os32, come già os16, ha due modalità di funzionamento: si può interagire direttamente con il kernel, oppure si può avviare il processo '**`init`**' e procedere con l'organizzazione consueta di un sistema Unix tradizionale. La modalità di colloquio diretto con il kernel è servita per consentire lo sviluppo di os32 e potrebbe essere utile per motivi di studio. Va osservato che durante l'interazione diretta con il kernel si dispone di una sola console, mentre quando si avvia '**`init`**' si possono avere delle console virtuali in base alla configurazione.

85.1 Disco fisso in un file-immagine

os32 è distribuito in modo da funzionare con l'ausilio di Qemu o Bochs, installato in un disco fisso virtuale, rappresentato da un file-immagine. Tale file-immagine, denominato '`disk.hda`' è suddiviso in almeno due partizioni: la prima è necessaria per l'avvio e il suo formato dipende dal sistema di avvio installato (dovrebbe essere SYSLINUX, pertanto il file system dovrebbe essere di tipo Dos-FAT); la seconda è di tipo Minix-1 e ospita il sistema os32.

La creazione e l'accesso alle partizioni contenute nel file '`disk.hda`' è un po' complicato da un sistema GNU/Linux, perché occorre estrapolarle di volta in volta in file indipendenti, per aggregarle poi nuovamente in un file-immagine unico. Per queste operazioni si usano degli script già predisposti:

Script	Descrizione
`file_image_functions`	Contiene delle funzioni utilizzate dagli altri script [94.1.5].
`fdisk` ***file***	Consente di utilizzare il programma '**`fdisk`**' sul file-immagine indicato, con l'ausilio delle funzioni contenute nel file '`file_image_functions`' [94.1.4].
`format` ***file n dos|minix***	Consente di inizializzare la partizione ***n***-esima del file-immagine indicato come primo argomento, con un file system di tipo Dos-FAT o di tipo Minix-1 [94.1.6].
`syslinux` ***file n***	Installa SYSLINUX nella partizione ***n*** del file-immagine indicato [94.1.10].
`mount` `umount`	Innesta o stacca le partizioni contenute nel file-immagine '`disk.hda`' utilizzando le directory '`/mnt/disk.hda.`***n***' che però devono già essere disponibili.

Quando si compila il sistema con lo script ***makeit.mer*** o con ***makeit.sep***, al termine del processo questo tenta di copiare il kernel '`kimage`' nella directory '`/mnt/disk.hda1/`' e poi il sistema nella directory '`/mnt/disk.hda2/`'. Queste due directory dovrebbero innestare rispettivamente le prime due partizioni del file-immagine '`disk.hda`'; tuttavia, occorre disporre dei privilegi necessari, come dire che occorre fare questo in qualità di utente '**`root`**'.

La copia del sistema nella seconda partizione non provvede però a produrre i file di dispositivo necessari nella directory '`dev/`'. Eventualmente, all'interno di tale directory si ottiene lo script '`.makedev`', da avviare attraverso il sistema GNU/Linux.

Va comunque ricordato che prima di avviare os32 è necessario distaccare gli innesti delle partizioni del file-immagine, perché os32 funziona utilizzando il file-immagine nel suo complesso, mentre l'innesto delle partizioni comporta l'estrapolazione delle partizioni e la successiva riaggregazione; inoltre, os32 non ha un alcun sistema di arresto del sistema, per cui è facile rendere incoerente il file system, quindi è bene fare spesso delle copie del file-immagine ed essere comunque pronti a ricostruirlo.

85.2 Configurazione dell'avvio e opzioni del kernel

Si utilizza SYSLINUX nella prima partizione per l'avvio di os32. Premesso che il kernel di os32 è contenuto nel file '`kimage`' e si colloca assieme ai file di SYSLINUX, nella configurazione contenuta nel file '`syslinux.cfg`' (relativo a SYSLINUX stesso) si leggono le istruzioni seguenti:

```
TIMEOUT 10
PROMPT 1
DEFAULT os32
#
LABEL os32
 KERNEL mboot.c32
 APPEND kimage net1=1,172.21.11.16,16 ↩
↪               route0=0.0.0.0,0,172.21.11.18,1 ↩
↪               route1=172.21.254.254,32,172.21.11.18,1
```

Con le opzioni passate al kernel di os32 si ottiene la configurazione dell'interfaccia di rete '**`net1`**' ('**`net0`**' corrisponde all'interfaccia *loopback*), con l'indirizzo 172.21.11.16, maschera di rete 255.255.0.0 e gli instradamenti necessari.

Opzione del kernel	Descrizione
`net`***i*****`=`*****i*****`,`** ***ipv4*****`,`** ***m***	Dichiara l'interfaccia di rete '**`net`*****i***', corrispondente all'indice ***i*** nella tabella delle interfacce, con l'indirizzo IPv4 specificato da ***ipv4*** e la maschera di rete lunga ***m*** bit.
`route`***n*****`=`*****dest*****`,`** ***m*****`,`** ***router*****`,`** ***i***	Dichiara la destinazione rappresentata dall'indirizzo ***dest*** e la maschera di rete composta da ***m*** bit, raggiungibile attraverso l'indirizzo ***router*** corrispondente all'interfaccia ***i*** nella tabella delle interfacce.

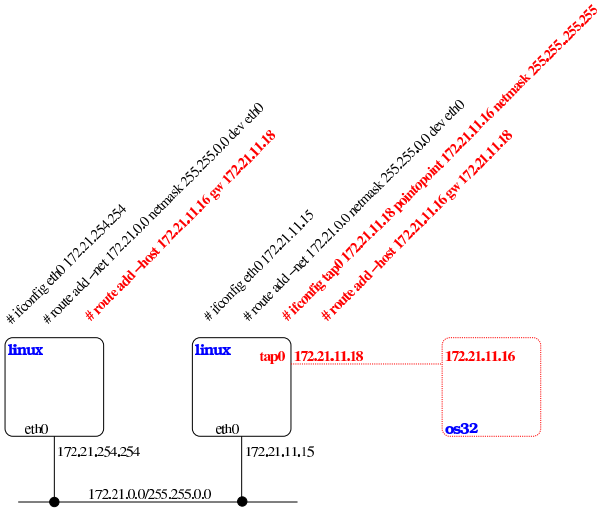
Volendo tradurre le opzioni che appaiono nell'esempio in comandi di un sistema GNU/Linux, sarebbe come se fosse stato scritto:

```
# ifconfig eth1 172.21.11.16 netmask 255.255.0.0 [Invio]

# route add -net 0.0.0.0 netmask 0.0.0.0 gw 172.21.11.18 ↩
↪       dev eth1 [Invio]

# route add -host 172.21.254.254 gw 172.21.11.18 ↩
↪       dev eth1 [Invio]
```

85.3 Configurazione per l'uso della rete

Per poter utilizzare le funzionalità di rete di os32, attraverso l'emulatore, è necessario predisporre nel sistema ospitante un'interfaccia di rete virtuale, connessa con quella di os32 attraverso una connessione punto-punto (sempre virtuale). Gli script che accompagnano os32 prevedono l'uso con Qemu o Bochs, in un sistema GNU/Linux, creando una connessione come quella schematizzata nella figura successiva.

Figura 85.4. Ipotesi di collegamento di os32 con il sistema ospitante e con il resto della rete locale: l'elaboratore che esegue os32 all'interno di un emulatore è quello corrispondente all'indirizzo 172.21.11.15. Le istruzioni che riguardano la configurazione necessaria a connettersi con os32 appaiono in rosso.



Il sistema os32, a sua volta, si configura esclusivamente attraverso le opzioni di avvio, come descritto nella sezione precedente.

85.4 Avvio di os32

L'avvio di os32 è un po' complicato, a causa della necessità di predisporre la rete nel modo appropriato, nel sistema ospitante, ma in pratica si usa lo script '`qemu`' o lo script '`bochs`', i quali si avvalgono a loro volta di '`tap0`', per creare l'interfaccia di rete virtuale '**`tap0`**' nel sistema ospitante. Tuttavia, l'avvio deve avvenire con i privilegi dell'utente '**`root`**' per poter predisporre le funzionalità di rete; pertanto va usato un programma come '**`gksu`**' per ottenere tali privilegi:

```
$ gksu ./qemu [Invio]
```

Oppure:

```
$ gksu ./bochs [Invio]
```

Negli esempi gli script vengono avviati indicando il percorso, per evitare che vengano confusi con i nomi dei file eseguibili di Qemu e di Bochs, i quali si trovano presumibilmente nella directory '`/usr/bin/`'.

Script	Descrizione
`tap0`	Predispone l'interfaccia '**`tap0`**' con l'instradamento necessario a raggiungere os32; questo script viene utilizzato da Bochs o da Qemu [94.1.11].
`qemu`	Avvia Qemu con le opzioni appropriate per eseguire os32 nel file-immagine '`disk.hda`', configurando la rete in modo appropriato. Tra le opzioni dell'avvio di Qemu si trova la richiesta di utilizzare lo script '`tap0`' per la configurazione della rete nel sistema ospitante [94.1.9].
`bochs`	Avvia Bochs con le opzioni appropriate per eseguire os32 nel file-immagine '`disk.hda`', configurando la rete in modo appropriato. Tra le opzioni dell'avvio di Bochs si trova la richiesta di utilizzare lo script '`tap0`' per la configurazione della rete nel sistema ospitante [94.1.2].

85.5 Interazione con il kernel

L'avvio di os32 passa, in ogni caso, per una prima fase di colloquio con il kernel. Si ottiene un menù e si possono premere semplicemente dei tasti, seguiti però da [*Invio*], secondo l'elenco previsto, per ottenere delle azioni molto semplici. In questa fase il disco da cui risulta avviato il kernel è già innestato ed è prevista la possibilità di avviare tre programmi: '`/bin/aaa`', '`/bin/bbb`' e '`/bin/ccc`'. In tal modo, si ha la possibilità di avviare qualcosa, a titolo diagnostico, prima dello stesso '**`init`**' ('`/bin/init`').

Figura 85.6. Aspetto di os32 in funzione, con il menù in evidenza.

```
os32 build 20AAMMGGHHmm ram 130048 Kibyte
[ata_init] ATA drive 0 size 8064 Kib
[ata_drq] ERROR: drive 2 error
[dm_init] ATA drive=0 total sectors=16128
[dm_init] partition type=0c start sector=63 total sectors=2961
[dm_init] partition type=81 start sector=3024 total sectors=13104
.-----------------------------------------------------------------.
| h    show this menu                                .--------------.|
| t    show internal timer values                    | all commands ||
| f    fork the kernel                               | followed by  ||
| m    memory map (HEX)                              | [Enter]      ||
| g|G  show GDT table first 21+21 items              '--------------'|
| i|I  show IDT table first 21+21 items                              |
| p    process status list                                         |
| s    super block list                                            |
| n    list of active inodes                                       |
| 1..9 kill process  1 to 9                                        |
| A..F kill process 10 to 15                                       |
| a..c run programs '/bin/aaa' to '/bin/ccc' in parallel           |
| x    exit interaction with kernel and start '/bin/init'          |
| q    quit kernel                                                 |
'-----------------------------------------------------------------'
```

Le funzioni principali disponibili in questa modalità diagnostica sono riassunte nella tabella successiva. È importante tenere presen-

te che, a differenza di os16, anche in questa fase i comandi sono conclusi con la pressione del tasto [*Invio*].

Comando	Risultato
`h`	Mostra il menù di funzioni disponibili.
`t`	Mostra i valori gestiti internamente dell'orologio del kernel.
`f`	Esegue una biforcazione del kernel, nella quale, il processo figlio si limita a mostrare ripetutamente il proprio numero di processo.
`m`	Mostra la mappa della memoria, elencando le aree continue utilizzate.
`g` `G`	Mostra le prime voci della tabella GDT, in binario.
`i` `I`	Mostra le prime voci della tabella IDT, in binario.
`p`	Mostra la situazione dei processi e altre informazioni.
`s`	Mostra delle informazioni sul super blocco.
`n`	Mostra l'elenco degli inode attivi.
`1`	Invia il segnale '**`SIGKILL`**' al processo numero uno.
`2`|`3`|`4`|`5`|`6`|`7`|`8`|`9`|`A`|`B`|`C`|`D`|`E`|`F`	Invia il segnale '**`SIGTERM`**' al processo con il numero corrispondente, da 2 a 15.
`a`|`b`|`c`	Avvia il programma '`/bin/aaa`', '`/bin/bbb`' o '`/bin/ccc`'.
`x`	Termina il ciclo e successivamente si passa all'avvio di '`/bin/init`'.
`q`	Ferma il sistema.

Figura 85.8. Aspetto di os32 in funzione mentre visualizza la tabella dei processi avviati e la mappa della memoria.

```
c
abaabaaba
p
pp  p pg                              T * 0x1000 D * 0x1000 stack
id id rp  tty  uid euid suid usage s addr  size addr  size pointer  name
 0  0  0 0000    0    0    0 00.03 R 00000 028e 00000 0000 028eb2c os32 kernel
 0  1  0 0000    0    0    0 00.09 r 0051e 000e 0052c 002d 002cf88 /bin/ccc
 1  2  0 0000   10   10   10 00.00 s 002bc 000e 002ca 002d 002cf34 /bin/aaa
 1  3  0 0000   11   11   11 00.00 s 002f7 000e 00305 002d 002cf34 /bin/bbb
ab
m
Hex mem map, blocks of 1000: 0-28f 2bc-332 51e-559
aabaab_
```

Con il comando '**`x`**' il ciclo termina e il kernel avvia '`/bin/init`', ma prima di farlo occorre che non ci siano altri processi in funzione, perché '**`init`**' deve assumere il ruolo di processo 1, ovvero il primo dopo il kernel.

Figura 85.9. Aspetto di os32 in funzione con il menù in evidenza, dopo aver dato il comando '**`x`**' per avviare '**`init`**'.

```
os32 build 20AAMMGGHHmm ram 130048 Kibyte
[ata_init] ATA drive 0 size 16 Kib
[ata_init] ATA drive 1 size 16 Kib
.-----------------------------------------------------------------.
| h    show this menu                              .--------------.|
| t    show internal timer values                  | all commands ||
| f    fork the kernel                             | followed by  ||
| m    memory map (HEX)                            | [Enter]      ||
| p    process status list                         '--------------'|
| s    super block list                                            |
| n    list of active inodes                                       |
| 1..9 kill process  1 to 9                                        |
| A..F kill process 10 to 15                                       |
| a..c run programs '/bin/aaa' to '/bin/ccc' in parallel           |
| x    exit interaction with kernel and start '/bin/init'          |
| q    quit kernel                                                 |
'-----------------------------------------------------------------'
x
init
os32: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change
console.
This is terminal /dev/console0
Log in as "root" or "user" with password "ciao" :-)
login:
```

85.6 Avvio e conclusione del sistema «normale»

Se non si intende operare direttamente con il kernel, come descritto nella sezione precedente, con il comando '**`x`**' si avvia '**`init`**'.

Il programma '**`init`**' legge il file '`/etc/inittab`' e sulla base del suo contenuto, avvia uno o più processi '**`getty`**', per la gestione dei vari terminali disponibili (si tratta comunque soltanto di console virtuali).

Il programma '**`getty`**' apre il terminale che gli viene indicato come opzione della chiamata (da '**`init`**' che lo determina in base al contenuto di '`/etc/inittab`'), facendo in modo che sia associato al descrittore zero (standard input). Quindi, dopo aver visualizzato il contenuto del file '`/etc/issue`', mostra un proprio messaggio e avvia il programma '**`login`**'.

Il programma '**`login`**' prende il posto di '**`getty`**' che così scompare dall'elenco dei processi. '**`login`**' procede chiedendo all'utente di identificarsi, utilizzando il file '`/etc/passwd`' per verificare le credenziali di accesso. Se l'identificazione ha successo, viene avviata la shell definita nel file '`/etc/passwd`' per l'utente, in modo da sostituirsi al programma '**`login`**', il quale scompare a sua volta dall'elenco dei processi.

Attraverso la shell è possibile interagire con il sistema operativo, secondo la modalità «normale», nei limiti delle possibilità di os32. Quando la shell termina di funzionare, '**`init`**' riavvia '**`getty`**'.

Per cambiare console virtuale si possono usare le combinazioni [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*], ma bisogna considerare che dipende dalla configurazione del file '`/etc/inittab`' se effettivamente vengono attivate tutte queste console.

Per concludere l'attività del sistema, basta concludere il funzionamento delle varie sessioni di lavoro (la shell finisce di funzionare con il comando interno '**`exit`**') e non c'è bisogno di altro; pertanto, non è previsto l'uso di comandi come '**`halt`**' o '**`shutdown`**' e, d'altro canto, le operazioni di scrittura nel file system sono sincrone, in modo tale da non richiedere accorgimenti particolari per la chiusura delle attività.