Parte xxxix

i86

Minix [1] è nato originariamente come sistema didattico. Minix non ha avuto il successo e la diffusione che avrebbe potuto avere a causa delle limitazioni della sua licenza iniziale. In seguito le cose sono cambiate, fortunatamente, perché Minix resta probabilmente l'unica possibilità reale per chi vuole utilizzare elaboratori con architettura i286 o inferiore.

Lo scopo di questo capitolo è introdurre all'uso di Minix per poter riutilizzare i vecchi elaboratori i286, in particolare, collegandoli a

una rete locale TCP/IP. Le informazioni seguenti si riferiscono alla versione 2.0.0.

Procurarsi il software

Minix è ottenibile dalla rete, precisamente a partire dalla sua pagina di presentazione ufficiale, quella del suo primo autore (La sigla «AST» rappresenta le iniziali di Andrew S. Tanenbaum), oltre che dai vari siti speculari del relativo FTP.

http://www.cs.vu.nl/~ast/minix.html

Minix è nato assieme a un libro che tuttora dovrebbe essere accompagnato da un CD-ROM contenente il sistema operativo:

- Andrew S. Tanenbaum, Alber S. Woodhull, *Operating Systems: Design and Implementation*, 2/e, Prentice-Hall

Pacchetti essenziali

Minix è un sistema molto piccolo e composto da pochi pacchetti. Questi hanno alcune particolarità: i nomi sono composti con lettere maiuscole e gli archivi compressi utilizzano la combinazione '**tar**'+'**compress**' e sono evidenziati dall'uso dell'estensione '`.TAZ`'.

Per prima cosa è necessario riprodurre la coppia di dischetti '`ROOT`' e '`USR`', a partire dai file omonimi. Il primo è in grado di avviarsi e contiene un file system minimo che si installa in un disco RAM, il secondo contiene una piccola serie di programmi da innestare nella directory '`/usr/`', che servono per poter installare Minix nel disco fisso. Se si ha poca memoria a disposizione (i classici 640 Kibyte

sono il minimo in assoluto per poter fare funzionare Minix), si può evitare l'utilizzo del disco RAM fondendo i due file in un solo dischetto. Data l'intenzione di questo capitolo viene descritta l'ultima di queste modalità di installazione.

Il mini sistema che si ottiene attraverso i due file appena citati, permette di installare, più o meno automaticamente, un insieme minimo di programmi contenuto nell'archivio 'USR.TAZ'

Esistono due versioni di questi tre file: una per architettura i386 o superiore e l'altra per i microprocessori inferiori. Date le intenzioni, si devono utilizzare i file della versione denominata '**i86**'.

Preparazione all'installazione

«

Il procedimento che viene descritto è valido sia per dischetti da 1440 Kibyte che da 1200 Kibyte. I due file 'ROOT' e 'USR' vanno copiati uno di seguito all'altro. Utilizzando GNU/Linux, si può fare nel modo seguente:

```
# cat ROOT USR > /dev/fd0 [Invio]
```

Dal punto di vista di Minix, il dischetto inserito nella prima unità a dischetti corrisponde al dispositivo '/dev/fd0', come per GNU/Linux, ma risulta diviso in partizioni: l'immagine 'ROOT' risulta essere '/dev/fd0a' e l'immagine 'USR' è '/dev/fd0c' (la partizione '**b**' è vuota).[2]

L'archivio 'USR.TAZ' deve essere suddiviso in diversi dischetti, con un procedimento un po' insolito: viene semplicemente tagliato a fettine della dimensione massima contenibile dal tipo di dischetti che si utilizza. Nel caso si tratti di dischetti da 1440 Kibyte, si può utilizzare GNU/Linux, o un altro sistema Unix, nel modo seguente:

```
# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=0 [Invio]

# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=1 [Invio]

# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=2 [Invio]
```

Se si trattasse di dischetti da 1200 Kibyte, occorrerebbe modificare la dimensione del blocco, come nell'esempio seguente:

```
# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=0 [Invio]

# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=1 [Invio]

# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=2 [Invio]
```

Nomi di dispositivo riferiti alle partizioni

Minix viene installato normalmente all'interno di una partizione primaria suddivisa in almeno due partizioni secondarie. La prima partizione serve a contenere il file system principale ed è di piccole dimensioni: 1440 Kibyte. La seconda serve per tutto il resto e viene innestata in corrispondenza della directory '`/usr/`'. Inizialmente le partizioni erano tre e '`/usr/`' era la terza. Attualmente, '`/usr/`' continua a essere la terza partizione e si finge che esista una seconda partizione senza alcuno spazio a disposizione.

Il primo disco fisso viene identificato dal dispositivo '`/dev/hd0`', il secondo da '`/dev/hd5`'. Le partizioni primarie del primo disco fisso vanno da '`/dev/hd1`' a '`/dev/hd4`'; quelle del secondo disco fisso da '`/dev/hd6`' a '`/dev/hd9`'. Le partizioni secondarie corrispondono al nome del dispositivo della partizione primaria con l'aggiunta di una lettera alfabetica che ne indica l'ordine.

- /dev/hd0

  - /dev/hd1
    * /dev/hd1a
    * /dev/hd1b
    * /dev/hd1c
    * ...
  - /dev/hd2
  - /dev/hd3
  - /dev/hd4
- /dev/hd5
  - /dev/hd6
  - /dev/hd7
  - /dev/hd8
  - /dev/hd9

Minix può essere installato in una partizione primaria qualunque, purché ci siano almeno 40 Mibyte a disposizione. Utilizzando la versione di Minix '**i86**', non conviene tentare di superare i 128 Mibyte.

Avvio

«

Minix utilizza un sistema di avvio piuttosto sofisticato; per fare un paragone con GNU/Linux, si tratta di qualcosa che compie le stesse funzioni di LILO, o di un cosiddetto *bootloader*.

Il sistema che svolge questa funzione in Minix si chiama *boot monitor* ed è importante capire subito come si utilizza se non si ha molta memoria RAM a disposizione, quanta ne richiederebbe un disco RAM per l'immagine '`ROOT`'.

Per cominciare, dopo aver preparato il dischetto ‘ROOT’+‘USR’, lo si inserisce senza la protezione contro la scrittura e si avvia l’elaboratore. Questo è ciò che appare.

```
Minix boot monitor 2.5

Press ESC to enter the monitor

Hit a key as follows:

    = Start Minix
```

Premendo il tasto [*Esc*] si attiva il *boot monitor*, mentre premendo [=] (si fa riferimento alla tastiera americana e questo simbolo si trova in corrispondenza della nostra lettera «ì») si avvia Minix con le impostazioni predefinite.

Dal momento che si immagina di avere a disposizione poca memoria (solo 1 Mibyte), non si può avviare Minix così, perché il contenuto dell’immagine ‘ROOT’ verrebbe caricato come disco RAM. È necessario utilizzare subito il *boot monitor*.

[*Esc*]

```
[ESC]
fd0>
```

Si ottiene un invito (*prompt*), attraverso il quale possono essere utilizzati alcuni comandi importanti per predisporre l’avvio del sistema Minix. Per ottenere aiuto si può utilizzare il comando ‘**help**’.

`fd0>` **help**[*Invio*]

Si ottiene un riepilogo dei comandi e del modo con cui possono essere utilizzati. Le cose più importanti che si possono fare con il *boot monitor* sono: l’avvio a partire da una partizione differente da quella prestabilita; l’assegnamento o il ripristino al valore predefinito di

una variabile.

Queste variabili sono solo entità riferite al sistema di avvio e la loro modifica permette di cambiare il modo con cui si avvia il kernel. Il comando '**set**' permette di elencare il contenuto di queste variabili.

```
fd0> set [Invio]
```

```
rootdev = (ram)
ramimagedev = (bootdev)
ramsize = (0)
processor = (286)
bus = (at)
memsize = (640)
emssize = (330)
video = (vga)
chrome = (mono)
image = (minix)
main() = (menu)
```

I valori appaiono tutti tra parentesi tonde perché rappresentano le impostazioni predefinite. Quando si cambia qualche valore, questo appare senza le parentesi.

La prima cosa da cambiare è il dispositivo di avvio, '**rootdev**'. Si deve assegnare il nome di dispositivo riferito all'immagine 'ROOT' su dischetto. Si tratta di '/dev/fd0a', come dire, la prima partizione secondaria del dischetto. In questo caso, il nome del dispositivo può anche essere indicato senza la parte iniziale, limitandolo al solo 'fd0a'.

```
fd0> rootdev=fd0a [Invio]

fd0> set [Invio]
```

```
rootdev = fd0a
ramimagedev = (bootdev)
ramsize = (0)
processor = (286)
bus = (at)
memsize = (640)
emssize = (330)
video = (vga)
chrome = (mono)
image = (minix)
main() = (menu)
```

Per avviare il sistema, basta utilizzare il comando '**boot**' senza argomenti.

`fd0>` **boot** [*Invio*]

In questo modo si lascia il *boot monitor* e si avvia il kernel. Una volta avviato il sistema, viene richiesto immediatamente l'innesto della seconda immagine, '`USR`', contenente gli strumenti necessari all'installazione. Avendo avviato senza disco RAM, il dischetto contenente l'immagine '`ROOT`' non può essere tolto e questo è il motivo per il quale deve essere contenuta nello stesso dischetto insieme a '`USR`'.[3]

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

Memory size = 970K   MINIX = 206K   RAM disk =    0K   Available = 765K

Mon Nov  3 15:24:15 MET 1997
Finish the name of device to mount as /usr: /dev/
```

Date le premesse, occorre specificare il nome del dispositivo corrispondente all'immagine '`USR`': si tratta di '`fd0c`'.

fd0c [*Invio*]

```
/dev/fd0c is read-write mounted on /usr
Starting standard daemons: update.

Login as root and run 'setup' to install Minix.

Minix  Release 2.0 Version 0

noname login:
```

root [*Invio*]

```
#
```

Installazione

«

L'installazione di Minix avviene in tre fasi:

1. preparazione della partizione di destinazione;
2. trasferimento del contenuto del dischetto, ovvero delle immagini 'ROOT' e 'USR';
3. dopo il riavvio, trasferimento dell'archivio 'USR.TAZ' e possibilmente, se si dispone di una partizione di almeno 40 Mibyte, anche di 'SYS.TAZ' e 'CMD.TAZ'.

Setup

«

Per iniziare l'installazione, dopo aver avviato il sistema Minix dal dischetto, si utilizza lo script '**setup**'.

```
# setup [Invio]
```

```
This is the Minix installation script.

Note 1: If the screen blanks suddenly then hit F3 to select "software
        scrolling".

Note 2: If things go wrong then hit DEL and start over.

Note 3: The installation procedure is described in the manual page
        usage(8).  It will be hard without it.

Note 4: Some questions have default answers, like this: [y]
        Simply hit RETURN (or ENTER) if you want to choose that answer.

Note 5: If you see a colon (:) then you should hit RETURN to continue.
:
```

[*Invio*]

Dopo la breve spiegazione, avendo premuto il tasto [*Invio*] si passa all'indicazione del tipo di tastiera. La scelta è ovvia, '**italian**', anche se non corrisponde esattamente: la barra verticale (quella per rappresentare i condotti) si trova al posto della lettera «ì» (i accentata). Durante questa fase di installazione conviene utilizzare la tastiera nazionale ('**italian**') per evitare spiacevoli incidenti quando si utilizza il programma di gestione delle partizioni.

```
What type of keyboard do you have?  You can choose one of:

    french  italian   latin-am  scandinavn  uk      us-swap
    german  japanese  olivetti  spanish     us-std

Keyboard type? [us-std]
```

italian [*Invio*]

```
Minix needs one primary partition of at least 30 Mb (it fits in 20 Mb, but
it needs 30 Mb if fully recompiled.  Add more space to taste.)

If there is no free space on your disk then you have to back up one of the
other partitions, shrink, and reinstall.  See the appropriate manuals of the
the operating systems currently installed.  Restart your Minix installation
after you have made space.

To make this partition you will be put in the editor "part".  Follow the
advice under the '!' key to make a new partition of type MINIX.  Do not
touch an existing partition unless you know precisely what you are doing!
Please note the name of the partition (hd1, hd2, ..., hd9, sd1, sd2, ...
sd9) you make.  (See the devices section in usage(8) on Minix device names.)
:
```

Il programma di Minix che permette di accedere alla tabella delle partizioni è '**`part`**' ed è ciò che sta per essere avviato. Come sempre, l'uso di un programma di questo genere è molto delicato: un piccolo errore mette fuori uso tutti i dati eventualmente contenuti in altre partizioni.

[*Invio*]

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device             Cyl Head Sec   Cyl Head Sec      Base      Size       Kb
    /dev/hd0                          ?    ?   ?
                         ?    ?   ?     ?    ?   ?         ?         ?        ?
Num Sort   Type
  ?  ?    ? ?            ?    ?   ?     ?    ?   ?         ?         ?        ?
  ?  ?    ? ?            ?    ?   ?     ?    ?   ?         ?         ?        ?
  ?  ?    ? ?            ?    ?   ?     ?    ?   ?         ?         ?        ?
  ?  ?    ? ?            ?    ?   ?     ?    ?   ?         ?         ?        ?

Type '+' or '-' to change, 'r' to read, '?' for more help, '!' for advice
```

Prima di utilizzare questo programma conviene leggere la sua guida interna, ottenibile con la pressione del tasto [*?*]. Il cursore si presenta inizialmente sull'indicazione del disco, '`/dev/hd0`', e può essere cambiato semplicemente premendo i tasti [+] o [−]. Una volta raggiunto il disco desiderato (in questo caso il primo disco va bene), si

deve leggere la sua tabella delle partizioni, in modo da rimpiazzare tutti i punti interrogativi che riempiono lo schermo.

[*r*]

```
   Select device       ----first----  --geom/last--  ------sectors-----
     Device             Cyl Head Sec   Cyl Head Sec      Base      Size       Kb
     /dev/hd0                          615    8  17
                          0    0   0   614    7  16         0     83640    41820
 Num Sort   Type
  1* hd1  86 DOS-BIG      0    1   0   613    7  16        17     83487    41743
  2  hd2  00 None         0    0   0     0    0  -1         0         0        0
  3  hd3  00 None         0    0   0     0    0  -1         0         0        0
  4  hd4  00 None         0    0   0     0    0  -1         0         0        0
```

Con questo esempio si suppone di avere solo un vecchio disco fisso MFM di circa 40 Mibyte, nel quale la prima partizione primaria risulta inizialmente dedicata al Dos. Così, basta cambiare il numero che identifica il tipo di partizione. Per farlo, vi si posiziona sopra il cursore, spostandolo con i tasti freccia, quindi si usano i tasti [+] o [−] fino a fare apparire il numero 81_{16}. Al primo intervento per cambiare un valore qualsiasi, viene richiesto esplicitamente se si intende modificare effettivamente i dati della tabella delle partizioni.

```
Do you wish to modify existing partitions (y/n) [y]
```

Una volta modificato il tipo, la prima partizione dovrebbe apparire così come segue:

```
 Num Sort   Type
  1* hd1  81 MINIX        0    1   0   613    7  16        17     83487    41743
```

Quindi si conclude.

[*q*]

```
Save partition table?  (y/n) [y]
```

Lo script di configurazione e installazione riprende richiedendo qua-

le sia la partizione su cui installare Minix. In questo caso si tratta della prima, cioè ‘`/dev/hd1`’.

```
Please finish the name of the primary partition you have created:

(Just type RETURN if you want to rerun "part")                    /dev/
```

hd1 [*Invio*]

```
You have created a partition named:     /dev/hd1
The following subpartitions are about to be created on /dev/hd3:

        Root subpartition:      /dev/hd1a       1440 kb
        /usr subpartition:      /dev/hd1c       rest of hd1

Hit return if everything looks fine, or hit DEL to bail out if you want to
think it over.  The next step will destroy /dev/hd1.
:
```

Come accennato in precedenza, Minix viene installato in due partizioni secondarie: la prima serve a contenere il file system principale, la seconda per il resto. In seguito si possono innestare anche partizioni successive.

```
Migrating from floppy to disk...


Scanning /dev/hd1c for bad blocks.  (Hit DEL to stop the scan if are absolutely
sure that there can not be any bad blocks.  Otherwise just wait.)
```

La scansione del disco fisso è necessaria se si utilizza un vecchio disco MFM, come si suppone di avere in questo esempio, mentre può essere inutile con un disco ATA. È importante fare attenzione: se ci sono settori inutilizzabili, vengono creati alcuni file ‘`/usr/.Bad_`$*$’ che **non vanno cancellati!** Alla fine, lo script procede a copiare il contenuto del dischetto nel disco fisso.

```
What is the memory size of this system in kilobytes? [4096 or more]
```

La dimensione di memoria RAM disponibile effettivamente è solo

di 970 Kibyte, quindi si inserisce questo valore; se la memoria a disposizione fosse maggiore o uguale a 4 Mibyte, non occorrerebbe indicare alcunché, basterebbe solo confermare.

970 [*Invio*]

```
Second level file system block cache set to 0 kb.
```

A questo punto, termina l'installazione del dischetto nel disco fisso e si può passare a riavviare il sistema da lì.

```
Please insert the installation ROOT floppy and type 'halt' to exit Minix.
You can type 'boot hd1' to try the newly installed Minix system.  See
"TESTING" in the usage manual.
```

halt [*Invio*]

```
System Halted
```

Avvio del sistema copiato nel disco fisso

Una volta conclusa l'esecuzione dello script di configurazione e installazione, si ritorna sotto il controllo del *boot monitor*, attraverso il quale è possibile avviare il sistema dalla prima partizione del disco fisso.

fd0> **boot /dev/hd1** [*Invio*]

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

at-hd0: 615x8x17

Memory size = 970K   MINIX = 206K   RAM disk =    0K   Available = 765K

Mon Nov  3 16:01:27 MET 1997
Starting standard daemons: update.

Login as root and run 'setup /usr' to install floppy sets.

Minix  Release 2.0 Version 0

noname login:
```

root [*Invio*]

Il suggerimento dato all'avvio ricorda che è possibile installare altre serie di dischetti, a cominciare da 'USR.TAZ', utilizzando il comando '**setup /usr**'.

Installazione delle serie di dischetti

«

Tra i pacchetti di Minix, 'USR.TAZ' è essenziale e cambia a seconda del tipo di architettura (i86 o i386). Però, dal momento che c'è spazio sufficiente nel disco fisso, conviene installare anche 'SYS.TAZ' per poter ricompilare il kernel e 'CMD.TAZ' che contiene i sorgenti dei vari programmi di servizio.

Tutti questi pacchetti devono essere suddivisi in dischetti nel modo visto in precedenza per il caso di 'USR.TAZ'.

setup /usr [*Invio*]

Lo script '**setup**' chiede una serie di conferme.

```
What is the size of the images on the diskettes? [all]
```

Premendo semplicemente [*Invio*] si intende che i dischetti vadano letti nella loro interezza.

[*Invio*]

```
What floppy drive to use? [0]
```

Premendo semplicemente [*Invio*] si fa riferimento alla prima unità, ‘`/dev/fd0`’.

[*Invio*]

```
Please insert input volume 1 and hit return
```

Si inserisce il primo dischetto e si conferma

[*Invio*]

Inizia la fase di estrazione di quanto contenuto nel primo dischetto, a partire dalla directory ‘`/usr/`’. Quando termina l’estrazione del primo dischetto, viene richiesto il successivo, fino alla conclusione.

Conviene ripetere la procedura fino a quando sono stati installati anche gli archivi ‘`SYS.TAZ`’ e ‘`CMD.TAZ`’.

Dischetto di avvio

Minix è molto semplice e non è necessario un dischetto di avvio realizzato appositamente. È sufficiente il dischetto utilizzato per iniziare l’installazione. Se si hanno difficoltà con l’avviamento di Minix dal disco fisso, si può avviare il *boot monitor* dal dischetto e con quello utilizzare il comando ‘**`boot /dev/hd1`**’.

Conclusione

«

Per chiudere l'attività di Minix, si può fare nel solito modo comune a quasi tutti i sistemi Unix.

```
# shutdown -h now[Invio]
```

Ricompilazione del kernel

«

Anche Minix, nella sua semplicità, richiede una ricompilazione del kernel per la sua ottimizzazione. In particolare, per poter attivare la gestione del TCP/IP occorre passare per la configurazione e ricompilazione.

Il file del kernel, secondo la tradizione di Minix, dovrebbe trovarsi nella directory radice e avere il nome '`minix`'. Se però, invece di trattarsi di un file, si tratta di una directory, nella fase di avvio viene eseguito il file più recente contenuto in tale directory. Il kernel normale, cioè quello che si trova dopo l'installazione, dovrebbe essere '`/minix/2.0.0`'.

Per poter ricompilare il kernel occorre avere installato il pacchetto '`SYS.TAZ`'. Si procede come segue:

1. si modifica il file '`/usr/include/minix/config.h`';
2. ci si posiziona nella directory '`/usr/src/tools/`';
3. si avvia la compilazione con il comando '**`make`**'.

Al termine si ottiene il file del kernel (o immagine) corrispondente a '`/usr/src/tools/image`' che si può copiare e rinominare come si ritiene più opportuno.

La configurazione che viene proposta deriva dagli esempi precedenti, in cui si ha una particolare penuria di memoria. Seguono solo alcuni pezzi.

```
/* If ROBUST is set to 1, writes of i-node, directory, and indirect blocks
 * from the cache happen as soon as the blocks are modified.  This gives a more
 * robust, but slower, file system.  If it is set to 0, these blocks are not
 * given any special treatment, which may cause problems if the system crashes.
 */
#define ROBUST              1    /* 0 for speed, 1 for robustness */
```

La macro '**ROBUST**' permette di sincronizzare le operazioni di accesso al disco. Nell'esempio mostrato si attiva questa opzione, in modo da poter utilizzare il sistema con tranquillità (e ovviamente con maggiore lentezza).

```
/* Number of slots in the process table for user processes. */
#define NR_PROCS          32
```

Il numero massimo dei processi eseguibili può essere una seria limitazione all'uso simultaneo dell'elaboratore da parte di più utenti, ma la scarsa memoria a disposizione consiglia di mantenere basso questo valore.

```
/* Enable or disable the second level file system cache on the RAM disk. */
#define ENABLE_CACHE2      0
```

Sempre a causa della carenza di memoria, è opportuno disabilitare la memoria cache.

```
/* Include or exclude device drivers.  Set to 1 to include, 0 to exclude. */
#define ENABLE_NETWORKING  1    /* enable TCP/IP code */
#define ENABLE_AT_WINI     1    /* enable AT winchester driver */
#define ENABLE_BIOS_WINI   1    /* enable BIOS winchester driver */
#define ENABLE_ESDI_WINI   1    /* enable ESDI winchester driver */
#define ENABLE_XT_WINI     0    /* enable XT winchester driver */
#define ENABLE_ADAPTEC_SCSI 0   /* enable ADAPTEC SCSI driver */
#define ENABLE_MITSUMI_CDROM 0  /* enable Mitsumi CD-ROM driver */
#define ENABLE_SB_AUDIO    0    /* enable Soundblaster audio driver */
```

In questa sezione è importante abilitare ciò che serve ed eliminare il resto. In particolare, è qui che si attiva la connettività TCP/IP, che non risulta attivata in modo predefinito.

```
/* NR_CONS, NR_RS_LINES, and NR_PTYS determine the number of terminals the
 * system can handle.
 */
#define NR_CONS             2    /* # system consoles (1 to 8) */
#define NR_RS_LINES         1    /* # rs232 terminals (0, 1, or 2) */
#define NR_PTYS             2    /* # pseudo terminals (0 to 64) */
```

Il numero predefinito di console virtuali è due, ma può essere espanso, sempre che ciò possa avere senso date le limitazioni del sistema. Invece è importante attivare gli pseudoterminali, cioè il numero massimo di connessioni remote. Volendo gestire la rete, è il caso di indicare almeno uno pseduoterminale.

Per modificare il file '`/usr/include/minix/config.h`' si può utilizzare '**`vi`**', che è un collegamento a '**`elvis`**', oppure '**`elle`**'.[4]

Si procede con la compilazione.

```
# cd /usr/src/tools [Invio]

# make [Invio]
```

Al termine della compilazione, se non sono occorsi incidenti, si ottiene il file '`image`'.

```
# cp image /minix/rete.0.1 [Invio]
```

Questo dovrebbe bastare, trattandosi del file più recente nella directory '`/minix/`', è anche quello che viene avviato la volta successiva.

```
# shutdown -h [Invio]
```

Quando si ricompila il kernel è probabile che si renda necessaria la creazione di file di dispositivo che in altre situazioni non sarebbero necessari. Nel caso della gestione della rete, sono necessari i file seguenti:

- '`/dev/eth`';
- '`/dev/ip`';
- '`/dev/tcp`';
- '`/dev/udp`';
- '`/dev/ttyp0`' e successivi;
- '`/dev/ptyp0`' e successivi.

Questo ragionamento vale anche per le console virtuali: se si vogliono molte console, forse è necessario aggiungere i file relativi.

Probabilmente c'è già tutto ciò di cui si può avere bisogno, ma se manca si può creare con lo script '**MAKEDEV**'.

```
MAKEDEV dispositivo
```

Per esempio, trovandosi già nella directory '`/dev/`', si può creare il dispositivo '`/dev/tcp`' nel modo seguente:

```
# MAKEDEV tcp [Invio]
```

Parametri di avvio

Anche Minix richiede alcuni parametri di avvio in presenza di hardware particolare. La gestione di questi avviene in modo molto semplice attraverso il *boot monitor*: basta definire una nuova variabile, assegnandole il valore corretto.

Scheda di rete

Per gestire una rete occorre una scheda di rete Ethernet. Nell'esempio seguente si immagina di disporre di una scheda compatibile con il modello NE2000 configurata con indirizzo di I/O 300_{16} e IRQ 11.

Il parametro di avvio per ottenere il riconoscimento della scheda Ethernet è '**DPETH*n***', dove ***n*** è il numero della scheda, a partire da zero.

```
DPETHn=indirizzo_i/o:irq:indirizzo_di_memoria
```

La scheda NE2000 non utilizza alcun indirizzo di memoria, quindi, per il nostro esempio occorre il parametro seguente:

```
DPETH0=300:11
```

Come si vede, l'indirizzo di I/O è espresso implicitamente in esadecimale e l'IRQ in decimale, mentre l'indirizzo di memoria viene omesso trattandosi di una NE2000. Per inserire tale parametro si utilizza il *boot monitor* nel modo seguente:

```
hd0> DPETH0=300:11 [Invio]

hd0> save [Invio]
```

L'ultima istruzione, '**save**', salva questo parametro che altrimenti dovrebbe essere indicato ogni volta che si avvia il sistema.

Se la scheda di rete viene riconosciuta, all'avvio appare il messaggio seguente:

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

ne2000: NE2000 at 300:11
```

Configurazione della rete

La configurazione della rete va fatta con cura, in modo da non avere bisogno di alcuni demoni che permettono una sorta di autoconfigurazione. Negli esempi seguenti si configura il nuovo sistema Minix tenendo conto di questa situazione:

- *dinkel.brot.dg*, IP 192.168.1.1, servizio DNS e router predefinito;
- *minix.brot.dg*, IP 192.168.1.25, elaboratore Minix.

Per quanto possibile, si fa in modo di non avere bisogno del DNS.

/etc/hosts

Volendo attivare localmente la risoluzione dei nomi e degli indirizzi è necessario il file '`/etc/hosts`', che va configurato come al solito, esattamente come si fa con GNU/Linux.

```
127.0.0.1       localhost
192.168.1.1     dinkel.brot.dg
192.168.1.25    minix.brot.dg
```

File «/etc/hostname.file»

«

Il file ‘`/etc/hostname.file`’ serve solo a definire il nome dell’elaboratore locale, in senso generale. Non ha niente a che vedere con le interfacce di rete.

```
# echo "minix.brot.dg" > /etc/hostname.file[Invio]
```

File «/etc/resolv.conf»

«

Il file ‘`/etc/resolv.conf`’ permette di indicare gli indirizzi dei nodi che forniscono un servizio DNS. Nell’esempio proposto si vuole fare in modo che il sistema di risoluzione dei nomi avvenga localmente, per mezzo di quanto contenuto nel file ‘`/etc/hosts`’. Per questo viene indicato come servente DNS anche l’indirizzo locale (*loopback*).

```
nameserver 127.0.0.1
nameserver 192.168.1.1
```

File «/etc/rc.net»

«

Lo script ‘`/etc/rc.net`’ viene utilizzato da ‘`/etc/rc`’ per attivare la rete. Lo si può utilizzare per attivare l’interfaccia di rete e per definire l’instradamento verso il router (l’instradamento verso la rete connessa all’interfaccia è predefinito).

```
# Attiva l'interfaccia e l'instradamento verso la sua rete.
ifconfig -h 192.168.1.25

# Definisce l'instradamento predefinito verso il router
add_route -g 192.168.1.1
```

«

Probabilmente, è utile ritoccare il file ‘/etc/rc’, per eliminare l’avvio automatico di alcuni demoni inutili dal momento che la rete è configurata. Quello che segue è il pezzo che attiva la gestione della rete.

```
# Network initialization.
(</dev/eth </dev/tcp) 2>/dev/null && net=true   # Is there a TCP/IP server?

if [ "$net" -a -f /etc/rc.net ]
then
        # There is a customized TCP/IP initialization script; run it.
        . /etc/rc.net
elif [ "$net" ] && [ "`hostaddr -e`" = 0:0:0:0:0:0 ]
then
        # No network hardware, configure a fixed address to run TCP/IP alone.
        ifconfig -h 192.9.200.1
fi

if [ "$net" ]
then
        echo -n "Starting network daemons: "
        for daemon in rarpd nonamed irdpd talkd
        do
                if [ -f /usr/bin/$daemon ]
                then
                        echo -n " $daemon"
                        $daemon &
                fi
        done
        echo .

        # Get the nodename from the DNS and set it.
        hostaddr -a >/etc/hostname.file || echo noname >/etc/hostname.file

        echo -n "Starting network services:"
        for pair in 'shell in.rshd' 'login in.rld' \
                        'telnet in.telnetd' 'ftp in.ftpd'
        do
                set $pair
                if [ -f /usr/bin/$2 ]
                then
```

```
                        echo -n " $1"
                        tcpd $1 /usr/bin/$2 &
                fi
        done
        echo .
fi
```

Vale la pena di modificare quanto segue:

```
if [ "$net" ]
then
        echo -n "Starting network daemons: "
        for daemon in nonamed talkd ### rarpd nonamed irdpd talkd
        do
        ...
```

Nel pezzo precedente non vengono avviati i demoni '**rarpd**' e '**irdpd**', che sono necessari rispettivamente per ottenere l'indirizzo IP in base all'indirizzo hardware della scheda Ethernet e a definire gli instradamenti verso i router. Eventualmente, si potrebbe anche evitare di avviare '**talkd**' se non si intende utilizzare '**talk**'. Il demone '**nonamed**' è necessario se non si vuole essere obbligati ad avere un servizio DNS esterno; in pratica è necessario perché venga interpretato il contenuto del file '`/etc/hosts`'.

Personalizzazione

Il sistema risulta configurato in maniera piuttosto disordinata, a cominciare dal fatto che la directory personale dell'utente '**root**' corrisponde alla directory radice; così, al suo interno si trovano i file di configurazione dell'amministratore. Probabilmente, la prima cosa da fare è quella di creare una directory '`/root/`', porvi al suo interno i file di configurazione (dovrebbe trattarsi di '`.ellepro.b1`', '`.exrc`' e '`.profile`'), modificando anche il file '`/etc/passwd`' in modo da assegnare all'utente '**root**' questa nuova directory.

File «/etc/passwd», «/etc/group» e «/etc/shadow»

Minix, nonostante la sua semplicità, utilizza le parole d'ordine oscurate (*shadow password*). Pertanto, se si tenta di inserire un utente manualmente, occorre intervenire anche su questo file, '`/etc/shadow`', altrimenti l'utente non può accedere.

Il file '`/etc/group`', se non va bene com'è, deve essere modificato manualmente, mentre per gli utenti conviene affidarsi allo script '**`adduser`**'.

```
adduser utente gruppo directory_home
```

Dopo aver creato un utente, come al solito è opportuno utilizzare il programma '**`passwd`**' per assegnare la parola d'ordine.[5]

Tastiera

La mappa della tastiera viene definita attraverso il programma '**`loadkeys`**' e il file contenente la mappa desiderata. Per cui,

```
# loadkeys ./tastiera.map [Invio]
```

permette di caricare la mappa del file '`tastiera.map`' contenuto nella directory corrente.

La mappa della tastiera, secondo la scelta fatta durante l'installazione di Minix, avviene per mezzo del file '`/etc/keymap`': se lo script '`/etc/rc`' lo trova durante la fase di avvio, lo carica attraverso '**`loadkeys`**'.

«

La configurazione della tastiera italiana, per quanto riguarda la versione 2.0 di Minix, non è perfetta. Per modificare la mappa occorre intervenire sul file '`/usr/src/kernel/keymaps/italian.src`'. Dopo la modifica si deve compilare il sorgente in modo da ottenere il file '`/usr/src/kernel/keymaps/italian.map`'. Al termine, questo file va copiato e rinominato in modo da sostituire '`/etc/keymap`'.

Il sorgente corretto potrebbe apparire come nell'esempio seguente, in particolare, per ottenere la tilde ('**~**') si deve usare la combinazione [*AltGr ì*], mentre per ottenere l'apostrofo inverso ('`') si deve usare la combinazione [*AltGr '*]. I caratteri che si trovano oltre il settimo bit, vengono rappresentati in ottale.

```
/* Modified by Daniele Giacomini  daniele @ swlibero.org    1998.12.22  */
/* Keymap for Italian standard keyboard, similar to Linux layout.       */

u16_t keymap[NR_SCAN_CODES * MAP_COLS] = {

/* scan-code            !Shift  Shift   Alt     AltGr   Alt+Sh  Ctrl    */
/* ========================================================================= */
/* 00 - none    */      0,      0,      0,      0,      0,      0,
/* 01 - ESC     */      C('['), C('['), CA('['),C('['), C('['), C('['),
/* 02 - '1'     */      '1',    '!',    A('1'), '1',    '!',    C('A'),
/* 03 - '2'     */      '2',    '"',    A('2'), '2',    '@',    C('@'),
/* 04 - '3'     */      '3',    0234,   A('3'), '3',    0234,   C('C'),
/* 05 - '4'     */      '4',    '$',    A('4'), '4',    '$',    C('D'),
/* 06 - '5'     */      '5',    '%',    A('5'), '5',    '%',    C('E'),
/* 07 - '6'     */      '6',    '&',    A('6'), '6',    '&',    C('F'),
/* 08 - '7'     */      '7',    '/',    A('7'), '{',    '/',    C('G'),
/* 09 - '8'     */      '8',    '(',    A('8'), '[',    '(',    C('H'),
/* 10 - '9'     */      '9',    ')',    A('9'), ']',    ')',    C('I'),
/* 11 - '0'     */      '0',    '=',    A('0'), '}',    '=',    C('@'),
/* 12 - '-'     */      '\'',   '?',    A('\''),'\`',   '?',    C('@'),
/* 13 - '='     */      0215,   '^',    0215,   '~',    '^',    C('^'),
/* 14 - BS      */      C('H'), C('H'), CA('H'),C('H'), C('H'), 0177,
/* 15 - TAB     */      C('I'), C('I'), CA('I'),C('I'), C('I'), C('I'),
```

```
/* 16 - 'q'      */      L('q'), 'Q',    A('q'), 'q',    'Q',    C('Q'),
/* 17 - 'w'      */      L('w'), 'W',    A('w'), 'w',    'W',    C('W'),
/* 18 - 'e'      */      L('e'), 'E',    A('e'), 'e',    'E',    C('E'),
/* 19 - 'r'      */      L('r'), 'R',    A('r'), 'r',    'R',    C('R'),
/* 20 - 't'      */      L('t'), 'T',    A('t'), 't',    'T',    C('T'),
/* 21 - 'y'      */      L('y'), 'Y',    A('y'), 'y',    'Y',    C('Y'),
/* 22 - 'u'      */      L('u'), 'U',    A('u'), 'u',    'U',    C('U'),
/* 23 - 'i'      */      L('i'), 'I',    A('i'), 'i',    'I',    C('I'),
/* 24 - 'o'      */      L('o'), 'O',    A('o'), 'o',    'O',    C('O'),
/* 25 - 'p'      */      L('p'), 'P',    A('p'), 'p',    'P',    C('P'),
/* 26 - '['      */      0212,   0202,   0212,   '[',    '{',    C('['),
/* 27 - ']'      */      '+',    '*',    A('+'), ']',    '}',    C(']'),
/* 28 - CR/LF    */      C('M'), C('M'), CA('M'),C('M'), C('M'), C('J'),
/* 29 - Ctrl     */      CTRL,   CTRL,   CTRL,   CTRL,   CTRL,   CTRL,
/* 30 - 'a'      */      L('a'), 'A',    A('a'), 'a',    'A',    C('A'),
/* 31 - 's'      */      L('s'), 'S',    A('s'), 's',    'S',    C('S'),
/* 32 - 'd'      */      L('d'), 'D',    A('d'), 'd',    'D',    C('D'),
/* 33 - 'f'      */      L('f'), 'F',    A('f'), 'f',    'F',    C('F'),
/* 34 - 'g'      */      L('g'), 'G',    A('g'), 'g',    'G',    C('G'),
/* 35 - 'h'      */      L('h'), 'H',    A('h'), 'h',    'H',    C('H'),
/* 36 - 'j'      */      L('j'), 'J',    A('j'), 'j',    'J',    C('J'),
/* 37 - 'k'      */      L('k'), 'K',    A('k'), 'k',    'K',    C('K'),
/* 38 - 'l'      */      L('l'), 'L',    A('l'), 'l',    'L',    C('L'),
/* 39 - ';'      */      0225,   0207,   0225,   '@',    '@',    C('@'),
/* 40 - '\''     */      0205,   0370,   0205,   '#',    '#',    C('@'),
/* 41 - '`'      */      '\\',   '|',    '\\',   '\\',   '|',    C('\\'),
/* 42 - l. SHIFT*/      SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,
/* 43 - '\\'     */      0227,   '|',    0227,   0227,   '|',    C('@'),
/* 44 - 'z'      */      L('z'), 'Z',    A('z'), 'z',    'Z',    C('Z'),
/* 45 - 'x'      */      L('x'), 'X',    A('x'), 'x',    'X',    C('X'),
/* 46 - 'c'      */      L('c'), 'C',    A('c'), 'c',    'C',    C('C'),
/* 47 - 'v'      */      L('v'), 'V',    A('v'), 'v',    'V',    C('V'),
/* 48 - 'b'      */      L('b'), 'B',    A('b'), 'b',    'B',    C('B'),
/* 49 - 'n'      */      L('n'), 'N',    A('n'), 'n',    'N',    C('N'),
/* 50 - 'm'      */      L('m'), 'M',    A('m'), 'm',    'M',    C('M'),
/* 51 - ','      */      ',',    ';',    A(','), ',',    ';',    C('@'),
/* 52 - '.'      */      '.',    ':',    A('.'), '.',    ':',    C('@'),
/* 53 - '/'      */      '-',    '_',    A('-'), '-',    '_',    C('_'),
/* 54 - r. SHIFT*/      SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,
/* 55 - '*'      */      '*',    '*',    A('*'), '*',    '*',    C('M'),
/* 56 - ALT      */      ALT,    ALT,    ALT,    ALT,    ALT,    ALT,
/* 57 - ' '      */      ' ',    ' ',    A(' '), ' ',    ' ',    C('@'),
/* 58 - CapsLck */      CALOCK, CALOCK, CALOCK, CALOCK, CALOCK, CALOCK,
/* 59 - F1       */      F1,     SF1,    AF1,    AF1,    ASF1,   CF1,
```

```
/* 60 - F2       */      F2,      SF2,     AF2,     AF2,     ASF2,    CF2,
/* 61 - F3       */      F3,      SF3,     AF3,     AF3,     ASF3,    CF3,
/* 62 - F4       */      F4,      SF4,     AF4,     AF4,     ASF4,    CF4,
/* 63 - F5       */      F5,      SF5,     AF5,     AF5,     ASF5,    CF5,
/* 64 - F6       */      F6,      SF6,     AF6,     AF6,     ASF6,    CF6,
/* 65 - F7       */      F7,      SF7,     AF7,     AF7,     ASF7,    CF7,
/* 66 - F8       */      F8,      SF8,     AF8,     AF8,     ASF8,    CF8,
/* 67 - F9       */      F9,      SF9,     AF9,     AF9,     ASF9,    CF9,
/* 68 - F10      */      F10,     SF10,    AF10,    AF10,    ASF10,   CF10,
/* 69 - NumLock  */      NLOCK,   NLOCK,   NLOCK,   NLOCK,   NLOCK,   NLOCK,
/* 70 - ScrLock  */      SLOCK,   SLOCK,   SLOCK,   SLOCK,   SLOCK,   SLOCK,
/* 71 - Home     */      HOME,    '7',     AHOME,   AHOME,   '7',     CHOME,
/* 72 - CurUp    */      UP,      '8',     AUP,     AUP,     '8',     CUP,
/* 73 - PgUp     */      PGUP,    '9',     APGUP,   APGUP,   '9',     CPGUP,
/* 74 - '-'      */      NMIN,    '-',     ANMIN,   ANMIN,   '-',     CNMIN,
/* 75 - Left     */      LEFT,    '4',     ALEFT,   ALEFT,   '4',     CLEFT,
/* 76 - MID      */      MID,     '5',     AMID,    AMID,    '5',     CMID,
/* 77 - Right    */      RIGHT,   '6',     ARIGHT,  ARIGHT,  '6',     CRIGHT,
/* 78 - '+'      */      PLUS,    '+',     APLUS,   APLUS,   '+',     CPLUS,
/* 79 - End      */      END,     '1',     AEND,    AEND,    '1',     CEND,
/* 80 - Down     */      DOWN,    '2',     ADOWN,   ADOWN,   '2',     CDOWN,
/* 81 - PgDown   */      PGDN,    '3',     APGDN,   APGDN,   '3',     CPGDN,
/* 82 - Insert   */      INSRT,   '0',     AINSRT,  AINSRT,  '0',     CINSRT,
/* 83 - Delete   */      0177,    '.',     A(0177), 0177,    '.',     0177,
/* 84 - Enter    */      C('M'),  C('M'),  CA('M'), C('M'),  C('M'),  C('J'),
/* 85 - ???      */      0,       0,       0,       0,       0,       0,
/* 86 - ???      */      '<',     '>',     A('<'),  '|',     '>',     C('@'),
/* 87 - F11      */      F11,     SF11,    AF11,    AF11,    ASF11,   CF11,
/* 88 - F12      */      F12,     SF12,    AF12,    AF12,    ASF12,   CF12,
/* 89 - ???      */      0,       0,       0,       0,       0,       0,
/* 90 - ???      */      0,       0,       0,       0,       0,       0,
/* 91 - ???      */      0,       0,       0,       0,       0,       0,
/* 92 - ???      */      0,       0,       0,       0,       0,       0,
/* 93 - ???      */      0,       0,       0,       0,       0,       0,
/* 94 - ???      */      0,       0,       0,       0,       0,       0,
/* 95 - ???      */      0,       0,       0,       0,       0,       0,
/* 96 - EXT_KEY  */      EXTKEY,  EXTKEY,  EXTKEY,  EXTKEY,  EXTKEY,  EXTKEY,
/* 97 - ???      */      0,       0,       0,       0,       0,       0,
/* 98 - ???      */      0,       0,       0,       0,       0,       0,
/* 99 - ???      */      0,       0,       0,       0,       0,       0,
/*100 - ???      */      0,       0,       0,       0,       0,       0,
/*101 - ???      */      0,       0,       0,       0,       0,       0,
/*102 - ???      */      0,       0,       0,       0,       0,       0,
/*103 - ???      */      0,       0,       0,       0,       0,       0,
```

```
/*104 - ???     */      0,      0,      0,      0,      0,      0,
/*105 - ???     */      0,      0,      0,      0,      0,      0,
/*106 - ???     */      0,      0,      0,      0,      0,      0,
/*107 - ???     */      0,      0,      0,      0,      0,      0,
/*108 - ???     */      0,      0,      0,      0,      0,      0,
/*109 - ???     */      0,      0,      0,      0,      0,      0,
/*110 - ???     */      0,      0,      0,      0,      0,      0,
/*111 - ???     */      0,      0,      0,      0,      0,      0,
/*112 - ???     */      0,      0,      0,      0,      0,      0,
/*113 - ???     */      0,      0,      0,      0,      0,      0,
/*114 - ???     */      0,      0,      0,      0,      0,      0,
/*115 - ???     */      0,      0,      0,      0,      0,      0,
/*116 - ???     */      0,      0,      0,      0,      0,      0,
/*117 - ???     */      0,      0,      0,      0,      0,      0,
/*118 - ???     */      0,      0,      0,      0,      0,      0,
/*119 - ???     */      0,      0,      0,      0,      0,      0,
/*120 - ???     */      0,      0,      0,      0,      0,      0,
/*121 - ???     */      0,      0,      0,      0,      0,      0,
/*122 - ???     */      0,      0,      0,      0,      0,      0,
/*123 - ???     */      0,      0,      0,      0,      0,      0,
/*124 - ???     */      0,      0,      0,      0,      0,      0,
/*125 - ???     */      0,      0,      0,      0,      0,      0,
/*126 - ???     */      0,      0,      0,      0,      0,      0,
/*127 - ???     */      0,      0,      0,      0,      0,      0
};
```

Dopo la modifica, si avvia la compilazione.

cd /usr/src/kernel/keymaps/ [*Invio*]

make [*Invio*]

Vengono generati tutti i file di configurazione che non siano già presenti (se si vuole ripetere la compilazione occorre prima rimuovere il file ‘`italian.map`’).

cp italian.map /etc/keymap [*Invio*]

Al prossimo riavvio si utilizza così la nuova mappa.

Altri programmi

Il software a disposizione per Minix non è molto e può essere trovato negli stessi FTP da cui si accede ai file del sistema operativo citati qui. In particolare, tra i programmi riferiti alla rete, vale la pena di ricordare i pacchetti ‘`HTTPD.TAZ`’ e ‘`FROG.TAZ`’. Il primo è un servente HTTP molto semplice e il secondo è un programma per il tracciamento dell’instradamento (simile a Traceroute). Sono entrambi molto utili e compilabili facilmente.

Pacchetto «HTTPD.TAZ»

Minix dispone di un servente HTTP elementare e lo si trova distribuito nel pacchetto ‘`HTTPD.TAZ`’. I pacchetti supplementari, come questo, vanno installati a partire dalla directory ‘`/usr/local/`’ e i sorgenti vanno collocati in ‘`/usr/local/src/`’.

```
# cd /usr/local/src[Invio]
```

Supponendo che il file ‘`HTTPD.TAZ`’ si trovi nel dischetto, innestato nella directory ‘`/mnt/`’, si può agire come segue:

```
# cat /mnt/HTTPD.TAZ | compress -d | tar xvf -[Invio]
```

Si ottiene la creazione della directory ‘`httpd/`’ a partire dalla posizione corrente, cioè ‘`/usr/local/src/`’.

```
# cd httpd[Invio]
```

Si procede con la compilazione.

```
# make[Invio]
```

Quindi si installa il programma.

```
# make install[Invio]
```

L'installazione non si occupa di copiare i file di configurazione: bisogna farlo manualmente.

```
# cp httpd.conf /etc[Invio]

# cp httpd.mtype /etc[Invio]
```

Il demone '**httpd**', installato in '`/usr/local/bin/`', ha bisogno di un utente '**www**', in base alla configurazione predefinita del file '`/etc/httpd.conf`' che non serve modificare.

```
# adduser www operator /usr/home/www[Invio]
```

Naturalmente la scelta della directory personale di questo utente fittizio è solo un fatto di gusto personale. Sempre in base alla configurazione predefinita, occorre aggiungere alla directory personale la directory '`exec/`' e all'interno di questa si devono collocare un paio di file.

```
# mkdir /usr/home/www/exec[Invio]

# cp dir2html /usr/home/www/exec[Invio]

# cp dir2html.sh /usr/home/www/exec[Invio]
```

Per ultimo, occorre avviare il demone. Per questo conviene ritoccare il file '`/etc/rc.net`' in modo da aggiungere la riga seguente:

```
tcpd   http   /usr/local/bin/httpd &
```

Pacchetto «FROG.TAZ»

«

'**frog**' è un programma per il tracciamento dell'instradamento. È semplice, ma anche molto importante. L'estrazione dell'archivio avviene nel modo solito, così come la compilazione e l'installazione.

```
# cd /usr/local/src [Invio]

# cat /mnt/FROG.TAZ | compress -d | tar xvf - [Invio]

# cd frog [Invio]

# make [Invio]

# make install [Invio]
```

Tutto qui.

Copie di sicurezza

Le copie di sicurezza possono essere fatte soltanto utilizzano '**tar**' e '**compress**'. Dal momento che il sistema è organizzato in modo piuttosto rigido, con una partizione principale molto piccola e una partizione '/usr/', normalmente conviene preoccuparsi solo di questa seconda partizione. Per la prima converrebbe realizzare un dischetto di avvio e installazione con gli stessi file di configurazione, compresi '/etc/passwd', '/etc/group' e '/etc/shadow'.

Archiviazione

Se si dispone di abbastanza spazio libero nella partizione '/usr/', se ne può fare la copia di sicurezza in un file collocato all'interno della stessa partizione. Successivamente si può scaricare su dischetti. Si può procedere nel modo seguente:

```
# cd /usr [Invio]

# tar cf - .  | compress -c > .BKP.TAZ [Invio]
```

Si ottiene il file '/usr/.BKP.TAZ' contenente la copia di quanto contenuto nella directory corrente '/usr/'. Successivamente si può

copiare il file ottenuto, a pezzi, su una serie di dischetti inizializzati in precedenza. Si comincia con l'inizializzazione e si suppone di disporre di dischetti da 1440 Kibyte.

```
# format /dev/fd0 1440 [Invio]
```

...

Una volta preparati i dischetti, si può scaricare il file nei dischetti.

```
# dd if=/usr/.BKP.TAZ of=/dev/fd0 bs=1440k count=1 skip=0
[Invio]

# dd if=/usr/.BKP.TAZ of=/dev/fd0 bs=1440k count=1 skip=2
[Invio]
```

...

Recupero

Per recuperare un sistema archiviato nel modo mostrato nella sezione precedente, si deve cominciare dall'installazione con il dischetto iniziale. La cosa migliore sarebbe l'utilizzo di un dischetto modificato opportunamente in modo che i file di configurazione corrispondano a quanto utilizzato nel proprio sistema.

Dopo l'installazione iniziale che consiste nel trasferimento di quanto contenuto nel dischetto iniziale nel disco fisso, si procede con l'installazione della copia (preparata in precedenza) della partizione collocata a partire da '`/usr/`'.

```
# setup /usr [Invio]
```

Uno dopo l'altro vengono richiesti tutti i dischetti.

Convivenza tra Minix e GNU/Linux

«

Se lo si desidera, si può fare convivere Minix assieme a GNU/Linux, nello stesso disco fisso, su partizioni distinte. Ma installare Minix in una partizione libera di un disco in cui GNU/Linux è già stato installato richiede prudenza e attenzione.

- L'installazione di Minix provoca l'alterazione dell'MBR del disco fisso, di conseguenza, al termine si avvia solo Minix. Quindi, prima di installare Minix occorre preparare uno o più dischi di avvio di GNU/Linux, in modo da poter in seguito ripristinare il sistema di avvio attraverso LILO.
- Quando si conclude il lavoro con Minix e si esegue un riavvio con un semplice [*Ctrl Alt Canc*], si ottiene un avvio a caldo (*warm boot*), ma se dopo si vuole avviare un kernel Linux, **questo non può essere caricato**. Pertanto, è necessario un riavvio a freddo, al limite attraverso lo spegnimento e la riaccensione dell'elaboratore.

LILO

«

Una volta che si è riusciti a fare riavviare il sistema GNU/Linux, con i dischetti di avvio a cui si faceva riferimento in precedenza, conviene modificare il file '`/etc/lilo.conf`' in modo che si possa scegliere tra l'avvio di GNU/Linux, Minix ed eventualmente altro.

Supponendo di avere installato Minix nella seconda partizione del primo disco fisso, le righe necessarie nel file '`/etc/lilo.conf`' sono quelle seguenti:

```
other=/dev/hda2
        label=minix
        table=/dev/hda
```

Volendo supporre che Minix sia stato installato nel secondo disco fisso, sempre nella seconda partizione, le righe sarebbero quelle seguenti:

```
other=/dev/hdb2
        label=minix
        table=/dev/hdb
        loader=/boot/chain.b
```

In pratica, è esattamente ciò che si fa quando si vuole controllare l'avvio del Dos.

Riferimenti

Minix è un sistema operativo molto limitato rispetto a GNU/Linux. Resta comunque l'unica opportunità, almeno per ora, di fronte a vecchi elaboratori i286 o inferiori.

Molti particolari importanti non sono stati descritti, ma le informazioni relative sono comunque accessibili dai siti FTP di distribuzione di Minix e dalla documentazione interna costituita delle pagine di manuale ('**man**').

- Andrew S. Tanenbaum, Alber S. Woodhull, *Operating Systems: Design and Implementation*, 2/e, Prentice-Hall
- *http://www.cs.vu.nl/~ast/minix.html*

[1] **Minix** licenza simile a BSD

[2] Se si trattasse della seconda unità a dischetti, si parlerebbe di '`/dev/fd1`', '`/dev/fd1a`' e '`/dev/fd1c`'.

[3] Una cosa da sapere subito è che Minix non utilizza la sequenza [*Ctrl C*] per interrompere un programma. Per questo si usa il tasto [*Canc*] da solo.

[4] Il programma '**elvis**' in particolare, è molto tradizionale: i tasti freccia generano proprio le lettere corrispondenti e quindi non possono essere usati durante la fase di inserimento. '**elle**' è un Emacs ridotto al minimo.

[5] Lo script '**adduser**' si avvale della directory personale dell'utente '**ast**' per inserire i file di configurazione iniziali. Questa directory, corrispondente a '`/usr/ast/`', svolge il ruolo di scheletro delle directory personali da creare. Volendo si può realizzare un proprio script per rendere la cosa più elegante.

ELKS: introduzione

ELKS, [1] ovvero *Embeddable linux kernel subset*, è un sistema operativo estremamente ridotto, in grado di funzionare con microprocessori i86 (16 bit), a partire dai primi (i8086 e i8088). Come dichiara il nome, si tratta di un sistema dove il kernel è derivato da Linux.

Le funzionalità disponibili sono minime e difficilmente il suo sviluppo può arrivare a un buon livello di affidabilità. Inoltre, sono sempre di meno i programmatori competenti e interessati a questo tipo di piattaforma; tuttavia, si tratta di un lavoro che potrebbe essere utile a livello didattico e sarebbe un vero peccato che venisse abbandonato del tutto.

Strumenti di sviluppo

Per compilare il kernel e i programmi di servizio che compongono il sistema operativo, è necessario un compilatore apposito, che però si usa su un sistema GNU/Linux standard. Il pacchetto del compilatore e degli strumenti di sviluppo associati è denominato Dev86 ed

è distribuito normalmente in forma sorgente, assieme ai sorgenti di ELKS.

Una volta scaricato il pacchetto Dev86, questo può essere espanso in una directory qualunque nell'elaboratore GNU/Linux, come mostrato dall'esempio seguente, dove però, successivamente possa acquisire i privilegi dell'utente '**root**':

```
$ cd [Invio]

$ mkdir ELKS [Invio]

$ cd ELKS [Invio]

# tar xzvf Dev86src-0.16.0.tar.gz [Invio]
```

Si ottiene la directory '`dev86-0.16.0/`' che si articola ulteriormente. Terminata l'installazione occorre compilare questi sorgenti e installarli. In questo caso si prevede di installare Dev86 a partire da '`/opt/dev86/`':

```
$ cd dev86-0.16.0 [Invio]

$ make PREFIX=/opt/dev86/ [Invio]
```

Viene richiesto di intervenire su alcuni indicatori (*flag*); in generale dovrebbe andare bene ciò che viene proposto in modo predefinito:

```
  1) (ON)    Library of bcc helper functions
  2) (ON)    Minimal syscalls for BIOS level
  3) (ON)    Unix error functions
  4) (ON)    Management for /etc/passwd /etc/group /etc/utmp
  5) (OFF)   Linux-i386 system call routines GCC
  6) (ON)    GNU termcap routines
  7) (ON)    Bcc 386 floating point
  8) (ON)    Linux-i386 system call routines
  9) (ON)    Example kernel include files and syscall.dat
 10) (ON)    Malloc routines
 11) (ON)    Various unix lib functions
 12) (ON)    Msdos system calls
 13) (ON)    Regular expression lib
 14) (ON)    Stdio package
 15) (ON)    String and memory manipulation
 16) (ON)    Linux-8086 system call routines
 17) (ON)    Termios functions
 18) (ON)    Unix time manipulation functions.
Select config option to flip [or quit] > quit[Invio]
```

Al termine della compilazione si passa all'installazione, cominciando dalla creazione della directory '`/opt/dev86/`'. Per fare questo occorrono i privilegi dell'utente '**`root`**':

```
$ su[Invio]
```

...

```
# mkdir -p /opt/dev86[Invio]
```

```
# make install[Invio]
```

Compilazione del kernel

La compilazione di un kernel ELKS procede in modo simile a quella di un kernel Linux. Anche in questo caso si segue lo stesso criterio usato per la compilazione degli strumenti di sviluppo, pertanto l'utente fa questo lavoro a partire dalla directory '`~/ELKS/`':

$ **cd** [*Invio*]

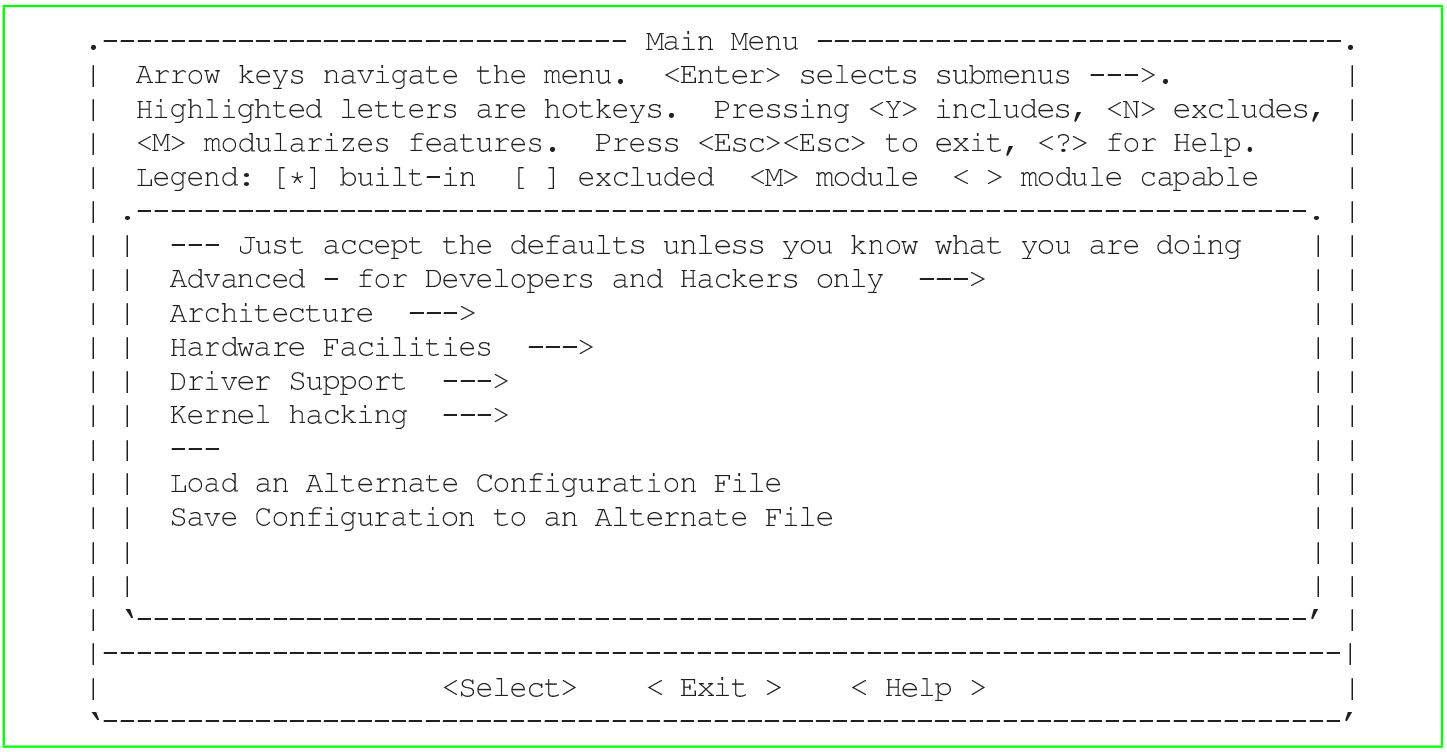
$ **cd ELKS** [*Invio*]

tar xzvf elks-0.1.1.tar.gz [*Invio*]

$ **cd elks-0.1.1** [*Invio*]

A questo punto si può procedere con la configurazione di ciò che si vuole nel kernel ELKS da compilare. Si può procedere con i comandi '**make config**' o '**make menuconfig**':

$ **make menuconfig** [*Invio*]



Terminata la fase di configurazione, si passa alla compilazione:

$ **make dep** [*Invio*]

$ **make clean** [*Invio*]

$ **make Image** [*Invio*]

Al termine della compilazione, se non ci sono problemi, si ottiene il file ‘`arch/i86/boot/Image`’.

Immagini di dischetti già pronti

Inizialmente si può cominciare tentando di avviare ELKS a partire da file-immagine già pronti. Per trasferire i file-immagine nei dischetti ci si comporta nel modo solito, esattamente come si fa per le immagini di dischetti di GNU/Linux, con la differenza che ci possono anche essere dischetti che incorporano sia il kernel, sia il file system.

Inizialmente, per fare esperimenti, è meglio usare il metodo tradizionale del kernel in un dischetto che si avvia e il file system principale in un altro, in modo da poter cambiare facilmente sia il kernel, sia il contenuto del file system.

Avvio di ELKS all’interno di DOSEMU

ELKS può essere avviato all’interno di DOSEMU. Ciò richiede un minimo di esperienza con DOSEMU, ma è utile quando si fanno delle modifiche al kernel, per fasi successive, perché l’avvio del sistema operativo è più rapido. A ogni modo, bisogna considerare che l’ambiente offerto da DOSEMU è diverso da quello reale; per esempio, se si sta modificando la mappa della tastiera, DOSEMU non va bene.

Spegnimento

Per concludere il funzionamento di un sistema ELKS può darsi che manchi sia il programma ‘**`halt`**’, sia ‘**`shutdown`**’. In questa situazione, un modo possibile di agire è il seguente:

```
# sync [Invio]

# umount / [Invio]
```

Dopo questa operazione, il file system principale dovrebbe essere ancora accessibile in sola lettura.

Riferimenti

«

- *ELKS: the embeddable Linux kernel system*
  http://elks.sourceforge.net/
- *Source Forge: ELKS*
  http://sourceforge.net/projects/elks/
  Il sorgente più recente del kernel e degli altri componenti può essere prelevato con CVS:

```
$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/elks
co elks [Invio]

$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/elks
co elksnet [Invio]

$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/elks
co elkscmd [Invio]
```

- Daniele Cono D'Elia, *ELKS Embeddable Linux Kernel Subset*
  http://www.valug.it/listing/documenti/ELKS/elks.pdf

[1] **ELKS** GNU GPL

ELKS: realizzazione personale

ELKS e i programmi di servizio del sistema sono distribuiti sia in forma sorgente, sia in dischetti già pronti per l'uso e l'installazione. Tuttavia, questi dischetti non funzionano bene con tutti i tipi elaboratore, per quanto questi possano essere vecchi, o anche antichi, pertanto conviene prepararsi alla realizzazione in proprio dei dischetti, in modo da poter raggirare eventuali ostacoli imprevisti.

In generale, per tutti i lavori necessari a preparare il proprio sistema ELKS, occorre utilizzare un elaboratore funzionante con un sistema GNU/Linux.

File system e dischetti

Salvo l'uso di estensioni particolari, ELKS è in grado di accedere soltanto a file system Minix con i nomi della lunghezza massima di 14 byte.

Quando si usa un sistema GNU/Linux per inizializzare i dischetti o i file-immagine dei dischetti da usare con ELKS, bisogna tenere presente che il programma '**`mkfs.minix`**' richiede l'uso di un'opzione appropriata, altrimenti genera un file system Minix incompatibile. Nell'esempio seguente si inizializza il dischetto corrispondente al file di dispositivo '`/dev/fd0`':

```
# mkfs.minix -n 14 /dev/fd0 [Invio]
```

File di dispositivo

«

I file di dispositivo usati da ELKS sono simili, ma non uguali a quelli di un sistema GNU/Linux comune. La situazione più importante da osservare riguarda i file di dispositivo per l'accesso ai dischetti e ai dischi fissi. Una problema abbastanza comune riguarda l'uso di '**`rdev`**' per modificare un kernel già compilato in modo che avvii un disco diverso da quello previsto in fase di compilazione: '**`rdev`**' può essere usato anche da un sistema GNU/Linux, ma i numeri primario e secondario dei file di dispositivo non corrispondono.

Tabella u178.1. File di dispositivo principali di un sistema ELKS.

Nome	Tipo	Numero primario	Numero secondario	Sigla in esadecimale
'`/dev/mem`'	caratteri	1	1	0101_{16}
'`/dev/kmem`'	caratteri	1	2	0102_{16}
'`/dev/null`'	caratteri	1	3	0103_{16}
'`/dev/zero`'	caratteri	1	5	0105_{16}
'`/dev/full`'	caratteri	1	7	0107_{16}

Nome	Tipo	Numero primario	Numero secondario	Sigla in esadecimale
‘/dev/bda’	blocchi	3	0	0300_{16}
‘/dev/bda1’	blocchi	3	1	0301_{16}
‘/dev/bda2’	blocchi	3	2	0302_{16}
‘/dev/bda3’	blocchi	3	3	0303_{16}
‘/dev/bda4’	blocchi	3	4	0304_{16}
‘/dev/bdb’	blocchi	3	64	0340_{16}
‘/dev/bdb1’	blocchi	3	65	0341_{16}
‘/dev/bdb2’	blocchi	3	66	0342_{16}
‘/dev/bdb3’	blocchi	3	67	0343_{16}
‘/dev/bdb4’	blocchi	3	68	0344_{16}
‘/dev/fd0’	blocchi	3	128	0380_{16}
‘/dev/fd1’	blocchi	3	192	$03C0_{16}$
‘/dev/tty1’	caratteri	4	0	0400_{16}
‘/dev/tty2’	caratteri	4	1	0401_{16}
‘/dev/tty3’	caratteri	4	2	0402_{16}
‘/dev/ttyS0’	caratteri	4	64	0440_{16}
‘/dev/ttyS1’	caratteri	4	65	0441_{16}

Nome	Tipo	Numero primario	Numero secondario	Sigla in esadecimale
‘/dev/ttyS2’	caratteri	4	66	0442_{16}
‘/dev/lp0’	caratteri	6	0	0600_{16}
‘/dev/lp1’	caratteri	6	1	0601_{16}
‘/dev/lp2’	caratteri	6	2	0602_{16}
‘/dev/tcpdev’	caratteri	8	0	0800_{16}

I file di dispositivo ‘`/dev/bd*`’ si riferiscono all’accesso al disco fisso, attraverso il BIOS. Eventualmente, sono previsti file di dispositivo per l’accesso diretto al disco fisso, con la denominazione consueta ‘`/dev/hd*`’, anche se inizialmente il kernel non è in grado di farlo, ma ugualmente il numero primario differisce da quello usato con i sistemi GNU/Linux.

Per la ricostruzione dei file di dispositivo è disponibile uno script ‘**`MAKEDEV`**’ già pronto, con descrizioni molto chiare, ma forse conviene riepilogare i comandi per i file di dispositivo elencati nella tabella:

```
#!/bin/sh

mknod mem       c 1 1
mknod kmem      c 1 2
mknod null      c 1 3

mknod zero      c 1 5
mknod full      c 1 7

mknod bda       b 3 0
mknod bda1      b 3 1
mknod bda2      b 3 2
mknod bda3      b 3 3
mknod bda4      b 3 4

mknod bdb       b 3 64
mknod bdb1      b 3 65
mknod bdb2      b 3 66
mknod bdb3      b 3 67
mknod bdb4      b 3 68

mknod fd0       b 3 128
mknod fd1       b 3 192

mknod tty1      c 4 0
mknod tty2      c 4 1
mknod tty3      c 4 2

mknod ttyS0     c 4 64
mknod ttyS1     c 4 65
mknod ttyS2     c 4 66

mknod lp0       c 6 0
mknod lp1       c 6 1
mknod lp2       c 6 2

mknod tcpdev    c 8 0
```

«

Un problema molto importante da risolvere quando si vuole mettere insieme un sistema operativo è il meccanismo di avvio. Il kernel ELKS, così come avviene nel caso di Linux, si avvia da solo, copiandolo in un dischetto come se fosse l'immagine dello stesso, offrendo così una base di partenza sicura.

Il fatto di avviare il kernel ELKS direttamente implica l'impossibilità di passargli dei parametri, pertanto bisogna utilizzare '**rdev**' o un programma simile per definire da quale file di dispositivo deve essere innestato il file system principale:

rdev ***file_kernel*** ***file_di_dispositivo*** [*Invio*]

Il programma '**rdev**' è disponibile sia in un sistema GNU/Linux comune, sia in un sistema ELKS; quello che cambia, purtroppo, sono i numeri primario e secondario dei file di dispositivo. Per esempio, se si sta operando attraverso un sistema GNU/Linux e si vuole impostare il file '`elks`', che si intende essere un kernel ELKS, occorre prima preparare il file di dispositivo appropriato, anche se questo lo si può fare in una directory temporanea:

mknod /tmp/fd0 b 3 128 [*Invio*]

rdev elks /tmp/fd0 [*Invio*]

rdev elks [*Invio*]

```
Root device 0x0380
```

Come si può intuire, il numero 0380_{16} rappresenta un file di dispositivo con numero primario pari a 03_{16}, ovvero 3_{10}, e numero secondario pari a 80_{16}, ovvero 128_{10}. Se invece si fa riferimento a un numero pri-

mario e secondario che in qualche modo sono previsti nel sistema, si può ottenere un'informazione un po' confusa:

```
# mknod /tmp/bda1 b 3 1 [Invio]

# rdev elks /tmp/bda1 [Invio]

# rdev elks [Invio]

  Root device /dev/hda1
```

Evidentemente occorre fare attenzione per non confondersi.

Bootblocks

Il pacchetto Bootblocks [1] consente di avviare un sistema ELKS contenuto in un dischetto o in una partizione del disco fisso, con il kernel inserito nello stesso file system. Il pacchetto viene distribuito assieme agli strumenti di sviluppo Dev86, ma non viene compilato automaticamente assieme a quelli. Si trova precisamente nella sottodirectory '`bootblocks/`' dei sorgenti di Dev86. Si compila in modo molto semplice con il comando '**`make`**':

```
# cd sorgenti_dev86/bootblocks [Invio]

# make [Invio]
```

Dalla compilazione si ottengono diversi file; per quanto riguarda il problema dell'avvio di ELKS sono utili:

File	Descrizione
'**`makeboot`**'	programma per l'installazione del settore di avvio, da usare attraverso un sistema GNU/Linux comune;
'**`makeboot.com`**'	programma analogo a '**`makeboot`**', da usare con un sistema Dos;

File	Descrizione
‘**minix_elks.bin**’	programma di appoggio per l’avvio di un kernel ELKS.

Per fare in modo che un dischetto, con file system Minix, contenente un sistema ELKS, completo di kernel e di programmi di servizio, occorre predisporre alcune cose come se le aspetta il sistema di avvio. Per la precisione si deve preparare la directory ‘`/boot/`’ contenente il kernel e il file ‘`minix_elks.bin`’, ma entrambi devono avere un nome appropriato; pertanto, il kernel deve essere ‘`/boot/linux`’ e il programma di appoggio (in origine ‘`minix_elks.bin`’) deve chiamarsi ‘`/boot/boot`’.

Supponendo di utilizzare un sistema GNU/Linux, supponendo di avere preparato il dischetto Minix (con i nomi al massimo di 14 byte) contenente tutto quello che serve, soprattutto con la directory ‘`/boot/`’ come spiegato, se questo dischetto risulta inserito nell’unità corrispondente al file di dispositivo ‘`/dev/fd0`’, **senza essere stato innestato**, si può eseguire il comando seguente, tenendo conto che il programma ‘**makeboot**’ si presume collocato in una directory prevista tra i vari percorsi della variabile di ambiente ‘**PATH**’:

```
# makeboot minix /dev/fd0 [Invio]

  Wrote sector 0
  Wrote sector 1
```

Se il programma si accorge che il settore di avvio del dischetto contiene già qualcosa, si rifiuta di procedere, a meno di usare l’opzione ‘**-f**’:

```
# makeboot -f minix /dev/fd0 [Invio]
```

```
Boot block isn't empty, zap it first
Wrote sector 0
Wrote sector 1
```

Riquadro u178.8. Come cambiare il nome del file del kernel.

Il sistema di avvio che si ottiene con l'uso di '**makeboot**' e del file '`minix_elks.bin`' prevede che il kernel sia precisamente il file '`/boot/linux`', mentre forse sarebbe più appropriato '`/boot/elks`'. Per ovviare a questa piccola incoerenza, basta intervenire nel sorgente, precisamente nel file '***sorgenti_dev86***`/bootblocks/minix_elks.c`'. A un certo punto, questo file contiene le righe seguenti:

```
elks_name:
  .asciz        "linux"
  .byte         0,0,0,0,0,0,0,0,0
```

Per fare in modo di avviare il kernel contenuto nel file '`/boot/elks`', si deve modificare e ricompilare:

```
elks_name:
  .asciz        "elks"
  .byte         0,0,0,0,0,0,0,0,0
```

Per installare ELKS in una partizione del disco fisso, le cose si complicano, perché manca la possibilità di usare il programma '**makeboot**' con il sistema ELKS stesso e purtroppo, il programma '`makeboot.com`', funzionante con un sistema Dos, si rifiuta di intervenire in partizioni che non siano Dos. Pertanto, occorre preparare una porzione di codice da incollare poi con il programma '**dd**' o simile:

```
# touch avvio[Invio]

# makeboot minix avvio[Invio]
```

```
Cannot read sector 0, clearing
Cannot read sector 1, clearing
Wrote sector 0
Wrote sector 1
```

In pratica, si crea un file vuoto, in questo caso il file ‘`avvio`’, quindi gli si inserisce il codice necessario, da incollare successivamente all’inizio della partizione da avviare.

Supponendo di avere inserito questo file in un dischetto contenente un sistema ELKS funzionante, con il quale si è riusciti a predisporre una partizione del disco fisso allo scopo di alloggiare il sistema stesso, basta copiare il file in questo modo:

```
# dd if=avvio of=/dev/bda1 [Invio]
```

Naturalmente, in questo caso si sta facendo riferimento alla prima partizione.

Bisogna ricordare che il file del kernel deve essere modificato con ‘**`rdev`**’, in modo da utilizzare il file system contenuto nella partizione stessa.

Perché il codice iniziale della partizione venga messo in funzione, è necessario che il primo settore del disco fisso (MBR) indichi la partizione stessa come avviabile.

Nonostante tutto, può succedere che il codice inserito all'inizio della partizione venga rifiutato dal BIOS, per mancanza della firma $55AA_{16}$. In presenza di difficoltà di questo tipo, rimane la possibilità di avviare a partire da un dischetto con il kernel ritoccato attraverso '**rdev**' in modo da utilizzare la partizione corretta.

Bootkit

Bootkit [2] è un sistema di avvio abbastanza buono, che in particolare può passare dei parametri di avvio al kernel ELKS.

Il problema di Bootkit è che non si trovano i sorgenti.

Si può trovare Bootkit incorporato nella distribuzione EDE (*ELKS distribution edition*) e si utilizza all'interno del sistema ELKS stesso.

```
bootkit opzioni file
```

Per ottenere il risultato, occorre predisporre la directory '`/boot/`' nel file system Minix che contiene il sistema ELKS da avviare. All'interno di questa directory si colloca il file '`boot.conf`', con una serie di direttive e di solito il file '`boot.txt`', il cui contenuto deve essere mostrato in fase di avvio. A titolo di esempio, il file '`boot.conf`' potrebbe contenere le righe seguenti:

```
message=boot.txt
prompt
timeout 100
image=elks
    label=fd0
    root=0x380
image=elks
    label=fd1
    root=0x381
image=elks
    label=bda1
    root=0x301
image=elks
    label=bda2
    root=0x302
image=elks
    label=bda3
    root=0x303
image=elks
    label=bda4
    root=0x304
#other=/dev/bda1
#    label=dos
```

Inizialmente, la direttiva '**`message=boot.txt`**' dichiara di usare il file '`boot.txt`' per mostrare un messaggio all'avvio, quindi, la direttiva '**`prompt`**' fa sì che venga mostrato un invito, con il quale poter scegliere tra le diverse possibilità di avvio. La direttiva '**`timeout 100`**' fa sì che la prima delle varie modalità di avvio sia scelta in mancanza di una risposta all'avvio, entro 10 s.

Le varie direttive '**`image=elks`**' delimitano l'inizio di una sezione, distinguibile in base al valore assegnato alla direttiva '**`label`**', che dichiara l'uso del kernel contenuto nel file '`elks`', che si deve trovare nella stessa directory ('`/boot/`'). Come si vede, all'interno di queste sezioni, la direttiva '**`root`**' consente di specificare il file system che il kernel deve innestare, anche se ciò deve essere fatto indicando i numeri primario e secondario del file di dispositivo, in forma

esadecimale.

Il file ‘`boot.txt`’ potrebbe contenere il messaggio seguente:

```
Please select one of the following root file systems:

fd0     (default)
fd1
bda1
bda2
bda3
bda4
```

Si completa il lavoro mettendo nella directory ‘`/boot/`’ il file del kernel, che in base agli esempi deve chiamarsi ‘`elks`’, quindi occorre creare il file ‘`boot`’, attraverso ‘`bootkit`’:

```
# bootkit -i boot /boot/boot [Invio]
```

Si osservi che questo comando va dato all’interno di un sistema ELKS già funzionante, ma una volta creato, il file può essere copiato così in altri dischetti.

Rimane il problema della creazione di un settore di avvio, che poi cerca il programma ‘`/boot/boot`’, il quale poi legge la configurazione del file ‘`/boot/boot.conf`’:

```
# bootkit -i minix -b copia_settore /dev/fd0 [Invio]
```

In questo modo si installano 1024 byte di codice all’inizio del dischetto (corrispondente al file di dispositivo ‘`/dev/fd0`’) e si salva una copia di quanto era presente prima nel file indicato come argomento dell’opzione ‘**`-b`**’. L’opzione ‘**`-b`**’ è obbligatoria in questo caso, quindi, se non si vuole salvare ciò che viene sovrascritto, si può usare ‘`/dev/null`’:

```
# bootkit -i minix -b /dev/null /dev/fd0 [Invio]
```

Teoricamente tutto questo potrebbe funzionare anche con una partizione di un disco fisso, per esempio la prima secondo il comando seguente:

```
# bootkit -i minix -b /dev/null /dev/bda1 [Invio]
```

Per funzionare, è comunque necessario che il settore iniziale del disco fisso (MBR) indichi la partizione come avviabile.

Anche Bootkit potrebbe fallire nel compito di avviare il sistema dal disco fisso, ma rimane il fatto che da dischetto è un meccanismo ottimo, che non richiede l'uso di '**rdev**' per dire al kernel quale disco o partizione utilizzare.

Installazione manuale nel disco fisso

«

L'installazione di un sistema ELKS nel disco fisso richiede di poter disporre di un dischetto funzionante e relativamente completo, possibilmente con una shell efficiente; inoltre, prima di iniziare occorre avere un quadro abbastanza chiaro di come questo dischetto è organizzato, cosa che si può fare utilizzando un sistema GNU/Linux, attraverso tutti gli strumenti a cui si è abituati.

Quando è tutto pronto, con dischetto di ELKS, o anche con un altro sistema operativo, se ciò è possibile e preferibile, occorre intervenire nella suddivisione delle partizioni del disco fisso. Considerato che l'accesso al disco avviene attraverso il BIOS, è bene che la partizione sia piccola e si trovi all'inizio o vicino all'inizio del disco stesso.

Teoricamente, dato il fatto che l'accesso avviene tramite le funzioni del BIOS, la partizione può arrivare a un massimo di 32 Mibyte, ma

in pratica, può darsi che si debba ridurre ancora di più. Comunque, la partizione può anche essere relativamente grande, ma poi, quando la si va a inizializzare, bisogna creare un file system piccolo.

La partizione deve essere di tipo *old minix*, corrispondente al codice 80_{16}.

Il sistema ELKS dovrebbe disporre del programma '**fdisk**' per modificare l'organizzazione delle partizioni:

```
# fdisk /dev/bda [Invio]
```

Il suo utilizzo è simile a quello dello stesso programma usato nei sistemi GNU/Linux (in particolare il comando [*?*] richiama la guida degli altri comandi disponibili), ma conviene armarsi di calcolatrice per fare i conti di quanti cilindri servono per la partizioni che si vogliono creare.

Ammesso che il settore di avvio del disco fisso contenga del codice corretto, è bene ricordare di rendere avviabile la partizione che si va a creare per ospitare ELKS.

Una volta sistemate le partizioni e salvate con il comando [*w*], si può abbandonare il programma '**fdisk**' ([*q*]) per passare a '**mkfs**' con il quale creare il file system Minix:

```
# mkfs /dev/bda1 5000 [Invio]
```

In questo caso si suppone di dover inizializzare la prima partizione del primo disco fisso, con un file system di 5000 Kibyte.

Si osservi che la dimensione in settori fornita da '**fdisk**' rappresenta una quantità espressa in unità da 512 byte, mentre il valore che si fornisce al programma '**mkfs**' esprime una quantità in unità da 1024 byte. In pratica, il file system va creato con un valore che non può eccedere la metà di quanto riporta '**fdisk**'.

È ormai chiaro che la partizione può essere più grande del file system Minix che si va a creare. Una volta creato, è bene provare a innestarlo, per esempio con il comando seguente ammettendo che sia disponibile la directory '`/mnt/`':

```
# mount /dev/bda1 /mnt [Invio]
```

Se si ottiene una segnalazione di errore nella quale viene affermato che il file system non è valido (non è Minix), bisogna provare a inizializzarlo con una dimensione minore.

Quando si arriva al punto di essere stati capaci di innestare il file system creato nella partizione del disco fisso, si può procedere con la copia del contenuto del dischetto, ma con prudenza. Infatti, il comando '**cp**' potrebbe essere molto poco amichevole, inoltre, fornendogli troppi argomenti (quando si usano dei modelli la shell li espande in elenchi che possono essere anche abbastanza numerosi) può bloccarsi, assieme a tutto il sistema.

Comunque, con un po' di prudenza, si possono ricreare le directory e al loro interno vi si possono copiare i file che si trovano nel dischetto (il programma '**cp**' potrebbe essere capace di copiare soltanto file normali); quindi occorre riprodurre una directory '`/dev/`' con i file di dispositivo necessari e infine si può cercare di risolvere il problema dell'avvio.

Come si può comprendere molto presto, quando ci si cimenta in un lavoro di questo tipo, occorre una strategia, che può consistere nella preparazione preventiva di qualche script che faccia buona parte di questo lavoro in modo automatico.

In queste spiegazioni ci sono molte piccole cose che sono omesse, soprattutto perché le situazioni che si presentano cambiano facilmente con una grande quantità di sfumature. È evidente che si tratta di un lavoro che può affrontare solo chi ha già una buona padronanza di un sistema GNU/Linux e non si lascia scoraggiare dai piccoli fallimenti a cui si va incontro sicuramente.

Adattamento della mappa della tastiera

Un problema che può presentarsi è quello di adattare la mappa della tastiera, cosa che richiede la ricompilazione del kernel selezionando il tipo corretto per ciò che si deve usare.

Se la mappa che serve non c'è, oppure se non funziona come ci si aspetterebbe, occorre predisporre in proprio un file con un nome che corrisponda al modello '`keys-`***xy***`.h`', dove ***xy*** sono due lettere che identificano la nazionalità. Questo file, insieme agli altri delle altre mappe, va collocato nella directory '***sorgenti_elks***`/arch/i86/drivers/char/KeyMaps/`'. Naturalmente, il file deve avere una certa forma, che si può intuire osservando quelli già esistenti.

Bisogna tenere in considerazione il fatto che il sistema è predisposto per un elaboratore con tastiera «XT», pertanto non può esistere il tasto [*AltGr*] e nemmeno si possono attuare tutte quelle combinazioni che invece sono disponibili con un sistema GNU/Linux comune.

Probabilmente si può usare con successo solo la parte alfanumerica, i tasti freccia, [*Esc*] e probabilmente i tasti funzionali. Quando si inserisce il [*Fissamaiuscole*] non è detto che la spia corrispondente si accenda; il tasto [*BlocNum*] non funziona e probabilmente i numeri sulla tastiera numerica si ottengono inserendo il [*Fissamaiuscole*].

Dovendo predisporre o modificare una mappa che comprende anche lettere accentate e altri simboli speciali che sono al di fuori del codice ASCII tradizionale, occorre considerare che la codifica usata è quella originale degli elaboratori 8086, ovvero quella che era nota come CP 437 (si veda la sezione 47.7.7 per trovare una copia completa della codifica CP 437).

Viene proposto un esempio di mappa per la tastiera italiana, che deve corrispondere al file '***sorgenti_elks***`/arch/i86/drivers/char/KeyMaps/keys-it.h`'. Si osservi in particolare l'intestazione che serve a uno script per fare in modo che il file venga preso in considerazione durante la configurazione del kernel.

Listato u178.14. Esempio di un file per la configurazione della tastiera secondo la disposizione italiana.

```
/* Keymap:IT:Italiano:Italy     */

#ifndef __KEYMAP_IT__
#define __KEYMAP_IT__

#if defined(CONFIG_KEYMAP_IT)

/*
    \ 1 2 3 4 5 6 7 8 9 0 ' ì
       q w e r t y u i o p è +
        a s d f g h j k l ò à ù
         z x c v b n m , . -
*/

static unsigned char xtkb_scan[] = {
```

```
  0,
  033,
  '1',  '2',  '3',  '4',  '5',  '6',  '7',  '8',  '9',  '0', '\'', 0215, '\b',
 '\t',  'q',  'w',  'e',  'r',  't',  'y',  'u',  'i',  'o',  'p', 0212,  '+',
  015,
 0202,  'a',  's',  'd',  'f',  'g',  'h',  'j',  'k',  'l', 0225, 0205,
 '\\',
 0200,
 0227,
  'z',  'x',  'c',  'v',  'b',  'n',  'm',  ',',  '.',  '-',
 0201,
  '*', 0203,  ' ', 0204,
 0241, 0242, 0243, 0244, 0245, 0246, 0247, 0250, 0251, 0252,
 0205,  '?',
  '7',  '8',  '9',
  '-',
  '4',  '5',  '6',
  '+',
  '1',  '2',  '3',
  '0',  '.'
};

/*
    | ! " £ $ % & / ( ) = ? ^
       Q W E R T Y U I O P é *
        A S D F G H J K L ç ° §
         Z X C V B N M ; : _
*/

static unsigned char xtkb_scan_shifted[] = {
    0,
  033,
  '!',  '"', 0234,  '$',  '%',  '&',  '/',  '(',  ')',  '=',  '?',  '^', '\b',
 '\t',  'Q',  'W',  'E',  'R',  'T',  'Y',  'U',  'I',  'O',  'P', 0202,  '*',
 '\r',
 0202,  'A',  'S',  'D',  'F',  'G',  'H',  'J',  'K',  'L', 0207, 0370,
  '|',
 0200,
  025,
  'Z',  'X',  'C',  'V',  'B',  'N',  'M',  ';',  ':',  '_',
 0201,
  '*', 0203,  ' ', 0204,
 0221, 0222, 0223, 0224, 0225, 0226, 0227, 0230, 0231, 0232,
 0204, 0213,
```

```
  '7',  '8',  '9',
  '-',
  '4',  '5',  '6',
  '+',
  '1',  '2',  '3',
  '0',  '.'
};

/*
    \ < > 3 4 5 6 { [ ] } ` ~
       q w e r t y u i o p [ ]
        a s d f g h j k l @ # ù
         « » c v b n m , . -
*/

static unsigned char xtkb_scan_ctrl_alt[] = {
    0,
  033,
  '<',  '>',  '3',  '4',  '5',  '6',  '{',  '[',  ']',  '}',  '`',  '~', '\b',
 '\t',  'q',  'w',  'e',  'r',  't',  'y',  'u',  'i',  'o',  'p',  '[',  ']',
 '\r',
 0202,  'a',  's',  'd',  'f',  'g',  'h',  'j',  'k',  'l',  '@',  '#',
 '\\',
 0200,
 0227,
 0256, 0257,  'c',  'v',  'b',  'n',  'm',  ',',  '.',  '-',
 0201,
  '*', 0203,  ' ', 0204,
 0241, 0242, 0243, 0244, 0245, 0246, 0247, 0250, 0251, 0252,
 0205,  '?',
  '7',  '8', '9',
  '-',
  '4',  '5', '6',
  '+',
  '1',  '2',  '3',
  '0',  '.'
};

/*
    \ 1 2 3 4 5 6 7 8 9 0 ' ì
       Q W E R T Y U I O P è +
        A S D F G H J K L ò à ù
         Z X C V B N M , . -
*/
```

```
static unsigned char xtkb_scan_caps[] = {
    0,
  033,
  '1',  '2',  '3',  '4',  '5',  '6',  '7',  '8',  '9',  '0', '\'', 0215, '\b',
 '\t',  'Q',  'W',  'E',  'R',  'T',  'Y',  'U',  'I',  'O',  'P', 0212,  '+',
 '\r',
 0202,  'A',  'S',  'D',  'F',  'G',  'H',  'J',  'K',  'L', 0225, 0205,
 '\\',
 0200,
 0227,
  'Z',  'X',  'C',  'V',  'B',  'N',  'M',  ',',  '.',  '-',
 0201,
  '*', 0203,  ' ', 0204,
 0221, 0222, 0223, 0224, 0225, 0226, 0227, 0230, 0231, 0232,
 0204, 0213,
  '7',  '8',  '9',
  '-',
  '4',  '5',  '6',
  '+',
  '1',  '2',  '3',
  '0',  '.'
};

#endif

#endif
```

Come si può intuire dai nomi degli array, si distingue tra quattro situazioni: la disposizione normale, la disposizione che entra in gioco quando si preme il tasto delle maiuscole, quando si inserisce il [*Fissamaiuscole*], infine quando si preme la combinazione [*Ctrl Alt* ***x***]. In pratica, per ottenere simboli come la chiocciola e il cancelletto, occorre usare la combinazione [*Ctrl Alt ò*] e [*Ctrl Alt à*]. Per qualche motivo, non è stato possibile collocare correttamente i simboli [<] e [>], che si ottengono invece con [*Ctrl Alt 1*] e [*Ctrl Alt 2*] rispettivamente. Inoltre, sono disponibili anche le parentesi graffe, l'accento rovesciato, la tilde e le virgolette

basse uncinate, come si fa di solito in un sistema GNU/Linux, ma sempre usando una combinazione del tipo [*Ctrl Alt* ***x***].

Si osservi che quanto mostrato vale come esempio e in caso di dubbio conviene verificare il funzionamento del kernel con la mappa americana standard.

Tabella u178.15. Estratto dalla codifica CP 437, per la definizione della mappa della tastiera.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
202_8	130_{10}	82_{16}	U+00E9	é
205_8	133_{10}	85_{16}	U+00E3	ã
207_8	135_{10}	87_{16}	U+00E7	ç
212_8	138_{10}	$8A_{16}$	U+00E8	è
215_8	141_{10}	$8D_{16}$	U+00EC	ì
225_8	149_{10}	95_{16}	U+00F2	ò
227_8	151_{10}	97_{16}	U+00F9	ù
234_8	156_{10}	$9C_{16}$	U+00A3	£
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»

[1] **Bootblocks** GNU GPL

[2] **Bootkit** GNU GPL