

14.11 Altre Note e Problemi

* Sia "fbset" che "startx" DEVONO essere lanciati dallo stesso frame buffer poiché questi ne son soggetti. Questo pone seri limiti a come questi comandi possano essere automatizzati tramite script.

* XFree86 4.0 avrà un "corretto" supporto per più di un'uscita video, ma attualmente il 3.3.1 non ce l'ha. Puoi lanciare due server con 3.3.1 ed usare x2x per passare da uno all'altro, in ogni caso... (vedi il prossimo paragrafo)

* Il frame buffer inattivo tratterrà l'ultima immagine di quand'era attivo, nessun aggiornamento cioè avverrà.

* Il monitor che non è selezionato non sempre mantiene la sua condizione quando non è attivo. (Ma di solito lo fa.)

* Geert Uytterhoeven (il mantenitore del frame buffer) e Linus Torvalds non sono d'accordo con gli attuali cambiamenti al supporto per console multi-head detti "frame buffer per VT" (ossia fbaddon) quindi questi non saranno mai nel principale ramo del kernel. (Questo è stato sentito di terza mano e potrebbe essere clamorosamente falso.)

* Se "non rispetti le regole" ed avvii il server X (lanci "startx") da uno schermo differente, la macchina potrebbe eventualmente crashare in malo modo con gli input del mouse e della tastiera tutti mischiati tra loro.

* La documentazione framebuffer.txt nelle sorgenti del kernel spiega che puoi usare le impostazioni Modeline nel file XF86Config direttamente quanto lanci X. L'utilizzare il frame buffer per Matrox sembra forzare il server X a cestinarle tutte quante. Quindi puoi solo avere una ("predefinita") impostazione per volta (lo stesso che hai nella modalità testuale).

* Il XF86_FBDev non è accelerato. Comunque, ci son patch per il supporto per Matrox accelerato presso

http://www.in-berlin.de/User/kraxel/xfree86/

14.11.1 Riuscire a far funzionare "init level five" (cioè xdm/gdm)

Non son ancora riuscito ad immaginarmi un modo per avviare nel livello 5 di init con una configurazione a doppio schermo (ed effettivamente avere il server su uno o l'altro degli schermi od entrambi). Mentre sembra abbastanza semplice aggiungere una riga al file dei server gdm/xdm, la costrizione di dover avviare il server

X dallo stesso frame buffer impedisce alla soluzione banale di funzionare. Se qualcuno trova un modo per favore mi contatti per e-mail e l'aggiungerò qui.

14.11.2 Utilizzare il programma x2x.

Esiste un simpatico programmino chiamato x2x che passa in automatico da un server X all'altro quando si arriva al bordo dello schermo. L'ultima casa conosciuta di questo programma è:

`http://ftp.digital.com/pub/DEC/SRC/x2x`

. Lo si trova anche tra i pacchetti opzionali della Debian. Non l'ho ancora provato, ma alcuni utenti han riferito d'averlo utilizzato con successo.

14.11.3 Altri comandi utili

Esistono alcuni comandi linux che vale la pena ricordare quando si ha a che fare con una configurazione a più uscite (specialmente quando si scrivono script).

* "chvt" permette di passare da una console virtuale ad un'altra

* "openvt" lancia un programma in un nuovo terminale virtuale (VT).

* "tty" riferisce il nome del corrente terminale.

14.11.4 Appendice A. Script cvtmode.m, in Octave

(notare l'impostazione bpp)

```
#!/usr/bin/octave -q
bpp = 16;
DCF = sscanf(argv(1,:), "%f");
HR  = sscanf(argv(2,:), "%f");
SH1 = sscanf(argv(3,:), "%f");
SH2 = sscanf(argv(4,:), "%f");
HFL = sscanf(argv(5,:), "%f");
VR  = sscanf(argv(6,:), "%f");
SV1 = sscanf(argv(7,:), "%f");
SV2 = sscanf(argv(8,:), "%f");
VFL = sscanf(argv(9,:), "%f");
pixclock = 1000000 / DCF;
left_margin = HFL - SH2;
right_margin = SH1 - HR;
hsync_len = SH2 - SH1;

# 3) regolazioni verticali:
upper_margin = VFL - SV2;
lower_margin = SV1 - VR;
vsync_len = SV2 - SV1;

RR = DCF / (HFL * VFL) *1e6;
HSF = DCF / HFL * 1e3;
```

```
printf("mode \"%dx%d\"\n",HR,VR);
printf("   # D: %3.2f MHz, H: %3.2f kHz, V: %2.2f Hz\n", DCF, HSF, RR);
printf("   geometry %d %d %d %d %d\n", HR, VR, HR, VR, bpp);
printf("   timings %d %d %d %d %d %d %d\n", ...
                                 pixclock, left_margin, right_margin, ...
                                 upper_margin, lower_margin, ...
                                 hsync_len, vsync_len);
printf("endmode\n");
```

14.11.5 Appendice B. Script "cvtfile", in Bourne Shell

(Che richiama lo script in octave "cvtmode")

```
#!/bin/sh

# Script della shell per convertire il file XF86Config in uno fb.modes.
# Utilizza lo script in octave cvtmode.m

if [ -z $1 ]; then
  FILE=/etc/X11/XF86Config
else
  FILE=$1
fi

i=1
LEN=`grep Modeline $FILE | wc -l`
while expr $i \< $LEN > /dev/null ;
do
  CURLINE=`grep Modeline $FILE | cut -d'"' -f 3-20 | head -$i | tail -1 `
  ./cvtmode.m $CURLINE
  echo " "
  i=`expr $i + 1`
done
```

15 Usare/Cambiare i caratteri

Per poter cambiare i caratteri si ha bisogno di kbd-0.99. Lo si può ottenere da

```
ftp://ftp.win.tue.nl/pub/linux/utils/kbd
```

.

Un vantaggio di scaricare ed installare kbd-0.99 è che si sarà in grado di caricare caratteri internazionali (es. il simbolo dell'Euro) nel proprio device console (L'avere tre simboli sulla tastiera, quello del Dollaro, quello della Sterlina e quello dell'Euro, è tres chic!).

16 Cambiare le modalità della console

Per essere in grado di cambiare le modalità (es. 640x480, 800x600, etc) si ha bisogno di fbset (attualmente fbset-19990118.tar.gz) - lo puoi recuperare in ftp da:

```
http://www.cs.kuleuven.ac.be/~geert/bin/fbset-19990118.tar.gz
```

Ha incluse un gran numero d'istruzioni su come utilizzarlo.

17 Impostare il driver X11 FBdev

Se non stai utilizzando XFree86 3.3.3.1 o successivi, ti conviene aggiornarti il prima possibile all'XFree86 3.3.3.1 - che include un driver per X FBdev per i device framebuffer. Altrimenti, segui i seguenti passi per come scaricare oppure creare il tuo proprio driver FBdev per vecchie versioni di XFree86 come le 3.3.2, 3.3.3 ecc.

Vai a

```
http://www.xfree86.org
```

e scarica il più recente archivio di sorgenti XServer, dearchivialo e configura i driver, seguendo questi passi:

- Modifica xc/config/cf/xf86site.def, decommenta il #define per XF68FBDevServer
- Commenta *tutti* i riferimenti a FB_VISUAL_STATIC_DIRECTCOLOR, in quanto è finto e non più usato. Se stai usando XFree86 3.3.3.1, non hai bisogno di questo passo - in quanto l'han già rimosso.
- Modifica xc/programs/Xserver/hw/xfree86/os-support/linux/lnx_io.c, e cambia K_RAW in K_MEDIUMRAW.

e quindi compila il driver. Non preoccuparti dei riferimenti a m68k, il driver supporta le piattaforme Intel. Quindi compila tutto l'insieme - ci vorrà parecchio tempo, credo, in quanto son sorgenti alquanto consistenti.

Alternativamente, se non hai tempo da perdere, puoi ottenere i binari dai siti indicati sotto. Tieni conto però che sono 'non ufficiali' e che l'utilizzo è un tuo proprio rischio.

Per libc5, usa quello presso:

```
http://user.cs.tu-berlin.de/~kraxel/linux/XF68_FBDev.gz
```

Per glibc2, scarica da questi URL.

```
http://user.cs.tu-berlin.de/~kraxel/linux/XF68_FBDev.libc6.gz
http://pobox.com/~brion/linux/fbxserver.html
```

Son pervenute informazioni riguardo ad X11 non funzionale su certe schede grafiche con il vesafb attivato, se questo accade, prova il nuovo driver XF86_FBdev per X11.

Questo driver, assieme con il vesafb può inoltre aiutare l'X11 a funzionare in alte risoluzioni grafiche su certi chipset grafici che non son supportati da nessuno degli attuali driver X11. Esempi sono la MGA G-200 ed altri.

Per configurare il driver XF86_FBdev per il tuo sistema X11, hai bisogno di modificare XF86Config come segue:

```
Section "Screen"
        Driver          "FBDev"
        Device          "Primary Card"
```

```
        Monitor         "Primary Monitor"
        SubSection      "Display"
                Modes           "default"
        EndSubSection
EndSection
```

Dovrai inoltre impostare XkbDisable nella sezione keyboard, oppure invocare il server XF86_FBDev con l'opzione '-kb' in modo da impostare la tastiera in un modo corretto. Se ti dimentichi d'impostare XkbDisable, dovrai mettere le seguenti righe nel tuo .Xmodmap per regolare le impostazioni della tastiera. Alternativamente, puoi modificare il tuo xkb in modo che rifletta la lista qui sotto.

Questo è stato corretto con XFree86 3.3.3.1 ed è una buona idea comunque l'aggiornarsi a questa versione in quanto un po' di bug son stati corretti ed anche in quanto include FBDev come uno dei driver, com'ho detto precedentemente.

```
! Keycode settings required
keycode 104 = KP_Enter
keycode 105 = Control_R
keycode 106 = KP_Divide
keycode 108 = Alt_R Meta_R
keycode 110 = Home
keycode 111 = Up
keycode 112 = Prior
keycode 113 = Left
keycode 114 = Right
keycode 115 = End
keycode 116 = Down
keycode 117 = Next
keycode 118 = Insert
keycode 119 = Delete
```

Potresti aver bisogno di smanettarci un po' su questa lista (prova a copiare la definizione originale dal driver X11 originale che stavi usando e modificando il nome del driver in FBDev), ma fondamentalmente questo è quel di cui hai bisogno per utilizzare il driver X11 vesafb.

Confido fortemente che i prossimi rilasci dell'X11 risolvano i problemi con le schede grafiche supportate.

18 Come posso convertire le "mode-line" di XFree86 nelle impostazioni per il device framebuffer?

Se hai XFree86 (X11) installato nella tua macchina, e lo puoi usare con successo, è un gioco da ragazzi convertire le "mode-line" del tuo XF86Config nelle impostazioni necessarie per il device framebuffer.

Il device framebuffer richiede i seguenti campi

- pixclock - il pixel clock in picosecondi
- left_margin - intervallo tra la sincronizzazione e l'immagine (a sinistra)
- right_margin - intervallo tra l'immagine e la sincronizzazione (a destra)
- upper_margin - intervallo tra la sincronizzazione e l'immagine (sopra)

- lower_margin - intervallo tra l'immagine e la sincronizzazione (sotto)
- hsync_len - lunghezza della sincronizzazione orizzontale
- vsync_len - lunghezza della sincronizzazione verticale

Una "mode-line" di XFree86 ha i seguenti campi

```
Modeline  "1280x1024" DCF HR SH1 SH2 HFL VR SV1 SV2 VFL
```

Alcuni semplici calcoli son necessari per tradurre le "mode-line" di XF86 in una serie d'impostazioni per il device framebuffer. Come esempio, esamineremo come convertire una "mode-line" presa dal mio file XF86Config.

```
Modeline  "1280x1024" 110.00 1280 1328 1512 1712 1024 1025 1028 1054
```

Prima, calcolare la velocità richiesta del pixclock. XFree86 usa megahertz mentre il framebuffer usa picosecondi (Perché, m'è ignoto). Dividi un milione per DCF. Ad esempio, 1.000.000 / 110,0 = 9090,9091

Quindi abbiam bisogno di calcolare le impostazioni orizzontali.

- left_margin = HFL - SH2
- right_margin = SH1 - HR
- hsync_len = SH2 - SH1

Nel nostro esempio, ciò sarà:

- left_margin = 1712 - 1512 = 200
- right_margin = 1328 - 1280 = 48
- hsync_len = 1512 - 1328 = 184

Ed ora tocca alle impostazioni verticali.

- upper_margin = VFL - SV2
- lower_margin = SV1 - VR
- vsync_len = SV2 - SV1

Che, nel nostro esempio, saranno:

- upper_margin = 1054 - 1028 = 26
- lower_margin = 1025 - 1024 = 1
- vsync_len = 1028 - 1025 = 3

Ora possiamo usare queste informazioni per impostare il framebuffer alla modalità desiderata. Per esempio, per il framebuffer matroxfb, è richiesto:

```
video=matrox:xres:<>,yres:<>,depth:<>,left:<>,right:<>,hslen:<>,upper:<>,lower:<>,vslen:<>
```

Metterò nel mio /etc/lilo.conf la seguente riga:

```
append = "video=matrox:xres:1280,yres:1024,depth:32,left:200,right:48,hslen:184,upper:26,lower:0,vslen:3"
```

Da notare che in questo caso non è usato il pixclock. Diviene necessario solo se non ti piace la velocità predefinita del pixclock. Puoi anche indicare questo parametro. Come impostarlo è documentato in altre parti di questo HOWTO.

19 Cambiare il logo di Linux

Può essere personalizzato cambiando il file linux_logo.h nella directory include/linux. Si tratta di un header c ed è piuttosto difficile cambiarlo a mano, comunque esiste una plug-in per gimp disponibile presso

```
http://registry.gimp.org/detailview.phtml?plugin=Linux+Logo
```

che ne creerà uno per te. Tutto quel di cui hai bisogno è un'immagine 80x80 con meno di 224 colori. Puoi sia lasciare che sia la plug-in a creare tre varietà (2,16,224) che creartele da te e poi usarle con la plug-in. Ti verrà chiesto dove salvare il file e se sei coraggioso lo puoi mettere in ($SRCDIR)/include/linux/linux_logo.h. Una volta che hai finito dovrai ricompilare il kernel come il solito, riavviare e se il framebuffer funziona vedrai il tuo nuovo logo all'avvio.

20 Cerchi altre informazioni?

Per quelli che sono interessati a lavorare col driver framebuffer, puntate i vostri browser a

```
http://www.linux-fbdev.org
```

per informazioni sulla programmazione.

Per le persone di lingua francese, esiste una traduzione presso

```
http://www.freenix.org/unix/linux/HOWTO/mini/Vesafb.html
```