Linux From Scratch

Versione 6.2

Gerard Beekmans

Linux From Scratch: Versione 6.2

di Gerard Beekmans

Estratto

Traduzione a cura del gruppo ILDP-LFS per l'Italian Linux Documentation Project (*http://ildp.pluto.it/*). Per maggiori informazioni, si visiti *http://ildp.pluto.it/lfs/*.

Sommario

Prefazione

1. Premessa

Le mie avventure in Linux iniziarono nel 1998, quando scaricai e installai la mia prima distribuzione. Dopo aver lavorato con essa per un po' scoprii problemi che sicuramente mi sarebbe piaciuto vedere migliorati. Ad esempio non mi piaceva la sistemazione degli script di avvio o come i programmi erano configurati di default. Provai un certo numero di distribuzioni alternative per sistemare questi problemi, ma ciascuna aveva i suoi pro e contro. Infine, realizzai che se volevo piena soddisfazione dal mio sistema Linux avrei dovuto costruirne uno mio da zero.

Cosa significa questo? Decisi di non usare pacchetti pre-compilati di nessun tipo, nemmeno CD-ROM o dischi di avvio che installassero utilità di base. Avrei usato il mio attuale sistema Linux per sviluppare il mio sistema personalizzato. Questo sistema Linux « perfetto» avrebbe avuto la forza di vari sistemi senza le loro debolezze associate. All'inizio l'idea faceva abbastanza paura, ma mi feci coinvolgere dall'idea di poter costruire un sistema che si sarebbe conformato ai miei bisogni e desideri, piuttosto che a uno standard che semplicemente non si sarebbe adattato a ciò che cercavo.

Dopo aver superato problemi come dipendenze circolari ed errori di time-out di compilazione creai un sistema Linux personalizzato pienamente operativo e adeguato ai bisogni individuali. Questo processo mi permise anche di creare sistemi Linux compatti che erano più veloci e prendevano meno spazio dei tradizionali sistemi operativi. Chiamai questo sistema un sistema Linux From Scratch [NdT: tradotto, Linux da zero], o, brevemente, un sistema LFS.

Quando condivisi i miei obiettivi ed esperienze con altri membri della comunità Linux divenne evidente che c'era un interesse elevato nelle idee alla base delle mie avventure con Linux. Questo sistema LFS personalizzato non solo viene incontro a specifiche e richieste degli utenti, ma serve anche come un'opportunità di apprendimento ideale per programmatori e amministratori di sistema per ampliare la loro conoscenza di Linux. Emerso questo ampio interesse, il progetto Linux From Scratch era nato.

Questo libro *Linux From Scratch* fornisce ai lettori il background e le istruzioni per disegnare e costruire sistemi Linux su misura. Questo libro evidenzia il progetto Linux from Scratch e i benefici dell'uso di questo sistema. Gli utenti possono dettare tutti gli aspetti del loro sistema, inclusi layout directory, setup di script, e sicurezza. Il sistema risultante verrà compilato completamente dal codice sorgente, e l'utente sarà in grado di specificare dove, perché, e come i programmi vengono installati. Questo libro permette ai lettori di adattare totalmente i sistemi Linux ai propri bisogni, e permette agli utenti un maggior controllo sul proprio sistema.

Spero che passerete bene il tempo lavorando sul vostro sistema LFS, e gradirete i numerosi benefici di avere un sistema che sia veramente *vostro*.

--
Gerard Beekmans
gerard@linuxfromscratch.org

2. Audience

Ci sono molte ragioni per voler leggere questo libro. La ragione principale è l'installazione di un sistema Linux dal codice sorgente. Una domanda che molti pongono è « Perché affrontare la seccatura della costruzione manuale di un sistema Linux da zero quando si può semplicemente scaricarne e installarne uno esistente?». Questa è una buona domanda ed è la ragione di questa sezione del libro.

Una ragione importante per l'esistenza di LFS è di aiutare la gente a imparare come funziona un sistema Linux dall'interno. Costruire un sistema LFS aiuta a dimostrare cosa rende Linux forte, come le cose funzionano insieme e dipendono l'una dall'altra. Una delle cose migliori che questa esperienza di studio fornisce è l'abilità di personalizzare Linux secono i propri gusti e necessità.

Un beneficio chiave di LFS è che si ha maggior controllo sul proprio sistema senza dipendere dall'implementazione Linux di qualcun altro. Con LFS, si è al posto di guida e si detta ogni aspetto del proprio sistema, come il layout delle directory e l'impostazione degli script di avvio. Si detta anche dove, perché e come sono installati i programmi.

Un altro beneficio di LFS è l'abilità di creare un sistema Linux molto compatto. Quando si installa una distribuzione regolare, di solito si è obbligati a includere molti programmi che probabilmente non verranno mai usati. Semplicemente si piazzano lì sprecando spazio disco (o peggio, cicli CPU). Non è difficile costruire un sistema LFS di meno di 100 MB, che è sostanzialmente più piccolo della maggior parte delle installazioni esistenti. Vi sembra ancora molto? Alcuni di noi hanno lavorato alla creazione di un sistema LFS embedded molto piccolo. Abbiamo costruito con successo un sistema che era giusto sufficiente ad eseguire il web server Apache con circa 8 MB di spazio disco usato. Ulteriori compattazioni possono ridurlo a 5 MB o meno. Si provi a farlo con una distribuzione regolare.

Compareremo una distribuzione Linux ad un hamburger che si compra ad un ristorante fast-food—non si ha idea di cosa si sta mangiando. LFS, d'altra parte, non fornisce l'hamburger, ma la ricetta per fare l' hamburger. Questo permette di rivederla, per omettere ingredienti indesiderati, e aggiungere i propri ingredienti che aumentano il gusto del proprio hamburger. Quando si è soddisfatti della ricetta, si prosegue e la si prepara. Lo si fa semplicemente come piace: arrostito, al forno, fritto, al barbecue, o crudo.

Un'altra analogia che possiamo usare è quella di comparare LFS con una casa finita. LFS darà il progetto della casa, ma sta a voi costruirla. Si ha la libertà di aggiustare il progetto in corsa, personalizzandolo secondo i bisogni e le preferenze.

Un ulteriore vantaggio di un sistema Linux personalizzato è la sicurezza. Compilando l'intero sistema dal codice sorgente, si ha il potere di controllare tutto e applicare tutte le patch di sicurezza che si sentono necessarie. Non si deve aspettare che qualcun altro compili i pacchetti binari per fissare un buco di sicurezza. A meno di esaminare la patch e implementarla da sé non si ha garanzia che il nuovo pacchetto binario sia costruito correttamente e corregga il problema (adeguatamente).

L'obiettivo di Linux From Scratch è di costruire un sistema completo e usabile dalle fondamenta. I lettori che non desiderano costruire il proprio sistema Linux da zero possono non trarre beneficio dalle informazioni in questo libro. Se si vuole solo sapere cosa succede mentre il computer si avvia, raccomandiamo l'HOWTO «From Power Up To Bash Prompt» che si trova presso *http://axiom.anu.edu.au/~okeefe/p2b/* o sul sito di The Linux Documentation Project (TLDP) su *http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html*. L'HOWTO costruisce un sistema simile a quello di questo libro, ma si focalizza strettamente sulla creazione di un sistema in grado di avviarsi fino a un prompt BASH. Si considerino i propri obiettivi. Se si vuole costruire un sistema Linux e imparare lungo la strada, allora questo libro è la miglior scelta.

Ci sono troppe buone ragioni per costruire il proprio sistema LFS per elencare tutte qui. Questa sezione è solo la punta dell'iceberg. Continuando nella propria esperienza con LFS, si scoprirà da sé il potere che informazione e conoscenza portano davvero.

3. Prerequisiti

Costruire un sistema LFS non è un lavoro semplice. Richiede un certo livello di conoscenza dell'amministrazione di un sistema Unix per poter risolvere i problemi ed eseguire correttamente i comandi elencati. In particolare, come minimo indispensabile, il lettore deve sapere già usare la linea di comando (shell) per copiare o spostare file e directory, elencare il contenuto di directory e file, e cambiare la directory corrente. Ci si aspetta anche che il lettore abbia una ragionevole conoscenza dell'uso e installazione di software Linux.

Poiché il libro LFS suppone *almeno* questo livello di conoscenza di base i vari supporti LFS saranno difficilmente in grado di fornire una grande assistenza; si scoprirà che le proprie domande riguardanti tali conoscenze di base rimarranno probabilmente senza risposta, o si verrà semplicemente rimandati all'elenco delle pre-letture essenziali di LFS.

Prima di costruire un sistema LFS si raccomanda di leggere i seguenti HOWTO:

- Software-Building-HOWTO
  http://www.tldp.org/HOWTO/Software-Building-HOWTO.html

  Questo è una guida completa alla costruzione e installazione di distribuzioni di software Unix «generiche» sotto Linux.

- The Linux Users' Guide
  http://www.linuxhq.com/guides/LUG/guide.html

  Questa guida copre l'uso di software Linux assortito.

- The Essential Pre-Reading Hint
  http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt

  Questo è un Hint LFS scritto specificamente per i nuovi utenti di Linux. È per lo più una lista di link a eccellenti sorgenti di informazione su un'ampia varietà di argomenti. Chiunque tenti di installare LFS deve almeno avere una comprensione di molti degli argomenti in questo hint.

4. Requisiti del sistema host

Il proprio sistema host deve avere il seguente software con almeno le versioni indicate. Questo non dovrebbe essere un problema per la maggior parte delle distribuzioni Linux moderne. Notare, inoltre, che molte distribuzioni metteranno gli header del software in pacchetti separati, spesso nella forma di «<package-name>-devel» o «<package-name>-dev». Assicurarsi di installare questi se la propria distribuzione li fornisce.

- **Bash-2.05a**
- **Binutils-2.12** (Versioni maggiori di 2.16.1 non sono consigliate, poiché non sono state testate)
- **Bzip2-1.0.2**
- **Coreutils-5.0** (o Sh-Utils-2.0, Textutils-2.0, e Fileutils-4.1)
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.0**
- **Gcc-2.95.3** (Versioni maggiori di 4.0.3 non sono consigliate, poiché non sono state testate)
- **Glibc-2.2.5** (Versioni maggiori della 2.3.6 non sono consigliate, poiché non sono state testate)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Linux Kernel-2.6.x** (compilato con GCC-3.0 o maggiore)

  La ragione del requisito di versione del kernel è che il supporto al thread-local storage in Binutils non verrà costruito e la suite di test del Native POSIX Threading Library (NPTL) terminerà con un segfault se il kernel dell'host non è almeno una versione 2.6.x compilata con una versione di GCC 3.0 o successiva.

  Se il kernel dell'host è precedente a 2.6.x, o se non è stato compilato usando un compilatore GCC-3.0 (o successivo) bisognerà sostituire il kernel con uno aderente alle specifiche. Si possono adottare due metodi per risolvere il problema. Primo, vedere se la propria distribuzione Linux fornisce un pacchetto kernel 2.6. Se lo fa lo si potrebbe installare. Se la propria distribuzione non offre un pacchetto kernel 2.6, o se si preferisce non installarlo, allora si può compilare da sé un kernel 2.6. Le istruzioni per la compilazione del kernel e la configurazione del bootloader (supponendo che l'host usi GRUB) si trovano nel Capitolo 8.
- **Make-3.79.1**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.14**

Per vedere se il proprio sistema host ha tutte le versioni di software appropriate, eseguire il seguente script:

```
cat > version-check.sh << "EOF"
#!/bin/bash

# Semplice script per elencare i numeri di versione degli strumenti di
sviluppo critici

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-4
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
gcc --version | head -n1
/lib/libc.so.6 | head -n1 | cut -d" " -f1-7
grep --version | head -n1
gzip --version | head -n1
cat /proc/version | head -n1 | cut -d" " -f1-3,5-7
make --version | head -n1
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1

EOF

bash version-check.sh
```

5. Tipografia

Per rendere le cose più facili da seguire ci sono alcune convenzioni tipografiche usate nel libro. Questa sezione contiene alcuni esempi dei formati tipografici che si trovano durante Linux from Scratch:

```
./configure --prefix=/usr
```

Questa forma di testo è disegnata per essere digitata esattamente come la si vede tranne ove altrimenti detto nel testo attorno. È anche usata nelle sezioni di spiegazione per identificare a quale dei comandi ci si riferisce.

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

Questa forma di testo (testo a larghezza fissa) mostra l'output dello schermo, probabilmente come risultato di comandi digitati. Questo formato è usato anche per mostrare nomi file, come `/etc/ld.so.conf`.

Enfasi

Questa forma di testo è usata per molti scopi nel libro, principalmente per enfatizzare punti importanti.

http://www.linuxfromscratch.org/

Questa forma di testo è usata per gli hyperlink, sia nel libro che verso pagine esterne come HOWTO, locazioni di download e siti web.

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
......
EOF
```

Questo formato è usato principalmente quando si creano file di configurazione. Il primo comando dice al sistema di creare il file `$LFS/etc/group` da ciò che è digitato nelle linee seguenti fino a quando viene incontrata la sequenza di fine file (EOF). Quindi questa intera sezione normalmente è digitata come vista.

<TESTO SOSTITUITO>

Questa forma di testo è usata per incapsulare testo che non deve essere digitato esattamente .

[TESTO OPZIONALE]

Questo formato è usato per incapsulare testo opzionale.

`passwd(5)`

Questo formato è usato per fare riferimento a una specifica pagina di manuale (d'ora in avanti, chiamata semplicemente pagina «man»). Il numero dentro le parentesi indica una specifica sezione dentro **man**. Per esempio **passwd** ha due pagine man. Per le istruzioni di installazione di LFS, queste due pagine man si troveranno in `/usr/share/man/man1/passwd.1` e `/usr/share/man/man5/passwd.5`. Entrambe le pagine man hanno diverse informazioni in esse. Quando il libro usa `passwd(5)` esso si riferisce specificamente a `/usr/share/man/man5/passwd.5`. **man passwd** stamperà la prima pagina man che trova che corrisponda a «passwd», che sarà `/usr/share/man/man1/passwd.1`. Per questo esempio bisognerà eseguire **man 5 passwd** per leggere la pagina specifica a cui si fa riferimento. Bisogna notare che la maggior parte delle pagine man non ha nomi di pagina duplicati in sezioni differenti. Pertanto, **man *<nome programma>*** generalmente è sufficiente.

6. Struttura

Questo libro è diviso nelle seguenti parti:

6.1. Parte I - Introduzione

La parte I spiega alcune cose importanti su come procedere con l'installazione di LFS. Fornisce anche meta-informazioni riguardanti il libro.

6.2. Parte II - Preparazione della costruzione

La parte II descrive come preparare il processo di costruzione: creazione di una partizione, scaricamento dei pacchetti e compilazione dei tool temporanei.

6.3. Parte III - Costruzione del sistema LFS

La parte III guida attraverso la costruzione del sistema LFS: compilazione ed installazione di tutti i pacchetti uno per uno, impostazione degli script di avvio e installazione del kernel. Il sistema Linux di base risultante è il fondamento su cui si può installare altro software, per estendere il proprio sistema nel modo che si preferisce. Al termine del libro, si trova un lista di tutti i programmi, librerie e file importanti che sono stati installati, come riferimento facile da usare.

7. Errata

Il software utilizzato per creare un sistema LFS è costantemente aggiornato e migliorato. Avvisi relativi alla sicurezza e correzioni di bug possono diventare disponibili dopo che il libro LFS è stato rilasciato. Per controllare se le versioni o le istruzioni di un pacchetto contenute in questa versione di LFS hanno bisogno di qualche modifica per sistemare le vulnerabilità di sicurezza o altre correzioni di bug, si visiti *http://www.linuxfromscratch.org/lfs/errata/6.2/* prima di procedere con la costruzione del sistema. Si tenga presente ogni modifica presentata e la si applichi alle sezioni appropriate del libro mano a mano che si procede con la costruzione del sistema LFS.

Parte I. Introduzione

Capitolo 1. Introduzione

1.1. Come costruire un Sistema LFS

Il sistema LFS verrà costruito utilizzando una distribuzione Linux precedentemente installata (come Debian, Mandrake, Red Hat o SuSE). Questo sistema Linux esistente (l'ospite o host) sarà usato come punto di partenza per fornire i programmi necessari, che includono un compilatore, un linker e una shell per costruire il nuovo sistema. Si selezioni l'opzione «development» durante l'installazione della propria distribuzione per poter accedere a questi strumenti.

Come alternativa all'installazione di un'intera distribuzione separata sulla propria macchina si potrebbe usare il Linux From Scratch LiveCD. Il CD funziona bene come sistema host, fornendo tutti gli strumenti necessari per seguire con successo le istruzioni in questo libro. Inoltre contiene tutti i pacchetti sorgenti, patche e una copia di questo libro. In questo modo una volta che si ha il CD non sono necessarie connessioni in rete o download aggiuntivi. Per ulteriori informazioni su LiveCD LFS o per scaricare una copia visitare *http://www.linuxfromscratch.org/livecd/*.

Il Capitolo 2 di questo libro descrive come creare una nuova partizione ed un file system nativi Linux, la sede in cui il nuovo sistema LFS sarà compilato ed installato. Il Capitolo 3 spiega quali pacchetti e patch scaricare per costruire un sistema LFS e come memorizzarli nel nuovo file system. Il Capitolo 4 discute come configurare un ambiente funzionante. Bisognerebbe leggere attentamente il Capitolo 4 perché spiega questioni importanti di cui uno sviluppatore dovrebbe essere consapevole prima di procedere attraverso il Capitolo 5 e oltre.

Il Capitolo 5 illustra l'installazione di alcuni pacchetti che costituiranno l'insieme degli strumenti fondamentali di sviluppo (la toolchain), usati per costuire il sistema LFS vero e proprio nel Capitolo 6. Alcuni di questi pacchetti sono necessari per la risoluzione di dipendenze circolari; ad esempio, per compilare un compilatore è necessario un compilatore.

Il Capitolo 5 mostra anche all'utente come costruire una prima passaggio della toolchain, comprendente Binutils e GCC (primo passaggio sostanzialmente significa che questi due pacchetti cruciali saranno reinstallati una seconda volta). Il passo successivo è la costruzione di Glibc, la libreria C. Glibc sarà compilata usando i programmi della toolchain costruiti nel primo passaggio. Poi, sarà costruito un secondo passaggio della toolchain. Questa volta la toolchain sarà linkata dinamicamente alla Glibc appena costruita. I pacchetti restanti del Capitolo 5 saranno tutti costruiti usando la toolchain di questo secondo passaggio. Al completamento di tutto questo l'installazione di LFS non dipenderà più dalla distribuzione ospite, con l'eccezione del kernel in esecuzione.

Questo sforzo per isolare il nuovo sistema dalla distribuzione host può sembrare eccessivo, ma all'inizio del capitolo Sezione 5.2, «Note tecniche sulla Toolchain» è fornita una spiegazione tecnica completa.

Nel Capitolo 6 viene costruito il sistema LFS completo. Il programma **chroot** (change root) viene usato per entrare in un ambiente virtuale ed avviare una nuova shell la cui directory root sarà la partizione LFS. Questo è molto simile a riavviare imponendo al kernel di montare la partizione LFS come partizione root. Il sistema non si riavvia davvero, si effettua solo il cambio di root perché la creazione di un sistema avviabile richiede del lavoro aggiuntivo che non è ancora necessario. Il vantaggio principale sta nel fatto che «cambiare root» permette di continuare ad usare il sistema ospite mentre si costruisce LFS. Nell'attesa della compilazione di un pacchetto si potrà passare ad una nuova console virtuale (CV) o ad un desktop X e continuare ad usare il computer normalmente.

Per completare l'installazione nel Capitolo 7 sono impostati gli LFS-Bootscript e nel Capitolo 8 sono impostati il kernel e il bootloader. Il Capitolo 9 contiene informazioni su come proseguire l'esperienza LFS oltre questo libro. Dopo aver eseguito i passi illustrati in questo libro, il computer sarà pronto per essere riavviato con il nuovo sistema LFS.

Questo è il processo in poche parole. Le informazioni dettagliate sui passi da fare sono trattate nei capitoli successivi e nelle descrizioni dei pacchetti. Le cose che possono sembrare complicate si chiariranno e tutto quadrerà man mano che si procede nell'avventura di LFS.

1.2. Risorse

1.2.1. FAQ

Se, durante la costruzione del sistema LFS, si incontrano degli errori, si hanno domande, o si pensa che ci sia un errore nel libro, si è pregati di consultare prima le Frequently Asked Questions (FAQ), che si trovano presso *http://www.linuxfromscratch.org/faq/*.

1.2.2. Mailing List

Il server `linuxfromscratch.org` ospita alcune mailing list usate per lo sviluppo del progetto LFS. Queste liste includono tra le altre le liste principali di sviluppo e suporto. Se la FAQ non risolve il proprio problema il passo successivo sarebbe la ricerca nelle mailing lista presso *http://www.linuxfromscratch.org/search.html*.

Per informazioni sulle diverse liste, su come iscriversi, sulla locazione degli archivi, e per ulteriori informazioni, visitare *http://www.linuxfromscratch.org/mail.html*.

1.2.3. IRC

Molti membri della comunità LFS offrono assistenza sulla nostra comunità Internet Relay Chat (IRC). Prima di usare questo suporto assicurarsi che la propria domanda non abbia già una risposta nelle FAQ LFS o negli archivi della mailing list. Si può trovare la rete IRC presso `irc.linuxfromscratch.org`. Il canale di supporto si chiama #LFS-support.

1.2.4. Riferimenti

Per ulteriori informazioni sui pacchetti, suggerimenti utili sono disponibili nella pagina di riferimento dei pacchetti LFS situata presso *http://www.linuxfromscratch.org/~matthew/LFS-references.html*.

1.2.5. siti mirror

Il progetto LFS ha numerosi mirror nel mondo per rendere più semplice l'accesso al sito e lo scaricamento dei pacchetti necessari. Visitare il sito web di LFS presso *http://www.linuxfromscratch.org/mirrors.html* per un elenco dei mirror correnti.

1.2.6. Contatti

Si è pregati di indirizzare tutte le proprie domande e commenti sulle mailing list di LFS (vedere sopra).

1.3. Aiuto

Se si incontrasse un problema o un dubbio mentre si lavora su questo libro controllare la pagina delle FAQ presso *http://www.linuxfromscratch.org/faq/#generalfaq*. Spesso le domande hanno già una risposta. Se così non fosse, si dovrebbe tentare di risalire all'origine del problema. Il seguente suggerimento dovrebbe fornire qualche idea per l'analisi dei problemi: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Se non si riesce a trovare il proprio roblema elencato nelle FAQ, cercare nelle mailing list su *http://www.linuxfromscratch.org/search.html*.

Abbiamo anche una meravigliosa comunità LFS che è disposta ad offrire assistenza attraverso le mailing list e IRC (si veda la sezione di questo libro Sezione 1.2, «Risorse»). Tuttavia riceviamo numerose richieste di supporto ogni giorno, e molte di esse possono trovare facilmente risposta andando sulle FAQ e cercando prima nelle mailing list. Perciò, perché noi possiamo offrire la miglior assistenza possibile, è meglio prima fare qualche ricerca da sé. Questo permette a noi di focalizzarci sulle necessità di supporto più inusuali. Se le proprie ricerche non producono una soluzione si è pregati di includere tutte le informazioni significative (menzionate in seguito) nella propria richiesta di aiuto.

1.3.1. Dati da indicare

Oltre ad una breve descrizione del problema, le informazioni essenziali da includere nella richiesta di aiuto sono:

- La versione del libro che si sta usando (in questo caso 6.2)
- La distribuzione e versione del sistema ospite usato per creare LFS
- Il pacchetto o la sezione in cui si è incontrato il problema
- L'esatto messaggio di errore o il sintomo rilevato
- Se ci si è scostati in qualche modo dal libro

Nota

Discostarsi dal libro *non* significa che non forniremo aiuto. Dopotutto LFS stesso si basa sulla preferenza personale. Essere sinceri sulle variazioni alla procedura stabilita ci aiuta a valutare ed individuare le possibili cause del problema.

1.3.2. Problemi dello script configure

Se qualcosa va storto durante l'esecuzione dello script **configure** esaminare il file `config.log`. Questo file potrebbe contenere degli errori incontrati durante **configure**, che non sono stati visualizzati sullo schermo. Se si decide di chiedere assistenza è bene includere le righe *rilevanti*.

1.3.3. Problemi di compilazione

Per facilitare l'individuazione della causa di problemi di compilazione, possono tornare utili sia l'output su schermo che il contenuto di vari file. Può essere utile l'output su schermo prodotto sia dallo script **configure** che dall'esecuzione di **make**. Non è necessario includere tutto l'output, ma bisogna includere abbastanza informazioni rilevanti. Ecco un esempio del tipo di informazione da includere dall'output su schermo di **make**:

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
```

```
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

In questo caso molte persone si limitano ad includere la sezione in basso:

```
make [2]: *** [make] Error 1
```

Questo non è sufficiente per diagnosticare il problema perché dice solamente che qualcosa è andato storto, non *cosa* è andato storto. L'intera sezione, come appare nell'esempio qui sopra, è quanto andrebbe incluso, perché riporta il comando che è stato eseguito ed il messaggio (o i messaggi) di errore di quel comando.

Un ottimo articolo su come chiedere aiuto su Internet è disponibile online presso *http://catb.org/~esr/faqs/smart-questions.html*. Leggendolo e mettendo in pratica i suggerimenti di questo documento si avranno molte più probabilità di ricevere l'aiuto che effettivamente serve.

Parte II. Preparazione della costruzione

Capitolo 2. Preparazione di una nuova partizione

2.1. Introduzione

In questo capitolo verrà preparata la partizione che ospiterà il sistema LFS. Verrà creata la partizione, verrà creato un file system su di essa e infine la partizione verrà montata.

2.2. Creazione di una nuova partizione

Come molti altri sistemi operativi, LFS è normalmente installato su una partizione dedicata. L'approccio raccomandato per costruire un sistema LFS è di usare una partizione vuota disponibile o, se si ha sufficiente spazio non partizionato, di crearne una. Comunque un sistema LFS (in effetti anche sistemi LFS multipli) può anche essere installato su una partizione già occupata da un altro sistema operativo e i diversi sistemi coesisteranno pacificamente. Il documento *http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt* spiega come implementarlo, mentre questo libro discute il metodo di usare una partizione nuova per l'installazione.

Un sistema minimale richiede una partizione di circa 1.3 gigabyte (GB), che sono sufficienti per contenere tutti i tarball dei sorgenti e compilare tutti i pacchetti. Ma se l'intenzione è di usare il sistema LFS come sistema Linux principale, probabilmente si vorrà installare del software aggiuntivo che richiederà più spazio (2-3 GB). Il sistema LFS di per sé non userà tutto tutto questo spazio. Gran parte di esso serve per fornire spazio ad operazioni temporanee. Compilare un pacchetto può richiedere molto spazio disco che sarà liberato dopo che il pacchetto è stato installato.

Dal momento che non c'è mai abbastanza Random Access Memory (RAM) per i processi di compilazione, è una buona idea usare una piccola partizione come spazio di `swap`. Questo spazio è usato dal kernel per memorizzare dati usati raramente e lasciare più memoria disponibile per i processi attivi. La partizione di `swap` del sistema LFS può essere la stessa usata dal sistema ospite, nel qual caso non è necessario crearne un'altra.

Avviare un programma di partizionamento come **cfdisk** o **fdisk** con un'opzione a linea di comando che nomini il disco su cui dovrà essere creata la nuova partizione. Per esempio `/dev/hda` per il disco IDE (Integrated Drive Electronics) primario. Creare quindi una partizione nativa Linux e, se necessaria, una partizione di `swap`. Se non si sa come usare questi programmi si può fare riferimento a `cfdisk(8)` o `fdisk(8)`.

Occorre ricordarsi la denominazione della nuova partizione (es. `hda5`). Questo libro farà riferimento ad essa come alla partizione LFS. Bisogna ricordare anche la denominazione della partizione di `swap`. Questi nomi saranno necessari in seguito per il file `/etc/fstab`.

2.3. Creazione di un file system sulla partizione

Ora che è stata creata una partizione vuota, si può creare il file system. Il sistema più usato nel mondo Linux è Ã¨ il file system second extended (`ext2`), ma con i nuovi dischi ad alta capacità i cosiddetti journaling file system stanno diventando sempre più popolari. Il file system third extended (`ext3`) rappresenta un sempre piú diffuso incremento dell'`ext2`; aggiunge un sistema di journalling ed è compatibile con le utilities E2fsprogs. Qui sarà creato un file system `ext3`. Istruzioni per creare altri file system possono essere trovate presso *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html*.

Per creare un file system `ext3` sulla partizione LFS lanciare il seguente comando:

```
mke2fs -jv /dev/<xxx>
```

Dove *<xxx>* va sostituito con il nome della partizione LFS (nell'esempio precedente `hda5`).

Nota

Alcune distribuzioni host usano caratteristiche personalizzate nei loro tool di creazione del filesystem (E2fsprogs). Ciò può causare problemi quando si avvierÃ nel proprio LFS nel Capitolo 9, poiché queste caratteristiche non saranno supportate dal programma E2fsprogs installato da LFS; si avrà un errore simile a «unsupported filesystem features, upgrade your e2fsprogs». Per verificare se il proprio sistema ospite usi espansioni personalizzate eseguire il seguente comando:

```
debugfs -R feature /dev/<xxx>
```

Se l'output contiene caratteristiche diverse da: `has_journal`, `dir_index`, `filetype`, `large_file`, `resize_inode`, `sparse_super` or `needs_recovery`, allora il proprio sistema ospite potrebbe avere espansioni personalizzate. In questo caso, per evitare successivi problemi, si può compilare il pacchetto E2fsprogs di LFS e usare i binari risultanti per ricreare il filesystem sulla propria partizione LFS:

```
cd /tmp
tar -xjvf /path/to/sources/e2fsprogs-1.39.tar.bz2
cd e2fsprogs-1.39
mkdir -v build
cd build
../configure
make #notare che qui non si esegue intenzionalmente 'make install'!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.39
```

Se è stata creata una partizione di `swap`, sarà anche necessario inizializzarla per l'uso lanciando il comando seguente. Se si sta usando una partizione di `swap` esistente, non c'è bisogno di formattarla.

```
mkswap /dev/<yyy>
```

Dove *<yyy>* va sostituito con il nome della partizione di swap.

2.4. Montaggio della nuova partizione

Ora che è stato creato un file system, la partizione deve essere resa accessibile. Per farlo, la partizione deve essere montata sotto un punto di mount a scelta. Per gli scopi di questo libro si assume che il file system venga montato sotto `/mnt/lfs`, ma la scelta della directory è lasciata al lettore.

Scegliere un punto di mount e assegnarlo alla variabile d'ambiente `LFS`, lanciando:

```
export LFS=/mnt/lfs
```

Creare il punto di mount e montare il file system LFS, lanciando:

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
```

Dove *<xxx>* va sostituito con il nome della partizione LFS.

Se si usano più partizioni per LFS (es. una per `/` e un'altra per `/usr`), montarle con:

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext3 /dev/<yyy> $LFS/usr
```

Dove *<xxx>* e *<yyy>* vanno sostituiti con i nomi delle relative partizioni.

Sarebbe bene anche assicurarsi che questa nuova partizione non sia montata con permessi troppo restrittivi (come le opzioni `nosuid`, `nodev`, o `noatime`). È possibile lanciare il comando **mount** senza alcun parametro per verificare quali opzioni sono state associata alla partizione LFS appena montata. Qualora fossero presenti le opzioni `nosuid`, `nodev`, e/o `noatime`, la partizione dovrà essere montata di nuovo.

Se si sta usando una partizione di `swap`, ci si assicuri che sia abilitata, lanciando il comando **swapon**:

```
/sbin/swapon -v /dev/<zzz>
```

Dove *<zzz>* va sostituito con il nome della partizione di swap.

Ora che si è creato uno spazio dove lavorare, è tempo di scaricare i pacchetti.

Capitolo 3. Pacchetti e patch

3.1. Introduzione

Questo capitolo include un elenco dei pacchetti da scaricare per costruire un sistema Linux di base. I numeri di versione riportati corrispondono alle versioni dei software che sappiamo funzionare, e questo libro è basato sul loro uso. Sconsigliamo fortemente di usare nuove versioni, poiché i comandi di compilazione per una versione potrebbero non funzionare con una versione più recente. Le versioni dei pacchetti più nuove potrebbero inoltre avere dei problemi che richiedono di lavorarci sopra. Questi lavori verranno sviluppati e stabilizzati nella versione di sviluppo del libro.

Non possiamo garantire che queste locazioni di download siano sempre accessibili. Nel caso in cui una locazione di download sia cambiata da quando questo libro è stato pubblicato, Google (*http://www.google.com/*) fornisce un utile motore di ricerca per molti pacchetti. Se questo tentativo non dovesse avere successo, si provi uno dei metodi alternativi di download discussi su *http://www.linuxfromscratch.org/lfs/packages.html*.

Sarà necessario archiviare i pacchetti scaricati e le patch in qualche posto che sia opportunamente disponibile durante tutta la costruzione. Servirà anche una directory di lavoro in cui scompattare i sorgenti e compilarli. `$LFS/sources` può essere utilizzata sia come posto per archiviare tarball e patch che come directory di lavoro. Usando questa directory, gli elementi richiesti si troveranno nella partizione LFS e saranno disponibili durante tutte le fasi del processo di costruzione.

Per creare questa directory si esegua il seguente comando, come utente `root`, prima di iniziare la sessione di download:

```
mkdir -v $LFS/sources
```

Rendere questa directory scrivibile e bloccata (Sticky). «Sticky» significa che anche se più utenti hanno permesso di scrittura su una directory solo il proprietario di un file può cancellare il file all'interno della directory sticky. Il seguente comando abiliterà le modalità scrittura e sticky:

```
chmod -v a+wt $LFS/sources
```

3.2. Tutti i pacchetti

Scaricare o procurarsi in altro modo i seguenti pacchetti:

- Autoconf (2.59) - 904 KB:
  Home page: *http://www.gnu.org/software/autoconf/*
  Download: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.59.tar.bz2*
  MD5 sum: `1ee40f7a676b3cfdc0e3f7cd81551b5f`
- Automake (1.9.6) - 748 KB:
  Home page: *http://www.gnu.org/software/automake/*
  Download: *http://ftp.gnu.org/gnu/automake/automake-1.9.6.tar.bz2*
  MD5 sum: `c11b8100bb311492d8220378fd8bf9e0`
- Bash (3.1) - 2,475 KB:
  Home page: *http://www.gnu.org/software/bash/*
  Download: *http://ftp.gnu.org/gnu/bash/bash-3.1.tar.gz*
  MD5 sum: `ef5304c4b22aaa5088972c792ed45d72`
- Bash Documentation (3.1) - 2,013 KB:
  Download: *http://ftp.gnu.org/gnu/bash/bash-doc-3.1.tar.gz*
  MD5 sum: `a8c517c6a7b21b8b855190399c5935ae`
- Berkeley DB (4.4.20) - 7,767 KB:
  Home page: *http://dev.sleepycat.com/*
  Download: *http://downloads.sleepycat.com/db-4.4.20.tar.gz*
  MD5 sum: `d84dff288a19186b136b0daf7067ade3`
- Binutils (2.16.1) - 12,256 KB:
  Home page: *http://sources.redhat.com/binutils/*
  Download: *http://ftp.gnu.org/gnu/binutils/binutils-2.16.1.tar.bz2*
  MD5 sum: `6a9d529efb285071dad10e1f3d2b2967`
- Bison (2.2) - 1,052 KB:
  Home page: *http://www.gnu.org/software/bison/*
  Download: *http://ftp.gnu.org/gnu/bison/bison-2.2.tar.bz2*
  MD5 sum: `e345a5d021db850f06ce49eba78af027`
- Bzip2 (1.0.3) - 654 KB:
  Home page: *http://www.bzip.org/*
  Download: *http://www.bzip.org/1.0.3/bzip2-1.0.3.tar.gz*
  MD5 sum: `8a716bebecb6e647d2e8a29ea5d8447f`
- Coreutils (5.96) - 4,948 KB:
  Home page: *http://www.gnu.org/software/coreutils/*
  Download: *http://ftp.gnu.org/gnu/coreutils/coreutils-5.96.tar.bz2*
  MD5 sum: `bf55d069d82128fd754a090ce8b5acff`

• DejaGNU (1.4.4) - 1,056 KB:
Home page: *http://www.gnu.org/software/dejagnu/*
Download: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz*
MD5 sum: `053f18fd5d00873de365413cab17a666`

• Diffutils (2.8.1) - 762 KB:
Home page: *http://www.gnu.org/software/diffutils/*
Download: *http://ftp.gnu.org/gnu/diffutils/diffutils-2.8.1.tar.gz*
MD5 sum: `71f9c5ae19b60608f6c7f162da86a428`

• E2fsprogs (1.39) - 3,616 KB:
Home page: *http://e2fsprogs.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.39.tar.gz?download*
MD5 sum: `06f7806782e357797fad1d34b7ced0c6`

• Expect (5.43.0) - 514 KB:
Home page: *http://expect.nist.gov/*
Download: *http://expect.nist.gov/src/expect-5.43.0.tar.gz*
MD5 sum: `43e1dc0e0bc9492cf2e1a6f59f276bc3`

• File (4.17) - 544 KB:
Download: *ftp://ftp.gw.com/mirrors/pub/unix/file/file-4.17.tar.gz*
MD5 sum: `50919c65e0181423d66bb25d7fe7b0fd`

Nota
File (4.17) potrebbe non essere più disponibile nella locazione elencata. Gli amministratori del sito di download principale occasionalmente rimuovono vecchie versioni quando ne vengono rilasciate di nuove. Una locazione alternativa di download che può avere disponibile la versione corretta è *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.2.27) - 1,097 KB:
Home page: *http://www.gnu.org/software/findutils/*
Download: *http://ftp.gnu.org/gnu/findutils/findutils-4.2.27.tar.gz*
MD5 sum: `f1e0ddf09f28f8102ff3b90f3b5bc920`

• Flex (2.5.33) - 680 KB:
Home page: *http://flex.sourceforge.net*
Download: *http://prdownloads.sourceforge.net/flex/flex-2.5.33.tar.bz2?download*
MD5 sum: `343374a00b38d9e39d1158b71af37150`

• Gawk (3.1.5) - 1,716 KB:
Home page: *http://www.gnu.org/software/gawk/*
Download: *http://ftp.gnu.org/gnu/gawk/gawk-3.1.5.tar.bz2*
MD5 sum: `5703f72d0eea1d463f735aad8222655f`

- GCC (4.0.3) - 32,208 KB:
  Home page: *http://gcc.gnu.org/*
  Download: *http://ftp.gnu.org/gnu/gcc/gcc-4.0.3/gcc-4.0.3.tar.bz2*
  MD5 sum: `6ff1af12c53cbb3f79b27f2d6a9a3d50`
- Gettext (0.14.5) - 6,940 KB:
  Home page: *http://www.gnu.org/software/gettext/*
  Download: *http://ftp.gnu.org/gnu/gettext/gettext-0.14.5.tar.gz*
  MD5 sum: `e2f6581626a22a0de66dce1d81d00de3`
- Glibc (2.3.6) - 13,687 KB:
  Home page: *http://www.gnu.org/software/libc/*
  Download: *http://ftp.gnu.org/gnu/glibc/glibc-2.3.6.tar.bz2*
  MD5 sum: `bfdce99f82d6dbcb64b7f11c05d6bc96`
- Glibc LibIDN add-on (2.3.6) - 99 KB:
  Download: *http://ftp.gnu.org/gnu/glibc/glibc-libidn-2.3.6.tar.bz2*
  MD5 sum: `49dbe06ce830fc73874d6b38bdc5b4db`
- Grep (2.5.1a) - 516 KB:
  Home page: *http://www.gnu.org/software/grep/*
  Download: *http://ftp.gnu.org/gnu/grep/grep-2.5.1a.tar.bz2*
  MD5 sum: `52202fe462770fa6be1bb667bd6cf30c`
- Groff (1.18.1.1) - 2,208 KB:
  Home page: *http://www.gnu.org/software/groff/*
  Download: *http://ftp.gnu.org/gnu/groff/groff-1.18.1.1.tar.gz*
  MD5 sum: `511dbd64b67548c99805f1521f82cc5e`
- GRUB (0.97) - 950 KB:
  Home page: *http://www.gnu.org/software/grub/*
  Download: *ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz*
  MD5 sum: `cd3f3eb54446be6003156158d51f4884`
- Gzip (1.3.5) - 324 KB:
  Home page: *http://www.gzip.org/*
  Download: *ftp://alpha.gnu.org/gnu/gzip/gzip-1.3.5.tar.gz*
  MD5 sum: `3d6c191dfd2bf307014b421c12dc8469`
- Iana-Etc (2.10) - 184 KB:
  Home page: *http://www.sethwklein.net/projects/iana-etc/*
  Download: *http://www.sethwklein.net/projects/iana-etc/downloads/iana-etc-2.10.tar.bz2*
  MD5 sum: `53dea53262b281322143c744ca60ffbb`
- Inetutils (1.4.2) - 1,019 KB:
  Home page: *http://www.gnu.org/software/inetutils/*
  Download: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.4.2.tar.gz*
  MD5 sum: `df0909a586ddac2b7a0d62795eea4206`

• IPRoute2 (2.6.16-060323) - 378 KB:
Home page: *http://linux-net.osdl.org/index.php/Iproute2*
Download: *http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.16-060323.tar.gz*
MD5 sum: `f31d4516b35bbfeaa72c762f5959e97c`

• Kbd (1.12) - 618 KB:
Download: *http://www.kernel.org/pub/linux/utils/kbd/kbd-1.12.tar.bz2*
MD5 sum: `069d1175b4891343b107a8ac2b4a39f6`

• Less (394) - 286 KB:
Home page: *http://www.greenwoodsoftware.com/less/*
Download: *http://www.greenwoodsoftware.com/less/less-394.tar.gz*
MD5 sum: `a9f072ccefa0d315b325f3e9cdbd4b97`

• LFS-Bootscripts (6.2) - 24 KB:
Download: *http://www.linuxfromscratch.org/lfs/downloads/6.2/lfs-bootscripts-6.2.tar.bz2*
MD5 sum: `45f9efc6b75c26751ddb74d1ad0276c1`

• Libtool (1.5.22) - 2,856 KB:
Home page: *http://www.gnu.org/software/libtool/*
Download: *http://ftp.gnu.org/gnu/libtool/libtool-1.5.22.tar.gz*
MD5 sum: `8e0ac9797b62ba4dcc8a2fb7936412b0`

• Linux (2.6.16.27) - 39,886 KB:
Home page: *http://www.kernel.org/*
Download: *http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.27.tar.bz2*
MD5 sum: `ebedfe5376efec483ce12c1629c7a5b1`

Nota
Il kernel di Linux è aggiornato relativamente spesso, molte volte questo è dovuto alla scoperta di vulnerabilità nella sicurezza. Dovrebbe essere utilizzata l'ultima versione del kernel 2.6.16.x, a meno che la pagina errata dica il contrario. Non utilizzare la versione 2.6.17 o successive per potenziali incompatibilità dei bootscript.

• Linux-Libc-Headers (2.6.12.0) - 2,481 KB:
Download: *http://ep09.pld-linux.org/~mmazur/linux-libc-headers/linux-libc-headers-2.6.12.0.tar.bz2*
MD5 sum: `eae2f562afe224ad50f65a6acfb4252c`

• M4 (1.4.4) - 376 KB:
Home page: *http://www.gnu.org/software/m4/*
Download: *http://ftp.gnu.org/gnu/m4/m4-1.4.4.tar.gz*
MD5 sum: `8d1d64dbecf1494690a0f3ba8db4482a`

• Make (3.80) - 900 KB:
Home page: *http://www.gnu.org/software/make/*
Download: *http://ftp.gnu.org/gnu/make/make-3.80.tar.bz2*
MD5 sum: `0bbd1df101bc0294d440471e50feca71`

- Man-DB (2.4.3) - 798 KB:
  Home page: *http://www.nongnu.org/man-db/*
  Download: *http://savannah.nongnu.org/download/man-db/man-db-2.4.3.tar.gz*
  MD5 sum: `30814a47f209f43b152659ba51fc7937`
- Man-pages (2.34) - 1,760 KB:
  Download: *http://www.kernel.org/pub/linux/docs/manpages/man-pages-2.34.tar.bz2*
  MD5 sum: `fb8d9f55fef19ea5ab899437159c9420`
- Mktemp (1.5) - 69 KB:
  Home page: *http://www.mktemp.org/*
  Download: *ftp://ftp.mktemp.org/pub/mktemp/mktemp-1.5.tar.gz*
  MD5 sum: `9a35c59502a228c6ce2be025fc6e3ff2`
- Module-Init-Tools (3.2.2) - 166 KB:
  Home page: *http://www.kerneltools.org/*
  Download: *http://www.kerneltools.org/pub/downloads/module-init-tools/module-init-tools-3.2.2.tar.bz2*
  MD5 sum: `a1ad0a09d3231673f70d631f3f5040e9`
- Ncurses (5.5) - 2,260 KB:
  Home page: *http://dickey.his.com/ncurses/*
  Download: *ftp://invisible-island.net/ncurses/ncurses-5.5.tar.gz*
  MD5 sum: `e73c1ac10b4bfc46db43b2ddfd6244ef`
- Patch (2.5.4) - 183 KB:
  Home page: *http://www.gnu.org/software/patch/*
  Download: *http://ftp.gnu.org/gnu/patch/patch-2.5.4.tar.gz*
  MD5 sum: `ee5ae84d115f051d87fcaaef3b4ae782`
- Perl (5.8.8) - 9,887 KB:
  Home page: *http://www.perl.com/*
  Download: *http://ftp.funet.fi/pub/CPAN/src/perl-5.8.8.tar.bz2*
  MD5 sum: `a377c0c67ab43fd96eeec29ce19e8382`
- Procps (3.2.6) - 273 KB:
  Home page: *http://procps.sourceforge.net/*
  Download: *http://procps.sourceforge.net/procps-3.2.6.tar.gz*
  MD5 sum: `7ce39ea27d7b3da0e8ad74dd41d06783`
- Psmisc (22.2) - 239 KB:
  Home page: *http://psmisc.sourceforge.net/*
  Download: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.2.tar.gz?download*
  MD5 sum: `77737c817a40ef2c160a7194b5b64337`
- Readline (5.1) - 1,983 KB:
  Home page: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
  Download: *http://ftp.gnu.org/gnu/readline/readline-5.1.tar.gz*
  MD5 sum: `7ee5a692db88b30ca48927a13fd60e46`

• Sed (4.1.5) - 781 KB:
Home page: *http://www.gnu.org/software/sed/*
Download: *http://ftp.gnu.org/gnu/sed/sed-4.1.5.tar.gz*
MD5 sum: `7a1cbbbb3341287308e140bd4834c3ba`

• Shadow (4.0.15) - 1,265 KB:
Download: *ftp://ftp.pld.org.pl/software/shadow/shadow-4.0.15.tar.bz2*
MD5 sum: `a0452fa989f8ba45023cc5a08136568e`

Nota
Shadow (4.0.15) potrebbe non essere più disponibile alla locazione elencata. Gli amministratori del sito della principale locazione di download occasionalmente rimuovono vecchie versioni quando ne vengono rilasciate di nuove. Una locazione di download alternativa che potrebbe avere disponibile la versione corretta è: *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Sysklogd (1.4.1) - 80 KB:
Home page: *http://www.infodrom.org/projects/sysklogd/*
Download: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.4.1.tar.gz*
MD5 sum: `d214aa40beabf7bdb0c9b3c64432c774`

• Sysvinit (2.86) - 97 KB:
Download: *ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.86.tar.gz*
MD5 sum: `7d5d61c026122ab791ac04c8a84db967`

• Tar (1.15.1) - 1,574 KB:
Home page: *http://www.gnu.org/software/tar/*
Download: *http://ftp.gnu.org/gnu/tar/tar-1.15.1.tar.bz2*
MD5 sum: `57da3c38f8e06589699548a34d5a5d07`

• Tcl (8.4.13) - 3,432 KB:
Home page: *http://tcl.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/tcl/tcl8.4.13-src.tar.gz?download*
MD5 sum: `c6b655ad5db095ee73227113220c0523`

• Texinfo (4.8) - 1,487 KB:
Home page: *http://www.gnu.org/software/texinfo/*
Download: *http://ftp.gnu.org/gnu/texinfo/texinfo-4.8.tar.bz2*
MD5 sum: `6ba369bbfe4afaa56122e65b3ee3a68c`

• Udev (096) - 190 KB:
Home page: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*
Download: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-096.tar.bz2*
MD5 sum: `f4effef7807ce1dc91ab581686ef197b`

• Udev Configuration Tarball - 4 KB:
Download: *http://www.linuxfromscratch.org/lfs/downloads/6.2/udev-config-6.2.tar.bz2*
MD5 sum: `9ff2667ab0f7bfe8182966ef690078a0`

- Util-linux (2.12r) - 1,339 KB:
  Download: *http://www.kernel.org/pub/linux/utils/util-linux/util-linux-2.12r.tar.bz2*
  MD5 sum: `af9d9e03038481fbf79ea3ac33f116f9`
- Vim (7.0) - 6,152 KB:
  Home page: *http://www.vim.org*
  Download: *ftp://ftp.vim.org/pub/vim/unix/vim-7.0.tar.bz2*
  MD5 sum: `4ca69757678272f718b1041c810d82d8`
- Vim (7.0) language files (optional) - 1,228 KB:
  Home page: *http://www.vim.org*
  Download: *ftp://ftp.vim.org/pub/vim/extra/vim-7.0-lang.tar.gz*
  MD5 sum: `6d43efaff570b5c86e76b833ea0c6a04`
- Zlib (1.2.3) - 485 KB:
  Home page: *http://www.zlib.net/*
  Download: *http://www.zlib.net/zlib-1.2.3.tar.gz*
  MD5 sum: `debc62758716a169df9f62e6ab2bc634`

Dimensione totale di questi pacchetti: circa 180 MB

3.3. Patch necessarie

In aggiunta ai pacchetti, sono richieste anche molte patch. Queste patch, oltre a correggere errori riscontrati nei pacchetti dai relativi manutentori, apportano anche piccole modifiche per rendere più facile lavorare con i pacchetti stessi. Saranno necessarie le seguenti patch per costruire un sistema LFS:

• Bash Upstream Fixes Patch - 23 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/bash-3.1-fixes-8.patch*
MD5 sum: `bc337045fa4c5839babf0306cc9df6d0`

• Bzip2 Bzgrep Security Fixes Patch - 1.2 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/bzip2-1.0.3-bzgrep_security-1.patch*
MD5 sum: `4eae50e4fd690498f23d3057dfad7066`

• Bzip2 Documentation Patch - 1.6 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/bzip2-1.0.3-install_docs-1.patch*
MD5 sum: `9e5dfbf4814b71ef986b872c9af84488`

• Coreutils Internationalization Fixes Patch - 101 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-i18n-1.patch*
MD5 sum: `3df2e6fdb1b5a5c13afedd3d3e05600f`

• Coreutils Suppress Uptime, Kill, Su Patch - 13 KB:
Download:
http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-suppress_uptime_kill_su-1.patch
MD5 sum: `227d41a6d0f13c31375153eae91e913d`

• Coreutils Uname Patch - 4.6 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-uname-1.patch*
MD5 sum: `c05b735710fbd62239588c07084852a0`

• Database (Berkeley) Upstream Fixes Patch - 3.8 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/db-4.4.20-fixes-1.patch*
MD5 sum: `32b28d1d1108dfcd837fe10c4eb0fbad`

• Diffutils Internationalization Fixes Patch - 18 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/diffutils-2.8.1-i18n-1.patch*
MD5 sum: `c8d481223db274a33b121fb8c25af9f7`

• Expect Spawn Patch - 6.8 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/expect-5.43.0-spawn-1.patch*
MD5 sum: `ef6d0d0221c571fb420afb7033b3bbba`

• Gawk Segfault Patch - 1.3 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/gawk-3.1.5-segfault_fix-1.patch*
MD5 sum: `7679530d88bf3eb56c42eb6aba342ddb`

• GCC Specs Patch - 15 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/gcc-4.0.3-specs-1.patch*
MD5 sum: `0aa7d4c6be50c3855fe812f6faabc306`

• Glibc Linux Types Patch - 1.1 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/glibc-2.3.6-linux_types-1.patch*
MD5 sum: `30ea59ae747478aa9315455543b5bb43`

- Glibc Inotify Syscall Functions Patch - 1.4 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/glibc-2.3.6-inotify-1.patch*
  MD5 sum: `94f6d26ae50a0fe6285530fdbae90bbf`
- Grep RedHat Fixes Patch - 55 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/grep-2.5.1a-redhat_fixes-2.patch*
  MD5 sum: `2c67910be2d0a54714f63ce350e6d8a6`
- Groff Debian Patch - 360 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/groff-1.18.1.1-debian_fixes-1.patch*
  MD5 sum: `a47c281afdda457ba4033498f973400d`
- GRUB Disk Geometry Patch - 28 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/grub-0.97-disk_geometry-1.patch*
  MD5 sum: `bf1594e82940e25d089feca74c6f1879`
- Gzip Security Patch - 2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/gzip-1.3.5-security_fixes-1.patch*
  MD5 sum: `f107844f01fc49446654ae4a8f8a0728`
- Inetutils GCC-4.x Fix Patch - 1.3 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/inetutils-1.4.2-gcc4_fixes-3.patch*
  MD5 sum: `5204fbc503c9fb6a8e353583818db6b9`
- Inetutils No-Server-Man-Pages Patch - 4.1 KB:
  Download:
  http://www.linuxfromscratch.org/patches/lfs/6.2/inetutils-1.4.2-no_server_man_pages-1.patch
  MD5 sum: `eb477f532bc6d26e7025fcfc4452511d`
- Kbd Backspace/Delete Fix Patch - 11 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/kbd-1.12-backspace-1.patch*
  MD5 sum: `692c88bb76906d99cc20446fadfb6499`
- Kbd GCC-4.x Fix Patch - 1.4 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/kbd-1.12-gcc4_fixes-1.patch*
  MD5 sum: `615bc1e381ab646f04d8045751ed1f69`
- Linux kernel UTF-8 Composing Patch - 11 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/linux-2.6.16.27-utf8_input-1.patch*
  MD5 sum: `d67b53e1e99c782bd28d879e11ee16c3`
- Linux Libc Headers Inotify Patch - 4.7 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/linux-libc-headers-2.6.12.0-inotify-3.patch*
  MD5 sum: `8fd71a4bd3344380bd16caf2c430fa9b`
- Mktemp Tempfile Patch - 3.5 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/mktemp-1.5-add_tempfile-3.patch*
  MD5 sum: `65d73faabe3f637ad79853b460d30a19`
- Module-init-tools Patch - 1.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/module-init-tools-3.2.2-modprobe-1.patch*
  MD5 sum: `f1e452fdf3b8d7ef60148125e390c3e8`
- Ncurses Fixes Patch - 8.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/ncurses-5.5-fixes-1.patch*

MD5 sum: 0e033185008f21578c6e4c7249f92cbb

- Perl Libc Patch - 1.1 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/perl-5.8.8-libc-2.patch*
  MD5 sum: 3bf8aef1fb6eb6110405e699e4141f99
- Readline Upstream Fixes Patch - 3.8 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/readline-5.1-fixes-3.patch*
  MD5 sum: e30963cd5c6f6a11a23344af36cfa38c
- Sysklogd 8-Bit Cleanness Patch - 0.9 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/sysklogd-1.4.1-8bit-1.patch*
  MD5 sum: cc0d9c3bd67a6b6357e42807cf06073e
- Sysklogd Fixes Patch - 27 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/sysklogd-1.4.1-fixes-1.patch*
  MD5 sum: 508104f058d1aef26b3bc8059821935f
- Tar GCC-4.x Fix Patch - 1.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-gcc4_fix_tests-1.patch*
  MD5 sum: 8e286a1394e6bcf2907f13801770a72a
- Tar Security Fixes Patch - 3.9 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-security_fixes-1.patch*
  MD5 sum: 19876e726d9cec9ce1508e3af74dc22e
- Tar Sparse Fix Patch - 0.9 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-sparse_fix-1.patch*
  MD5 sum: 9e3623f7c88d8766878ecb27c980d86a
- Texinfo Multibyte Fixes Patch - 1.5 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/texinfo-4.8-multibyte-1.patch*
  MD5 sum: 6cb5b760cfdd2dd53a0430eb572a8aaa
- Texinfo Tempfile Fix Patch - 2.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/texinfo-4.8-tempfile_fix-2.patch*
  MD5 sum: 559bda136a2ac7777ecb67511227af85
- Util-linux Cramfs Patch - 2.8 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/util-linux-2.12r-cramfs-1.patch*
  MD5 sum: 1c3f40b30e12738eb7b66a35b7374572
- Vim Upstream Fixes Patch - 42 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-fixes-7.patch*
  MD5 sum: d274219566702b0bafcb83ab4685bbde
- Vim Man Directories Patch - 4.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-mandir-1.patch*
  MD5 sum: b6426eb4192faba1e867ddd502323f5b
- Vim Spellfile Patch - 1.2 KB:
  Download: *http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-spellfile-1.patch*
  MD5 sum: 98e59e34cb6e16a8d4671247cebd64ee

Dimensione totale di queste patch: circa 775.9 KB

In aggiunta alle patch richieste in precedenza, esiste un certo numero di patch opzionali create dalla comunità LFS. Queste patch opzionali risolvono problemi minori o abilitano funzionalità che non sono abilitate per default. Si è liberi di esaminare il database delle patch che si trova su *http://www.linuxfromscratch.org/patches/* e acquisire ogni patch aggiuntiva che sia adatta alle richieste del sistema.

Capitolo 4. Ultimi preparativi

4.1. $LFS

In questo libro la variabile di ambiente `LFS` verrà usata più volte. È fondamentale che questa variabile sia sempre definita. Dovrebbe essere impostata al punto di mount scelto per la partizione LFS. Controllare che la variabile `LFS` sia definita correttamente con:

```
echo $LFS
```

Verificare che sullo schermo compaia il percorso del punto di mount della partizione LFS, che è `/mnt/lfs` se si segue l'esempio dato in precedenza. Se l'output non è corretto, si può impostare la variabile con:

```
export LFS=/mnt/lfs
```

Avere questa variabile definita è utile perché comandiy come **mkdir $LFS/tools** possono essere digitati letteralmente. La shell sostituirà automaticamente «$LFS» con «/mnt/lfs» (o con ciò che è stato impostato nella variabile) quando processerà la linea di comando.

Non dimenticare di verificare che `$LFS` sia definita qualora si lasci e si rientri nell'ambiente di lavoro corrente (come quando si fa un **su** verso `root` o un altro utente).

4.2. Creazione della directory $LFS/tools

Tutti i programmi compilati nel Capitolo 5 verranno installati sotto `$LFS/tools` per tenerli separati dai programmi compilati nel Capitolo 6. I programmi compilati qui sono tool temporanei e non saranno parte del sistema LFS finale. Tenendoli in una directory separata, potranno essere facilmente eliminati più tardi dopo il loro uso. Questo inoltre impedisce loro di finire nelle directory di produzione dell'host (è facile che succeda accidentalmente nel Capitolo 5).

Creare la directory richiesta eseguendo quanto segue come `root`:

```
mkdir -v $LFS/tools
```

Il prossimo passo è creare un link simbolico `/tools` sul sistema host. Questo punterà alla directory appena creata sulla partizione LFS. Eseguire anche questo comando come `root`:

```
ln -sv $LFS/tools /
```

Nota

Il comando precedente è corretto. Il comando **ln** ha poche variazioni sintattiche, perciò accertarsi di controllare **info coreutils ln** e `ln(1)` prima di riportare quello che si potrebbe credere un errore.

Il link simbolico creato permette di compilare la toolchain in modo che faccia sempre riferimento a `/tools` . Ciò significa che compilatore, assemblatore e linker lavoreranno sia in questo capitolo (quando si sta ancora usando qualche strumento del sistema ospite), sia nel prossimo (quando si accede con **chroot** alla partizione LFS).

4.3. Aggiunta dell'utente LFS

Quando si accede al sistema come utente `root`, fare un solo errore può danneggiare o distruggere il sistema. Perciò raccomandiamo di costruire i pacchetti in questo capitolo come utente senza privilegi. Si può usare il proprio user name, ma per rendere più facile l'impostazione di un ambiente di lavoro pulito, creare un nuovo utente chiamato `lfs` come membro di un nuovo gruppo (chiamato ugualmente `lfs`) e usare questo utente durante il processo di installazione. Come `root`, inserire i comandi seguenti per aggiungere il nuovo utente:

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

Significato delle opzioni nella linea di comando:

`-s /bin/bash`
Questo rende **bash** la shell di default per l'utente `lfs`.

`-g lfs`
Questa opzione aggiunge l'utente `lfs` al gruppo `lfs`.

`-m`
Questo crea una directory home per `lfs`.

`-k /dev/null`
Questo parametro inibisce la possibilità di copiare file da una directory modello (che per default è `/etc/skel`) cambiando la locazione di input verso il dispositivo speciale null.

`lfs`
Questo è il nome effettivo del gruppo e dell'utente creati.

Per accedere come `lfs` (contrariamente a quando ci si sposta sull'utente `lfs` quando si è connessi come `root`, il che non richiede all'utente `lfs` di avere una password), dare a `lfs` una password:

```
passwd lfs
```

Dare a `lfs` pieno accesso a `$LFS/tools` rendendo `lfs` il proprietario della directory:

```
chown -v lfs $LFS/tools
```

Se era stata creata una directory di lavoro separata come suggerito, dare all'utente `lfs` la proprietà di questa directory:

```
chown -v lfs $LFS/sources
```

Quindi, accedere con l'utente `lfs`. Questo può essere fatto usando una console virtuale, attraverso un display manager, o con il seguente comando di sostituzione utente:

```
su - lfs
```

Il «`-`» dice a **su** di avviare una shell di login invece di una shell non-login. La differenza tra questi due tipi di shell può essere trovata in dettaglio in `bash(1)` e **info bash**.

4.4. Configurazione dell'ambiente

Impostare un buon ambiente di lavoro creando due nuovi file di avvio per la shell **bash**. Mentre si è connessi come utente `lfs`, digitare il seguente comando per creare un nuovo `.bash_profile`:

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

Quando si accede come utente `lfs`, di solito la shell iniziale è una shell di *login*, che legge il file `/etc/profile` del sistema ospite (il quale probabilmente contiene alcune impostazioni e variabili d'ambiente) e quindi `.bash_profile`. Il comando **exec env -i.../bin/bash** nel file `.bash_profile` sostituisce la shell in esecuzione con una nuova con un ambiente completamente vuoto, tranne che per le variabili `HOME`, `TERM` e `PS1`. Questo garantisce che nessuna variabile d'ambiente non voluta e potenzialmente pericolosa si infiltri dal sistema ospite nell'ambiente costruito. La tecnica qui usata raggiunge l'obiettivo di assicurare un ambiente pulito.

La nuova istanza della shell è una shell *non-login*, che non legge i file `/etc/profile` o `.bash_profile`, ma legge invece il file `.bashrc`. Creare ora il file `.bashrc`:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL PATH
EOF
```

Il comando **set +h** disabilita la funzione hash della **bash**. Normalmente l'hashing è una caratteristica utile: **bash** usa una tabella hash per ricordare i percorsi completi dei file eseguibili ed evitare di cercare ripetutamente nel `PATH` per trovare lo stesso eseguibile. Tuttavia i nuovi tool dovrebbero essere utilizzati non appena vengono installati. Disattivando la funzione hash, la shell cercherà sempre nel `PATH` quando un programma sta per essere eseguito. In questo modo, la shell troverà i nuovi tool compilati in `$LFS/tools` non appena sono disponibili senza ricordare una precedente versione dello stesso programma in una posizione diversa.

Impostare la maschera di creazione file dell'utente (umask) a 022 garantisce che i nuovi file e directory creati siano scrivibili solo dal loro proprietario, ma siano leggibili ed eseguibili da chiunque (supponendo che i modi di default siano usati dalla chiamata di sistema `open(2)`, i nuovi file finiranno per avere i permessi con la modalità 644 e le directory con la modalità 755).

La variabile `LFS` dovrebbe venire impostata sul punto di mount scelto.

La variabile `LC_ALL` controlla la localizzazione di certi programmi, facendo sì che i loro messaggi seguano le convenzioni di un paese specifico. Se il sistema ospite usa una versione di Glibc più vecchia della 2.2.4, avere `LC_ALL` impostata a qualcosa di diverso da «POSIX» o «C» (durante questo capitolo) può causare problemi se si esce dall'ambiente chroot e vi si vuole rientrare successivamente. Impostare `LC_ALL` a «POSIX» o «C» (i due sono equivalenti) assicura che ogni cosa funzioni come ci si aspetta nell'ambiente chroot.

Mettendo `/tools/bin` all'inizio nel `PATH` standard, tutti i programmi installati in Capitolo 5 vengono trovati dalla shell immediatamente dopo la loro installazione. Questo, combinato con la disabilitazione dell'hashing, limita il rischio che vecchi programmi vengano usati dall'ospite quando gli stessi programmi sono disponibili nell'ambiente del capitolo 5.

Infine, per avere l'ambiente completamente preparato per costruire i tool temporanei, caricare il profilo utente appena creato:

```
source ~/.bash_profile
```

4.5. SBU

Molta gente vorrebbe sapere in anticipo quanto tempo occorre approssimativamente per compilare e installare ciascun pacchetto. Poiché si può costruire Linux from Scratch su tanti sistemi differenti, è impossibile fare una stima accurata dei tempi. Il pacchetto più grosso (Glibc) richiederà approssimativamente 20 minuti sui sistemi più veloci, ma potrebbe richiedere fino a tre giorni sui sistemi più lenti. Invece di dare tempi, verrà usata la misura Standard Build Unit (SBU).

La misura SBU funziona nel modo seguente. Il primo pacchetto da compilare in questo libro è Binutils nel Capitolo 5. Il tempo che occorre per compilare questo pacchetto è ciò a cui si farà riferimento come Standard Build Unit o SBU. Tutti gli altri tempi di compilazione saranno espressi relativamente a questo tempo.

Ad esempio, consideriamo un pacchetto il cui tempo di compilazione sia di 4,5 SBU. Questo significa che se un sistema impiega 10 minuti per compilare ed installare il primo passo della Binutils, esso impiegherà *approssimativamente* 45 minuti per costruire questo pacchetto. Fortunatamente molti tempi sono più brevi di quello di Binutils.

In generale le SBU non sono molto accurate, poiché esse dipendono da molti fattori, inclusa la versione di GCC dell'host. Si noti che su macchine basate su Symmetric Multi-Processor (SMP) gli SBU sono ancora meno accurati. Sono forniti qui per dare una stima di quanto tempo può impiegare l'installazione di un pacchetto, tuttavia i numeri possono variare anche di dozzine di minuti in certi casi.

Se si vogliono vedere i tempi attuali per macchine specifiche si raccomanda la LinuxFromScratch SBU Home Page presso *http://www.linuxfromscratch.org/~sbu/*.

4.6. Le suite di test

Molti pacchetti forniscono una suite di test. Eseguire la suite di test per un pacchetto appena costruito è una buona idea, poiché può fornire un «controllo d'integrità » per indicare che tutto sia stato compilato correttamente. Una suite di test che supera il suo insieme di verifiche normalmente garantisce che il pacchetto funzioni così come lo sviluppatore indendeva. Non garantisce, tuttavia, che il pacchetto sia totalmente privo di errori.

Alcune suite di test sono più importanti di altre. Per esempio le suite di test per i pacchetti chiave della toolchain —GCC, Binutils e Glibc— sono di grande importanza, per via del loro ruolo centrale in un sistema che funziona correttamente. Le suite di test per GCC e Glibc possono richiedere molto tempo per completarsi, specialmente sulle macchine più lente, ma sono fortemente raccomandate.

Nota

L'esperienza ha dimostrato che c'è poco da guadagnare eseguendo le suite di test nel Capitolo 5. Non si può sfuggire al fatto che il sistema ospite esercita sempre una qualche influenza sui test di quel capitolo, spesso causando fallimenti inspiegabili. Poiché i tool costruiti nel Capitolo 5 sono temporanei e in seguito scartati, non si raccomanda al lettore medio di eseguire le suite di test nel Capitolo 5. Le istruzioni per eseguire queste suite di test sono fornite a beneficio di tester e sviluppatori, ma sono strettamente facoltative.

Un problema comune quando si eseguono le suite di test per Binutils e GCC è l'esaurimento degli pseudo terminali (PTY). Questo può dare luogo a un numero altissimo di fallimenti durante il test. Tutto questo può accadere per molte ragioni, ma la causa più probabile è che il sistema ospite non abbia il file system `devpts` impostato correttamente. Questo problema è discusso con maggiori dettagli nel Capitolo 5.

A volte le suite di test dei paccheti falliranno, ma per ragioni di cui gli sviluppatori sono a conoscenza e sono state giudicate non critiche. Consultare i log situati presso *http://www.linuxfromscratch.org/lfs/build-logs/6.2/* per verificare che questi fallimenti siano normali. Questo principio si applica a tutti i test durante il libro.

Capitolo 5. Costruzione di un sistema temporaneo

5.1. Introduzione

In questo capitolo verrà compilato e installato un sistema Linux minimale. Questo sistema conterrà sufficienti strumenti da permettere di costruire il sistema LFS finale nel Capitolo 6 e consentire un ambiente di lavoro un po' più agevole per l'utente di un ambiente minimo.

La costruzione di questo sistema minimo è eseguita in due fasi. Il primo passo è la costruzione di una nuova toolchain indipendente dall'host (compilatore, assemblatore, linker, librerie e qualche altro tool). Il secondo passo usa questa toolchain per costruire gli altri tool essenziali.

I file compilati in questo capitolo verranno installati nella directory `$LFS/tools` per tenerli separati dai file installati nel prossimo capitolo e dalle directory dell'host. Dal momento che questi pacchetti compilati sono puramente temporanei, non vogliamo inquinare il nascente sistema LFS.

> **Importante**
>
> Prima di digitare le istruzioni di costruzione di un pacchetto, il pacchetto deve essere scompattato come utente `lfs`, e deve essere eseguito un **cd** nella directory creata. Le istruzioni di costruzione suppongono che si stia usando la shell **bash**.

Molti pacchetti sono patchati prima della compilazione, ma solo quando la patch è necessaria per aggirare un problema. Spesso una patch è necessaria sia in questo che nel prossimo capitolo, ma talvolta solo in uno dei due. D'altra parte, non ci si preoccupi se le istruzioni per una patch scaricata sembrano mancare. Applicando una patch, si potrebbero incontrare messaggi di avviso riguardanti *offset* o *fuzz*. Questi warning non sono nulla di cui preoccuparsi, in quanto la patch è stata applicata con successo.

Durante la compilazione di molti pacchetti, ci saranno molti warning che scorreranno sullo schermo. Sono normali, e possono essere ignorati senza problemi. Questi avvisi sono ciò che appaiono: avvisi riguardanti usi deprecati, ma non invalidati, della sintassi C o C++. Gli standard C cambiano piuttosto spesso e certi pacchetti usano ancora il vecchio standard. Questo non è un problema, ma fa apparire l'avviso.

> **Importante**
>
> Dopo aver installato ciascun pacchetto bisogna cancellare le sue directory dei sorgenti e di costruzione, tranne ove richiesto diversamente. Cancellare i sorgenti fa risparmiare spazio, ma previene anche errate configurazioni quando lo stesso pacchetto viene reinstallato in seguito.

Controllare un'ultima volta che la variabile ambiente `LFS` sia definita correttamente:

```
echo $LFS
```

Verificare che l'output mostri il percorso del punto di mount della partizione, che è `/mnt/lfs`, usando il nostro esempio.

5.2. Note tecniche sulla Toolchain

Questa sezione prova a spiegare alcuni dei dettagli tecnici e logici dietro al metodo di costruzione complessivo. Non è importante che qui si comprenda ogni cosa immediatamente. Molto di questo avrà senso quando si sarà realizzata una compilazione. Questa sezione può essere ripresa in ogni momento durante il procedimento

Lo scopo complessivo del Capitolo 5 è di fornire un ambiente provvisorio in cui si possa accedere con chroot, e dal quale si possa produrre una costruzione pulita e priva di problemi del sistema LFS nel Capitolo 6. Lungo la strada ci si separa il più possibile dal sistema host, e nel farlo si costruisce una toolchain auto-contenuta che si auto-ospita. Bisogna notare che il processo di costruzione è stato disegnato per minimizzare il rischio per i nuovi lettori e fornire un minimo valore didattico nello stesso tempo.

Importante

Prima di continuare bisogna essere certi del nome della propria piattaforma, a cui spesso si fa riferimento anche come tripletta target. Spesso la tripletta target sarà probabilmente *i686-pc-linux-gnu*. Un modo semplice per determinare la propria piattaforma è quello di eseguire lo script **config.guess** fornito con i sorgenti di molti pacchetti. Scompattare i sorgenti delle Binutils ed eseguire lo script: **`./config.guess`** e annotare l'output.

Bisogna anche essere a conoscenza del nome del linker dinamico, cui spesso si fa riferimento come dynamic loader (da non confondere con il linker standard **ld** che è parte delle Binutils). Il linker dinamico fornito da Glibc trova e carica le librerie condivise necessarie a un programma, prepara il programma all'esecuzione, e quindi lo esegue. Il nome del linker dinamico solitamente sarà `ld-linux.so.2`. Su piattaforme meno recenti, il nome potrebbe essere `ld.so.1`, e le nuove piattaforme a 64 bit potrebbero avere nomi completamente diversi. Il nome del linker dinamico della piattaforma può venire determinato guardando nella directory `/lib` sul sistema host. Un sistema per determinare il nome a colpo sicuro è di controllare un binario a caso del sistema host eseguendo: **`readelf -l <nome del binario> | grep interpreter`** e annotando l'output. Il riferimento autoritativo che copre tutte le piattaforme è nel file `shlib-versions` nella root dell'albero dei sorgenti di Glibc.

Alcuni punti tecnici su come funziona il metodo di costruzione del Capitolo 5:

- Il processo è simile, nel principio, al cross-compiling in cui i tool installati nello stesso prefisso lavorano in cooperazione, e in questo modo utilizzano una piccola «magia» GNU.
- Manipolazione attenta del percorso di ricerca delle librerie standard del linker per assicurare che i programmi siano linkati solo verso le librerie scelte.
- Manipolazione attenta del file `specs` di **gcc** per dire al compilatore quale linker dinamico verrà usato

Le binutils sono installate per prime perché il file **configure** eseguito sia da GCC che da Glibc effettua vari tipi di test sull'assemblatore e il linker per determinare quali caratteristiche del software abilitare e disabilitare. Ciò è molto più importante di quanto uno possa pensare inizialmente. Una configurazione scorretta di GCC o Glibc può portare a una toolchain corrotta, dove l'impatto di una tale corruzione potrebbe non evidenziarsi fin quasi al termine della costruzione dell'intera distribuzione. Un fallimento della suite di test normalmente evidenzierà questo errore prima di aver perso eseguito troppo lavoro aggiuntivo.

Le Binutils installano il loro assemblatore e linker in due locazioni, `/tools/bin` e `/tools/$TARGET_TRIPLET/bin`. I tool in ogni locazione sono collegati stabilmente all'altra. Un aspetto importante del linker è il suo ordine di ricerca della libreria. Informazioni dettagliate si possono

ottenere da **ld** passandogli il flag *--verbose*. Ad esempio, un **ld --verbose | grep SEARCH** illustrerà i percorsi di ricerca correnti e il loro ordine. Esso mostra quali file sono linkati da **ld** compilando un programma dummy e passando lo switch *--verbose* al linker. Ad esempio, **gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** mostrerà tutti i file aperti con successo durante la fase di link.

Il prossimo pacchetto installato è GCC. Un esempio di ciò che si può vedere durante la sua esecuzione di **configure** è:

```
checking what assembler to use...
        /tools/i686-pc-linux-gnu/bin/as
checking what linker to use... /tools/i686-pc-linux-gnu/bin/ld
```

Ciò è importante per le ragioni sopra menzionate. Dimostra anche che lo script di configurazione di GCC non cerca le directory del PATH per trovare quali tool usare. Tuttavia durante le attuali operazioni dello stesso **gcc** non vengono necessariamente usati gli stessi percorsi di ricerca. Per scoprire quale linker standard sarà usato da **gcc** eseguire: **gcc -print-prog-name=ld**.

Informazioni dettagliate possono essere ottenute da **gcc** passandogli l'opzione da linea di comando *-v* quando compila un programma dummy. Ad esempio, **gcc -v dummy.c** mostrerà informazioni dettagliate riguardanti preprocessore, compilazione, e fase di assemblaggio, includendo i percorsi **gcc** di ricerca e il loro ordine.

Il pacchetto installato successivamente è Glibc. Le considerazioni più importanti per costruire Glibc riguardano il compilatore, i tool binari e gli header del kernel. Il compilatore normalmente non è un problema perché Glibc userà sempre il **gcc** trovato in una directory del PATH. I tool binari e gli header del kernel possono essere un po' più complicati. Pertanto non si prendano rischi e si utilizzino gli switch di configurazione disponibili per forzare le selezioni corrette. Dopo 'esecuzione di **configure**, verificare i contenuti del file `config.make` nella directory `glibc-build` per tutti i dettagli importanti. Notare l'uso di *CC="gcc -B/tools/bin/"* per controllare quali tool binari sono usati e l'uso dei flag *-nostdinc* e *-isystem* per controllare il percorso di ricerca dell'include del compilatore. Queste cose evidenziano un importante aspetto del pacchetto Glibc: è molto autosufficiente nel suo motore di costruzione e generalmente non fa riferimento ai default della toolchain.

Dopo l'installazione di Glibc, apportare alcuni aggiustamenti per garantire che ricerca e collegamento prendano posto solo nel prefisso `/tools`. Installare un **ld** aggiustato, che ha un percorso di ricerca incluso in esso limitato a `/tools/lib`. Quindi correggere il file specs di **gcc** per farlo puntare al nuovo linker dinamico in `/tools/lib`. Questo ultimo passo è vitale per l'intero processo. Come citato in precedenza, un percorso fisso verso un linker dinamico è incluso in ogni eseguibile Executable and Link Format (ELF) condiviso. Ciò può essere controllato eseguendo: **readelf -l <nome del binario> | grep interpreter**. Correggendo il file specs di gcc si garantisce che ogni programma compilato da qui alla fine di questo capitolo usi il nuovo linker dinamico in `/tools/lib`.

La necessità di usare il nuovo linker dinamico è anche la ragione per cui la patch Specs è applicata per il secondo passo di GCC. Non fare questo significa far sì che gli stessi programmi GCC abbiano il nome del linker dinamico dalla directory del sistema host `/lib` incluso in essi, cosa che vanificherebbe l'oiettivo di allontanarci dall'host.

Durante il secondo passo di Binutils, si può utilizzare lo switch di configurazione *--with-lib-path* per controllare il percorso di ricerca delle librerie di **ld**. Da questo punto in poi, la toolchain principale è auto-contenuta e si auto-ospita. I rimanenti pacchetti del Capitolo 5 si costruiscono tutti con la nuova Glibc in `/tools`.

Una volta entrati nell'ambiente chroot nel Capitolo 6, il primo grosso pacchetto che si installa è Glibc, per via della sua natura auto-sufficiente cui si è accennato prima. Una volta installato Glibc in `/usr`, si realizza un rapido cambio dei default della toolchain, e quindi si procede alla vera costruzione del resto del sistema

LFS.

5.3. Binutils-2.16.1 - Passo 1

Il pacchetto Binutils contiene un linker, un assemblatore e altri tool per manipolare file oggetto.

Tempo di costruzione approssimativo: 1 SBU
Spazio necessario su disco: 189 MB

5.3.1. Installazione di Binutils

È importante che Binutils sia il primo pacchetto compilato, poiché sia Glibc che GCC eseguono diversi test sul linker e l'assemblatore disponibili per determinare quale delle loro caratteristiche abilitare.

La documentazione di Binutils raccomanda di costruire Binutils al di fuori della directory dei sorgenti, in una directory di costruzione dedicata:

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Nota

Perché i valori di SBU elencati nel resto del libro siano di qualche utilità misurare il tempo occorrente a costruire questo pacchetto dalla configurazione, fino a, e inclusa, la prima installazione. Per ottenere questo facilmente, racchiudere tre comandi in un comando **time** in questo modo: **`time { ./configure ... && make && make install; }`**.

Preparare Binutils per la compilazione:

```
../binutils-2.16.1/configure --prefix=/tools --disable-nls
```

Significato delle opzioni di configurazione:

`--prefix=/tools`
Questo dice allo script di configurazione di prepararsi a installare i programmi delle Binutils nella directory `/tools`.

`--disable-nls`
Questo disabilita l'internazionalizzazione, poiché i18n non è necessario per i tool temporanei.

Continuare con la compilazione del pacchetto:

```
make
```

La compilazione è ora completa. Normalmente ora eseguiremmo la suite di test, ma in questa fase preliminare il framework della suite di test (Tcl, Expect e DejaGNU) non è ancora al suo posto. I benefici derivanti da un'esecuzione dei test a questo punto sono minimi, poiché i programmi di questo primo passo verranno presto rimpiazzati da quelli del secondo.

Installare il pacchetto:

```
make install
```

Quindi preparare il linker per la successiva fase di «aggiustamento»:

```
make -C ld clean
make -C ld LIB_PATH=/tools/lib
cp -v ld/ld-new /tools/bin
```

Significato dei parametri di make:

`-C ld clean`
Questo dice al programma make di rimuovere tutti i file compilati nella sottodirectory `ld` .

`-C ld LIB_PATH=/tools/lib`
Questa opzione ricostruisce ogni cosa nella sottodirectory `ld`. Specificare la variabile del Makefile `LIB_PATH` nella linea di comando ci permette di sovrascrivere il valore di default e puntarlo alla locazione temporanea dei tool. Il valore di questa variabile specifica il percorso di ricerca di default delle librerie da parte del linker. Questa preparazione viena usata più avanti nel capitolo.

Dettagli su questo pacchetto si trovano nella Sezione 6.11.2, «Contenuti di Binutils.»

5.4. GCC-4.0.3 - Passo 1

Il pacchetto GCC contiene la collezione di compilatori GNU, che include i compilatori C e C++.

Tempo di costruzione approssimativo: 8.2 SBU
Spazio necessario su disco: 514 MB

5.4.1. Installazione di GCC

La documentazione di GCC raccomanda di costruire GCC al di fuori della directory dei sorgenti, in una directory di costruzione dedicata:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Preparare GCC per la compilazione:

```
../gcc-4.0.3/configure --prefix=/tools \
    --with-local-prefix=/tools --disable-nls --enable-shared \
    --enable-languages=c
```

Significato delle opzioni di configurazione:

`--with-local-prefix=/tools`
Lo scopo di questa opzione è di rimuovere `/usr/local/include` dal percorso di ricerca di include di **gcc**. Ciò non è assolutamente essenziale, tuttavia aiuta a minimizzare l'influenza del sistema host.

`--enable-shared`
Questa opzione permette la costruzione di `libgcc_s.so.1` e `libgcc_eh.a`. Avere `libgcc_eh.a` disponibile assicura che lo script configure per Glibc (il prossimo pacchetto che compiliamo) produca risultati appropriati.

`--enable-languages=c`
Questa opzione assicura che venga costruito solo il compilatore C.

Continuare con la compilazione del pacchetto:

```
make bootstrap
```

Significato delle opzioni di configurazione:

`bootstrap`
Questo target non si limita a compilare GCC, ma lo compila molte volte. Usa i programmi compilati in un primo round per compilare se stesso una seconda volta, e quindi di nuovo una terza volta. Quindi confronta la seconda e la terza compilazione per essere sicuro di potersi riprodurre perfettamente. Questo presuppone anche che sia compilato correttamente.

La compilazione è ora completa. A questo punto normalmente verrebbe eseguita la suite di test, ma, come citato in precedenza, il framework della suite di test non è ancora al suo posto. I benefici derivanti dall'esecuzione dei test a questo punto sono minimi, poiché i programmi del primo passo verranno presto rimpiazzati.

Installare il pacchetto:

```
make install
```

Come tocco finale creare un symlink. Molti programmi e script eseguono **cc** invece di **gcc**, che è usato per avere programmi generici e solitamente usabili su tutti i tipi di sistemi UNIX, dove il compilatore C GNU non è sempre installato. Eseguire **cc** lascia l'amministratore di sistema libero di decidere quale compilatore C installare.

```
ln -vs gcc /tools/bin/cc
```

Dettagli su questo pacchetto si trovano nella Sezione 6.12.2, «Contenuti di GCC.»

5.5. Linux-Libc-Headers-2.6.12.0

Il pacchetto Linux-Libc-Headers contiene gli header del kernel «sterilizzati».

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 27 MB

5.5.1. Installazione di Linux-Libc-Headers

Per anni è stata pratica comune usare i cosiddetti header del kernel «grezzi»(direttamente da un tarball del kernel) in `/usr/include`, ma nel corso degli ultimi anni c'è stata una forte presa di posizione degli sviluppatori del kernel sul fatto che queste cose non devono essere fatte. Perciò è nato il progetto Linux-Libc-Headers, progettato per mantenere una versione stabile delle Application Programming Interface (API) degli header di Linux.

Installazione dei file header:

```
cp -Rv include/asm-i386 /tools/include/asm
cp -Rv include/linux /tools/include
```

Se la propria architettura non è i386 (compatibile), aggiustare di conseguenza il primo comando.

Dettagli su questo pacchetto si trovano nella Sezione 6.7.2, «Contenuti di Linux-Libc-Headers.»

5.6. Glibc-2.3.6

Il pacchetto Glibc contiene la libreria C principale. Questa libreria fornisce tutte le routine di base per allocare memoria, cercare directory, aprire e chiudere file, leggere e scrivere file, manipolare stringhe, individuare pattern, aritmetica, e così via.

Tempo di costruzione approssimativo: 6 SBU
Spazio necessario su disco: 325 MB

5.6.1. Installazione di Glibc

La documentazione di Glibc raccomanda di costruire Glibc al di fuori della directory dei sorgenti, in una directory di costruzione dedicata:

```
mkdir -v ../glibc-build
cd ../glibc-build
```

Preparare Glibc per la compilazione:

```
../glibc-2.3.6/configure --prefix=/tools \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --with-binutils=/tools/bin \
    --without-gd --with-headers=/tools/include \
    --without-selinux
```

Significato delle opzioni di configurazione:

`--disable-profile`
Questa costruisce le librerie senza informazioni di profilo. Omettere questa opzione se è necessaria la profilazione nei tool temporanei.

`--enable-add-ons`
Questa dice a Glibc di usare l'add-on NPTL come propria libreria di threading.

`--enable-kernel=2.6.0`
Questa dice a Glibc di compilare la libreria con il supporto per i kernel Linux 2.6.x.

`--with-binutils=/tools/bin`
Sebbene non richiesto, questa opzione assicura che non ci siano errori riguardo quali programmi di Binutils usare durante la costruzione di Glibc.

`--without-gd`
Questa impedisce la costruzione del programma **memusagestat**, che si ostina a linkarsi verso le librerie dell'host (libgd, libpng, libz, ecc.).

`--with-headers=/tools/include`
Questo dice a Glibc di compilarsi verso gli header recentemente installati nella directory tools, così che sappia esattamente quali sono le caratteristiche del kernel e possa ottimizzare se stessa di conseguenza.

`--without-selinux`
Quando si costruisce da host che includono funzionalità SELinux (es. Fedora Core 3) Glibc verrà costruito con supporto per SELinux. Poiché gli strumenti dell'ambiente LFS non contengono supporto per SELinux una Glibc compilata con tale supporto non riuscirà ad operare correttamente.

Durante questo stage può apparire il seguente avviso:

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.
```

Il programma mancante o incompatibile **msgfmt** è generalmente innocuo, ma talvolta può causare problemi quando si esegue la suite di test. Questo programma **msgfmt** è parte del pacchetto Gettext che la distribuzione dovrebbe fornire. Se **msgfmt** è presente ma giudicato incompatibile, aggiornare il pacchetto Gettext del sistema o continuare senza di esso e vedere se la suite di test funziona ugualmente senza problemi.

Compilare il pacchetto:

```
make
```

La compilazione è ora completa. Come citato in precedenza, eseguire le suite di test per i tool temporanei installati in questo capitolo non è obbligatorio. Per eseguire la suite di test di Glibc (se lo si desidera), digitare il seguente comando:

```
make check
```

Per una discussione dei fallimenti dei test che sono di particolare importanza vedere Sezione 6.9, «Glibc-2.3.6.»

In questo capitolo, alcuni test possono venire condizionati negativamente da tool esistenti o problemi di ambiente del sistema in uso. I fallimenti dei test di Glibc in questo capitolo sono tipicamente non preoccupanti. La Glibc installata nel Capitolo 6 è quella che finirà per essere utilizzata, ed è questa che avrà bisogno di superare la maggior parte dei test (anche nel Capitolo 6, alcuni fallimenti avverranno comunque, ad esempio, con i math test).

Quando si incontrano fallimenti, prenderne nota, quindi continuare ridando il comando **make check**. La suite di test dovrebbe riprendere da dove si era interrotta e continuare. Questa sequenza stop-start può essere aggirata digitando un comando **make -k check**. Se si usa questa opzione, accertarsi di fare il log dell'output così da poter esaminare più tardi i fallimenti nel file log.

La fase di installazione di Glibc emetterà un avviso innocuo al termine riguardante l'assenza di `/tools/etc/ld.so.conf`. Evitare questo avviso con:

```
mkdir -v /tools/etc
touch /tools/etc/ld.so.conf
```

Installare il pacchetto:

```
make install
```

Diversi paesi e culture hanno diverse convenzioni su come comunicare. Queste convenzioni spaziano dal formato per rappresentare date e ore a questioni più complesse, come la lingua parlata. L'«internazionalizzazione» dei programmi GNU funziona con le localizzazioni.

Nota

Se non sono state eseguite le suite di test in questo capitolo (come da raccomandazione), non c'è bisogno di installare ora le localizzazioni. Le localizzazioni appropriate verranno installate nel prossimo capitolo. Per installare comunque le localizzazioni di Glibc, usare le istruzioni da

Sezione 6.9, «Glibc-2.3.6.»

Dettagli su questo pacchetto si trovano nella Sezione 6.9.4, «Contenuti di Glibc.»

5.7. Regolazione della toolchain

Ora che le librerie C provvisorie sono state installate vogliamo che tutti gli strumenti compilati nel resto di questo capitolo siano linkati verso queste librerie. Per fare questo bisogna sistemare i file specs del linker e del compilatore.

Il linker, sistemato al termine del primo passo di Binutils, deve essere rinominato per essere propriamente rintracciato ed usato. Prima, salvare il linker originale, poi sostituirlo con il linker sistemato. Verrà creato anche un link al proprio omologo in `/tools/$(gcc -dumpmachine)/bin`:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

D'ora in poi ogni cosa verrà linkata solo verso le librerie in `/tools/lib`.

Il prossimo lavoro è far puntare GCC al nuovo linker dinamico. Questo è ottenuto dal dump del file «specs» del GCC in una locazione dove GCC potrà accederci di default. Poi, una semplice sostituzione con **sed** modifica il linker dinamico che GCC utilizzerà:

```
SPECFILE=`dirname $(gcc -print-libgcc-file-name)`/specs &&
gcc -dumpspecs > $SPECFILE &&
sed 's@^/lib/ld-linux.so.2@/tools&@g' $SPECFILE > tempspecfile &&
mv -vf tempspecfile $SPECFILE &&
unset SPECFILE
```

In alternativa il file specs può essere editato a mano. Questo lo si fa sostituendo ogni occorrenza di «/lib/ld-linux.so.2» con «/tools/lib/ld-linux.so.2»

Accertarsi di ispezionare visivamente il file specs per verificare che i cambiamenti voluti siano stati apportati.

> **Importante**
>
> Se si lavora su una piattaforma nella quale il nome del linker dinamico sia diverso da `ld-linux.so.2`, sostituire «ld-linux.so.2» con il nome del linker dinamico della propria piattaforma nel comando precedente. Fare riferimento alla Sezione 5.2, «Note tecniche sulla Toolchain,», se necessario.

Durante il processo di compilazione GCC esegue uno script (**fixincludes**) che esamina il sistema alla ricerca di file header che possono aver bisogno di essere corretti (per esempio, potrebbero contenere errori di sintassi), e installa le versioni corrette in una direttory include privata. C'è la possibilità che, come risultato di questo processo, alcuni file header del sistema host abbiano trovato una strada nella directory include privata di GCC. Dal momento che il resto di questo capitolo richiede solo gli header dal GCC e Glibc, che a questo punto sono entrambi stati installati, qualsiasi header «corretto» può essere rimosso in modo sicuro. Questo aiuta a evitare qualsiasi inquinamento negli header dell'host dell'ambiente di compilazione. Eseguire i seguenti comandi per rimuovere i file header nella directory include privata di GCC :

```
GCC_INCLUDEDIR=`dirname $(gcc -print-libgcc-file-name)`/include &&
find ${GCC_INCLUDEDIR}/* -maxdepth 0 -xtype d -exec rm -rvf '{}' \; &&
rm -vf `grep -l "DO NOT EDIT THIS FILE" ${GCC_INCLUDEDIR}/*` &&
unset GCC_INCLUDEDIR
```

Attenzione

A questo punto è imperativo fermarsi ed assicurarsi che le funzioni di base (compilazione e link) della nuova toolchain funzionino come previsto. Per eseguire una verifica di integrità eseguire i seguenti comandi:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

Se funziona tutto correttamente non ci devono essere errori, e l'output dell'ultimo comando sarà del tipo:

```
[Requesting program interpreter:
    /tools/lib/ld-linux.so.2]
```

Notare che `/tools/lib` appare come prefisso del linker dinamico.

Se l'output non appare come sopra o non c'è stato proprio output, allora c'è qualcosa di sbagliato. Investigare e ripercorrere i passi per trovare dove è il problema e correggerlo. Questo problema deve essere risolto prima di continuare. Dapprima eseguire di nuovo la verifica di integrità, usando **gcc** invece di **cc**. Se questo funziona, allora manca il symlink `/tools/bin/cc`. Rivedere la Sezione 5.4, «GCC-4.0.3 - Passo 1,» e installare il symlink. In seguito assicurarsi che il `PATH` sia corretto. Questo può essere verificato eseguendo **echo $PATH** e verificando che `/tools/bin` sia in cima alla lista. Se il `PATH` è sbagliato ciò può significare che non si è connessi come utente `lfs` o che qualcosa è andato storto nella precedente Sezione 4.4, «Configurazione dell'ambiente.» Un'altra possibilità è che qualcosa possa essere andato male con il file specs modificato in precedenza. In questo caso, rifare la modifica al file specs .

Una volta che tutto va bene, togliere i file di test:

```
rm -v dummy.c a.out
```

Nota

La costruzione di TCL nella prossima sezione servirà come verifica aggiuntiva che la toolchain è stata costruita correttamente. Se la costruzione di TCL fallisce, è un'indicazione che qualcosa è andato storto con l'installazione di Binutils, GCC, o Glibc, ma non con TCL in sè

5.8. Tcl-8.4.13

Il pacchetto Tcl contiene il Tool Command Language.

Tempo di costruzione approssimativo: 0.3 SBU
Spazio necessario su disco: 24 MB

5.8.1. Installazione di Tcl

Questo pacchetto e i prossimi due (Expect e DejaGNU) sono installati per supportare l'esecuzione delle suite di test per GCC e Binutils. Installare tre pacchetti a scopo di test può sembrare eccessivo, ma è molto rassicurante, se non essenziale, sapere che i tool più importanti funzionano correttamente. Anche se le suite di test non vengono eseguite in questo capitolo (non sono obbligatorie), questi pacchetti sono necessari per eseguire le suite di test nel Capitolo 6.

Preparare Tcl per la compilazione:

```
cd unix
./configure --prefix=/tools
```

Costruire il pacchetto:

```
make
```

Per testare i risultati digitare: **`TZ=UTC make test`**. È noto che la suite di test di Tcl subisce fallimenti sotto certe condizioni del sistema in uso che non sono pienamente comprese. Pertanto fallimenti della suite di test qui non sono sorprendenti, e non sono considerati critici. Il parametro *`TZ=UTC`* imposta la time zone al Coordinated Universal Time (UTC), noto anche come Greenwich Mean Time (GMT), ma solo per la durata dell'esecuzione della suite di test. Questo assicura che i test sull'orologio vengano eseguiti correttamente. Dettagli sulla variabile d'ambiente `TZ` sono forniti nel Capitolo 7.

Installare il pacchetto:

```
make install
```

Installare gli header di Tcl. Il prossimo pacchetto, Expect, li richiede per la compilazione.

```
make install-private-headers
```

Ora fare un link simbolico necessario:

```
ln -sv tclsh8.4 /tools/bin/tclsh
```

5.8.2. Contenuti di Tcl

Programmi installati: tclsh (link a tclsh8.4) e tclsh8.4
Libreria installata: libtcl8.4.so

Brevi descrizioni

tclsh8.4	La shell di comando Tcl
tclsh	Un link a tclsh8.4
`libtcl8.4.so`	La libreria Tcl

5.9. Expect-5.43.0

Il pacchetto Expect contiene un programma che permette il dialogo con altri programmi interattivi.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 4 MB

5.9.1. Installazione di Expect

Prima correggere un bug che può portare a falsi fallimenti durante l'esecuzione della suite di test di GCC:

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

Preparare Expect per la compilazione:

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include --with-x=no
```

Significato delle opzioni di configurazione:

`--with-tcl=/tools/lib`
Questa assicura che lo script configure trovi l'installazione di Tcl nella locazione dei tool temporanei e non vada a trovarne una esistente sul sistema host.

`--with-tclinclude=/tools/include`
Questo dice esplicitamente a Expect dove trovare gli header interni di Tcl. L'uso di questa opzione evita condizioni in cui **configure** fallisce perché non ha rilevato automaticamente la locazione della directory dei sorgenti di Tcl.

`--with-x=no`
Questa dice allo script configure di non cercare Tk (il componente GUI di Tcl) o le librerie dell'X Window System, ciascuno dei quali può risiedere sul sistema host, ma non esiste nell'ambiente temporaneo.

Costruire il pacchetto:

```
make
```

Per testare i risultati digitare: **`make test`**. Notare che la suite di test di Expect è nota per sperimentare fallimenti sotto certe condizioni del sistema in uso, le quali non sono sotto controllo. Pertanto fallimenti della suite di test qui non sono sorprendenti, e non sono considerati critici.

Installare il pacchetto:

```
make SCRIPTS="" install
```

Significato dei parametri di make:

`SCRIPTS=""`
Questo previene l'installazione degli script supplementari di expect, che non sono necessari.

5.9.2. Contenuti di Expect

Programma installato: expect
Libreria installata: libexpect-5.43.a

Brevi descrizioni

expect	Comunica con altri programmi interattivi conformemente a uno script
`libexpect-5.43.a`	Contiene funzioni che permettono a Expect di essere usato come una estensione di Tcl o di essere usato direttamente da C o C++ (senza Tcl)

5.10. DejaGNU-1.4.4

Il pacchetto DejaGNU contiene un framework per testare altri programmi.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 6.2 MB

5.10.1. Installazione di DejaGNU

Preparare DejaGNU per la compilazione:

```
./configure --prefix=/tools
```

Costruire e installare il pacchetto:

```
make install
```

Per testare i risultati, digitare: **`make check`**.

5.10.2. Contenuti di DejaGNU

Programma installato: runtest

Breve descrizione

runtest Uno script wrapper che individua la shell **expect** appropriata, e quindi esegue DejaGNU

5.11. GCC-4.0.3 - Passo 2

Il pacchetto GCC contiene la collezione di compilatori GNU, che include i compilatori C e C++.

Tempo di costruzione approssimativo: 4.2 SBU
Spazio necessario su disco: 443 MB

5.11.1. Reinstallazione di GCC

I tool richiesti per testare GCC e Binutils (Tcl, Expect e DejaGNU) sono ora installati. GCC e Binutils possono ora essere ricostruiti, linkandoli verso la nuova Glibc e testandoli appropriatamente (se si esegue la suite di test in questo capitolo). Prendere nota che queste suite di test sono altamente dipendenti da PTY funzionanti correttamente, che sono forniti dal sistema in uso. I PTY solitamente sono implementati attraverso il file system `devpts`. Verificare che il sistema in uso sia impostato correttamente a questo riguardo eseguendo un test veloce:

```
expect -c "spawn ls"
```

La risposta può essere:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Se viene ricevuto il precedente messaggio il sistema in uso non ha i suoi PTY impostati correttamente. In questo caso non c'è modo di eseguire le suite di test per GCC e Binutils fino a quando questo problema non viene risolto. Consultare la FAQ di LFS su *http://www.linuxfromscratch.org//lfs/faq.html#no-ptys* per ulteriori informazioni su come far funzionare i PTY.

Come spiegato precedentemente in Sezione 5.7, «Regolazione della toolchain», in normali circostanze lo script GCC **fixincludes** viene eseguito per correggere potenziali corruzioni di file header. Siccome a questo punto sono giá stati installati GCC-4.0.3 e Glibc-2.3.6 e i loro rispettivi file header si sà che non richiedono correzioni, lo script **fixincludes** non è richiesto. Come detto precedentemente, lo script può in effetti inquinare l'ambiente di compilazione installando header corretti dal sistema host nella directory include privata di GCC. L'esecuzione dello script **fixincludes** può essere eliminata immettendo i seguenti comandi:

```
cp -v gcc/Makefile.in{,.orig} &&
sed 's@\./fixinc\.sh@-c true@' gcc/Makefile.in.orig > gcc/Makefile.in
```

La compilazione con target bootstrap eseguita in Sezione 5.4, «GCC-4.0.3 - Passo 1» ha costruito GCC con il flag di compilazione `-fomit-frame-pointer`. La compilazione senza target bootstrap omette questo flag di default, così eseguire il seguente **sed** per usarlo al fine di assicurare una costruzione uniforme del compilatore.

```
cp -v gcc/Makefile.in{,.tmp} &&
sed 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in
```

Applicare la seguente patch per modificare la locazione del linker dinamico di default del GCC (tipicamente `ld-linux.so.2`):

```
patch -Np1 -i ../gcc-4.0.3-specs-1.patch
```

La patch suddetta rimuove anche `/usr/include` dal path di ricerca degli include dui GCC. Applicare la patch adesso piuttosto che sistemare il file specs dopo l'installazione assicura che sia utilizzato il nuovo linker dinamico durante l'effettiva compilazione di GCC. Ossia, tutti i binari creati durante la compilazione punteranno alla nuova Glibc.

> **Importante**
>
> Le precedenti patch sono critiche per assicurare il successo della costruzione complessiva. Non dimenticare di applicarle.

Creare di nuovo una directory separata di costruzione:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Prima di avviare la costruzione di GCC, ricordare di disallocare ogni variabile ambiente che sovrascriva i flag di ottimizzazione di default.

Preparare GCC per la compilazione:

```
../gcc-4.0.3/configure --prefix=/tools \
--with-local-prefix=/tools --enable-clocale=gnu \
--enable-shared --enable-threads=posix \
--enable-__cxa_atexit --enable-languages=c,c++ \
--disable-libstdcxx-pch
```

Significato delle nuove opzioni di configurazione:

`--enable-clocale=gnu`
Questa opzione assicura che venga selezionato il corretto modello locale per le librerie C++ in tutte le circostanze. Se lo script configure trova la localizzazione *de_DE* installata, selezionerà il modello gnu corretto di localizzazione. Tuttavia se la localizzazione *de_DE* non è installata, c'è il rischio di costruire delle librerie C++ incompatibili con le Application Binary Interface (ABI), poiché potrebbe venire selezionato il modello scorretto di localizzazione generica.

`--enable-threads=posix`
Questa abilita la gestione dell'eccezione C++ per il codice multi-threaded.

`--enable-__cxa_atexit`
Questa opzione permette l'uso di *__cxa_atexit*, piuttosto che *atexit* per registrare i distruttori C++ per oggetti statici e globali. Questa opzione essenzialmente è per una gestione dei distruttori pienamente conforme agli standard. Incide anche sull'ABI C++, pertanto il risultato saranno librerie condivise C++ e programmi C++ interoperabili con altre distribuzioni Linux.

`--enable-languages=c,c++`
Questa opzione assicura che entrambi i compilatori C e C++ siano costruiti.

`--disable-libstdcxx-pch`
Non costruisce l'header precompilato (PCH) per `libstdc++`. Prende un sacco di spazio, e non ce ne facciamo nulla.

Compilare il pacchetto:

```
make
```

Non c'è bisogno di usare il target *bootstrap* ora, poiché il compilatore usato per compilare GCC è stato costruito esattamente a partire dalla stessa versione dei sorgenti GCC usati in precedenza.

La compilazione è ora completa. Come citato precedentemente eseguire le suite di test per i tool temporanei compilati in questo capitolo non è obbligatorio. Per eseguire comunque la suite di test di GCC usare il seguente comando:

```
make -k check
```

Il flag *-k* è usato per far sì che la suite di test funzioni fino al completamento e non si fermi al primo fallimento. La suite di test di GCC è molto completa ed è praticamente sicuro che generi qualche fallimento.

Per una discussione sui fallimenti dei test di particolare importanza, prego vedere Sezione 6.12, «GCC-4.0.3.»

Installare il pacchetto:

```
make install
```

Attenzione

A questo punto è imperativo fermarsi ed assicurarsi che le funzioni di base (compilazione e link) della nuova toolchain funzionino come previsto. Per eseguire una verifica di integrità eseguire i seguenti comandi:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

Se funziona tutto correttamente non ci devono essere errori, e l'output dell'ultimo comando sarà del tipo:

```
[Requesting program interpreter:
    /tools/lib/ld-linux.so.2]
```

Notare che `/tools/lib` appare come prefisso del linker dinamico.

Se l'output non appare come sopra o non c'è stato proprio output, allora c'è qualcosa di sbagliato. Investigare e ripercorrere i passi per trovare dove è il problema e correggerlo. Questo problema deve essere risolto prima di continuare. Dapprima eseguire di nuovo la verifica di integrità, usando **gcc** invece di **cc**. Se questo funziona, allora manca il symlink `/tools/bin/cc`. Rivedere la Sezione 5.4, «GCC-4.0.3 - Passo 1,» e installare il symlink. In seguito assicurarsi che il `PATH` sia corretto. Questo può essere verificato eseguendo **echo $PATH** e verificando che `/tools/bin` sia in cima alla lista. Se il `PATH` è sbagliato ciò può significare che non si è connessi come utente `lfs` o che qualcosa è andato storto nella precedente Sezione 4.4, «Configurazione dell'ambiente.» Un'altra possibilità è che qualcosa possa essere andato male con il file specs modificato in precedenza. In questo caso, rifare la modifica al file specs .

Una volta che tutto va bene, togliere i file di test:

```
rm -v dummy.c a.out
```

Dettagli su questo pacchetto si trovano nella Sezione 6.12.2, «Contenuti di GCC.»

5.12. Binutils-2.16.1 - Passo 2

Il pacchetto Binutils contiene un linker, un assemblatore e altri tool per manipolare file oggetto.

Tempo di costruzione approssimativo: 1.1 SBU
Spazio necessario su disco: 154 MB

5.12.1. Re-installazione di Binutils

Creare nuovamente una directory di costruzione separata:

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Preparare Binutils per la compilazione:

```
../binutils-2.16.1/configure --prefix=/tools \
    --disable-nls --with-lib-path=/tools/lib
```

Significato delle nuove opzioni di configurazione:

`--with-lib-path=/tools/lib`
Questa dice allo script configure di specificare il percorso di ricerca della libreria durante la compilazione delle Binutils, che significa passare `/tools/lib` al linker. Questo impedisce che il linker cerchi nelle directory delle librerie sull'host.

Compilare il pacchetto:

```
make
```

La compilazione è ora completa. Come discusso in precedenza, eseguire la suite di test non è obbligatorio per i tool temporanei in questo capitolo. Per eseguire comunque la suite di test delle Binutils, digitare il seguente comando:

```
make check
```

Installare il pacchetto:

```
make install
```

Ora preparare il linker per la fase di «risistemazione» nel prossimo capitolo:

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin
```

Dettagli su questo pacchetto si trovano nella Sezione 6.11.2, «Contenuti di Binutils.»

5.13. Ncurses-5.5

Il pacchetto Ncurses contiene librerie per la gestione indipendente dal terminale di schermi caratteri.

Tempo di costruzione approssimativo: 0.7 SBU
Spazio necessario su disco: 30 MB

5.13.1. Installazione di Ncurses

Preparare Ncurses per la compilazione:

```
./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

Significato delle opzioni di configurazione:

`--without-ada`
Questo assicura che Ncurses non costruisca il supporto per il compilatore Ada, che potrebbe essere presente sull'host, ma non sarebbe disponibile una volta entrati nell'ambiente **chroot**.

`--enable-overwrite`
Questa dice a Ncurses di installare i suoi file header in `/tools/include`, invece che in `/tools/include/ncurses`, per garantire che altri pacchetti possano trovare con successo gli header di Ncurses.

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.18.2, «Contenuti di Ncurses.»

5.14. Bash-3.1

Il pacchetto Bash contiene la Bourne-Again SHell.

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 22 MB

5.14.1. Installazione di Bash

Gli sviluppatori originari hanno corretto diversi problemi dal rilascio iniziale di Bash-3.1. Applicare queste correzioni:

```
patch -Np1 -i ../bash-3.1-fixes-8.patch
```

Preparare Bash per la compilazione:

```
./configure --prefix=/tools --without-bash-malloc
```

Significato delle opzioni di configurazione:

`--without-bash-malloc`
Questa opzione disattiva l'uso della funzione di allocazione di memoria di Bash (`malloc`) che è nota per causare errori di segmentation fault. Disattivando questa opzione, Bash userà le funzioni `malloc` dalle Glibc, che sono più stabili.

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make tests`**.

Installare il pacchetto:

```
make install
```

Fare un link per i programmi che usano **sh** come shell:

```
ln -vs bash /tools/bin/sh
```

Dettagli su questo pacchetto si trovano nella Sezione 6.27.2, «Contenuti di Bash.»

5.15. Bzip2-1.0.3

Il pacchetto Bzip2 contiene programmi per comprimere e decomprimere file. La compressione di file di testo con **bzip2** raggiunge una migliore percentuale di compressione rispetto al tradizionale **gzip**.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 4.2 MB

5.15.1. Installazione di Bzip2

Il pacchetto Bzip2 non contiene uno script **configure**. Compilarlo e testarlo con:

```
make
```

Installare il pacchetto:

```
make PREFIX=/tools install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.28.2, «Contenuti di Bzip2.»

5.16. Coreutils-5.96

Il pacchetto Coreutils contiene utilità per visualizzare e impostare le caratteristiche di base del sistema.

Tempo di costruzione approssimativo: 0.6 SBU
Spazio necessario su disco: 56.1 MB

5.16.1. Installazione di Coreutils

Preparare Coreutils per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make RUN_EXPENSIVE_TESTS=yes check`**. Il parametro *`RUN_EXPENSIVE_TESTS=yes`* dice alla test suite di eseguire numerosi test aggiuntivi che sono considerati relativamente dispendiosi (in termini di potenza CPU e uso della memoria) su alcune piattaforme, ma generalmente non sono un problema su Linux.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.14.2, «Contenuti di Coreutils.»

5.17. Diffutils-2.8.1

Il pacchetto Diffutils contiene programmi che mostrano le differenze tra file o directory.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 6.2 MB

5.17.1. Installazione di Diffutils

Preparare Diffutils per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.29.2, «Contenuti di Diffutils.»

5.18. Findutils-4.2.27

Il pacchetto Findutils contiene programmi per trovare file. Questi programmi sono fatti per cercare ricorsivamente attraverso un albero di directory e per creare, mantenere e cercare in un database (spesso più velocemente della ricerca ricorsiva, ma inapplicabile se il database non è stato aggiornato recentemente).

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 12 MB

5.18.1. Installazione di Findutils

Preparare Findutils per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.32.2, «Contenuti di Findutils.»

5.19. Gawk-3.1.5

Il pacchetto Gawk contiene programmi per la manipolazione dei file di testo.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 18.2 MB

5.19.1. Installazione di Gawk

Preparare Gawk per la compilazione:

```
./configure --prefix=/tools
```

A causa di un bug nello script **configure**, Gawk fallisce la rilevazione di certi aspetti del supporto locale in Glibc. Questo bug porta, per esempio, al fallimento della testsuite di Gettext. Risolvere il problema aggiungendo la definizione della macro mancante nel `config.h`:

```
cat >>config.h <<"EOF"
#define HAVE_LANGINFO_CODESET 1
#define HAVE_LC_MESSAGES 1
EOF
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.35.2, «Contenuti di Gawk.»

5.20. Gettext-0.14.5

Il pacchetto Gettext contiene utilità per l'internazionalizzazione e la localizzazione. Questo permette ai programmi di essere compilati con il NLS (Native Language Support), abilitandoli ad emettere messaggi nel linguaggio nativo dell'utente.

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 43 MB

5.20.1. Installazione di Gettext

Per il provvisorio insieme di tool, occorre soltanto compilare e installare un binario da Gettext.

Preparare Gettext per la compilazione:

```
cd gettext-tools
./configure --prefix=/tools --disable-shared
```

Significato delle opzioni di configurazione:

`--disable-shared`
Al momento non occorre installare nessuna delle librerie condivise di Gettext, perciò non c'è bisogno di compilarle.

Compilare il pacchetto:

```
make -C lib
make -C src msgfmt
```

Siccome è stato compilato soltanto un binario, non è possibile eseguire la testsuite senza compilare le librerie addizionali di supporto dal pacchetto Gettext. È perciò sconsigliato tentare di eseguire la testsuite in questa fase.

Installare il binario **msgfmt**:

```
cp -v src/msgfmt /tools/bin
```

Dettagli su questo pacchetto si trovano nella Sezione 6.36.2, «Contenuti di Gettext.»

5.21. Grep-2.5.1a

Il pacchetto Grep contiene programmi per la ricerca nei file.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 4.8 MB

5.21.1. Installazione di Grep

Preparare Grep per la compilazione:

```
./configure --prefix=/tools \
    --disable-perl-regexp
```

Significato delle opzioni di configurazione:

`--disable-perl-regexp`
Questa fa sì che il programma **grep** non venga linkato verso una libreria Perl Compatible Regular Expression (PCRE) che può essere presente sul sistema in uso e non sarebbe disponibile una volta entrati nell'ambiente **chroot**.

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.37.2, «Contenuti di Grep.»

5.22. Gzip-1.3.5

Il pacchetto Gzip contiene programmi per compattare e scompattare i file.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 2.2 MB

5.22.1. Installazione di Gzip

Preparare Gzip per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.39.2, «Contenuti di Gzip.»

5.23. M4-1.4.4

Il pacchetto M4 contiene un processore macro.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 3 MB

5.23.1. Installazione di M4

Preparare M4 per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.16.2, «Contenuti di M4.»

5.24. Make-3.80

Il pacchetto Make contiene un programma per compilare pacchetti.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 7.8 MB

5.24.1. Installazione di Make

Preparare Make per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.44.2, «Contenuti di Make.»

5.25. Patch-2.5.4

Il pacchetto Patch contiene un programma per modificare o creare file applicando un file «patch» tipicamente creato dal programma **diff**.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 1.6 MB

5.25.1. Installazione di Patch

Preparare Patch per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.48.2, «Contenuti di Patch.»

5.26. Perl-5.8.8

Il pacchetto Perl contiene il Practical Extraction and Report Language

Tempo di costruzione approssimativo: 0.7 SBU
Spazio necessario su disco: 84 MB

5.26.1. Installazione di Perl

Prima adattare alcuni path alla libreria C inclusi nel codice applicando la seguente patch:

```
patch -Np1 -i ../perl-5.8.8-libc-2.patch
```

Preparare Perl per la compilazione (assicurarsi di riportare correttamente la parte 'Data/Dumper Fcntl IO POSIX' del comando, sono tutte lettere):

```
./configure.gnu --prefix=/tools -Dstatic_ext='Data/Dumper Fcntl IO POSIX'
```

Significato delle opzioni di configurazione:

`-Dstatic_ext='Data/Dumper Fcntl IO POSIX'`
Questa dice a Perl di costruire l'insieme minimo di estensioni statiche necessarie per installare e testare i pacchetti Coreutils e Glibc nel prossimo capitolo.

Solo alcune delle utilità contenute in questo pacchetto devono essere costruite:

```
make perl utilities
```

Sebbene Perl abbia una suite di test, non si raccomanda la sua esecuzione a questo punto. È stata costruita solo una parte di Perl e eseguire **make test** ora farebbe sì che anche il resto di Perl venga costruito, che in questa fase è inutile. La suite di test può essere eseguita nel prossimo capitolo se lo si desidera.

Installare questi tool e le loro librerie:

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.8.8
cp -Rv lib/* /tools/lib/perl5/5.8.8
```

Dettagli su questo pacchetto si trovano nella Sezione 6.22.2, «Contenuti di Perl.»

5.27. Sed-4.1.5

Il pacchetto Sed contiene uno stream editor.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 6.1 MB

5.27.1. Installazione di Sed

Preparare Sed per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.20.2, «Contenuti di Sed.»

5.28. Tar-1.15.1

Il pacchetto Tar contiene un programma per l'archiviazione.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 13.7 MB

5.28.1. Installazione di Tar

Se si desidera eseguire la suite di test, applicare la seguente patch per correggere alcuni problemi con GCC-4.0.3:

```
patch -Np1 -i ../tar-1.15.1-gcc4_fix_tests-1.patch
```

Preparare Tar per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.53.2, «Contenuti di Tar.»

5.29. Texinfo-4.8

Il pacchetto Texinfo contiene programmi per leggere, scrivere e convertire pagine info.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 16.3 MB

5.29.1. Installazione di Texinfo

Preparare Texinfo per la compilazione:

```
./configure --prefix=/tools
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Dettagli su questo pacchetto si trovano nella Sezione 6.54.2, «Contenuti di Texinfo.»

5.30. Util-linux-2.12r

Il pacchetto Util-linux contiene una serie di programmi di utilità. Fra di loro ci sono utilità per gestire i file system, le console, le partizioni e i messaggi.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 8.9 MB

5.30.1. Installazione di Util-linux

Util-linux non usa gli header e librerie appena installati nella directory `/tools`. Questo viene corretto alterando lo script di configurazione:

```
sed -i 's@/usr/include@/tools/include@g' configure
```

Preparare Util-linux per la compilazione:

```
./configure
```

Compilare alcune routine di supporto:

```
make -C lib
```

Solo alcune delle utilità contenute in questo pacchetto devono essere costruite:

```
make -C mount mount umount
make -C text-utils more
```

Questo pacchetto non è provvisto di una suite di test.

Copiare questi programmi nella directory dei tool temporanea:

```
cp mount/{,u}mount text-utils/more /tools/bin
```

Dettagli su questo pacchetto si trovano nella Sezione 6.56.3, «Contenuti di Util-linux.»

5.31. Eseguire lo strip

I passi in questa sezione sono facoltativi, ma se la partizione LFS è piuttosto piccola è bene sapere che si possono rimuovere alcune cose non necessarie. Gli eseguibili e le librerie costruiti poco fa contengono circa 130 MB di simboli di debug inutili. Rimuovere questi simboli con:

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
```

L'ultimo dei comandi precedenti salterà circa venti file, affermando che non riconosce il loro formato di file. Molti di questi sono script invece che binari.

Si faccia attenzione a *non* usare *`--strip-unneeded`* sulle librerie. Quelle statiche verrebbero distrutte e bisognerebbe ricostruire di nuovo tutti i pacchetti dell'albero toolchain.

Per risparmiare altri 30 MB rimuovere la documentazione:

```
rm -rf /tools/{info,man}
```

Ora ci saranno almeno 850 MB di spazio libero nel `$LFS` che possono essere usati per costruire e installare Glibc nella prossima fase. Se è possibile costruire e installare Glibc, si può costruire e installare anche il resto.

5.32. Cambio del proprietario

Nota

Per il resto di questo libro i comandi devono essere eseguiti mentre si è autenticati come utente `root` e non più come utente `lfs`. Inoltre controllare attentamente che `$LFS` sia impostata nell'ambiente di `root`.

In questo momento la directory `$LFS/tools` appartiene all'utente `lfs`, un utente che esiste solo sul sistema host. Se si tenesse la directory `$LFS/tools` così come è i file apparterrebbero a un ID utente senza il corrispondente account. Questo è pericoloso, poichè un utente creato successivamente potrebbe avere questo stesso ID utente e diverrebbe il proprietario della directory `$LFS/tools` e di tutti i file in essa contenuti, il che esporrebbe questi file a possibili manipolazioni.

Per evitare questa eventualità si può aggiungere l'utente `lfs` al nuovo sistema LFS più tardi, quando si creerà il file `/etc/passwd`, preoccupandosi di assegnare gli stessi ID utente e gruppo come nel proprio sistema. Meglio ancora, modificare il proprietario della directory `$LFS/tools` all'utente `root` eseguendo il seguente comando:

```
chown -R root:root $LFS/tools
```

Sebbene la directory `$LFS/tools` possa essere cancellata una volta che il sistema LFS è completato, essa può essere conservata per compilare sistemi LFS aggiuntivi *della stessa versione del libro*. Come salvare al meglio `$LFS/tools` è questione di gusti personali, ed è lasciato come esercizio per il lettore.

Parte III. Costruzione del sistema LFS

Capitolo 6. Installazione del software di sistema di base

6.1. Introduzione

In questo capitolo entriamo nella sezione di costruzione e iniziamo a creare sul serio il nostro sistema LFS. Questo significa che accediamo con chroot nel nostro mini sistema Linux provvisorio, facciamo pochi ultimi preparativi e quindi iniziamo a installare i pacchetti.

L'installazione di questo software è chiara. Anche se in molti casi le istruzioni di installazione avrebbero potuto essere più corte e generiche, abbiamo deciso di fornire le istruzioni complete per ogni pacchetto, in modo da rendere minime le possibilitÃ di sbagliare. La chiave per imparare cosa fa funzionare un sistema Linux è conoscere per cosa è utilizzato ciascun pacchetto e perché l'utente (o il sistema) ne ha bisogno. Per ogni pacchetto installato viene dato un sommario dei suoi contenuti, seguito da descrizioni concise di ciascun programma e libreria installati.

Se si intende utilizzare ottimizzazioni di compilazione, rivedere i suggerimenti sull'ottimizzazione in *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*. Le ottimizzazioni di compilazione possono far sì che un programma giri un po' più veloce, ma potrebbero anche causare difficoltà di compilazione e problemi quando il programma è in funzione. Se un pacchetto si rifiuta di compilarsi usando l'ottimizzazione, provare a compilarlo senza ottimizzazione e vedere se questo risolve il problema. Anche se il pacchetto si compila usando l'ottimizzazione, c'à il rischio che non sia stato compilato correttamente a causa delle complesse interazioni tra il codice e i tool di costruzione. Inoltre notare che le opzioni `-march` e `-mtune` possono causare problemi con i pacchetti toolchain (Binutils, GCC and Glibc). I piccoli guadagni potenziali raggiunti usando le ottimizzazioni del compilatore sono spesso superati dai rischi. Consigliamo coloro che costruiscono LFS per la prima volta di costruirlo senza ottimizzazioni personalizzate. Il sistema che si otterrà sarà ugualmente molto veloce e stabile nello stesso tempo.

L'ordine in cui i pacchetti sono installati in questo capitolo deve essere seguito con precisione, per assicurare che nessun programma accidentalmente si trovi cablato dentro un percorso che fa riferimento a `/tools`. Per la stessa ragione non compilare pacchetti in parallelo. La compilazione in parallelo può fare risparmiare tempo (specialmente su macchine con due CPU), ma potrebbe produrre un programma contenente percorsi fissi verso `/tools` , cosa che farà arrestare il programma quando questa directory verrà rimossa.

Prima delle istruzioni di installazione ciascuna pagina di installazione dà informazioni sul pacchetto: una descrizione concisa di ciò che contiene, quanto tempo occorrerà approssimativamente per costruirlo e quanto spazio su disco è necessario durante questo processo di costruzione. Dopo le istruzioni di installazione segue una lista di programmi e librerie che il pacchetto installa (insieme a brevi descrizioni degli stessi).

6.2. Preparazione dei file system virtuali del kernel

Diversi file system esportati dal kernel sono usati per comunicare a e dal kernel stesso. Questi file system sono virtuali nel senso che per essi non viene utilizzato spazio su disco. Il contenuto del file system risiede in memoria.

Cominciare creando directory sulle quali verranno montati i file system:

```
mkdir -pv $LFS/{dev,proc,sys}
```

6.2.1. Creazione dei device nodes iniziali

Quando il kernel esegue il boot del sistema, richiede solo la presenza di alcuni device nodes, in particolare i device `console` e `null`. I device nodes saranno creati sul disco fisso in modo da essere disponibili prima che **udevd** sia stato avviato e anche quando Linux è avviato con *`init=/bin/bash`*. Creare i device eseguendo i seguenti comandi:

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

6.2.2. Montaggio e popolamento di /dev

Il metodo raccomandato per popolare la directory `/dev` con i devices consiste nel montare un file system virtuale (ad esempio `tmpfs`) nella directory `/dev` e permettere ai devices di essere creati dinamicamente su questo file system virtuale mano a mano che essi sono individuati o utilizzati. Generalmente questo viene fatto durante il processo di boot da Udev. Pioché questo nuovo sistema non ha ancora Udev e non è stato ancora avviato, è necessario montare e popolare `/dev` manualmente. Ciò si ottiene con un bind mount della directory `/dev` del sistema ospite. Un bind mount è un tipo speciale di montaggio, che permette di creare un'immagine di una directory o di un punto di mount in qualche altra locazione. Usare il seguente comando per ottenere questo:

```
mount --bind /dev $LFS/dev
```

6.2.3. Montaggio dei file system virtuali del kernel

Ora montare i rimanenti file system virtuali del kernel:

```
mount -vt devpts devpts $LFS/dev/pts
mount -vt tmpfs shm $LFS/dev/shm
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

6.3. Gestione dei pacchetti

La gestione dei pacchetti è un'aggiunta richiesta spesso al libro LFS. Un gestore di pacchetti permette di tracciare l'installazione dei file, rendendo semplice la rimozione e l'aggiornamento dei pacchetti. Prima che ci si meravigli, NO—questa sezione non tratterà, né raccomanderà, nessun particolare gestore di pacchetti. Ciò che offre è una collezione delle più popolari tecniche e di come esse funzionino. Il gestore di pacchetti perfetto per il lettore potrebbe trovarsi tra queste tecniche, oppure potrebbe essere una combinazione di due o più di esse. Questa sezione accenna brevemente ai problemi che possono sorgere quando si aggiornano i pacchetti.

Alcune ragioni del perché in LFS o BLFS non sia menzionato nessun gestore di pacchetti sono:

- La trattazione della gestione dei pacchetti distoglie l'attenzione dallo scopo di questo libro—insegnare come sia costruito un sistema Linux.
- Ci sono molteplici soluzioni per la gestione dei pacchetti, ognuna con i suoi pregi e difetti. Inserirne una che soddisfi tutti i lettori è difficile.

Ci sono alcuni hint scritti sull'argomento della gestione dei pacchetti. Visitare *Hints subproject* e vedere se uno di essi soddisfa le proprie necessità.

6.3.1. Problemi nell'aggiornamento

Un gestore di pacchetti rende semplice l'aggiornamento a versioni più recenti quando vengono rilasciate. Generalmente le istruzioni dei libri LFS e BLFS possono essere usate per aggiornare a varsioni più recenti. Qui sono elencati alcuni punti di cui si dovrebbe essere consapevoli quando si aggiornano i pacchetti, specialmente su un sistema in funzione.

- Se uno dei pacchetti toolchain (Glibc, GCC o Binutils) ha bisogno di essere aggiornato ad una versione minore più recente, è più sicuro ricostruire LFS. Sebbene si *potrebbe* essere capaci di ottenere tutti i pacchetti nel loro ordine di dipendenze con una ricostruzione, non la raccomandiamo. Per esempio, se glibc-2.2.x ha bisogno di essere aggiornato a glibc-2.3.x, è più sicuro ricostruire. Per aggiornamenti di micro versione, una semplice reinstallazione di solito funziona, ma non è garantita. Per esempio, aggiornare da glibc-2.3.4 a glibc-2.3.5 di solito non causa nessun problema.
- Se un pacchetto che contiene una libreria condivisa è aggiornato e se il nome della libreria cambia, allora tutti i pacchetti con un link dinamico alla libreria devono essere ricompilati per creare il link alla libreria più recente. (Notare che non c'è correlazione tra la versione di un pacchetto e il nome della libreria). Per esempio, si consideri un pacchetto foo-1.2.3 che installa una libreria condivisa di nome `libfoo.so.1`. Supponiamo che si aggiorni il pacchetto ad una nuova versione foo-1.2.4 che installa una libreria condivisa di nome `libfoo.so.2`. In questo caso, tutti i pacchetti che hanno un link dinamico alla `libfoo.so.1` devono essere ricompilati per creare il link a `libfoo.so.2`. Notare che non si dovrebbero rimuovere le librerie precedenti finché i pacchetti dipendenti non siano stati ricompilati.

6.3.2. Tecniche di gestione dei pacchetti

Le seguenti sono alcune tecniche comuni di gestione dei pacchetti. Prima di prendere una decisione per un gestore di pacchetti effettuare alcune ricerche tra le varie tecniche, specialmente sugli svantaggi della strategia in questione.

6.3.2.1. Ci si ricorda di tutto

Sì, questa è una tecnica di gestione dei pacchetti. Alcune persone non sentono la necessità di un gestore di pacchetti perché conoscono in profondità i pacchetti e sanno quali file sono installati da ogni pacchetto. Inoltre alcuni utenti non hanno bisogno di nessuna gestione di pacchetti perché pianificano la ricostruzione dell'intero sistema quando un pacchetto è cambiato.

6.3.2.2. Installare in directory separate

Questa è una gestione di pacchetti semplicistica che non necessita di nessun pacchetto in più per gestire le installazioni. Ogni pacchetto è installato in una directory separata. Per esempio, il pacchetto foo-1.1 è installato in `/usr/pkg/foo-1.1` e viene creato un link simbolico da `/usr/pkg/foo` a `/usr/pkg/foo-1.1`. Quando si installa una nuova versione foo-1.2, questa viene installata in `/usr/pkg/foo-1.2` e il precedente link simbolico è sostituito da un link simbolico alla nuova versione.

Le variabili d'ambiente come `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` e `CPPFLAGS` devono essere ampliate per includere `/usr/pkg/foo`. Quando i pacchetti non sono più pochi, questa strategia diventa ingestibile.

6.3.2.3. Gestione dei pacchetti stile link simbolico

Questa è una variante della precedente tecnica di gestione dei pacchetti. Ogni pacchetto è installato in modo simile alla strategia precedente. Ma invece di fare il link simbolico, per ogni file viene creato un link simbolico nella gerarchia `/usr`. Questo toglie la necessità di ampliare le variabili d'ambiente. Anche se i link simbolici possono essere creati dall'utente, per automatizzarne la creazione sono stati scritti molti gestori di pacchetti che usano questo approccio. Alcuni tra i più popolari sono Stow, Epkg, Graft e Depot.

L'installazione deve essere manipolata, in modo tale che il pacchetto creda di essere installato in `/usr` mentre in realtà è installato nella gerarchia `/usr/pkg`. Installare in questo modo di solito non è un compito banale. Per esempio, supponiamo che si stia installando un pacchetto libfoo-1.1. Le istruzioni seguenti potrebbero non installare il pacchetto nel modo giusto:

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

L'installazione funzionerà, ma i pacchetti dipendenti potrebbero non avere il link a libfoo come ci si sarebbe aspettati. Se si compila un pacchetto che ha un link a libfoo, si dovrebbe controllare che il link sia a `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` invece che a `/usr/lib/libfoo.so.1`, come ci si sarebbe aspettati. L'approccio corretto è quello di usare la strategia `DESTDIR` per manipolare l'installazione del pacchetto. Questo approccio funziona così:

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

Molti pacchetti supportano questo approccio, ma alcuni no. Per i pacchetti che non si compilano, si potrebbe aver bisogno di installare manualmente il pacchetto, oppure ci si potrebbe accorgere che è più facile installare alcuni pacchetti problematici in `/opt`.

6.3.2.4. Gestione basata sulla datazione

In questa tecnica un file viene contrassegnato dalla data prima dell'installazione del pacchetto. Dopo l'installazione, un semplice uso del comando **find** con le opzioni appropriate può generare un log di tutti i file installati dopo la creazione del file contrassegnato. Un gestore di pacchetti scritto con questo approccio è install-log.

Anche se questa strategia ha il vantaggio di essere semplice, ha due difetti. Se durante l'installazione i file sono installati con una data diversa dalla data attuale, questi file non saranno rintracciati dal gestore di pacchetti. Inoltre questa strategia può essere usata solo quando si installa un pacchetto alla volta. I log non sono attendibili se due pacchetti sono installati su due console differenti.

6.3.2.5. Gestione basata su LD_PRELOAD

In questo approccio una libreria viene caricata prima dell'installazione. Durante l'installazione questa libreria tiene traccia dei pacchetti che vengono installati collegandosi ai vari eseguibili come **cp**, **install**, **mv** e tracciando le chiamate di sistema che modificano il file system. Perché questo sistema funzioni, si deve creare dinamicamente il link a tutti gli eseguibili senza il bit suid o sgid. Caricare prima la libreria può causare qualche effetto collaterale indesiderato durante l'installazione. Perciò è consigliato eseguire alcuni test per assicurarsi che il gestore di pacchetti non interrompa niente e faccia un log di tutti i file appropriati.

6.3.2.6. Creazione di archivi di pacchetti

In questa strategia l'installazione dei pacchetti è manipolata in un albero separato, come descritto nella gestione dei pacchetti stile link simbolico. Dopo l'installazione è creato un archivio di pacchetti usando i file installati. Questo archivio poi è usato per installare il pacchetto sulla macchina locale o può anche essere usato per installarlo su altre macchine.

Questo approccio è usato dalla maggior parte dei gestori di pacchetti che si trovano nelle distribuzioni commerciali. Esempi di gestori di pacchetti che seguono questo approccio sono RPM (che, tra l'altro, è richiesto da *Linux Standard Base Specification*), pkg-utils, apt di Debian e Portage system di Gentoo. Un'indicazione che descrive come adottare questo stile di gestione dei pacchetti per sistemi LFS si trova in *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

6.3.2.7. Gestione basata sull'utente

Questo approccio, unicamente per LFS, è stato inventato da Matthias Benkmann ed è disponibile in *Hints Project*. In questo approccio ogni pacchetto è installato nelle locazioni standard come un utente separato. I file che appartengono ad un pacchetto sono facilmente identificati controllando l'ID dell'utente. Le caratteristiche e i difetti di questo approccio sono troppo complesse da descrivere in questa sezione. Per i dettagli si vedano le indicazioni in *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

6.4. Accesso all'ambiente chroot

È il momento di accedere all'ambiente chroot per iniziare a costruire e installare il sistema LFS definitivo. Come utente `root` eseguire il comando seguente per entrare nell'ambiente che è, al momento, popolato solamente dai tool provvisori:

```
chroot "$LFS" /tools/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
    /tools/bin/bash --login +h
```

L'opzione *`-i`* data al comando **env** cancellerà tutte le variabili dell'ambiente chroot. Dopo questo, solo le variabili `HOME`, `TERM`, `PS1` e `PATH` vengono di nuovo definite. Il costrutto *`TERM=$TERM`* imposterà la variabile `TERM` all'interno di chroot dandole lo stesso valore che ha fuori da chroot. Questa variabile è necessaria perché programmi come **vim** e **less** funzionino correttamente. Se sono necessarie altre variabili, come `CFLAGS` o `CXXFLAGS`, questo è un momento buono per definirle di nuovo.

D'ora in poi non c'è più bisogno di usare la variabile `LFS`, perché tutto il lavoro sarà ristretto al file system LFS. Ciò perché alla shell Bash viene detto che adesso `$LFS` è la directory root (`/`).

Notare come `/tools/bin` sia all'ultimo posto nel `PATH`. Questo significa che un tool provvisorio non verrà più utilizzato una volta che sarà installata la sua versione finale. Questo accade perché la shell non «ricorda» le locazioni dei binari eseguiti—per questa ragione l'hashing viene disabilitato passando l'opzione *`+h`* a **bash**.

Notare che il prompt **bash** dirà `I have no name!` Questo è normale, poiché il file `/etc/passwd` non è stato ancora creato.

Nota

È importante che tutti i comandi nel resto di questo capitolo e nei seguenti capitoli siano avviati dall'interno dell'ambiente chroot. Se si dovesse lasciare questo ambiente per qualsiasi ragione (ad esempio per riavviare), assicurarsi che i file system virtuali del kernel siano montati come spiegato in Sezione 6.2.2, «Montaggio e popolamento di /dev» e in Sezione 6.2.3, «Montaggio dei file system virtuali del kernel» ed entrare di nuovo in chroot prima di continuare con l'installazione.

6.5. Creazione delle directory

È il momento di creare alcune strutture nel file system LFS. Creare un albero di directory standard inserendo i seguenti comandi :

```
mkdir -pv /{bin,boot,etc/opt,home,lib,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
mkdir -v /var/{lock,log,mail,run,spool}
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

Le directory sono, di default, create con il permesso in modalità 755, ma questo non è desiderabile per tutte le directory. Nei comandi precedenti vengono fatti due cambiamenti: uno alla directory home dell'utente `root` e un altro alle directory per i file temporanei.

Il primo cambiamento di modalità assicura che non tutti possano entrare nella directory `/root`—la stessa cosa che un utente normale avrebbe fatto con la propria home directory. Il secondo cambio di modalità assicura che tutti gli utenti possano scrivere nelle directory `/tmp` e `/var/tmp`, ma non possano rimuovere da queste i file di altri utenti. Quest'ultima operazione è proibita dal cosiddetto «sticky bit», il bit più alto nella maschera 1777.

6.5.1. Nota sulla conformità a FHS

L'albero delle directory si basa sul Filesystem Hierarchy Standard (FHS) (disponibile su *http://www.pathname.com/fhs/*). Oltre a FHS, si creano dei link simbolici di compatibilitlà per le directory `man`, `doc` e `info`, poiché molti pacchetti provano ancora a installare la loro documentazione in `/usr/<directory>` o `/usr/local/<directory>` invece che in `/usr/share/<directory>` o `/usr/local/share/<directory>`. FHS non è preciso sulla struttura della sottodirectory `/usr/local/share`, quindi creiamo solo le directory necessarie. Tuttavia si è liberi di creare queste directory se si preferisce conformarsi più strettamente allo standard FHS.

6.6. Creazione dei file e dei link simbolici essenziali

Certi programmi hanno cablati dentro dei collegamenti a programmi che non esistono ancora. Per soddisfare questi programmi, creare un certo numero di link simbolici che verranno rimpiazzati da file veri durante il corso di questo capitolo, dopo che il software è stato installato.

```
ln -sv /tools/bin/{bash,cat,grep,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv bash /bin/sh
```

Un sistema Linux corretto conserva un elenco dei file system montati nel file `/etc/mtab`. Normalmente questo file dovrebbe essere creato quando si monta un nuovo file system. Poiché non monteremo nessun file system nel nostro ambiente chroot, creare un file vuoto per quelle utility che si aspettano la presenza di `/etc/mtab`:

```
touch /etc/mtab
```

Affinché l'utente `root` sia in grado di effettuare il login e il nome «root» sia riconosciuto, devono esserci voci appropriate nei file `/etc/passwd` e `/etc/group`.

Creare il file `/etc/passwd` eseguendo il seguente comando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
EOF
```

La vera password per `root` sarà impostata successivamente (la «x» urata quì è solo un segnaposto).

Creare il file `/etc/group` eseguendo il seguente comando:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
EOF
```

I gruppi creati non fanno parte di nessuno standard—sono gruppi dettati in parte dalle necessità della configurazione Udev in questo capitolo e in parte dalla convenzione comune adottata da un certo numero di distribuzioni Linux esistenti. Lo standard Linux di base (Linux Standard Base, LSB, reperibile in *http://www.linuxbase.org*) raccomanda solo che, oltre al gruppo `root` con Group ID (GID) 0, sia presente un gruppo `bin` con GID 1. Tutti gli altri nomi e GID di gruppo possono essere scelti liberamente dall'amministratore di sistema, pioché i programmi ben scritti non dipendono dal numero GID, ma piuttosto usano il nome del gruppo.

Per rimuovere il prompt «I have no name!», iniziare una nuova shell. Poiché un completo Glibc è stato installato in Capitolo 5 e i file `/etc/passwd` e `/etc/group` sono stati creati, la risoluzione del nome utente e del nome di gruppo adesso funzionerà.

```
exec /tools/bin/bash --login +h
```

Notare l'uso della direttiva *+h*. Essa dice a **bash** di non usare il suo percorso interno di hash. Senza questa direttiva **bash** ricorderebbe i percorsi dei binari che ha eseguito. Per assucurare l'uso dei binari compilati di recente non appena vengono installati, la direttiva *+h* sarà usata per tutta la durata di questo capitolo.

I programmi **login**, **agetty** e **init** (e altri) usano un certo numero di file di log per memorizzare informazioni, come chi è entrato nel sistema e quando. Tuttavia questi programmi non scriveranno nei file di log se questi non esistono già. Inizializzare i file di log e assegnare loro i permessi appropriati:

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/run/utmp /var/log/lastlog
chmod -v 664 /var/run/utmp /var/log/lastlog
```

Il file `/var/run/utmp` registra gli utenti che al momento hanno effettuato il log in. Il file `/var/log/wtmp` registra tutti gli accessi e le uscite. Il file `/var/log/lastlog` registra quando ogni utente ha effettuato l'ultimo log in. Il file `/var/log/btmp` registra i tentativi non riusciti di log in.

6.7. Linux-Libc-Headers-2.6.12.0

Il pacchetto Linux-Libc-Headers contiene gli header del kernel «sterilizzati».

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 27 MB

6.7.1. Installazione di Linux-Libc-Headers

Per anni è stata pratica comune usare gli header del kernel «grezzi» (direttamente da un tarball del kernel) in `/usr/include`, ma nel corso degli ultimi anni c'è stata una forte presa di posizione degli sviluppatori del kernel sul fatto che queste cose non devono essere fatte. Perciò è nato il progetto Linux-Libc-Headers, disegnato per mantenere una versione stabile delle API degli header di Linux.

Aggiungere un header userspace e il supporto per le chiamate di sistema per la caratteristica inotify disponibile nei kernel Linux più recenti:

```
patch -Np1 -i ../linux-libc-headers-2.6.12.0-inotify-3.patch
```

Installare i file header:

```
install -dv /usr/include/asm
cp -Rv include/asm-i386/* /usr/include/asm
cp -Rv include/linux /usr/include
```

Assicurarsi che tutti gli header siano di proprietà di root:

```
chown -Rv root:root /usr/include/{asm,linux}
```

Assicurarsi che tutti gli utenti possano leggere gli header:

```
find /usr/include/{asm,linux} -type d -exec chmod -v 755 {} \;
find /usr/include/{asm,linux} -type f -exec chmod -v 644 {} \;
```

6.7.2. Contenuti di Linux-Libc-Headers

Header installati: /usr/include/{asm,linux}/*.h

Brevi descrizioni

`/usr/include/{asm,linux}/*.h` Gli header delle API di Linux

6.8. Man-pages-2.34

Il pacchetto Man-pages contiene oltre 1,200 pagine man.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 18.4 MB

6.8.1. Installazione di Man-pages

Installare Man-pages eseguendo:

```
make install
```

6.8.2. Contenuti di Man-pages

File installati: varie pagine man

Brevi descrizioni

`pagine man` Descrive le funzioni del linguaggio di programmazione C, importanti file di dispositivo e file di configurazione significativi

6.9. Glibc-2.3.6

Il pacchetto Glibc contiene la libreria C principale. Questa libreria fornisce tutte le routine di base per allocare memoria, cercare directory, aprire e chiudere file, leggere e scrivere file, manipolare stringhe, individuare pattern, aritmetica, e così via.

Tempo di costruzione approssimativo: 13.5 SBU inclusa la suite di test
Spazio necessario su disco: 510 MB inclusa la suite di test

6.9.1. Installazione di Glibc

Nota

Alcuni pacchetti al di fuori di LFS suggeriscono di installare GNU libiconv per poter tradurre i dati da una codifica ad un'altra. La home page del progetto (*http://www.gnu.org/software/libiconv/*) dice: « Questa libreria fornisce un'implementazione di `iconv()`, da usare in sistemi che non ne hanno una, o la cui implementazione non può fare conversioni da/a Unicode». Glibc fornisce un'implementazione di `iconv()` e può convertire da/a Unicode, pertanto libiconv non è necessaria su un sistema LFS.

Il sistema di costruzione di Glibc è auto-contenuto e si installerà perfettamente, nonostante gli specs file del compilatore e il linker puntino ancora a `/tools`. Specs e linker non si possono correggere prima dell'installazione di Glibc, poiché i test autoconf di Glibc darebbero falsi risultati e questo vanificherebbe lo scopo di ottenere una realizzazione pulita.

Il tarball di glibc-libidn aggiunge a Glibc il supporto per i nomi di dominio internazionalizzati names (internationalized domain names, IDN). Molti programmi che supportano IDN richiedono l'intera libreria `libidn`, non questa aggiunta (si veda *http://www.linuxfromscratch.org/blfs/view/svn/general/libidn.html*). Estrarre il tarball dalla directory dei sorgenti di Glibc:

```
tar -xf ../glibc-libidn-2.3.6.tar.bz2
```

Applicare la patch seguente per risolvere degli errori di costruzione nei pacchetti che includono `linux/types.h` dopo `sys/kd.h`:

```
patch -Np1 -i ../glibc-2.3.6-linux_types-1.patch
```

Aggiungere un header per definire le funzioni syscall per la caratteristica inotify disponibile nei kernel Linux più recenti:

```
patch -Np1 -i ../glibc-2.3.6-inotify-1.patch
```

Nella localizzazione vi_VN.TCVN, in fase di avvio **bash** entra in un loop infinito. Non si sa se ciò è dovuto a un bug di **bash** o a un problema di Glibc. Disabilitare l'installazione di questa localizzazione per evitare il problema:

```
sed -i '/vi_VN.TCVN/d' localedata/SUPPORTED
```

Quando si esegue **make install**, uno script chiamato `test-installation.pl` esegue un piccolo test di integrità sulla nostra Glibc appena installata. Tuttavia, poiché la nostra toolchain punta ancora alla directory `/tools`, il test di integrità potrebbe essere effettuato sulla Glibc sbagliata. Possiamo forzare lo script perché controlli la Glibc che abbiamo appena installato con il seguente comando:

```
sed -i \
's|libs -o|libs -L/usr/lib -Wl,-dynamic-linker=/lib/ld-linux.so.2 -o|' \
       scripts/test-installation.pl
```

La documentazione di Glibc raccomanda di costruire Glibc fuori dalla directory dei sorgenti, in una directory dedicata:

```
mkdir -v ../glibc-build
cd ../glibc-build
```

Preparare Glibc per la compilazione:

```
../glibc-2.3.6/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

Significato delle nuove opzioni di configurazione

`--libexecdir=/usr/lib/glibc`
Questa cambia la locazione del programma **pt_chown** dal suo default, `/usr/libexec`, a `/usr/lib/glibc`.

Compilare il pacchetto:

```
make
```

Importante

In questa sezione la suite di test per Glibc è considerata critica. Non saltarla per nessuna ragione.

Testare i risultati:

```
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

Probabilmente si verificherà un fallimento previsto (trascurato) nel test *posix/annexc*. Inoltre la suite di test di Glibc dipende molto dal sistema host. Questo è un elenco dei problemi più comuni:

- I test *nptl/tst-clock2* e *tst-attr3* a volte falliscono. La ragione non è completamente conosciuta, ma ci sono indicazioni che un carico pesante del sistema può causare questi fallimenti.
- I test math a volte falliscono su sistemi ove la CPU non è una Intel originale relativamente nuova o un processore AMD autentico.
- Se si ha la partizione LFS montata con l'opzione *`noatime`*, il test *atime* fallirà. Come menzionato in Sezione 2.4, «Montaggio della nuova partizione», non usare l'opzione *`noatime`* durante la costruzione di LFS.
- Su hardware più vecchi e lenti o su sistemi in fase di caricamento certi test potrebbero fallire a causa del superamento dei tempi di time out dei test.

Sebbene sia un messaggio innocuo, la fase di installazione di Glibc si lamenterà per l'assenza di `/etc/ld.so.conf`. Prevenire questo avviso con:

```
touch /etc/ld.so.conf
```

Installare il pacchetto:

```
make install
```

Installare l'header inotify nella locazione degli header di sistema:

```
cp -v ../glibc-2.3.6/sysdeps/unix/sysv/linux/inotify.h \
    /usr/include/sys
```

Le localizzazioni che permettono al sistema di rispondere in un altro linguaggio non sono state installate dal precedente comando. Nessuna delle localizzazioni è necessaria, ma se ne mancassero alcune, le suite di test dei pacchetti futuri salterebbero passaggi importanti del test.

Le singole localizzazioni possono essere installate usando il programma **localedef**. Ad es. il primo comando **localedef** che segue unisce la definizione del set di caratteri independenti della localizzazione `/usr/share/i18n/locales/de_DE` con la definizione della mappa di caratteri `/usr/share/i18n/charmaps/ISO-8859-1.gz` e accoda il risultato al file `/usr/lib/locale/locale-archive`. Le istruzioni seguenti installeranno il set minimo di localizzazioni necessarie per la copertura ottimale di test:

```
mkdir -pv /usr/lib/locale
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Inoltre installare la localizzazione per la propria nazione, lingua e set di caratteri.

In alternativa, installare in una volta sola tutte le localizzazioni elencate nel file `glibc-2.3.6/localedata/SUPPORTED` (che include tutte le localizzazioni elencate sopra e molte altre) con il comando seguente, che impiegherà molto tempo:

```
make localedata/install-locales
```

Usare quindi il comando **localedef** per creare e installare le localizzazioni non elencate nel file `glibc-2.3.6/localedata/SUPPORTED` nel caso improbabile in cui fossero necessarie.

6.9.2. Configurazione di Glibc

Si deve creare il file `/etc/nsswitch.conf`, poiché, nonostante Glibc lo fornisca di default quando questo file è mancante o corrotto, i default di Glibc non funzionano bene in un ambiente di rete. Inoltre bisogna configurare la fascia del fuso orario.

Creare un nuovo file `/etc/nsswitch.conf` eseguendo il seguente comando:

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

Un modo per sapere in quale fuso orario ci si trovi, è eseguire il seguente script:

```
tzselect
```

Dopo aver risposto a qualche domanda sulla locazione, lo script darà il nome della fascia del fuso orario (ad es. *America/Edmonton*). C'è anche qualche altra possibile fascia elencata in `/usr/share/zoneinfo` come *Canada/Eastern* o *EST5EDT* che non è identificata dallo script ma che si può usare.

Creare poi il file `/etc/localtime` eseguendo:

```
cp -v --remove-destination /usr/share/zoneinfo/<xxx> \
    /etc/localtime
```

Sostituire *<xxx>* con il nome della fascia oraria scelta (es., Canada/Eastern).

Significato delle opzioni di cp:

`--remove-destination`
: Questa è necessaria per forzare la rimozione di link simbolici già esistenti. La ragione per cui si copia il file invece di usare un link simbolico è di coprire la situazione in cui `/usr` sia su una partizione separata. Questo potrebbe essere importante quando si avvia il sistema in modalità singolo utente.

6.9.3. Configurazione del Dynamic Loader

Per default il dynamic loader (`/lib/ld-linux.so.2`) cerca in `/lib` e in `/usr/lib` le librerie dinamiche richieste dai programmi quando vengono eseguiti. Tuttavia, se ci fossero librerie in directory diverse da `/lib` e `/usr/lib`, occorre aggiungerle al file `/etc/ld.so.conf` perché il dynamic loader le trovi. Due directory che di solito contengono librerie aggiuntive sono `/usr/local/lib` e `/opt/lib`, perciò aggiungere queste directory al percorso di ricerca del dynamic loader.

Creare un nuovo file `/etc/ld.so.conf` eseguendo quanto segue:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf

/usr/local/lib
/opt/lib

# End /etc/ld.so.conf
EOF
```

6.9.4. Contenuti di Glibc

Programmi installati: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump e zic
Librerie installate: ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.a, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so e libutil.{a,so}

Brevi descrizioni

catchsegv	Può essere usato per creare una traccia dello stack quando un programma termina con un segmentation fault
gencat	Genera elenchi di messaggi
getconf	Visualizza i valori di configurazione del sistema per variabili specifiche del file system
getent	Acquisisce valori da un database amministrativo
iconv	Esegue una conversione del set di caratteri
iconvconfig	Crea i file di configurazione a caricamento rapido del modulo **iconv**
ldconfig	Configura i riferimenti runtime del linker dinamico
ldd	Riporta quali librerie condivise sono richieste da ciascun dato programma o libreria condivisa
lddlibc4	Assiste **ldd** con i file oggetto
locale	Stampa diverse informazioni sulla localizzazione corrente
localedef	Compila le specifiche di localizzazione
mtrace	Legge e interpreta un file di traccia memoria e visualizza un sommario in formato

	leggibile
nscd	Un demone che fornisce una cache per le richieste più comuni del servizio dei nomi
nscd_nischeck	Verifica se il modo sicuro è necessario o no per il NIS+ lookup
pcprofiledump	Scarica le informazioni generate dal PC profiling
pt_chown	Un programma helper per **grantpt** per definire proprietario, gruppo e permessi di accesso di uno pseudo terminale slave
rpcgen	Genera codice C per implementare il protocollo Remote Procedure Call (RPC)
rpcinfo	Fa una chiamata RPC a un server RPC
sln	Un programma **ln** linkato staticamente
sprof	Legge e visualizza dati di profilo degli oggetti condivisi
tzselect	Chiede all'utente la locazione del sistema e riporta la descrizione della corrispondente fascia del fuso orario
xtrace	Traccia l'esecuzione di un programma stampando la funzione correntemente eseguita
zdump	Scarica la fascia del fuso orario
zic	Il compilatore della fascia del fuso orario
`ld.so`	Il programma helper per le librerie condivise eseguibili
`libBrokenLocale`	Usato internamente da Glibc come uno strumento grossolano per conoscere i programmi interrotti in esecuzione (ad es. alcune applicazioni Motif). Per maggiorni informazioni si veda il commento in `glibc-2.3.6/locale/broken_cur_max.c`
`libSegFault`	Il gestore del segnale di segmentation fault usato da **catchsegv**
`libanl`	Una libreria asincrona di ricerca nomi
`libbsd-compat`	Fornisce la portabilità necessaria ad eseguire certi programmi Berkey Software Distribution (BSD) sotto Linux
`libc`	La libreria C principale
`libcidn`	Usata internamente da Glibc per gestire i nomi di dominio internazionalizzati nella funzione `getaddrinfo()`
`libcrypt`	La libreria crittografica
`libdl`	La libreria di interfaccia per il collegamento dinamico
`libg`	Libreria fittizia che non contiene funzioni. Precedentemente era una libreria runtime per **g++**
`libieee`	Il link a questo modulo obbliga le funzioni math a seguire le regole di gestione dell'errore definite dell'Institute of Electrical and Electronic Engineers (IEEE). Il default è la gestione dell'errore POSIX.1
`libm`	La libreria matematica
`libmcheck`	Il link a questa libreria abilita il controllo dell'allocazione di memoria
`libmemusage`	Usato da **memusage** per aiutare a raccogliere informazioni sull'uso della memoria

	da parte di un programma
`libnsl`	La libreria dei servizi di rete
`libnss`	Sono le librerie Name Service Switch, contenenti funzioni per la risoluzione di nomi di host, nomi utente, nomi di gruppo, alias, servizi, protocolli, ecc.
`libpcprofile`	Contiene funzioni di profiling utilizzate per tracciare l'ammontare di tempo CPU speso in linee specifiche di codice sorgente
`libpthread`	La libreria dei thread POSIX
`libresolv`	Contiene funzioni per creare, inviare e interpretare pacchetti dei server dei nomi di dominio di Internet
`librpcsvc`	Contiene funzioni che forniscono diversi servizi RPC
`librt`	Contiene funzioni che forniscono la maggior parte delle interfacce specificate da POSIX.1b Realtime Extension
`libthread_db`	Contiene funzioni utili per costruire debugger per programmi multi-thread
`libutil`	Contiene codice per le funzioni «standard» usate in molte utility Unix

6.10. Riaggiustamento della Toolchain

Ora che le librerie C finali sono state installate, è tempo di aggiustare di nuovo la toolchain. La toolchain verrà messa a punto in modo che ogni nuovo programma compilato si collegherà a queste nuove librerie. Questo è lo stesso processo usato nella fase «Messa a punto» all'inizio del Capitolo 5, ma con le sistemazioni invertite. Nel Capitolo 5 la chain è stata guidata dalle directory `/{,usr/}lib` dell'host alla nuova directory `/tools/lib`. Ora la chain verrà guidata dalla stessa directory `/tools/lib` alle directory di LFS `/{,usr/}lib`.

Per prima cosa eseguire un backup del linker `/tools` e sostituirlo con il linker aggiustato che abbiamo creato nel capitolo 5. Creeremo anche un link ai suoi equivalenti in `/tools/$(gcc -dumpmachine)/bin`.

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

Poi correggere il file specs di GCC in modo che punti al nuovo linker dinamico e così GCC sappia dove trovare il suoi file di avvio. Un comando **perl** si occupa di questo:

```
gcc -dumpspecs | \
perl -p -e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g;' \
    -e 's@\*startfile_prefix_spec:\n@$_/usr/lib/ @g;' > \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

È una buona idea controllare visivamente il file specs per verificare che i cambiamenti voluti siano stati apportati.

> **Importante**
>
> Se si sta lavorando su una piattaforma in cui il nome del linker dinamico è diverso da `ld-linux.so.2`, nel comando precedente sostituire «ld-linux.so.2» con il nome del linker dinamico della piattaforma. Se necessario si rimanda a Sezione 5.2, «Note tecniche sulla Toolchain,».

È obbligatorio a questo punto assicurarsi che le funzioni di base (di compilazione e di link) della toolchain modificata funzionino come previsto. Per farlo, eseguire i seguenti controlli di integrità:

```
echo 'main(){}' > dummy.c
cc dummy.c -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà (ad eccezione di differenze nel nome del linker dinamico dipendenti dalla piattaforma):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Notare che `/lib` è ora il prefisso del nostro linker dinamico.

Ora assicurarsi di essere configurati in modo da usare gli startfile corretti:

```
grep -o '/usr/lib.*/crt[1in].* .*' dummy.log
```

Se tutto funziona correttamente non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
/usr/lib/crt1.o succeeded
/usr/lib/crti.o succeeded
/usr/lib/crtn.o succeeded
```

Quindi verificare che il nuovo linker venga usato con i path di ricerca corretti:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

Poi assicurarsi che stiamo usando la libc corretta:

```
grep "/lib/libc.so.6 " dummy.log
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
attempt to open /lib/libc.so.6 succeeded
```

Infine assicurarsi che GCC stia usando il linker dinamico corretto:

```
grep found dummy.log
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà (ad eccezione di differenze nel nome del linker dinamico dipendenti dalla piattaforma):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

Se l'output non appare come mostrato sopra o non si è ricevuto alcun output, allora si è verificato qualche errore grave. Bisogna investigare e ripercorrere i propri passi per trovare dove è il problema e correggerlo. La ragione più probabile è che qualcosa sia andato storto nella correzione del file specs. Tutti i problemi dovranno essere risolti prima di poter continuare nel processo.

Appena tutto funziona correttamente cancellare i file di test:

```
rm -v dummy.c a.out dummy.log
```

6.11. Binutils-2.16.1

Il pacchetto Binutils contiene un linker, un assemblatore e altri tool per manipolare file oggetto.

Tempo di costruzione approssimativo: 1.5 SBU inclusa la suite di test
Spazio necessario su disco: 172 MB inclusa la suite di test

6.11.1. Installazione di Binutils

Verificare che i PTY funzionino correttamente nell'ambiente chroot. Controllare che le impostazioni siano corrette eseguendo un semplice test:

```
expect -c "spawn ls"
```

Se compare il seguente messaggio, l'ambiente chroot non è impostato per una corretta operatività corretta PTY:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Questo problema deve essere risolto prima di eseguire le suite di test per Binutils e GCC.

La documentazione di Binutils raccomanda di costruire Binutils fuori dalla directory dei sorgenti, in una directory di costruzione dedicata:

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Preparare Binutils per la compilazione:

```
../binutils-2.16.1/configure --prefix=/usr \
    --enable-shared
```

Compilare il pacchetto:

```
make tooldir=/usr
```

Il significato del parametro di make:

`tooldir=/usr`
Normalmente la tooldir (la directory dove alla fine verranno collocati gli eseguibili) è impostata a `$(exec_prefix)/$(target_alias)`. Per esempio le macchine i686 la espanderanno in `/usr/i686-pc-linux-gnu`. Poiché questo è un sistema personalizzato, questa directory di destinazione specifica in `/usr` non è necessaria. `$(exec_prefix)/$(target_alias)` verrebbe utilizzata se il sistema fosse usato per il cross-compile (per esempio compilare un pacchetto su una macchina Intel che genera codice che può essere eseguito su macchine PowerPC).

Importante

La suite di test per Binutils in questa sezione è considerata critica. Non saltarla per nessuna ragione.

Testare i risultati:

```
make check
```

Installare il pacchetto:

```
make tooldir=/usr install
```

Installare il file header `libiberty` che è richiesto da alcuni pacchetti:

```
cp -v ../binutils-2.16.1/include/libiberty.h /usr/include
```

6.11.2. Contenuti di Binutils

Programmi installati: addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings e strip
Librerie installate: libiberty.a, libbfd.{a,so} e libopcodes.{a,so}

Brevi descrizioni

addr2line	Traduce indirizzi di programmi in nomi di file e numeri di linea; dato un indirizzo e il nome di un eseguibile, usa le informazioni di debug nell'eseguibile per determinare quale file sorgente e numero di linea siano associati all'indirizzo
ar	Crea, modifica e estrae da archivi
as	Un assemblatore che assembla l'output di **gcc** in file oggetto
c++filt	Utilizzato dal linker per decodificare simboli C++ e Java ed evitare che funzioni sovraccaricate crollino
gprof	Mostra i dati del profilo grafico di chiamata
ld	Un linker che combina un certo numero di file oggetto e archivi in un singolo file, riallocando i loro dati e collegando i riferimenti dei simboli
nm	Elenca i simboli occorrenti in un dato file oggetto
objcopy	Utilizzato per tradurre un certo tipo di file oggetto in un altro
objdump	Visualizza informazioni su un dato file oggetto, con opzioni che permettono di controllare quali informazioni evidenziare; l'informazione mostrata è utile ai programmatori che stanno lavorando ai tool di compilazione
ranlib	Genera un indice dei contenui di un archivio e lo memorizza nell'archivio; l'indice elenca tutti i simboli definiti dai membri dell'archivio che sono file oggetto riallocabili
readelf	Mostra informazioni sui binari di tipo ELF
size	Visualizza le dimensioni delle sezioni e la dimensione totale per i file oggetto dati
strings	Emette, per ciascun file dato, la sequenza di caratteri stampabili che siano almeno di una specifica lunghezza (per default è 4); per i file oggetto stampa, per default, solo le stringhe

	delle sezioni di inizializzazione e caricamento, mentre per altri tipi di file scansiona l'intero file
strip	Elimina i simboli dai file oggetto
`libiberty`	Contiene routine usate da diversi programmi GNU, inclusi **getopt**, **obstack**, **strerror**, **strtol** e **strtoul**
`libbfd`	La libreria Binary File Descriptor
`libopcodes`	Una libreria per gestire gli opcodes—le versioni «testo leggibile» delle istruzioni per il processore; è usata per la costruzione di utilità come **objdump**

6.12. GCC-4.0.3

Il pacchetto GCC contiene la collezione di compilatori GNU, che include i compilatori C e C++.

Tempo di costruzione approssimativo: 22 SBU inclusa la suite di test
Spazio necessario su disco: 566 MB inclusa la suite di test

6.12.1. Installazione di GCC

Applicare una sostituzione **sed** che sopprimerà l'installazione di `libiberty.a`. In sostituzione verrà utilizzata la versione di `libiberty.a` fornita da Binutils:

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

La costruzione bootstrap eseguita in Sezione 5.4, «GCC-4.0.3 - Passo 1» ha creato GCC con il flag di compilazione `-fomit-frame-pointer`. Le costruzioni non-bootstrap omettono questo flag per default, quindi applicare il **sed** seguente per usarlo e così garantire delle costruzioni di compilazione coerenti.

```
sed -i 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in
```

È noto che a volte lo script **fixincludes** cerca erroneamente di "correggere" gli header di sistema finora installati. Poiché si sa che gli header installati da GCC-4.0.3 e Glibc-2.3.6 non richiedono correzioni, digitare il seguente comando per impedire l'esecuzione dello script **fixincludes**:

```
sed -i 's@\./fixinc\.sh@-c true@' gcc/Makefile.in
```

GCC fornisce uno script **gccbug** che durante la compilazione rileva se è presente mktemp e codifica il risultato in un test. Questo farà sì che lo script tornerà ad usare meno nomi casuali per i file temporanei. Instelleremo mktemp più tardi, perciò il seguente sed simulerà la sua presenza.

```
sed -i 's/@have_mktemp_command@/yes/' gcc/gccbug.in
```

La documentazione di GCC raccomanda di costruire GCC fuori dalla directory dei sorgenti in una directory dedicata:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Preparare GCC per la compilazione:

```
../gcc-4.0.3/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++
```

Compilare il pacchetto:

```
make
```

Importante

La suite di test per GCC in questa sezione è considerata critica. Non saltarla per nessuna ragione.

Testare i risultati, ma senza bloccarsi sugli errori:

```
make -k check
```

Per avere un sommario dei risultati della suite di test, eseguire:

```
../gcc-4.0.3/contrib/test_summary
```

Solo per i sommari, eseguire il pipe dell'output con **`grep -A7 Summ`**.

I risultati possono essere confrontati con quelli in *http://www.linuxfromscratch.org/lfs/build-logs/6.2/*.

Non sempre è possibile evitare alcuni fallimenti. Gli sviluppatori di GCC di solito sono consapevoli di questi problemi, ma non li hanno ancora risolti. In particolare, è noto che i test `libmudflap` sono particolarmente problematici a causa di un bug in GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). A meno che i risultati dei test non siano enormemente diversi da quelli all'URL precedente, si può continuare tranquillamente.

Installare il pacchetto:

```
make install
```

Alcuni pacchetti si aspettano che il preprocessore C sia installato nella directory `/lib`. Per supportare questi pacchetti creare questo link simbolico:

```
ln -sv ../usr/bin/cpp /lib
```

Molti pacchetti usano il nome **cc** per chiamare il compilatore C. Per soddisfare questi pacchetti creare un link simbolico:

```
ln -sv gcc /usr/bin/cc
```

Ora che la nostra toolchain finale è posizionata, è importante assicurarsi di nuovo che la compilazione e il linking funzionino come previsto. Facciamo questo eseguendo gli stessi controlli di integrità che abbiamo fatto prima nel capitolo:

```
echo 'main(){}' > dummy.c
cc dummy.c -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà (ad eccezione di differenze nel nome del linker dinamico dipendenti dalla piattaforma):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Ora assicurarsi di essere configurati in modo da usare gli startfile corretti:

```
grep -o '/usr/lib.*/crt[1in].* .*' dummy.log
```

Se tutto funziona correttamente non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crtn.o succeeded
```

Quindi verificare che il nuovo linker venga usato con i path di ricerca corretti:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Poi assicurarsi che stiamo usando la libc corretta:

```
grep "/lib/libc.so.6 " dummy.log
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà:

```
attempt to open /lib/libc.so.6 succeeded
```

Infine assicurarsi che GCC stia usando il linker dinamico corretto:

```
grep found dummy.log
```

Se tutto funziona correttamente, non dovrebbero esserci errori e l'output dell'ultimo comando sarà (ad eccezione di differenze nel nome del linker dinamico dipendenti dalla piattaforma):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

Se l'output non appare come mostrato sopra o non si è ricevuto alcun output, allora si è verificato qualche errore grave. Bisogna investigare e ripercorrere i propri passi per trovare dove è il problema e correggerlo. La ragione più probabile è che qualcosa sia andato storto nella correzione del file specs. Tutti i problemi dovranno essere risolti prima di poter continuare nel processo.

Appena tutto funziona correttamente cancellare i file di test:

```
rm -v dummy.c a.out dummy.log
```

6.12.2. Contenuti di GCC

Programmi installati: c++, cc (link a gcc), cpp, g++, gcc, gccbug e gcov
Librerie installate: libgcc.a, libgcc_eh.a, libgcc_s.so, libstdc++.[a,so] e libsupc++.a

Brevi descrizioni

cc	Il compilatore C
cpp	Il preprocessore C; è usato dal compilatore per espandere le dichiarazioni #include, #define e simili nei file sorgenti
c++	Il compilatore C++
g++	Il compilatore C++
gcc	Il compilatore C
gccbug	Uno shell script usato per aiutare a creare utili report dei bug
gcov	Tool di coverage testing; è usato per analizzare i programmi e determinare dove le ottimizzazioni avranno il maggior effetto
`libgcc`	Contiene il supporto run-time per **gcc**
`libstdc++`	La libreria standard C++
`libsupc++`	Fornisce routine di supporto per il linguaggio di programmazione C++

6.13. Berkeley DB-4.4.20

Il pacchetto Berkeley DB contiene programmi e utility usate da molte altre applicazioni per le funzioni legate ai database.

Tempo di costruzione approssimativo: 1.2 SBU
Spazio necessario su disco: 77 MB

Altre possibilità di installazione

Nel libro BLFS ci sono delle istruzioni per costruire questo pacchetto, se si ha la necessità di costruire il server RPC o binding di linguaggio aggiuntivi. I binding di linguaggio aggiuntivi richiederanno pacchetti ulteriori per essere installati. Si veda *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db* per le istruzioni di installazione consigliate.

Inoltre GDBM *potrebbe* essere usato al posto di Berkeley DB per soddisfare Man-DB. Tuttavia, poiché Berkeley DB è considerato una parte essenziale della costruzione di LFS, esso non sarà elencato come una dipendenza per nessun pacchetto nel libro BLFS. Allo stesso modo si impiegano molte ore per testare LFS con installato Berkeley DB, non con GDBM. Se si conoscono a pieno i rischi contro i benefici nell'usare GDBM e si desidera usarlo comunque, vedere le istruzioni BLFS situate in *http://www.linuxfromscratch.org/blfs/view/svn/general/gdbm.html*.

6.13.1. Installazione di Berkeley DB

Eseguire la patch sul pacchetto per eliminare potenziali interruzioni:

```
patch -Np1 -i ../db-4.4.20-fixes-1.patch
```

Preparare Berkeley DB per la compilazione:

```
cd build_unix &&
../dist/configure --prefix=/usr --enable-compat185 --enable-cxx
```

Significato delle opzioni di configurazione:

`--enable-compat185`
Questa opzione abilita la costruzione delle API di compatibilità con Berkeley DB 1.85.

`--enable-cxx`
Questa opzione abilita la costruzione delle librerie API di C++.

Compilare il pacchetto:

```
make
```

Non è possibile testare il pacchetto completamente, perché questo comporterebbe la costruzione di binding TCL. I binding TCL non possono essere costruiti in modo corretto ora, perché TCL ha un link a Glibc in `/tools` e non a Glibc in `/usr`.

Installare il pacchetto:

```
make docdir=/usr/share/doc/db-4.4.20 install
```

Significato del parametro di make:

docdir=...
Questa variabile specifica il posto corretto per la documentazione.

Correggere i proprietari dei file installati:

```
chown -v root:root /usr/bin/db_* \
    /usr/lib/libdb* /usr/include/db* &&
chown -Rv root:root /usr/share/doc/db-4.4.20
```

6.13.2. Contenuti di Berkeley DB

Programmi installati: db_archive, db_checkpoint, db_deadlock, db_dump, db_hotbackup, db_load, db_printlog, db_recover, db_stat, db_upgrade e db_verify
Librerie installate: libdb.{so,ar} e libdb_cxx.r{o,ar}

Brevi descrizioni

db_archive	Stampa i nomi dei path dei file di log che non sono più in uso
db_checkpoint	Un demone usato per monitorare e controllare i log di database
db_deadlock	Un demone usato per terminare richieste di tipo lock quando sono rilevati dei deadlock
db_dump	Converte i file di database in un file di formato plain-text leggibile da **db_load**
db_hotbackup	Crea istantanee «hot backup» o «hot failover» di database Berkeley DB
db_load	È usato per ricavare file di database da file di testo
db_printlog	Converte i file di log del database in testi leggibili
db_recover	È usato per ripristinare un database ad uno stato consistente dopo un fallimento
db_stat	Mostra le statistiche per i database Berkeley
db_upgrade	È usato per aggiornare i file di database ad una nuova versione di Berkeley DB
db_verify	È usato per eseguire controlli di coerenza su file di database
`libdb.{so,a}`	Contiene funzioni per manipolare file di database da programmi C
`libdb_cxx.{so,a}`	Contiene funzioni per manipolare file di database da programmi C++

6.14. Coreutils-5.96

Il pacchetto Coreutils contiene utilità per visualizzare e impostare le caratteristiche di base del sistema.

Tempo di costruzione approssimativo: 1.1 SBU
Spazio necessario su disco: 58.3 MB

6.14.1. Installazione di Coreutils

Un problema noto con il programma **uname** di questo pacchetto è che il parametro *-p* restituisce sempre `unknown`. La seguente patch corregge questo comportamento per le architetture Intel:

```
patch -Np1 -i ../coreutils-5.96-uname-1.patch
```

Inibire l'installazione da parte di Coreutils di binari che verranno installati da altri programmi più tardi:

```
patch -Np1 -i ../coreutils-5.96-suppress_uptime_kill_su-1.patch
```

POSIX richiede che i programmi di Coreutils delimitino i caratteri correttamente anche in localizzazioni di tipo multibyte. La patch seguente corregge questa non conformità e altri bug legati all'internazionalizzazione:

```
patch -Np1 -i ../coreutils-5.96-i18n-1.patch
```

Affinché siano superati i test aggiunti da questa patch, devono essere modificati i permessi per il file di test:

```
chmod +x tests/sort/sort-mb-tests
```

Nota

Nel passato sono stati trovati molti bug in questa patch. Prima di riportare nuovi bug ai maintainer di Coreutils, controllare se siano riproducibili senza questa patch.

Si è scoperto che a volte i messaggi tradotti generano un buffer overflow nel comando **who -Hu**. Aumentare la dimensione del buffer:

```
sed -i 's/_LEN 6/_LEN 20/' src/who.c
```

Ora preparare Coreutils per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

La suite di test di Coreutils fa numerose assunzioni circa la presenza di utenti e gruppi di sistema che non sono valide all'interno dell'ambiente minimale esistente al momento. Pertanto si dovranno eseguire dei passi aggiuntivi prima di eseguire i test. Se si decide di non eseguire i test, saltare a «Installazione del pacchetto».

Creare due gruppi dummy e un utente dummy:

```
echo "dummy1:x:1000:" >> /etc/group
echo "dummy2:x:1001:dummy" >> /etc/group
echo "dummy:x:1000:1000::/bin/bash" >> /etc/passwd
```

Ora la suite di test è pronta per essere eseguita. Dapprima avviare alcuni test pensati per essere eseguiti come `root`:

```
make NON_ROOT_USERNAME=dummy check-root
```

Quindi eseguire i rimanenti test come utente `dummy`:

```
src/su dummy -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Una volta terminati i test, rimuovere utente e gruppi dummy:

```
sed -i '/dummy/d' /etc/passwd /etc/group
```

Installare il pacchetto:

```
make install
```

Spostare i programmi nelle locazioni specificate dal FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

Alcuni degli script nel pacchetto LFS-Bootscripts dipendono da **head**, **sleep** e **nice**. Poiché `/usr` potrebbe non essere disponibile durante le prime fasi di avvio, questi binari devono essere sulla partizione di root:

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.14.2. Contenuti di Coreutils

Programmi installati: basename, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, shred, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami e yes

Brevi descrizioni

basename Toglie ogni percorso e ogni suffisso specificato dai nomi dei file

cat Concatena file allo standard output

chgrp Cambia il gruppo di appartenenza di file e directory

chmod Cambia i permessi di ciascun file nella modalità specificata. La modalità può essere sia una rappresentazione simbolica dei cambiamenti da apportare, sia un numero ottale che rappresenti i nuovi permessi

chown Cambia utente e/o gruppo proprietari di file e directory

chroot Esegue un dato comando con la directory specificata come directory `/`

cksum Stampa il checksum CRC (Cyclic Redundancy Check) e il numero dei byte di ciascun file specificato

comm Confronta due file ordinati, stampando su tre colonne le linee che sono univoche, e le linee che sono comuni

cp Copia i file

csplit Divide un dato file in più file nuovi, separandoli secondo pattern o numeri di linee specificati e visualizzando il numero dei byte di ciascun nuovo file

cut Stampa parti di linee, selezionando le parti in accordo con i campi o le posizioni richieste

date Visualizza l'orario corrente nel formato indicato, o definisce la data di sistema

dd Copia un file usando il numero e la dimensione di blocco indicato, opzionalmente può anche apportarci delle conversioni

df Riporta l'ammontare di spazio di disco disponibile (e usato) su tutti i file system montati, o solo sui file system contenenti i file indicati

dir Elenca i contenuti di ciascuna directory specificata (come il comando **ls**)

dircolors Invia comandi per impostare la variabile ambiente `LS_COLOR` e cambiare lo schema di colore usato da **ls**

dirname Toglie il suffisso non-directory dal nome di un file

du Riporta l'ammontare dello spazio di disco utilizzato dalla directory corrente, da ciascuna delle directory specificate (includendo tutte le sottodirectory) o da ciascuno dei file indicati

echo Visualizza le stringhe specificate

env Esegue un comando in un ambiente modificato

expand Converte tabulazioni in spazi

expr Valuta espressioni

factor Stampa i fattori primi di tutti i numeri interi specificati

false Non fa nulla, senza successo. Esce sempre con un codice di stato che indica l'errore

fmt Riformatta i paragrafi nei file specificati

fold Racchiude le linee nei file specificati

groups Riporta il gruppo di appartenenza di un utente

head Stampa le prime dieci linee (o il numero di linee indicato) di ciascun file specificato

hostid Riporta l'identificatore numerico (in esadecimale) dell'host

hostname Riporta o imposta il nome dell'host

id Riporta l'effettivo ID utente, ID gruppo e gruppo di appartenenza dell'utente corrente o di un utente specificato

install Copia file, mentre definisce i loro permessi e, se possibile, il loro proprietario e gruppo

join Unisce le linee che hanno campi unione identici da due file separati

link	Crea un link fisico con il nome specificato ad un file
ln	Crea link fisici o simbolici tra file
logname	Riporta il nome di login dell'utente corrente
ls	Elenca il contenuto di ciascuna directory specificata
md5sum	Riporta o verifica i checksum MD5 (Message Digest 5)
mkdir	Crea directory con il nome specificato
mkfifo	Crea FIFO (First-In, First-Out), una «named pipe» in gergo UNIX, con i nomi specificati
mknod	Crea nodi di dispositivo con i nomi specificati; un nodo di dispositivo è un file speciale a caratteri, o un file speciale a blocchi, o un FIFO
mv	Sposta o rinomina file o directory
nice	Esegue un programma con la priorità di scheduling modificata
nl	Numera le linee dei file specificati
nohup	Esegue un comando non interrompibile, con output rediretto a un file di log
od	Mostra i file in ottale e in altri formati
paste	Unisce i file specificati, unendo in sequenza le linee corrispondenti l'una all'altra, separate da caratteri tab
pathchk	Controlla se i nomi di file sono validi o portabili
pinky	È un client finger leggero; riporta alcune informazioni sugli utenti specificati
pr	Impagina e incolonna i file per la stampa
printenv	Stampa l'ambiente
printf	Stampa gli argomenti specificati secondo il formato specificato, in modo molto simile alla funzione printf del C
ptx	Genera un indice permutato del contenuto dei file specificati, con ciascuna parola chiave nel suo contesto
pwd	Riporta il nome della directory di lavoro corrente
readlink	Riporta il valore di un link simbolico specificato
rm	Rimuove file o directory
rmdir	Rimuove directory, se sono vuote
seq	Stampa una sequenza di numeri con un dato range e un dato incremento
sha1sum	Stampa o verifica il checksum 160-bit Secure Hash Algorithm 1 (SHA1)
shred	Sovrascrive ripetutamente i file specificati con pattern complessi, per rendere difficile il recupero dei dati
sleep	Pausa per uno specifico ammontare di tempo
sort	Ordina le linee dei file specificati
split	Divide il file specificato in parti, per dimensione o per numero di linee

stat	Visualizza lo stato del file o del filesystem
stty	Imposta o restituisce le impostazioni della riga di terminale
sum	Stampa il checksum e conteggia i blocchi di ciascun file specificato
sync	Svuota i buffer del file system; forza sul disco i blocchi modificati e aggiorna il super block
tac	Concatena e visualizza al contrario i file specificati
tail	Stampa le ultime dieci linee (o il dato numero di linee) di ciascun file specificato
tee	Legge dall'input standard, mentre scrive sia sull'output standard, sia nei file specificati
test	Confronta valori e verifica tipi di file
touch	Cambia data e ora del file, impostando l'ora di accesso e modifica dei file specificati con l'ora attuale; i file che non esistono sono creati con lunghezza zero
tr	Traduce, estrae e cancella i caratteri specificati dall'input standard
true	Non fa nulla, con successo; esce sempre con un codice di stato che indica successo (vero)
tsort	Esegue un ordinamento topologico; scrive una lista completamente ordinata in accordo con l'ordinamento parziale di un file specificato
tty	Riporta il nome del file del terminale connesso all'input standard
uname	Riporta informazioni di sistema
unexpand	Converte spazi in tabulazioni
uniq	Scarta tutte le linee identiche consecutive tranne una
unlink	Rimuove il file specificato
users	Riporta i nomi degli utenti attualmente connessi al sistema
vdir	È identico a **ls -l**
wc	Riporta il numero di linee, parole e byte per ciascun dato file e il totale di linee quando viene specificato più di un file
who	Riporta chi è connesso al sistema
whoami	Riporta il nome utente associato con l'attuale effettivo ID utente
yes	Emette ripetutamente «y» o una data stringa fino a quando non viene interrotto

6.15. Iana-Etc-2.10

Il pacchetto Iana-Etc fornisce dati per servizi di rete e protocolli.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 2.1 MB

6.15.1. Installazione di Iana-Etc

Il seguente comando converte i dati grezzi forniti da IANA nei formati corretti per i file dati `/etc/protocols` e `/etc/services`:

```
make
```

Installare il pacchetto:

```
make install
```

6.15.2. Contenuti di Iana-Etc

File installati: /etc/protocols e /etc/services

Brevi descrizioni

`/etc/protocols`	Descrive i vari protocolli Internet DARPA che sono disponibili dal sottosistema TCP/IP
`/etc/services`	Fornisce una mappatura tra nomi testuali di servizi internet e i loro numeri di porta e tipi di protocollo sottostanti assegnati

6.16. M4-1.4.4

Il pacchetto M4 contiene un processore macro.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 3 MB

6.16.1. Installazione di M4

Preparare M4 per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.16.2. Contenuti di M4

Programma installato: m4

Brevi descrizioni

m4 Copia i file specificati espandendo nel contempo le macro che essi contengono. Queste macro sono o incluse o definite dall'utente e possono comprendere un numero qualsiasi di argomenti. Oltre a fare espansione macro **m4** contiene funzioni per includere nomi di file, eseguire comandi Unix, fare aritmetica intera, manipolare testo, ricorsività e così via. Il programma **m4** può essere usato sia come front-end per il compilatore sia come processore macro in sé.

6.17. Bison-2.2

Il pacchetto Bison contiene un generatore di analizzatori sintattici.

Tempo di costruzione approssimativo: 0.6 SBU
Spazio necessario su disco: 11.9 MB

6.17.1. Installazione di Bison

Preparare Bison per la compilazione:

```
./configure --prefix=/usr
```

Il sistema di configurazione causa la costruzione di bison senza il supporto per l'internazionalizzazione dei messaggi di errore se **bison** non si trova già in $PATH. La seguente aggiunta correggerà questa mancanza.

```
echo '#define YYENABLE_NLS 1' >> config.h
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.17.2. Contenuti di Bison

Programmi installati: bison e yacc
Libreria installata: liby.a

Brevi descrizioni

bison	Genera, da una serie di regole, un programma per analizzare la struttura dei file di testo. Bison è un sostituto di Yacc (Yet Another Compiler Compiler).
yacc	Wrapper per **bison**, pensato per programmi che chiamano ancora **yacc** invece di **bison**. Chiama **bison** con l'opzione *`-y`*.
`liby.a`	Libreria Yacc contenente implementazioni di yyerror e funzioni principali compatibili con Yacc. Questa libreria normalmente non è molto utile, ma POSIX la richiede.

6.18. Ncurses-5.5

Il pacchetto Ncurses contiene librerie per la gestione indipendente dal terminale di schermi caratteri.

Tempo di costruzione approssimativo: 0.7 SBU
Spazio necessario su disco: 31 MB

6.18.1. Installazione di Ncurses

Dalla release di Ncurses-5.5, sono stati trovati e corretti una falla di memoria e qualche bug di display. Applicare questi fix:

```
patch -Np1 -i ../ncurses-5.5-fixes-1.patch
```

Preparare Ncurses per la compilazione:

```
./configure --prefix=/usr --with-shared --without-debug --enable-widec
```

Significato delle opzioni di configurazione:

`--enable-widec`
Questa scelta fa sì che siano costruite le librerie wide-character (es. `libncursesw.so.5.5`) al posto di quelle normali (es. `libncurses.so.5.5`). Queste librerie wide-character si possono usare sia in localizzazioni multibyte sia traditionali a 8-bit, mentre le librerie normali lavorano in modo corretto solo in localizzazioni a 8-bit. Le librerie wide-character e quelle normali sono compatibili a livello di sorgente, ma non come binari.

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Dare alle librerie Ncurses i permessi di esecuzione:

```
chmod -v 755 /usr/lib/*.5.5
```

Correggere una libreria che non dovrebbe essere eseguibile:

```
chmod -v 644 /usr/lib/libncurses++.a
```

Spostare le librerie nella directory `/lib`, dove ci si aspetta che si trovino:

```
mv -v /usr/lib/libncurses.so.5* /lib
```

Poiché le librerie sono state spostate, un link simbolico punta su un file non esistente. Ricrearlo:

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Molte applicazioni aspettano ancora il linker per riuscire a trovare le librerie Ncurses non-wide-character. Costringere tali applicazioni a fare il link alle librerie wide-character per mezzo di link simbolici e script

linker:

```
for lib in curses ncurses form panel menu ; do \
    rm -vf /usr/lib/lib${lib}.so ; \
    echo "INPUT(-l${lib}w)" >/usr/lib/lib${lib}.so ; \
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \
done &&
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

Infine assicurarsi che sia ancora possibile costruire le vecchie applicazioni che cercano `-lcurses` durante la costruzione:

```
echo "INPUT(-lncursesw)" >/usr/lib/libcursesw.so &&
ln -sfv libncurses.so /usr/lib/libcurses.so &&
ln -sfv libncursesw.a /usr/lib/libcursesw.a &&
ln -sfv libncurses.a /usr/lib/libcurses.a
```

Nota

Le istruzioni precedenti non creano le librerie Ncurses non-wide-character fintanto che nessun pacchetto installato con la compilazione da sorgenti non avrà un link ad esse in runtime. Se si devono avere tali librerie a causa di qualche applicazione solo binaria, costruirle con i seguenti comandi:

```
make distclean &&
./configure --prefix=/usr --with-shared --without-normal \
  --without-debug --without-cxx-binding &&
make sources libs &&
cp -av lib/lib*.so.5* /usr/lib
```

6.18.2. Contenuti di Ncurses

Programmi installati: captoinfo (link a tic), clear, infocmp, infotocap (link a tic), reset (link a tset), tack, tic, toe, tput e tset
Librerie installate: libcursesw.{a,so} (symlink e linker script a libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} e i loro corrispettivi non-wide-character senza "w" nei nomi delle librerie.

Brevi descrizioni

captoinfo	Converte una descrizione termcap in una descrizione terminfo
clear	Pulisce lo schermo, se possibile
infocmp	Confronta o stampa descrizioni terminfo
infotocap	Converte una descrizione terminfo in una descrizione termcap
reset	Reinizializza un terminale con i suoi valori di default
tack	Il terminfo action checker; è usato principalmente per testare l'accuratezza di un inserimento nel database terminfo
tic	Il compilatore delle descrizioni degli inserimenti in terminfo, che traduce un file terminfo dal formato sorgente al formato binario necessario per le routine delle librerie ncurses. Un file terminfo contiene informazioni sulle capacità di un certo terminale

toe	Elenca tutti i tipi di terminale disponibili, dando per ciascuno il suo nome primario e la sua descrizione.
tput	Rende disponibili alla shell i valori delle risorse dipendenti da terminale; può anche essere usato per resettare o inizializzare un terminale o riportare il suo nome lungo
tset	Può essere usato per inizializzare terminali
`libcurses`	Un link verso `libncurses`
`libncurses`	Contiene funzioni per visualizzare il testo in molti modi complessi su uno schermo terminale; un buon esempio dell'uso di queste funzioni è il menu visualizzato durante il **make menuconfig** del kernel
`libform`	Contiene funzioni per implementare i form
`libmenu`	Contiene funzioni per implementare i menu
`libpanel`	Contiene funzioni per implementare i panel

6.19. Procps-3.2.6

Il pacchetto Procps contiene programmi per monitorare i processi.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 2.3 MB

6.19.1. Installazione di Procps

Compilare il pacchetto:

```
make
```

Questo pacchetto non ha una suite di test.

Installare il pacchetto:

```
make install
```

6.19.2. Contenuti di Procps

Programmi installati: free, kill, pgrep, pkill, pmap, ps, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w e watch
Libreria installata: libproc.so

Brevi descrizioni

free	Riporta l'ammontare di memoria libera e usata nel sistema, sia della memoria fisica che di swap
kill	Invia segnali ai processi
pgrep	Cerca processi sulla base del loro nome e di altri attributi
pkill	Invia segnali a processi sulla base del loro nome e di altri attributi
pmap	Riporta la mappa di memoria del dato processo
ps	Elenca i processi attualmente in esecuzione
skill	Invia segnali a processi che corrispondono ai criteri specificati
slabtop	Mostra in tempo reale informazioni dettagliate sulla slap cache del kernel
snice	Cambia la priorità di schedulazione di processi che corrispondono ai criteri dati
sysctl	Modifica i parametri del kernel in run time
tload	Stampa un grafico del carico medio del sistema corrente
top	Mostra un elenco dei processi più intensivi della CPU; fornisce una panoramica dell'attività del processore in tempo reale
uptime	Riporta da quanto tempo il sistema è in funzione, quanti utenti sono connessi e il carico medio del sistema
vmstat	Riporta statistiche sulla memoria virtuale, dando informazioni su processi, memoria, paginazione, blocchi Input/Output (IO), traps e attività della CPU

w Mostra quali utenti sono attualmente connessi, dove e da quando

watch Esegue ripetutamente un dato comando, visualizzando la prima schermata piena del suo output; questo permette all'utente di verificare come cambia l'output nel tempo

`libproc` Contiene le funzioni usate dalla maggior parte dei programmi in questo pacchetto

6.20. Sed-4.1.5

Il pacchetto Sed contiene uno stream editor.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 6.4 MB

6.20.1. Installazione di Sed

Preparare Sed per la compilazione:

```
./configure --prefix=/usr --bindir=/bin
```

Significato delle nuove opzioni di configurazione:

`--enable-html`
Costruisce la documentazione HTML.

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.20.2. Contenuti di Sed

Programma installato: sed

Brevi descrizioni

sed Filtra e trasforma file di testo in un singolo passaggio

6.21. Libtool-1.5.22

Il pacchetto Libtool contiene lo script di supporto alle librerie generiche GNU. Racchiude la complessità dell'uso di librerie condivise in una interfaccia consistente e portabile.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 16.6 MB

6.21.1. Installazione di Libtool

Preparare Libtool per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.21.2. Contenuti di Libtool

Programmi installati: libtool e libtoolize
Librerie installate: libltdl.[a,so]

Brevi descrizioni

libtool	Fornisce servizi di supporto generalizzato alla costruzione delle librerie
libtoolize	Fornisce un metodo standard per aggiungere supporto **libtool** a un pacchetto
`libltdl`	Nasconde le diverse difficoltà delle librerie dlopening

6.22. Perl-5.8.8

Il pacchetto Perl contiene il Practical Extraction and Report Language

Tempo di costruzione approssimativo: 1.5 SBU
Spazio necessario su disco: 143 MB

6.22.1. Installazione di Perl

Per prima cosa creare il file di base `/etc/hosts` al quale si riferirà uno dei file di configurazione di Perl e che sarà usato dalla suite di test se verrà eseguita.

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Per avere il pieno controllo su come Perl viene configurato, eseguire lo script interattivo **Configure** e modificare manualmente il modo in cui il pacchetto è costruito. Se i default che esso auto-rileva sono adatti, preparare Perl per la compilazione con:

```
./configure.gnu --prefix=/usr \
    -Dman1dir=/usr/share/man/man1 \
    -Dman3dir=/usr/share/man/man3 \
    -Dpager="/usr/bin/less -isR"
```

Significato delle opzioni di configurazione:

`-Dpager="/usr/bin/less -isR"`
Questo corregge un errore nel modo in cui **perldoc** invoca il programma **less**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Da quando Groff non è più installato, **Configure** crede che non si vogliono le man page per Perl. Inserendo questi parametri si annulla questa decisione.

Compilare il pacchetto:

```
make
```

Per testare i risultati, eseguire: **`make test`**.

Installare il pacchetto:

```
make install
```

6.22.2. Contenuti di Perl

Programmi installati: a2p, c2ph, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.8.8 (link a perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, psed (link a s2p), pstruct (link a c2ph), s2p, splain e xsubpp
Librerie installate: Diverse centinaia, che non possono essere elencate qui

Brevi descrizioni

a2p	Traduce awk in Perl
c2ph	Fa il dump di strutture C come se fossero generate da **cc -g -S**
dprofpp	Visualizza dati di profilo Perl
enc2xs	Costruisce un'estensione Perl per il modulo Encode dagli Unicode Character Mappings o dai Tcl Encoding Files
find2perl	Traduce in Perl comandi **find**
h2ph	Converte file header C `.h` in file header Perl `.ph`
h2xs	Converte file header C `.h` in estensioni Perl
instmodsh	Uno script shell per esaminare i moduli Perl installati; può anche creare un tarball da un modulo installato
libnetcfg	Può essere usato per configurare la `libnet`
perl	Combina alcune delle migliori caratteristiche di C, **sed**, **awk** e **sh** in un singolo linguaggio che è come un coltellino svizzero
perl5.8.8	Un link fisico a **perl**
perlbug	Usato per generare dei report di bug su Perl o sui moduli che ne fanno parte e inviarli via email
perlcc	Genera eseguibili da programmi Perl
perldoc	Visualizza una parte di documentazione in formato pod che è inclusa nell'albero di installazione di Perl o in uno script Perl
perlivp	La Perl Installation Verification Procedure (procedura di verifica dell'installazione di Perl); può essere usata per verificare che Perl e le sue librerie siano state installate correttamente
piconv	Una versione Perl del convertitore della codifica caratteri **iconv**
pl2pm	Un tool rudimentale per convertire file Perl4 `.pl` in moduli Perl5 `.pm`
pod2html	Converte file dal formato pod al formato HTML
pod2latex	Converte file dal formato pod al formato LaTeX
pod2man	Converte dati pod in input formattato *roff
pod2text	Converte dati pod in testo formattato ASCII
pod2usage	Stampa su file i messaggi di uso a partire da documenti pod inclusi
podchecker	Verifica la sintassi dei file di documentazione in formato pod
podselect	Visualizza le sezioni selezionate di documentazione pod

psed	Una versione Perl dello stream editor **sed**
pstruct	Scarica strutture C come se fossero generate da comandi **cc -g -S**
s2p	Traduce script **sed** in Perl
splain	Usato per forzare in Perl la dignostica di avviso prolissa
xsubpp	Converte codice Perl XS in codice C

6.23. Readline-5.1

Il pacchetto Readline è un insieme di librerie che offre editing a linea di comando e capacità di history.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 10.2 MB

6.23.1. Installazione di Readline

In origine gli sviluppatori hanno corretto molti problemi a partire dalla release iniziale di Readline-5.1. Applicare questi fix:

```
patch -Np1 -i ../readline-5.1-fixes-3.patch
```

La reinstallazione di Readline farà sì che le vecchie librerie siano spostate in <libraryname>.old. Se questo normalmente non è un problema, in qualche caso può causare un bug di link in **ldconfig**. Ciò si può evitare eseguendo i due sed seguenti:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Preparare Readline per la compilazione:

```
./configure --prefix=/usr --libdir=/lib
```

Compilare il pacchetto:

```
make SHLIB_XLDFLAGS=-lncurses
```

Significato delle opzioni di make:

`SHLIB_LIBS=-lncurses`
Questa opzione forza il collegamento di Readline alla libreria `libncurses`(in realtà `libncursesw`).

Questo pacchetto non ha una suite di test.

Installare il pacchetto:

```
make install
```

Dare alle librerie dinamiche di Readline permessi più appropriati:

```
chmod -v 755 /lib/lib{readline,history}.so*
```

Ora spostare le librerie statiche in una locazione più appropriata:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Successivamente rimuovere i file `.so` in `/lib` e ricollegarli in `/usr/lib`.

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.5 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

6.23.2. Contenuti di Readline

Librerie installate: libhistory.{a,so} e libreadline.{a,so}

Brevi descrizioni

`libhistory`	Fornisce un'interfaccia utente consistente per richiamare linee o history
`libreadline`	Aiuta nella consistenza dell'interfaccia utente tra diversi programmi che devono fornire un'interfaccia a linea di comando

6.24. Zlib-1.2.3

Il pacchetto Zlib contiene routine di compressione e decompressione usate da alcuni programmi.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 3.1 MB

6.24.1. Installazione di Zlib

> **Nota**
>
> Zlib è nota per la costruzione scorretta della propria libreria condivisa se CFLAGS è specificato nell'ambiente. Se si usa una specifica variabile CFLAGS, assicurarsi di aggiungere la direttiva *-fPIC* alla variabile CFLAGS per la durata del comando configure precedente, quindi rimuoverla successivamente.

Preparare Zlib per la compilazione:

```
./configure --prefix=/usr --shared --libdir=/lib
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **make check**.

Installare la libreria condivisa:

```
make install
```

Il comando precedente ha installato un file .so in /lib. Lo rimuoveremo e lo ricollegheremo in /usr/lib:

```
rm -v /lib/libz.so
ln -sfv ../../lib/libz.so.1.2.3 /usr/lib/libz.so
```

Costruire la libreria statica:

```
make clean
./configure --prefix=/usr
make
```

Per testare di nuovo i risultati digitare: **make check**.

Installare la libreria statica:

```
make install
```

Correggere i permessi sulla libreria statica:

```
chmod -v 644 /usr/lib/libz.a
```

6.24.2. Contenuti di Zlib

Librerie installate: libz.{a,so}

Brevi descrizioni

`libz` Contiene funzioni di compressione e decompressione usate da alcuni programmi

6.25. Autoconf-2.59

Il pacchetto autoconf contiene programmi per produrre script di shell che possono automaticamente configurare il codice sorgente.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 7.2 MB

6.25.1. Installazione di Autoconf

Preparare Autoconf per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**. Questo richiede molto tempo, circa 3 SBU. Inoltre 2 test che usano Automake non vengono eseguiti. Per eseguire un controllo completo Autoconf può essere testato nuovamente dopo che Automake è stato installato.

Installare il pacchetto:

```
make install
```

6.25.2. Contenuti di Autoconf

Programmi installati: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames

Brevi descrizioni

autoconf Produce script di shell che configurano automaticamente il codice sorgente dei pacchetti software per adattarli alle varie tipologie di sistemi Unix-like. Gli script di configurazione prodotti sono indipendenti; per avviarli non è necessario il programma Autoconf

autoheader Crea file modello della dichiarazione C *#define* da usare col configuratore

autom4te È un wrapper per il processore di macro M4

autoreconf Esegue automaticamente **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, e **libtoolize** nell'ordine corretto per risparmiare tempo quando vengono modificati i file template **autoconf** e **automake**

autoscan Può aiutare a creare un file `configure.in` per un pacchetto software. Esamina i file sorgenti in un albero di directory, cercando problemi di portabilità, e crea un file `configure.scan` che serve come un `configure.in` preliminare per il pacchetto

autoupdate Modifica un file `configure.in` che richiama macro di **autoconf** con il loro vecchio nome per utilizzare i nomi di macro correnti

ifnames Può essere di aiuto quando si scrive un `configure.in` per un pacchetto software. Evidenzia gli identificatori che il pacchetto utilizza nel preprocessore C. Se un pacchetto è già stato configurato per avere una certa portabilità questo programma può aiutare a determinare quale **configure** deve essere deve essere controllato. Può colmare delle

lacune nel file `configure.in` generato da **autoscan**

6.26. Automake-1.9.6

Il pacchetto Automake contiene programmi per generare Makefile da usare con Autoconf.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 7.9 MB

6.26.1. Installazione di Automake

Preparare Automake per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**. Questo richiede molto tempo, circa 10 SBU.

Installare il pacchetto:

```
make install
```

6.26.2. Contenuti di Automake

Programmi installati: acinstall, aclocal, aclocal-1.9.6, automake, automake-1.9.6, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree e ylwrap

Brevi descrizioni

acinstall	Uno script che installa file stile aclocal M4
aclocal	Genera file `aclocal.m4` basati sui contenuti dei file `configure.in`
aclocal-1.9.6	Link fisico a **aclocal**
automake	Tool per la generazione automatica di file `Makefile.in` da file `Makefile.am`. Per creare tutti i file `Makefile.in` per un pacchetto, avviare questo programma nella top-level directory. Scansionando il file `configure.in` trova automaticamente ogni `Makefile.am` e genera il corrispondente `Makefile.in`
automake-1.9.6	Link fisico a **automake**
compile	Un wrapper per compilatori
config.guess	Script che tenta di prevedere la tripletta canonica per il dato host, compilazione o architettura target
config.sub	Script di validazione delle soubroutine di configurazione

depcomp	Script per compilare un programma in modo che in aggiunta all'output desiderato vengano generate anche informazioni di dipendenze
elisp-comp	Compilatore di codice Lisp per Emacs
install-sh	Script che installa un programma, uno script o un file dati
mdate-sh	Script che stampa gli orari di modifica di un file o directory
missing	Script che agisce come un sostituto comune per programmi GNU mancanti durante una installazione
mkinstalldirs	Script che crea un albero di directory
py-compile	Compila un programma Python
symlink-tree	Script che crea un albero di link simbolici verso un albero di directory
ylwrap	Wrapper per **lex** e **yacc**

6.27. Bash-3.1

Il pacchetto Bash contiene la Bourne-Again SHell.

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 25.8 MB

6.27.1. Installazione di Bash

Se si è scaricato il tarball della documentazione Bash, e si desidera installare la documentazione HTML, digitare i seguenti comandi:

```
tar -xvf ../bash-doc-3.1.tar.gz &&
sed -i "s|htmldir = @htmldir@|htmldir = /usr/share/doc/bash-3.1|" \
    Makefile.in
```

Gli sviluppatori del programma hanno corretto svariati problemi dal rilascio iniziale di Bash-3.1. Applicare le correzioni:

```
patch -Np1 -i ../bash-3.1-fixes-8.patch
```

Preparare Bash per la compilazione:

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc --with-installed-readline
```

Significato delle opzioni di configurazione:

`--with-installed-readline`
Questa opzione dice a Bash di usare la libreria `readline` già installata nel sistema piuttosto che usare la propria versione di readline.

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make tests`**.

Installare il pacchetto:

```
make install
```

Eseguire il programma **bash** appena compilato (sostituendo quello si sta attualmente eseguendo):

```
exec /bin/bash --login +h
```

Nota

I parametri usati rendono il processo **bash** una shell di login interattiva e continuano a disabilitare l'hashing, così che i nuovi programmi siano trovati non appena sono disponibili.

6.27.2. Contenuti di Bash

Programmi installati: bash, bashbug e sh (link a bash)

Brevi descrizioni

bash Un interprete di comandi largamente utilizzato. Esegue molti tipi di espansioni e sostituzioni in una data linea di comando prima di eseguirla, il che rende questo interprete un potente strumento

bashbug Script di shell che aiuta l'utente a comporre e spedire rapporti standard di bug riguardanti **bash**

sh Link simbolico al programma **bash**. Quando invocato come **sh**, **bash** prova a simulare l'ambiente di avvio della versione storica di **sh** il più fedelmente possibile, rimanendo al contempo conforme allo standard POSIX

6.28. Bzip2-1.0.3

Il pacchetto Bzip2 contiene programmi per comprimere e decomprimere file. La compressione di file di testo con **bzip2** raggiunge una migliore percentuale di compressione rispetto al tradizionale **gzip**.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 5.3 MB

6.28.1. Installazione di Bzip2

Applicare una patch per installare la documentazione per questo pacchetto:

```
patch -Np1 -i ../bzip2-1.0.3-install_docs-1.patch
```

Il comando **bzgrep** non interpreta i caratteri escape '|' e '&' nei nomi di file che gli vengono passati. Questo permette a comandi arbitrari di essere eseguiti con i privilegi dell'utente che lancia **bzgrep**. Applicare la patch seguente per risolvere questo:

```
patch -Np1 -i ../bzip2-1.0.3-bzgrep_security-1.patch
```

Lo script **bzdiff** usa ancora il deprecato programma **tempfile**. Aggiornarlo affinché venga usato **mktemp** al suo posto:

```
sed -i 's@tempfile -d /tmp -p bz@mktemp -p /tmp@' bzdiff
```

Preparare Bzip2 per la compilazione con:

```
make -f Makefile-libbz2_so
make clean
```

Significato del parametro di make:

`-f Makefile-libbz2_so`
: Questo farà sì che Bzip2 venga costruito usando un diverso file `Makefile`, in questo caso il file `Makefile-libbz2_so`, che crea una libreria dinamica `libbz2.so` e collega le utilità di Bzip2 verso questo.

Compilare e testare il pacchetto:

```
make
```

Se si sta reinstallando Bzip2, bisogna prima dare **`rm -vf /usr/bin/bz*`**, altrimenti il successivo **make install** fallirà.

Installare i programmi:

```
make install
```

Ora installare i binari **bzip2** condivisi nella directory `/bin`, quindi creare qualche necessario link simbolico, e pulire:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.28.2. Contenuti di Bzip2

Programmi installati: bunzip2 (link a bzip2), bzcat (link a bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless e bzmore
Librerie installate: libbz2.{a,so}

Brevi descrizioni

bunzip2	Decomprime file bzippati.
bzcat	Decomprime verso lo standard output.
bzcmp	Esegue **cmp** su file bzippati.
bzdiff	Esegue **diff** su file bzippati.
bzgrep	Esegue **grep** su file bzippati.
bzegrep	Esegue **egrep** su file bzippati
bzfgrep	Esegue **fgrep** su file bzippati
bzip2	Comprime file utilizzando l'algoritmo di compressione del testo con ordinamento a blocchi Burros-Wheeler con codifica Huffman. Il tasso di compressione in genere è considerevolmente migliore di quello raggiunto da compressori più convenzionali che utilizzano «Lempel-Ziv», come **gzip**.
bzip2recover	Tenta di recuperare dati da file bzippati danneggiati.
bzless	Esegue **less** su file bzippati.
bzmore	Esegue **more** su file bzippati.
`libbz2*`	La libreria che implementa la compressione con ordinamento a blocchi senza perdita di dati, usando l'algoritmo Burrows-Wheeler.

6.29. Diffutils-2.8.1

Il pacchetto Diffutils contiene programmi che mostrano le differenze tra file o directory.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 6.3 MB

6.29.1. Installazione di Diffutils

POSIX richiede il comando **diff** per gestire i caratteri di spazio bianco secondo la localizzazione corrente. La patch seguente fissa i problemi di non compilazione:

```
patch -Np1 -i ../diffutils-2.8.1-i18n-1.patch
```

La patch precedente farà sì che il sistema di costruzione di Diffutils provi a ricostruire la pagina man `diff.1` usando il programma non disponibile **help2man**. Il risultato è una pagina man che **diff** non riesce a leggere. Ciò può essere evitato aggiornando l'orario del file `man/diff.1`:

```
touch man/diff.1
```

Preparare Diffutils per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

6.29.2. Contenuti di Diffutils

Programmi installati: cmp, diff, diff3 e sdiff

Brevi descrizioni

cmp	Confronta due file e riporta se o in quali byte differiscono
diff	Confronta due file o directory e riporta in quali linee i file differiscono
diff3	Confronta tre file linea per linea
sdiff	Unisce due file e interattivamente mostra i risultati

6.30. E2fsprogs-1.39

Il pacchetto E2fsprogs contiene le utilità per la gestione del file system `ext2`. Supporta anche il file system journaling `ext3`.

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 31.2 MB

6.30.1. Installazione di E2fsprogs

Si raccomanda di costruire E2fsprogs in una sottodirectory dell'albero dei sorgenti:

```
mkdir -v build
cd build
```

Preparare E2fsprogs per la compilazione:

```
../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs --disable-evms
```

Significato delle opzioni di configurazione:

`--with-root-prefix=""`
Certi programmi (come il programma **e2fsck**) sono considerati essenziali. Quando, per esempio, `/usr` non è montata, questi programmi devono sempre essere disponibili. Essi risiedono in una directory come `/lib` o `/sbin`. Se questa opzione non viene passata al configuratore di E2fsprogs, i programmi vengono installati nella directory `/usr`.

`--enable-elf-shlibs`
Crea le librerie condivise usate da alcuni programmi in questo pacchetto.

`--disable-evms`
Disabilita la costruzione del plugin Enterprise Volume Management System (EVMS). Questo plugin non è aggiornato con le ultime interfacce interne di EVMS e EVMS non è installato come parte di un sistema LFS base, perciò il plugin non è richiesto. Vedere il sito web di EVMS *http://evms.sourceforge.net/* per ulteriori informazioni riguardanti lo stesso EVMS.

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Uno dei test di E2fsprogs cercherà di allocare 256 MB di memoria. Se oltre a questa non si dispone di una quantità significativa di memoria RAM, si raccomanda di abilitare uno spazio swap sufficiente per il test. Vedere Sezione 2.3, «Creazione di un file system sulla partizione» e Sezione 2.4, «Montaggio della nuova partizione» per i dettagli sulla creazione e l' abilitazione di spazio swap.

Installare binari e documentazione:

```
make install
```

Installare le librerie condivise:

```
make install-libs
```

6.30.2. Contenuti di E2fsprogs

Programmi installati: badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, filefrag, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs e uuidgen.
Librerie installate: libblkid.{a,so}, libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libss.{a,so} e libuuid.{a,so}

Brevi descrizioni

badblocks	Cerca blocchi corrotti su di un dispositivo (di solito una partizione del disco)
blkid	Utilità a linea di comando per localizzare e stampare gli attributi dei blocchi del dispositivo
chattr	Cambia gli attributi dei file in un file system `ext2`; cambia anche i file system `ext3` e la versione journaling dei file system `ext2`
compile_et	Un compilatore di tabella di errore; converte una tabella di nomi e messaggi di codici errore in un file sorgente C utilizzabile con la libreria `com_err`
debugfs	Un debugger del file system; può essere utilizzato per esaminare e cambiare lo stato di un file system `ext2`
dumpe2fs	Stampa le informazioni su super blocchi e gruppi di blocchi per il file system presente in un dato dispositivo
e2fsck	È utilizzato per verificare, e eventualmente riparare, file system `ext2` e file system `ext3`
e2image	Utilizzato per salvare dati critici del file system `ext2` in un file
e2label	Visualizza o cambia l'etichetta del file system `ext2` presente in un dato dispositivo
filefrag	Riporta lo stato di frammentazione di un file particolare
findfs	Trova un file system attraverso l'etichetta o l'UUID (Universally Unique Identifier)
fsck	È usato per verificare, e opzionalmente riparare, file system
fsck.ext2	Per default verifica i file system `ext2`
fsck.ext3	Per default verifica i file system `ext3`
logsave	Salva l'output di un comando in un file di log
lsattr	Elenca gli attributi dei file in un file system `ext2`
mk_cmds	Converte una tabella di nomi comando e messaggi di aiuto in un file sorgente C utilizzabile con la libreria di sottosistema `libss`
mke2fs	Crea un file system `ext2` o `ext3` sul dispositivo indicato
mkfs.ext2	Per default crea file system `ext2`
mkfs.ext3	Per default crea file system `ext3`
mklost+found	è usato per creare una directory `lost+found` su un file system `ext2`; pre-alloca blocchi disco su questa directory per alleggerire il lavoro di **e2fsck**
resize2fs	Può essere utilizzato per espandere o ridurre un file system `ext2`

tune2fs	Utilizzato per aggiustare parametri regolabili del file system in un file system `ext2`
uuidgen	Crea un nuovo UUID. Ciascun nuovo UUID può essere ragionevolmente considerato unico tra tutti gli UUID creati, sul sistema locale e su altri sistemi, in passato e in futuro
`libblkid`	Contiene routine per l'identificazione dei dispositivi e l'estrazione di token
`libcom_err`	Routine per visualizzare gli errori comuni
`libe2p`	Usato da **dumpe2fs**, **chattr** e **lsattr**
`libext2fs`	Contiene rutine per abilitare programmi utente a manipolare un file system `ext2`
`libss`	Usato da **debugfs**
`libuuid`	Contiene routine per generare identificatori unici per oggetti che possono essere accessibili al di fuori del sistema locale

6.31. File-4.17

Il pacchetto File contiene una utilità per determinare il tipo di uno o più file.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 7.5 MB

6.31.1. Installazione di File

Preparare File per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è dotato di una suite di test.

Installare il pacchetto:

```
make install
```

6.31.2. Contenuti di File

Programma installato: file
Libreria installata: libmagic.{a,so}

Brevi descrizioni

file	Prova a classificare ciascun file dato; lo fa eseguendo diversi test— test del file system, test del magic number e test di linguaggio
`libmagic`	Contiene routine per il riconoscimento del magic number, usata dal programma **file**

6.32. Findutils-4.2.27

Il pacchetto Findutils contiene programmi per trovare file. Questi programmi sono fatti per cercare ricorsivamente attraverso un albero di directory e per creare, mantenere e cercare in un database (spesso più velocemente della ricerca ricorsiva, ma inapplicabile se il database non è stato aggiornato recentemente).

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 12 MB

6.32.1. Installazione di Findutils

Preparare Findutils per la compilazione:

```
./configure --prefix=/usr --libexecdir=/usr/lib/findutils \
    --localstatedir=/var/lib/locate
```

Significato delle opzioni di configurazione:

`--localstatedir`
Questa opzione cambia la locazione del database **locate** in modo che sia in `/var/lib/locate`, locazione conforme a FHS.

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Alcuni script nel pacchetto LFS-Bootscripts dipendono da **find**. Poiché `/usr` potrebbe non essere disponibile nelle prime fasi dell'avvio, questo programma deve risiedere nella partizione root. Inoltre lo script **updatedb** deve essere modificato per correggere un percorso esplicito:

```
mv -v /usr/bin/find /bin
sed -i -e 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.32.2. Contenuti di Findutils

Programmi installati: bigram, code, find, frcode, locate, updatedb e xargs

Brevi descrizioni

bigram Era utilizzato per produrre i database **locate**

code Era utilizzato per produrre database **locate**; è l'antenato di **frcode**.

find Cerca negli alberi di directory specificati i file rispondenti a determinati criteri

frcode È chiamato da **updatedb** per comprimere la lista di nomi di file; usa la front-compression, riducendo la dimensione del database di un fattore da 4 a 5.

locate Cerca in un database di nomi di file e riporta i nomi che contengono una data stringa o

corrispondono ad un dato pattern

updatedb Aggiorna il database **locate**; scansiona l'intero file system (inclusi altri file system che siano attualmente montati, tranne se specificato diversamente) e inserisce nel database ogni nome di file che trova

xargs Può essere usato per applicare un dato comando ad una lista di file

6.33. Flex-2.5.33

Il pacchetto Flex contiene un'utilità per generare programmi che riconoscono pattern nel testo.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 8.4 MB

6.33.1. Installazione di Flex

Preparare Flex per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Ci sono alcuni pacchetti che si aspettano di trovare la libreria `lex` in `/usr/lib`. Creare un link simbolico per risolvere questo problema:

```
ln -sv libfl.a /usr/lib/libl.a
```

Alcuni programmi non sono ancora a conoscenza di **flex** e cercano di eseguire il suo predecessore, **lex**. Per supportare questi programmi, creare un script wrapper chiamato `lex` che chiama `flex` in modalità emulazione di **lex**:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

6.33.2. Contenuti di Flex

Programmi installati: flex e lex
Libreria installata: libfl.a

Brevi descrizioni

flex	Un tool per generare programmi che riconoscono pattern nel testo; permette la versatilità di specificare le regole per trovare pattern, eliminando il bisogno di sviluppare un programma specializzato
lex	Uno script che esegue **flex** in modalità emulazione **lex**

`libfl.a` La libreria di `flex`

6.34. GRUB-0.97

Il pacchetto GRUB contiene il GRand Unified Bootloader.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 10.2 MB

6.34.1. Installazione di GRUB

Questo pacchetto è noto per avere problemi quando vengono cambiati i suoi flag di ottimizzazione di default (incluse le opzioni *-march* e *-mcpu*). Se dovessero essere state definite variabili di ambiente che disabilitano le ottimizzazioni di default, come i flag `CFLAGS` e `CXXFLAGS`, eliminarle quando si costruisce GRUB.

Iniziare applicando la patch seguente per permettere un miglior riconoscimento dei drive, correggere alcuni problemi GCC 4.x e fornire un supporto SATA migliore per alcuni disk controller:

```
patch -Np1 -i ../grub-0.97-disk_geometry-1.patch
```

Preparare GRUB per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
mkdir -v /boot/grub
cp -v /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

Sostituire `i386-pc` con qualunque directory sia appropriata per il proprio hardware.

La directory `i386-pc` contiene alcuni file `*stage1_5`, uno per ogni file system. Controllare i file disponibili e copiare quelli appropriati nella directory `/boot/grub`. Molti utenti copieranno i file `e2fs_stage1_5` e/o `reiserfs_stage1_5`.

6.34.2. Contenuti di GRUB

Programmi installati: grub, grub-install, grub-md5-crypt, grub-set-default, grub-terminfo e mbchk

Brevi descrizioni

grub	La shell di comando del Grand Unified Bootloader
grub-install	Installa GRUB sul dispositivo specificato
grub-md5-crypt	Esegue la cifratura di una password in formato MD5
grub-set-default	Imposta la voce boot di default per GRUB
grub-terminfo	Genera un comando terminfo da un nome terminfo; può essere impiegato se si usa

	un terminale sconosciuto
mbchk	Verifica il formato di un kernel multi-boot

6.35. Gawk-3.1.5

Il pacchetto Gawk contiene programmi per la manipolazione dei file di testo.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 18.2 MB

6.35.1. Installazione di Gawk

In alcune circostanze, Gawk-3.1.5 cerca di liberare un grosso banco di memoria non allocato. Questo bug è corretto dalla patch seguente:

```
patch -Np1 -i ../gawk-3.1.5-segfault_fix-1.patch
```

Preparare Gawk per la compilazione:

```
./configure --prefix=/usr --libexecdir=/usr/lib
```

A causa di un bug nello script **configure**, Gawk non riesce a rilevare alcuni aspetti del supporto per la localizzazione di Glibc. Questo bug porta ad es. a fallimenti della suite di test di Gettext. Aggirare il problema aggiungendo a `config.h` le macro definizioni che mancano:

```
cat >>config.h <<"EOF"
#define HAVE_LANGINFO_CODESET 1
#define HAVE_LC_MESSAGES 1
EOF
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.35.2. Contenuti di Gawk

Programmi installati: awk (link a gawk), gawk, gawk-3.1.5, grcat, igawk, pgawk, pgawk-3.1.5 e pwcat

Brevi descrizioni

awk	Un link verso **gawk**
gawk	Un programma per manipolare file di testo; è l'implementazione GNU di **awk**
gawk-3.1.5	Un link fisico verso **gawk**
grcat	Fa il dump del database dei gruppi `/etc/group`
igawk	Dà a **gawk** la capacità di includere file
pgawk	La versione profilata di **gawk**
pgawk-3.1.5	Link fisico verso **pgawk**

pwcat Fa il dump del database delle password `/etc/passwd`

6.36. Gettext-0.14.5

Il pacchetto Gettext contiene utilità per l'internazionalizzazione e la localizzazione. Questo permette ai programmi di essere compilati con il NLS (Native Language Support), abilitandoli ad emettere messaggi nel linguaggio nativo dell'utente.

Tempo di costruzione approssimativo: 1 SBU
Spazio necessario su disco: 65 MB

6.36.1. Installazione di Gettext

Preparare Gettext per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**. Questo richiede molto tempo, circa 5 SBU.

Installare il pacchetto:

```
make install
```

6.36.2. Contenuti di Gettext

Programmi installati: autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext e xgettext
Librerie installate: libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so} e libgettextsrc.so

Brevi descrizioni

autopoint	Copia i file dell'infrastruttura standard di Gettext in un pacchetto sorgente
config.charset	Emette una tabella dipendente dal sistema di alias di codifica dei caratteri
config.rpath	Emette un set di variabili dipendenti dal sistema, che decrivono come definire il path di ricerca runtime delle librerie condivise in un eseguibile
envsubst	Sostituisce variabili di ambiente in stringhe formato shell
gettext	Traduce un messaggio dal linguaggio naturale al linguaggio dell'utente, cercando la traduzione in un catalogo di messaggi
gettext.sh	Serve principalmente come libreria di funzioni di shell per gettext
gettextize	Copia tutti i file standard Gettext nella data directory di primo livello di un pacchetto, per iniziare ad internazionalizzarlo
hostname	Visualizza il nome host del network in varie forme
msgattrib	Filtra i messaggi di un catalogo di traduzione in accordo con i loro attributi e manipola gli attributi
msgcat	Concatena ed unisce i file `.po` specificati

msgcmp	Confronta due file `.po` per verificare che entrambi contengano lo stesso set di stringhe msgid
msgcomm	Trova i messaggi che sono comuni tra i file `.po` specificati
msgconv	Converte un catalogo di traduzione in una diversa codifica caratteri
msgen	Crea un catalogo di traduzione in inglese
msgexec	Applica un comando a tutte le traduzioni di un catalogo di traduzioni
msgfilter	Applica un filtro a tutte le traduzioni di un catalogo di traduzioni
msgfmt	Genera un catalogo di messaggi binari da un catalogo di traduzioni
msggrep	Estrae tutti i messaggi di un catalogo traduzioni che rispettano un dato pattern o appartengono a degli specificati file sorgenti
msginit	Crea un nuovo file `.po`, inizializzando le meta-informazioni con valori dall'ambiente dell'utente
msgmerge	Combina due traduzioni grezze in un singolo file
msgunfmt	Decompila un catalogo di messaggi binari in un testo di traduzione grezzo
msguniq	Unifica traduzioni duplicate in un catalogo di traduzioni
ngettext	Visualizza traduzioni in linguaggio nativo di un messaggio testuale la cui forma grammaticale dipende da un numero
xgettext	Estrae le linee traducibili del messaggio dal dato file sorgente per fare una prima bozza di traduzione
`libasprintf`	Definisce la classe *autosprintf*, che rende le routine di output formattate C usabili in programmi C++, per l'uso con le stringhe <*string*> e i flussi <*iostream*>
`libgettextlib`	Una libreria privata contenente routine comuni usate da vari programmi Gettext; queste non sono pensate per uso generale
`libgettextpo`	Usato per scrivere programmi specializzati che processano file `.po`; questa libreria è usata quando le applicazioni standard fornite con Gettext (come **msgcomm**, **msgcmp**, **msgattrib** e **msgen**) non bastano
`libgettextsrc`	Una libreria privata contenente routine comuni usate da vari programmi Gettext; non sono pensate per un uso generale

6.37. Grep-2.5.1a

Il pacchetto Grep contiene programmi per la ricerca nei file.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 4.8 MB

6.37.1. Installazione di Grep

L'attuale pacchetto Grep ha molti bug, specialmente nel supporto per le localizzazioni multibyte. RedHat ne ha corretti alcuni con la patch seguente:

```
patch -Np1 -i ../grep-2.5.1a-redhat_fixes-2.patch
```

Affinché siano superati i test aggiunti da questa patch, si devono cambiare i permessi per il file di test:

```
chmod +x tests/fmbtest.sh
```

Preparare Grep per la compilazione:

```
./configure --prefix=/usr --bindir=/bin
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.37.2. Contenuti di Grep

Programmi installati: egrep (un link a grep), fgrep (un link a grep) e grep

Brevi descrizioni

egrep Stampa le linee corrispondenti ad una espressione regolare estesa

fgrep Stampa le linee corrispondenti ad una lista di stringhe fissate

grep Stampa le linee corrispondenti ad una espressione regolare di base

6.38. Groff-1.18.1.1

Il pacchetto Groff contiene programmi per processare e formattare testo

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 39.2 MB

6.38.1. Installazione di Groff

Applicare la patch che aggiunge a Groff i device «ascii8» e «nippon»:

```
patch -Np1 -i ../groff-1.18.1.1-debian_fixes-1.patch
```

Nota

Questi device sono usati da Man-DB quando formatta le pagine di manuale non inglesi che non sono nella codifica ISO-8859-1. Attualmente non c'è nessuna patch funzionante per Groff-1.19.x che aggiunga questa funzionalità.

Molti font per lo schermo non contengono doppi apici Unicode e trattini. Modificare Groff per usare in alternativa gli equivalenti ASCII:

```
sed -i -e 's/2010/002D/' -e 's/2212/002D/' \
    -e 's/2018/0060/' -e 's/2019/0027/' font/devutf8/R.proto
```

Groff si aspetta che la variabile ambiente *PAGE* contenga la dimensione di default della carta. Per utenti negli Stati Uniti *PAGE=letter* è appropriata. Altrove *PAGE=A4* dovrebbe essere più adatta. Anche se la dimensione di default della carta è configurata durante la compilazione, può essere annullata più tardi inserendo o «A4» o «letter» al file `/etc/papersize`.

Preparare Groff per la compilazione:

```
PAGE=<paper_size> ./configure --prefix=/usr --enable-multibyte
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non ha una suite di test.

Installare il pacchetto:

```
make install
```

Alcuni programmi di documentazione, come **xman**, non funzioneranno correttamente senza i seguenti link simbolici:

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.38.2. Contenuti di Groff

Programmi installati: addftinfo, afmtodit, eqn, eqn2graph, geqn (link a eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link a tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit e troff

Brevi descrizioni

addftinfo	Legge un file di font troff e aggiunge informazioni in più sulla metrica dei font utilizzata dal sistema **groff**
afmtodit	Crea un font file per l'uso con **groff** e **grops**
eqn	Compila descrizioni di equazioni incluse entro file di input di troff in comandi che sono capiti da **troff**
eqn2graph	Converte una troff EQN (equazione) in una immagine tagliata
geqn	Un link verso **eqn**
grn	Un preprocessore **groff** per file gremlin
grodvi	Un driver per **groff** che produce il formato dvi di TeX
groff	Un front-end per il sistema di formattazione documenti di groff; normalmente esegue il programma **troff** e un post-processore appropriato per i dispositivi selezionati
groffer	Visualizza file groff e pagine man su terminali X e tty
grog	Legge file e deduce quali delle opzioni **groff** *-e*, *-man*, *-me*, *-mm*, *-ms*, *-p*, *-s* e *-t* sono necessarie per stampare i file e riporta il comando **groff** includendo queste opzioni
grolbp	È un driver **groff** per stampanti Canon CAPSL (stampanti laser delle serie LBP-4 e LBP-8)
grolj4	È un driver **groff** che produce output in formato PCL5 adatto ad una stampante HP LaserJet 4.
grops	Traduce l'output del GNU **troff** in PostScript
grotty	Traduce l'output del GNU **troff** in una forma adatta a dispositivi tipo terminale
gtbl	Un link a **tbl**
hpftodit	Crea un file font per l'uso con **groff -Tlj4** da un file metrico di font marcati HP
indxbib	Crea un indice invertito per i database bibliografici in uno specifico file per l'uso con **refer**, **lookbib** e **lkbib**
lkbib	Cerca in database bibliografici riferimenti che contengano chiavi specificate e riporta ogni riferimento trovato
lookbib	Stampa un prompt sullo standard-error (a meno che lo standard-input non sia un terminale), legge dallo standard-input una linea contenente un set di parole chiave, cerca nei database bibliografici di uno specificato file i riferimenti contenenti queste parole chiave, stampa sullo standard-output ogni riferimento trovato e ripete questo processo fino alla fine dell'input
mmroff	Un semplice preprocessore per **groff**

neqn	Formatta equazioni per output ASCII (American Standard Code for Information Interchange)
nroff	Uno script che emula il comando **nroff** usando **groff**
pfbtops	Traduce un font PostScript da formato `.pfb` ad ASCII
pic	Compila le descrizioni di immagini integrate in file di input di tipo troff o Tex in comandi comprensibili da TeX o **troff**
pic2graph	Converte un diagramma PIC in una immagine tagliata
post-grohtml	Traduce l'output del GNU **troff** in HTML
pre-grohtml	Traduce l'output del GNU **troff** in HTML
refer	Copia i contenuti di un file nello standard output, ad eccezione delle linee tra *.[* e *.]* che sono interpretate come citazioni e le linee tra *.R1* and *.R2* che sono interpretate come comandi su come processare le citazioni
soelim	Legge i file e sostituisce le linee del tipo *.so file* con il contenuto del *file* menzionato
tbl	Compila le descrizioni delle tabelle integrate nei file input di troff in comandi comprensibili da **troff**
tfmtodit	Crea un file font da usare con **groff -Tdvi**
troff	Un **troff** altamente compatibile con Unix; di solito deve essere invocato usando il comando **groff**, che avvierà anche preprocessori e post-processori nell'ordine appropriato e con le opzioni appropriate

6.39. Gzip-1.3.5

Il pacchetto Gzip contiene programmi per compattare e scompattare i file.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 2.2 MB

6.39.1. Installazione di Gzip

Gzip ha 2 vulnerabilità di sicurezza note. La seguente patch le risolve entrambe:

```
patch -Np1 -i ../gzip-1.3.5-security_fixes-1.patch
```

Preparare Gzip per la compilazione:

```
./configure --prefix=/usr
```

Lo script **gzexe** ha la locazione del binario **gzip** incorporata. Poiché in seguito sarà cambiata la locazione del binario, il seguente comando assicura che la nuova locazione venga inclusa nello script:

```
sed -i 's@"BINDIR"@/bin@g' gzexe.in
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Spostare il programma **gzip** nella directory `/bin` e creare alcuni link simbolici ad esso comunemente utilizzati:

```
mv -v /usr/bin/gzip /bin
rm -v /usr/bin/{gunzip,zcat}
ln -sv gzip /bin/gunzip
ln -sv gzip /bin/zcat
ln -sv gzip /bin/compress
ln -sv gunzip /bin/uncompress
```

6.39.2. Contenuti di Gzip

Programmi installati: compress (link a gzip), gunzip (link a gzip), gzexe, gzip, uncompress (link a gunzip), zcat (link a gzip), zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore e znew

Brevi descrizioni

compress	Comprime e decomprime file compressi
gunzip	Scompatta file gzippati
gzexe	Crea file eseguibili autoestraenti
gzip	Compatta i file forniti, usando la codifica Lempel-Ziv (LZ77)

uncompress	Decomprime file compressi
zcat	Scompatta i file forniti gzippati nello standard output
zcmp	Esegue **cmp** su file gzippati
zdiff	Esegue **diff** su file gzippati
zegrep	Esegue **egrep** su file gzippati.
zfgrep	Esegue **fgrep** su file gzippati
zforce	Forza un'estensione `.gz` su tutti i file forniti che sono gzippati, così che **gzip** non li compatterà di nuovo; questo può essere utile quando i nomi file sono stati troncati durante un trasferimento file
zgrep	Esegue **grep** su file gzippati
zless	Esegue **less** su file gzippati
zmore	Esegue **more** su file gzippati
znew	Ricompatta file dal formato **compress** al formato **gzip**: `.Z` diventa `.gz`

6.40. Inetutils-1.4.2

Il pacchetto Inetutils contiene programmi di base per il networking.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 8.9 MB

6.40.1. Installazione di Inetutils

Appicare una patch ad Inetutils per abilitarlo a compilare con GCC-4.0.3:

```
patch -Np1 -i ../inetutils-1.4.2-gcc4_fixes-3.patch
```

Non verranno installati tutti i programmi inclusi in Inetutils. Ciononostante il sistema di installazione di Inteutils cercherà di installare ugualmente tutte le pagine man. La patch seguente correggerà questa situazione:

```
patch -Np1 -i ../inetutils-1.4.2-no_server_man_pages-1.patch
```

Preparare Inetutils per la compilazione:

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
    --sysconfdir=/etc --localstatedir=/var \
    --disable-logger --disable-syslogd \
    --disable-whois --disable-servers
```

Significato delle opzioni di configurazione:

`--disable-logger`
questa opzione disabilita l'installazione da parte di Inetutils del programma **logger**, che è usato dagli script per passare messaggi al System Log Daemon. Non lo si installa perché Util-linux più tardi ne installa una versione migliore.

`--disable-syslogd`
Questa opzione inibisce l'installazione da parte di Inetutils del System Log Daemon, che è installato con il pacchetto Sysklogd.

`--disable-whois`
Questa opzione inibisce la costruzione del client **whois** di Inetutils, che è tristemente anziano. Istruzioni per un migliore client **whois** si trovano nel libro BLFS.

`--disable-servers`
Questa inibisce l'installazione dei vari server di rete inclusi come parte del pacchetto Inetutils. Questi server sono giudicati non appropriati in un sistema LFS di base. Alcuni non sono sicuri per natura, e sono considerati sicuri solo su reti accreditate. Ulteriori informazioni possono essere trovate su *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html*. Notare che per molti di questi server sono disponibili sostituti migliori.

Compilare il pacchetto:

```
make
```

Questo pacchetto non è fornito di una suite di test.

Installarlo:

```
make install
```

Spostare il programma **ping** al suo posto conformemente a FHS:

```
mv -v /usr/bin/ping /bin
```

ù

6.40.2. Contenuti di Inetutils

Programmi installati: ftp, ping, rcp, rlogin, rsh, talk, telnet e tftp

Brevi descrizioni

ftp	Il programma di trasferimento file
ping	Invia pacchetti echo-request e riporta quanto tempo impiega la replica
rcp	Esegue una copia remota di file
rlogin	Esegue login remoto
rsh	Essegue una shell remota
talk	Usato per conversare con un altro utente
telnet	Un'interfaccia al protocollo TELNET
tftp	Un programma leggero di trasferimento file

6.41. IPRoute2-2.6.16-060323

Il pacchetto IPRoute2 contiene programmi per il networking di base e avanzato basato su IPV4.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 4.8 MB

6.41.1. Installazione di IPRoute2

Compilare il pacchetto:

```
make SBINDIR=/sbin
```

Significato dell'opzione di make:

SBINDIR=/sbin
Questo assicura che i binari di IPRoute2 si installino in `/sbin`. Questa secondo FHS è la locazione corretta, poiché alcuni dei binari di IPRoute2 sono usati dal pacchetto LFS-Bootscripts.

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make SBINDIR=/sbin install
```

Il binario **arpd** collega le librerie Berkeley DB che risiedono in `/usr` e utilizza un database in `/var/lib/arpd/arpd.db`. Così, in conformità al FHS, esso deve stare in `/usr/sbin`. Muoverlo qui:

```
mv -v /sbin/arpd /usr/sbin
```

6.41.2. Contenuti di IPRoute2

Programmi installati: arpd, ctstat (link a lnstat), ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link a lnstat), ss, e tc.

Brevi descrizioni

arpd Il demone ARP in userspace, utile nelle reti veramente grandi, dove l'implementazione in userspace di ARP è insufficente, o quando si imposta un honeypot (trappola)

ctstat Utilità per lo stato della connessione

ifcfg Uno script di shell wrapper per il comando **ip**

ifstat Mostra le statistiche delle interfacce, incluso l'ammontare dei pacchetti trasmessi e ricevuti dall'interfaccia

ip L'eseguibile principale. Ha molte differenti funzioni:

ip link *<device>* permette agli utenti di vedere lo stato del dispositivo e di cambiarlo

ip addr permette agli utenti di vedere gli indirizzi e le loro proprietà, aggiungere nuovi indirizzi e cancellare quelli vecchi

ip neighbor permette agli utenti di vedere i neighbour binding e le loro proprietà, aggiungere nuove registrazioni e cancellare quelle vecchie

ip rule permette agli utenti di vedere le politiche di routing e di cambiare le regole della tabelle di routiong

ip route permette di vedere la tabella di routing e cambiare le regole della tabela di routing

ip tunnel permette agli utenti di vedere i tunnel ip e le loro proprietà, e cambiarli

ip maddr permette agli utenti di vedere gli indirizzi multicast e le loro proprietà, e cambiarli

ip mroute permette agli utenti di impostare, cambiare o cancellare il routing mutlicast

ip monitor permette agli utenti di monitorare lo stato dei dispositivi, indirizzi e route in modo continuo

lnstat Fornisce statistiche di rete per Linux. È un sostituto generalizzato e più completo per il vecchio programma **rtstat**

nstat Mostra statistiche di rete

routef Un componente di **ip route**. Serve per svuotare le tabelle di routing

routel Un componente di **ip route**. Serve per elencare le tabelle di routing

rtacct Mostra il contenuto di `/proc/net/rt_acct`

rtmon Uutilità di monitoraggio delle route

rtpr Converte l'output di **ip -o** in una forma leggibile

rtstat Utilità di stato delle route

ss Simile al comando **netstat**; mostra le connessioni attive

tc Eseguibile di controllo del traffico; è un'implementazione per il Quality Of Service (QOS) e Class Of Service (COS)

tc qdisc permette agli utenti di impostare la disciplina di queueing

tc class permette agli utenti di impostare classi basate sulla schedulazione della disciplina delle code

tc estimator permette agli utenti di stimare il flusso di rete in una rete

tc filter permette agli utenti di impostare il filtraggio dei pacchetti QOS/COS

tc policy permette agli utenti di impostare le politiche QOS/COS

6.42. Kbd-1.12

Il pacchetto Kbd contiene file mappa e utilità per la tastiera.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 12.3 MB

6.42.1. Installazione di Kbd

Il comportamento dei tasti Backspace e Delete non è lo stesso tra le mappe di tastiera nel pacchetto Kbd. La seguente patch corregge questo problema per le mappe i386:

```
patch -Np1 -i ../kbd-1.12-backspace-1.patch
```

Dopo aver applicato la patch, il tasto Backspace genera il carattere con il codice 127, e il tasto Delete genera una ben nota sequenza escape.

Applicare una patch a Kbd per correggere un bug nel **setfont** che viene innescato quando si compila con GCC-4.0.3:

```
patch -Np1 -i ../kbd-1.12-gcc4_fixes-1.patch
```

Preparare Kbd per la compilazione:

```
./configure --datadir=/lib/kbd
```

Significato delle opzioni di configurazione:

`--datadir=/lib/kbd`
Questa opzione inserisce i dati del layout della tastiera in una directory che sarà sempre nella partizione di root invece del default `/usr/share/kbd`.

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

Nota

Per alcune lingue (es., Bielorusso) il pacchetto Kbd non fornisce una mappa di tastiera utile dove la keymap «by» adotta la codifica ISO-8859-5, e normalmente viene utilizzata la mappa tastiera CP1251. Gli utenti con tali linguaggi devono scaricare separatamente delle mappe tastiera funzionanti.

Alcuni script nel pacchetto LFS-Bootscripts dipendono dal **kbd_mode**, **openvt**, e **setfont**. Poiché `/usr` può non essere disponile durante la prima parte dell'avvio, questi binari devono trovarsi nella partizione di root:

```
mv -v /usr/bin/{kbd_mode,openvt,setfont} /bin
```

6.42.2. Contenuti di Kbd

Programmi installati: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link a psfxtable), psfgettable (link a psfxtable), psfstriptable (link a psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start, e unicode_stop

Brevi descrizioni

chvt	Cambia il terminale virtuale in primo piano
deallocvt	Dealloca i terminali virtuali non usati
dumpkeys	Fa il dump delle tabelle di conversione tastiera
fgconsole	Stampa il numero di terminali virtuali attivi
getkeycodes	Stampa la tabella di mappatura del kernel da scancode a keycode
kbd_mode	Riporta o definisce il modo tastiera
kbdrate	Imposta ripetizione e ritardo della tastiera
loadkeys	Carica le tabelle di conversione della tastiera
loadunimap	Carica la tabella di mappatura del kernel da unicode a font
mapscrn	Un programma obsoleto usato per caricare una tabella di mappatura caratteri definita dall'utente nel driver della console; questo ora è fatto da **setfont**
openvt	Avvia un programma su un nuovo terminale virtuale (VT)
psfaddtable	Un link a **psfxtable**
psfgettable	Un link a **psfxtable**
psfstriptable	Un link a **psfxtable**
psfxtable	Gestisce tabelle carattere Unicode per font console
resizecons	Cambia l'idea del kernel sulla dimensione della console
setfont	Cambia i font Enhanced Graphic Adapter (EGA) e Video Graphics Array (VGA) sulla console
setkeycodes	Carica le registrazioni sulla tabella di mappatura del kernel da scancode a keycode, utile nel caso si abbiano dei tasti insoliti sulla tastiera in uso
setleds	Definisce flag e Light Emitting Diodes (LED) della tastiera
setmetamode	Definisce la gestione dei meta-tasti della tastiera
showconsolefont	Mostra il font corrente dello schermo EGA/VGA della console
showkey	Riporta scancode e keycode e codici ASCII dei tasti premuti sulla tastiera
unicode_start	Pone tastiera e console in modalità UNICODE. Non usare questo programma a

	meno che il proprio file di mappatuta tastiera sia con codifica ISO-8859-1. Per altre codifiche, questa utility produce risultati non corretti
unicode_stop	Fa uscire tastiera e console dalla modalità UNICODE

6.43. Less-394

Il pacchetto Less contiene un visualizzatore di file di testo.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 2.6 MB

6.43.1. Installazione di Less

Preparare Less per la compilazione:

```
./configure --prefix=/usr --sysconfdir=/etc
```

Significato delle opzioni di configurazione:

`--sysconfdir=/etc`
Questa opzione dice ai programmi creati dal pacchetto di guardare in `/etc` per i loro file di configurazione.

Compilare il pacchetto:

```
make
```

Questo pacchetto non viene fornito di una suite di test.

Installare il pacchetto:

```
make install
```

6.43.2. Contenuti di Less

Programmi installati: less, lessecho e lesskey

Brevi descrizioni

less	Un visualizzatore di file o paginatore; visualizza i contenuti dei file forniti, permettendo di scrollare su e giù, trovare stringhe e saltare verso marcatori
lessecho	Necessario per espandere meta-caratteri, come * e *?*, nei nomi dei file su sistemi Unix
lesskey	Usato per specificare i vincoli dei caratteri per **less**

6.44. Make-3.80

Il pacchetto Make contiene un programma per compilare pacchetti.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 7.8 MB

6.44.1. Installazione di Make

Preparare Make per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.44.2. Contenuti di Make

Programma installato: make

Brevi descrizioni

make Determina automaticamente quali pezzi di un pacchetto devono essere (ri)compilati, e quindi inserisce i comandi appropriati

6.45. Man-DB-2.4.3

Il pacchetto Man-DB contiene programmi per la ricerca e la visualizzazione di pagine man.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 9 MB

6.45.1. Installazione di Man-DB

Devono essere apportati tre aggiustamenti ai sorgenti di Man-DB.

Il primo modifica la locazione delle pagine di manuale tradotte che vengono fornite con Man-DB, al fine di essere accessibili in entranbe le localizzazioni tradizionale e UTF-8:

```
mv man/de{_DE.88591,} &&
mv man/es{_ES.88591,} &&
mv man/it{_IT.88591,} &&
mv man/ja{_JP.eucJP,} &&
sed -i 's,\*_\*,??,' man/Makefile.in
```

La seconda modifica è una sostituzione **sed** per cancellare le righe «/usr/man» dentro il file `man_db.conf` per prevenire risultati ridondanti quando si usano programmi come **whatis**:

```
sed -i '/\t\/usr\/man/d' src/man_db.conf.in
```

La terza modifica per tener conto di programmi che Man-DB dovrebbe poter cercare al momento dell'esecuzione, ma che non sono stati ancora installati:

```
cat >>include/manconfig.h.in <<"EOF"
#define WEB_BROWSER "exec /usr/bin/lynx"
#define COL "/usr/bin/col"
#define VGRIND "/usr/bin/vgrind"
#define GRAP "/usr/bin/grap"
EOF
```

Il programma **col** fa parte del pacchetto Util-linux, **lynx** è un browser web testuale (vedere BLFS per le istruzioni d'installazione), **vgrind** converte i sorgenti di programma in input Groff, e **grap** è utile per la composizione grafica nei documenti in Groff. I programmi **vgrind** e **grap** non sono normalmente necessari per la visualizzazione della pagine manuale. Essi non fanno parte di LFS o BLFS, ma si dovrebbe poterli installare da se dopo aver terminato LFS, se lo si desidera.

Preparare Man-DB per la compilazione:

```
./configure --prefix=/usr --enable-mb-groff --disable-setuid
```

Significato delle opzioni di compilazione:

`--enable-mb-groff`
Questo dice al programma **man** di usare «ascii8» e «nippon» come dispositivi di Groff per la formattazione delle pagine di manuale non-ISO-8859-1.

`--disable-setuid`
Questo disabilita l'impostazione del programma **man** con setuid all'utente `man`.

Compilare il pacchetto:

```
make
```

Questo pacchetto non viene fornito di una suite di test.

Installare il pacchetto:

```
make install
```

Alcuni pacchetti forniscono pagine man UTF-8 che questa versione di **man** non può visualizzare. Il seguente script permetterà ad alcuni di questi di essere convertiti codifica che ci si aspetta, mostrata nella tabella di sotto. Man-DB si aspetta che le pagine di manuale siano nella codifica descritta nella tabella, e le convertirà ove necessario nella corrente codifica di localizzazione quando le visualizza, così che esse saranno visualizzate in entrambe le localizzazioni UTF-8 e tradizionale. Siccome questo script è inteso per un uso limitato durante la costruzione del sistema, per informazione, non si disturberà con il controllo degli errori, né si userà un nome file temporaneo non prevedibile.

```
cat >>convert-mans <<"EOF"
#!/bin/sh -e
FROM="$1"
TO="$2"
shift ; shift
while [ $# -gt 0 ]
do
        FILE="$1"
        shift
        iconv -f "$FROM" -t "$TO" "$FILE" >.tmp.iconv
        mv .tmp.iconv "$FILE"
done
EOF
install -m755 convert-mans  /usr/bin
```

Informazioni aggiuntive riguardanti la compressione delle pagine man e info può essere trovata nel libro BLFS presso *http://www.linuxfromscratch.org/blfs/view/cvs/postlfs/compressdoc.html*.

6.45.2. Pagine Manuale di LFS non in Inglese

Le distribuzioni Linux hanno differenti politiche riguardanti la codifica dei caratteri nelle quali sono memorizzate nel filesystem le pagine di manuale. Es., RedHat memorizza tutte le pagine manuale in UTF-8, mentre Debian usa codifiche specifiche della lingua (spesso a 8-bit). Questo porta all'incompatibilità dei pacchetti con pagine manuale progettate per le diverse distribuzioni.

LFS utilizza le stesse convenzioni di Debian. È stato scelto questo perché Man-DB non transcodifica le pagine man memorizzate in UTF-8. E, per i propri scopi, Man-DB è preferibile a Man per il suo funzionamento senza extra configurazioni in ogni localizzazione. Per ultimo, per ora, non ci sono implementazioni pienamente funzionanti con le convenzioni di RedHat. **Groff** di RedHat è conosciuto per mal-formattare i testi.

La relazione tra i codici di lingua e la codifica aspettata delle pagine di manuale è elencata qui sotto. Man-DB le converte automaticamente nella codifica locale durante la visualizzazione.

Tabella 6.1. Codifica prevista dei caratteri delle pagine manuale

Lingua (codice)	Codifica
Danese (da)	ISO-8859-1
Tedesco (de)	ISO-8859-1
Inglese (en)	ISO-8859-1
Spagnolo (es)	ISO-8859-1
Finlandese (fi)	ISO-8859-1
Francese (fr)	ISO-8859-1
Irlandese (ga)	ISO-8859-1
Galiziano (gl)	ISO-8859-1
Indonesiano (id)	ISO-8859-1
Islandese (is)	ISO-8859-1
Italiano (it)	ISO-8859-1
Olandese (nl)	ISO-8859-1
Norvegese (no)	ISO-8859-1
Portoghese (pt)	ISO-8859-1
Svedese (sv)	ISO-8859-1
Ceco (cs)	ISO-8859-2
Croato (hr)	ISO-8859-2
Ungherese (hu)	ISO-8859-2
Giapponese (ja)	EUC-JP
Coreano (ko)	EUC-KR
Polacco (pl)	ISO-8859-2
Russo (ru)	KOI8-R
Slovacco (sk)	ISO-8859-2
Turco (tr)	ISO-8859-9

Nota

Le pagine Manuale in linguaggi non presenti nella lista non sono supportati. Il Norvegese non funziona per adesso per la transizione dalla localizzazione no_NO alla nb_NO, il Coreano non funziona per l'incompletezza di una patch di Groff.

Se la fonte distribuisce le pagine manuale nella stessa codifica come si aspetta Man-DB, le pagine manuale possono essere copiate in `/usr/share/man/<language code>`. Es., le pagine manuale Francesi (*http://ccb.club.fr/man/man-fr-1.58.0.tar.bz2*) possono essere installate con il seguente comando:

```
mkdir -p /usr/share/man/fr &&
cp -rv man? /usr/share/man/fr
```

Se la fonte distribuisce le pagine manuale in UTF-8 (es., «per RedHat») invece della codifica elencata nella tabella sopra, queste devono essere convertite da UTF-8 alla codifica elencata nella suddetta tabella per la loro installazione. Questo puo; essere ottenuto con **convert-mans**, es., le pagine manuale in Spagnolo (*http://ditec.um.es/~piernas/manpages-es/man-pages-es-1.55.tar.bz2*) possono essere installate attraverso i seguenti comandi:

```
mv man7/iso_8859-7.7{,X}
convert-mans UTF-8 ISO-8859-1 man?/*.?
mv man7/iso_8859-7.7{X,}
make install
```

Nota

La necessità di escludere il file `man7/iso_8859-7.7` dal processo di conversione è dovuto al fatto che questo è già in ISO-8859-1 è un bug del pacchetto man-pages-es-1.55. Le versioni future non dovrebbero richiedere questo aggiustamento.

6.45.3. Contenuti di Man-DB

Programmi installati: accessdb, apropos, catman, convert-mans,lexgrog, man, mandb, manpath, whatis e zsoelim

Brevi Descrizioni

accessdb	Esegue un dump del contenuto del database di **whatis** in una forma leggibile
apropos	Ricerca nel database di **whatis** e visualizza le brevi descrizioni dei comandi di sistema che contengono la stringa fornita
catman	Crea o aggiorna le pagine di manuale pre-formattate
convert-mans	Riformatta le pagine di manuale così che Man-DB possa visualizzarle
lexgrog	Visualizza in una-riga informazioni sommarie circa una data pagina di manuale
man	Formatta e visualizza la richiesta pagina di manuale
mandb	Crea o aggiorna il database di **whatis**
manpath	Visualizza il contenuto di $MANPATH o (se $MANPATH non è impostata) esegue un appropriata ricerca basata sui path della configurazione in man.conf e sull'ambiente dell'utente
whatis	Ricerca nel database di **whatis** e visualizza le brevi descrizioni dei comandi di sistema che contengono la data parola chiave come una parola separata
zsoelim	Legge file e sostituisce righe di file dalla forma *file .so* con i contenuti del suddetto *file*

6.46. Mktemp-1.5

Il pacchetto Mktemp contiene programmi usati per creare file temporanei sicuri negli script di shell.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 0.4 MB

6.46.1. Installazione di Mktemp

Molti script usano ancora il programma deprecato **tempfile**, che ha funzionalità simili a **mktemp**. Applicare una patch a mktemp per includere un wrapper **tempfile**:

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-3.patch
```

Preparare Mktemp per la compilazione:

```
./configure --prefix=/usr --with-libc
```

Significato delle opzioni di configurazione:

`--with-libc`
Questo fa sì che il programma **mktemp** usi le funzioni *mkstemp* e *mkdtemp* dalla libreria C di sistema invece che le proprie implementazioni delle stesse.

Compilare il pacchetto:

```
make
```

Questo pacchetto non viene fornito di una suite di test.

Installare il pacchetto:

```
make install
make install-tempfile
```

6.46.2. Contenuti di Mktemp

Programmi installati: mktemp e tempfile

Brevi descrizioni

mktemp	Crea file temporanei in maniera sicura; è usato negli script
tempfile	Crea file temporanei in un maniera meno sicura di **mktemp**; è installato per retrocompatibilità

6.47. Module-Init-Tools-3.2.2

Il pacchetto Module-Init-Tools contiene programmi per gestire i moduli del kernel in kernel Linux maggiori o uguali alla versione 2.5.47.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 7 MB

6.47.1. Installazione di Module-Init-Tools

Per primo correggere un potenziale problema quando i moduli sono specificati utilizzando espressioni reglari:

```
patch -Np1 -i ../module-init-tools-3.2.2-modprobe-1.patch
```

Digitare i seguenti comandi per eseguire i test (notare che il comando **make distclean** è richiesto per pulire l'albero dei sorgenti, perché i sorgenti vengono ricompilati come parte del processo di test):

```
./configure &&
make check &&
make distclean
```

Preparare Module-Init-Tools per la compilazione:

```
./configure --prefix=/ --enable-zlib
```

Compilare il pacchetto:

```
make
```

Installare il pacchetto:

```
make INSTALL=install install
```

Significato dei parametri di compilazione:

`INSTALL=install`
Normalmante, **make install** non installerà i binari se esistono di già. Questa opzione esclude tutte le azioni chiamando **install** invece di usare il wrapper script predefinito.

6.47.2. Contenuti di Module-Init-Tools

Programmi installati: depmod, generate-modprobe.conf, insmod, insmod.static, lsmod, modinfo, modprobe, e rmmod

Brevi descrizioni

depmod	Crea un file di dipendenza basato sui simboli che trova nel set di moduli esistente; questo file di dipendenza è usato da **modprobe** per caricare automaticamente i moduli richiesti
generate-modprobe.conf	Crea un file modprobe.conf da un esistente modulo 2.2 o 2.4 di setup
insmod	Installa un modulo caricabile nel kernel in esecuzione

insmod.static	Una versione di **insmod** compilata staticamente
lsmod	Elenca i moduli correntemente caricati
modinfo	Esamina un file oggetto associato ad un modulo kernel e visualizza ogni informazione che può ricavare
modprobe	Usa un file di dipendenza, creato da **depmod**, per caricare automaticamente i moduli rilevanti
rmmod	Scarica i moduli dal kernel in esecuzione

6.48. Patch-2.5.4

Il pacchetto Patch contiene un programma per modificare o creare file applicando un file «patch» tipicamente creato dal programma **diff**.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 1.6 MB

6.48.1. Installazione di Patch

Preparare Patch per la compilazione.

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make install
```

6.48.2. Contenuti di Patch

Programma installato: patch

Brevi descrizioni

patch Modifica i file secondo un file patch. Un file patch normalmente è una lista di differenze creata con il programma **diff**. Applicando queste differenze ai file originali, **patch** crea la versione "riparata".

6.49. Psmisc-22.2

Il pacchetto Psmisc contiene programmi per visualizzare informazioni sui processi in esecuzione.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 2.2 MB

6.49.1. Installazione di Psmisc

Preparare Psmisc per la compilazione:

```
./configure --prefix=/usr --exec-prefix=""
```

Significato delle opzioni di configurazione:

`--exec-prefix=""`
Fa sì che i binari di Psmisc si installino in `/bin` invece che in `/usr/bin`. Questa è la corretta collocazione conforme a FHS, poiché alcuni dei binari di Psmisc sono usati dal pacchetto LFS-Bootscripts.

Compilare il pacchetto:

```
make
```

Questo pacchetto non è dotato di una suite di test.

Installare il pacchetto:

```
make install
```

Non c'è ragione perché i programmi **pstree** e **pstree.x11** risiedano in `/bin`. Pertanto spostarli in `/usr/bin`:

```
mv -v /bin/pstree* /usr/bin
```

Per default il programma di Psmisc **pidof** non è installato. Generalmente questo non è un problema, poiché viene installato più tardi nel pacchetto Sysvinit, che fornisce un programma **pidof** migliore. Se, per un particolare sistema, Sysvinit non sarà usato, completare l'installazione di Psmisc creando il seguente link simbolico:

```
ln -sv killall /bin/pidof
```

6.49.2. Contenuti di Psmisc

Programmi installati: fuser, killall, pstree e pstree.x11 (link a pstree)

Brevi descrizioni

fuser	Riporta gli ID di Processo (PID) di processi che usano i file o file system dati
killall	Uccide i processi per nome; manda un segnale a tutti i processi eseguendo uno dei comandi specificati
oldfuser	Riporta gli ID di Processo (PID) di processi che usano i file o file system indicati

pstree	Visualizza i processi in esecuzione con un albero
pstree.x11	Uguale a **pstree**, con la differenza che aspetta conferma prima di uscire

6.50. Shadow-4.0.15

Il pacchetto Shadow contiene programmi per gestire le password in modo sicuro.

Tempo di costruzione approssimativo: 0.3 SBU
Spazio necessario su disco: 18.6 MB

6.50.1. Installazione di Shadow

Nota

Se si vuole obbligare l'uso di password forti fare riferimento a *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* per installare Cracklib prima di costruire Shadow. Quindi aggiungere *`--with-libcrack`* al comando **configure** seguente.

Preparare Shadow per la compilazione:

```
./configure --libdir=/lib --enable-shared --without-selinux
```

Significato delle opzioni di configurazione:

`--without-selinux`
Il supporto per selinux è abilitato di default, ma selinux non è costruito nel sistema base di LFS. Se questa opzione non è usata lo script **configure** fallirà.

Disabilitare l'installazione del programma **groups**, e le sue pagine man, poiché Coreutils ne fornisce una versione migliore:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile
find man -name Makefile -exec sed -i '/groups/d' {} \;
```

Disabilitare l'installazione delle pagine man in Cinese e Coreano, dal momento che Man-DB non può formattarle in modo corretto:

```
sed -i -e 's/ ko//' -e 's/ zh_CN zh_TW//' man/Makefile
```

Shadow fornisce altre pagine man nella codifica UTF-8. Man-DB può visualizzarle nelle codifiche raccomandate utilizzando lo script **convert-mans** che viene installato.

```
 for i in de es fi fr id it pt_BR; do
    convert-mans UTF-8 ISO-8859-1 man/${i}/*.?
done

for i in cs hu pl; do
    convert-mans UTF-8 ISO-8859-2 man/${i}/*.?
done

convert-mans UTF-8 EUC-JP man/ja/*.?
convert-mans UTF-8 KOI8-R man/ru/*.?
convert-mans UTF-8 ISO-8859-9 man/tr/*.?
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non è fornito di una suite di test.

Installare il pacchetto:

```
make install
```

Shadow usa due file per configurare le impostazioni di autenticazione per il sistema. Installare questi due file di configurazione:

```
cp -v etc/{limits,login.access} /etc
```

Invece di utilizzare il metodo di default *crypt*, sarebbe meglio utilizzare il più sicuro *MD5* per la cifratura delle password, che permette anche l'uso di password più lunghe di 8 caratteri. È necessario anche cambiare la vecchia locazione `/var/spool/mail` per le caselle di posta degli utenti che Shadow invece tiene di default su `/var/mail` usata correntemente. Per risolvere entrambi i problemi bisogna modificare il file di configurazione mentre lo si copia nella sua destinazione:

```
sed -e's@#MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \
          -e 's@/var/spool/mail@/var/mail@' \
          etc/login.defs > /etc/login.defs
```

Nota

Se si è costruito Shadow con il supporto Cracklib, eseguire il seguente:

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
    /etc/login.defs
```

Spostare un programma nella cartella giusta:

```
mv -v /usr/bin/passwd /bin
```

Spostare le librerie di Shadow in locazioni più appropriate:

```
mv -v /lib/libshadow.*a /usr/lib
rm -v /lib/libshadow.so
ln -sfv ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

L'opzione *-D* del programma **useradd** richiede la directory `/etc/default` per funzionare correttamente:

```
mkdir -v /etc/default
```

6.50.2. Configurazione di Shadow

Questo pacchetto contiene strumenti per aggiungere, modificare e cancellare utenti e gruppi, impostare e cambiare le loro password, e altri strumenti amministrativi simili. Per una spiegazione completa su cosa significa *password shadowing* si veda il file `doc/HOWTO` all'interno dell'albero dei sorgenti scompattato. Bisogna tenere una cosa a mente se si decide di usare il supporto di Shadow: che i programmi che servono per verificare le password (gestori di finestre, programmi per ftp, demoni pop3, e simili) devono essere conformi a Shadow. Cioé occorre che siano in grado di funzionare con le password shadowed.

Per abilitare le password shadowed eseguire il seguente comando:

```
pwconv
```

Per abilitare le password shadowed di gruppo eseguire:

```
grpconv
```

6.50.3. Definizione della password root

Scegliere una password per l'utente *root* e definirla usando:

```
passwd root
```

6.50.4. Contenuti di Shadow

Programmi installati: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link a newgrp), su, useradd, userdel, usermod, vigr (link a vipw), e vipw
Librerie installate: libshadow.{a,so}

Brevi descrizioni

chage	Viene usato per modificare il numero massimo di giorni entro i quali bisogna cambiare la propria password
chfn	Viene usato per modificare il nome completo di un utente ed altre informazioni
chgpasswd	Viene usato per aggiornare le password di un gruppo in modo batch
chpasswd	Viene usato per aggiornare le password di utente in modo batch
chsh	Viene usato per modificare la shell di login predefinita per un utente
expiry	Verifica e fa rispettare i termini di scadenza della password corrente
faillog	Viene usato per esaminare il log dei login falliti, per impostare il numero massimo di tentativi prima che l'account venga bloccato e per azzerare il contatore dei tentativi
gpasswd	Viene usato per aggiungere e cancellare membri ed amministratori dai gruppi
groupadd	Crea un gruppo con il nome indicato
groupdel	Cancella il gruppo col nome indicato
groupmod	Viene usato per modificare il nome o il GID del gruppo indicato
grpck	Verifica l'integrità dei file dei gruppi `/etc/group` e `/etc/gshadow`
grpconv	Crea o aggiorna il file dei gruppi shadow dal normale file dei gruppi
grpunconv	Aggiorna `/etc/group` da `/etc/gshadow` e infine cancella il secondo
lastlog	Visualizza i login più recenti per ogni utente, o per un dato utente
login	Viene usato dal sistema per permettere agli utenti di autenticarsi
logoutd	È un demone usato per far rispettare le restrizioni sul tempo e sulle porte per il log-in
newgrp	Viene usato per modificare il GID corrente durante una sessione di login
newusers	Viene usato per creare o aggiornare una serie di account utenti
nologin	Visualizza un messaggio che un account non è disponibile. Progettato per essere usato come

	shell di default per account che sono stati disabilitati
passwd	Viene usato per modificare la password di un utente o di un gruppo
pwck	Verifica l'integrità dei file delle password `/etc/passwd` e `/etc/shadow`
pwconv	Crea o aggiorna il file delle password shadow dal normale file delle password
pwunconv	Aggiorna `/etc/passwd` da `/etc/shadow` e infine cancella il secondo
sg	Esegue un dato comando mentre il GID dell'utente è impostato su quello del gruppo indicato
su	Esegue una shell sostituendo ID utente e gruppo
useradd	Crea un nuovo utente con il nome indicato, o aggiorna le informazioni di default del nuovo utente
userdel	Cancella l'account dell'utente indicato
usermod	Viene usato per modificare il nome di login, l'UID (User Identification), la shell, il gruppo iniziale, la home directory, etc.
vigr	Edita i file `/etc/group` o `/etc/gshadow`
vipw	Edita i file `/etc/passwd` o `/etc/shadow`
`libshadow`	Contiene funzioni usate dalla maggior parte dei programmi di questo pacchetto

6.51. Sysklogd-1.4.1

Il pacchetto Sysklogd contiene programmi per fare il log dei messaggi di sistema, come quelli dati dal kernel quando accadono cose insolite.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 0.6 MB

6.51.1. Installazione di Sysklogd

La seguente patch corregge diversi problemi, incluso un problema di costruzione di Sysklogd con i kernel della serie Linux 2.6

```
patch -Np1 -i ../sysklogd-1.4.1-fixes-1.patch
```

La patch seguente permette a sysklogd di trattare i byte nel range 0x80--0x9f letteralmente quando vengono loggati, invece di sostituirli con il relativi codici in ottale. Non applicare la patch a sysklogd nella codifica UTF-8 potrebbe danneggiare i messaggi:

```
patch -Np1 -i ../sysklogd-1.4.1-8bit-1.patch
```

Compilare il pacchetto:

```
make
```

Questo pacchetto non ha una suite di test.

Installare il pacchetto:

```
make install
```

6.51.2. Configurazione di Sysklogd

Creare un nuovo file `/etc/syslog.conf` eseguendo il seguente:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# End /etc/syslog.conf
EOF
```

6.51.3. Contenuti di Sysklogd

Programmi installati: klogd e syslogd

Brevi descrizioni

klogd Un demone di sistema per l'intercettazione e il log dei messaggi del kernel

syslogd Fa il log dei messaggi che i programmi di sistema offrono per il log. Ogni messaggio loggato contiene almeno una marcatura di data e un nome host, e normalmente anche il nome del programma, ma questo dipende dalla politica di trust applicata al demone di log

6.52. Sysvinit-2.86

Il pacchetto Sysvinit contiene programmi per controllare l'avvio, il funzionamento e l'arresto del sistema.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 1 MB

6.52.1. Installazione di Sysvinit

Quando i run-level vengono modificati (per esempio quando viene arrestato il sistema), **init** invia segnali di terminazione a quei processi che sono stati lanciati da **init** medesimo e che non devono rimanere in esecuzione nel nuovo run-level. Nel fare questo, **init** emette messaggi quali «Sending processes the TERM signal» che sembra implicare che vengano inviati questi segnali a tutti i processi che sono in esecuzione al momento. Per evitare questo errore di interpretazione, è possibile modificare i sorgenti in modo che questo messaggio invece diventi «Sending processes started by init the TERM signal»

```
sed -i 's@Sending processes@& started by init@g' \
    src/init.c
```

Compilare Sysvinit:

```
make -C src
```

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make -C src install
```

6.52.2. Configurazione di Sysvinit

Creare un nuovo file `/etc/inittab` eseguendo il seguente:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
```

```
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

6.52.3. Contenuti di Sysvinit

Programmi installati: bootlogd, halt, init, killall5, last, lastb (link a last), mesg, mountpoint, pidof (link a killall5), poweroff (link a halt), reboot (link a halt), runlevel, shutdown, sulogin, telinit (link a init), utmpdump e wall

Brevi descrizioni

bootlogd	Registra i messaggi di avvio in un file di log
halt	Normalmente invoca **shutdown** con il flag *-h*, tranne quando si trova già nel run-level 0, quindi dice al kernel di arrestare il sistema; annota nel file `/var/log/wtmp` che il sistema si sta per arrestare
init	È il primo processo ad essere avviato quando il kernel ha inizializzato l'hardware dal quale rileva il processo di boot e avvia tutti i processi ai quali è istruito
killall5	Invia un segnale a tutti i processi, esclusi quei processi presenti nella sua stessa sessione, evitando in tal modo di uccidere la shell che sta eseguendo lo script che lo ha chiamato
last	Mostra quali utenti hanno fatto gli ultimi login (e logout), ricercando all'indietro nel file `/var/log/wtmp`; mostra anche gli avvi e gli arresti del sistema, e i cambi di run-level
lastb	Mostra i tentativi di login falliti, come registrati in `/var/log/btmp`
mesg	Controlla quali altri utenti possono inviare messaggi al terminale dell'utente corrente
mountpoint	Verifica se la directory è un punto di mount
pidof	Riporta i PID dei programmi specificati
poweroff	Dice al kernel di arrestare il sistema e spegnere il computer (si veda anche **halt**)
reboot	Dice al kernel di riavviare il sistema (si veda anche **halt**)
runlevel	Riporta i precedenti e l'attuale run-level, come registrato nell'ultimo record di run-level in `/var/run/utmp`
shutdown	Spegne il sistema in modo sicuro, inviando segnali a tutti i processi e notificando lo spegnimento a tutti gli utenti connessi
sulogin	Consente a `root` di effettuare il login; normalmente è invocato da **init** quando il sistema va in modalità singolo utente
telinit	Dice a **init** in quale run-level entrare
utmpdump	Mostra il contenuto del file di login specificato in un formato più leggibile
wall	Invia messaggi a tutti gli utenti connessi

6.53. Tar-1.15.1

Il pacchetto Tar contiene un programma per l'archiviazione.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 13.7 MB

6.53.1. Installazione di Tar

Applicare una patch per correggere alcuni problemi con la suite di test quando si usa GCC-4.0.3:

```
patch -Np1 -i ../tar-1.15.1-gcc4_fix_tests-1.patch
```

Tar ha un bug quando l'opzione *-S* è usata con file di oltre 4 GB. La seguente patch corregge appropriatamente questo problema

```
patch -Np1 -i ../tar-1.15.1-sparse_fix-1.patch
```

Le recenti versioni di Tar sono vulnerabili ad un buffer overflow da archivi costruiti in modo particolare. La seguente patch si riferisce a questo:

```
patch -Np1 -i ../tar-1.15.1-security_fixes-1.patch
```

Preparare Tar per la compilazione:

```
./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/sbin
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

6.53.2. Contenuti di Tar

Programmi installati: rmt e tar

Brevi descrizioni

rmt è usato per gestire in remoto un drive di nastri magnetici, per mezzo di una connessione di comunicazione interprocesso.

tar Crea, estrae file da, ed elenca il contenuto degli archivi, noti anche come tarball.

6.54. Texinfo-4.8

Il pacchetto Texinfo contiene programmi per leggere, scrivere e convertire pagine info.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 16.6 MB

6.54.1. Installazione di Texinfo

Il programma **info** fa dei presupposti, come quello che una stringa occupi lo stesso numero di celle di caratteri sullo schermo e byte in memoria e che si possa interrompere una stringa in qualiasi punto, ciò fallisce nelle localizzazioni basate su UTF-8. La patch di sotto li rende validi tornando ai messaggi Inglesi quando ` in uso una localizzazione multibyte:

```
patch -Np1 -i ../texinfo-4.8-multibyte-1.patch
```

Texinfo permette agli utenti locali di sovrascrivere file arbitrari attraverso un attacco symlink su file temporanei. Applicare la seguente patch per correggere il problema:

```
patch -Np1 -i ../texinfo-4.8-tempfile_fix-2.patch
```

Preparare Texinfo per la compilazione:

```
./configure --prefix=/usr
```

Compilare il pacchetto:

```
make
```

Per testare i risultati, digitare: **`make check`**.

Installare il pacchetto:

```
make install
```

Opzionalmente, installare i componenti appartenenti ad una installazione TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

Significato dei parametri di make:

`TEXMF=/usr/share/texmf`
La variabile `TEXMF` del makefile contiene la locazione della radice dell'albero delle directory di TeX se, per esempio, si avesse intenzione di installare successivamente un pacchetto TeX.

Il sistema di documentazione Info utilizza semplici file di testo per conservare la lista delle voci di menu. Il file è posto nella directory `/usr/share/info/dir`. Sfortunatamente, a causa di problemi occasionali presenti nei Makefile di alcuni pacchetti, questo file può non risultare più allineato con le pagine info installate sul sistema. Se dovesse essere necessario ricreare il file `/usr/share/info/dir`, i seguenti comandi opzionali risolveranno il problema:

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.54.2. Contenuti di Texinfo

Programmi installati: info, infokey, install-info, makeinfo, texi2dvi, texi2pdf e texindex

Brevi descrizioni

info	È utilizzato per leggere pagine info, che sono molto simili alle pagine man, ma spesso approfondiscono molto, non limitandosi alla sola spiegazione delle opzioni della linea di comando. Si paragonino ad esempio **man bison** e **info bison**.
infokey	Compila un file sorgente contenente personalizzazioni info in un formato binario
install-info	Utilizzato per installare pagine info; aggiorna le voci presenti nel file indice di **info**
makeinfo	Traduce i documenti sorgenti Texinfo prescelti in file info, semplice testo o HTML
texi2dvi	Impiegato per formattare il documento Texinfo prescelto in un file, indipendente dalla periferica, che possa poi essere stampato
texi2pdf	Usato per formattare il documento Texinfo dato in un file Portable Document Format (PDF).
texindex	È utilizzato per ordinare i file indice di Texinfo

6.55. Udev-096

Il pacchetto Udev contiene programmi per la creazione dinamica di nodi di dispositivo.

Tempo di costruzione approssimativo: 0.1 SBU
Spazio necessario su disco: 6.8 MB

6.55.1. Installazione di Udev

Il tarball udev-config contiene file specifici di LFS usati per configurare Udev. Decomprimerlo dentro la directory dei sorgenti di Udev:

```
tar xf ../udev-config-6.2.tar.bz2
```

Creare alcuni dispositivi e directory che Udev non potrebbe gestire dovuto al fatto che sono richiesti molto presto nel processo di avvio:

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}
mknod -m0666 /lib/udev/devices/null c 1 3
ln -sv /proc/self/fd /lib/udev/devices/fd
ln -sv /proc/self/fd/0 /lib/udev/devices/stdin
ln -sv /proc/self/fd/1 /lib/udev/devices/stdout
ln -sv /proc/self/fd/2 /lib/udev/devices/stderr
ln -sv /proc/kcore /lib/udev/devices/core
```

Compilare il pacchetto:

```
make EXTRAS="extras/ata_id extras/cdrom_id extras/edd_id \
            extras/firmware extras/floppy extras/path_id \
            extras/scsi_id extras/usb_id extras/volume_id"
```

The meaning of the make option:

EXTRAS=...
Questo compila parecchi helper binari che possono aiutare nella personalizzazione delle regole di Udev.

Per testare i risultati digitare: **`make test`**.

Notare che la suite di test di Udev produrrà numerosi messaggi nei log di sistema dell'host. Questi sono innoqui e possono essere ignorati.

Installare il pacchetto:

```
make DESTDIR=/ \
    EXTRAS="extras/ata_id extras/cdrom_id extras/edd_id \
            extras/firmware extras/floppy extras/path_id \
            extras/scsi_id extras/usb_id extras/volume_id" install
```

Significato dell'opzione di make:

DESTDIR=/
Questo impedisce al processo di costruzione di Udev di uccidere qualunque processo **udevd** che dovesse essere in esecuzione sul sistema host.

Udev deve ssere configurato al fine di funzionare nella maniera giusta, per questo non installare alcun file di configurazione di default. Installare i file di configurazione specifici di LFS:

```
cp -v udev-config-6.2/[0-9]* /etc/udev/rules.d/
```

Installare la documentazione che spiega come greare le regole di Udev:

```
install -m644 -D -v docs/writing_udev_rules/index.html \
    /usr/share/doc/udev-096/index.html
```

6.55.2. Contenuti di Udev

Programmi installati: ata_id, cdrom_id, create_floppy_devices, edd_id, firmware_helper, path_id, scsi_id, udevcontrol, udevd, udevinfo, udevmonitor, udevsettle, udevtest, udevtrigger, usb_id, vol_id, and write_cd_aliases
Directory installata: /etc/udev

Brevi descrizioni

ata_id	Fornisce a Udev una stringa unica e informazioni addizionali (uuid, label) per un driver ATA
cdrom_id	Fornisce a Udev le capacità di un drive CD-ROM o DVD-ROM
create_floppy_devices	Crea tuttu i possibili dispositivi floppy basati sul tipo CMOS
edd_id	Fornisce a Udev l' EDD ID per un BIOS disk drive
firmware_helper	Upload il firmware ai dispositivi
path_id	Fornisce il path unico più corto possibile a un dispositivo
scsi_id	Fornisce a Udev un indentificativo SCSI unico basato sui dati ritornati da l'invio di un comando SCSI INQUIRY al dispositivo specificato
udevcontrol	Configura un numero di opzioni per il demone **udevd** in esecuzione, come il livello di log.
udevd	Un demone in ascolto di uevent sul socket netlink, crea dispositivi e esegue i programmi esterni configurati in resposta a questi uevent
udevinfo	Permette agli utenti di interrogare il database di Udev per informazioni su qualsiasi dispositivo correntemente presente sul sistema; fornisce anche un modo di interrogare qualsiasi dispositivo nell'albero `sysfs` per aiutare nella creazione delle regole di udev
udevmonitor	Stampa gli eventi ricevuti dal kernel e l'ambiente che Udev spedisce dopo il processo delle regole
udevsettle	Osserva la coda degli eventi Udev e esce se tutti gli uevent correnti sono stati gestiti

udevtest	Simula un uevent per un dato dispositivo, e stampa il nome del nodo che il reale **udevd** avrebbe creato, o il nome dell' interfaccia di rete rinominata
udevtrigger	Attiva nel kernel la ripetizione degli uevents dei dispositivi
usb_id	Fornisce a Udev le informazioni sui dispositivi USB
vol_id	Fornisce a Udev la label e l'uuid di un filesystem
`/etc/udev`	Contiene i file di configurazione di Udev, i permessi dei dispositivi, e le regole per la nomenclatura dei dispositivi

6.56. Util-linux-2.12r

Il pacchetto Util-linux contiene una serie di programmi di utilità. Fra di loro ci sono utilità per gestire i file system, le console, le partizioni e i messaggi.

Tempo di costruzione approssimativo: 0.2 SBU
Spazio necessario su disco: 17.2 MB

6.56.1. Note sulla conformità con FHS

Il FHS raccomanda di utilizzare la directory `/var/lib/hwclock`, invece dell'usuale `/etc`, come ubicazione per il file `adjtime`. Per rendere il programma **hwclock** conforme a FHS, eseguire:

```
sed -i 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    hwclock/hwclock.c
mkdir -p /var/lib/hwclock
```

6.56.2. Installazione di Util-linux

Util-linux fallisce la compilazione con le versioni più recenti di Linux-libc-headers. La seguente patch corregge il problema:

```
patch -Np1 -i ../util-linux-2.12r-cramfs-1.patch
```

Preparare Util-linux per la compilazione:

```
./configure
```

Compilare il pacchetto:

```
make HAVE_KILL=yes HAVE_SLN=yes
```

Significato dei parametri di make:

HAVE_KILL=yes
Questo previene la compilazione e l'installazione del programma **kill** (già installato da Procps).

HAVE_SLN=yes
Questo previene la compilazione e l'installazione del programma **sln** (un **ln** linkato staticamente già installato da Glibc).

Questo pacchetto non è provvisto di una suite di test.

Installare il pacchetto:

```
make HAVE_KILL=yes HAVE_SLN=yes install
```

6.56.3. Contenuti di Util-linux

Programmi installati: agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, flock, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, pg, pivot_root, ramsize (link to rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (link to rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (link to swapon), swapon, tailf, tunelp, ul, umount, vidmode (link to rdev), whereis, and write

Brevi descrizioni

agetty	Apre una porta tty, richiede un nome utente, e invoca il programma **login**
arch	Riporta l'architettura della macchina
blockdev	Permette di chiamare dispositivi a blocchi ioctls dalla riga di comando
cal	Visualizza un semplice calendario
cfdisk	Manipola la tabella delle partizioni del dispositivo indicato
chkdupexe	Trova eseguibili doppi
col	Filtra i line feed inversi
colcrt	Filtra l'output di **nroff** per i terminali che mancano di alcune caratteristiche come l'overstriking e le mezze righe
colrm	Filtra le colonne specificate
column	Formatta un file in colonne multiple
ctrlaltdel	Imposta la funzione della combinazione di tasti Ctrl+Alt+Del in un riavvio a caldo o a freddo
cytune	Affina i parametri dei driver della linea seriale per schede Cyclades
ddate	Restituisce la data Discordiana, o converte una data Gregoriana in una Discordiana
dmesg	Scarica i messaggi di avvio del kernel
elvtune	Regola performance e interattività di un dispositivo a blocchi
fdformat	Formatta a basso livello un floppy
flock	Acquisisce un lock di file e poi esegue un comando mantenendo il lock
fdisk	Manipola la tabella delle partizioni del dispositivo specificato
fsck.cramfs	Esegue un controllo di consistenza in un file system Cramfs nel dispositivo selezionato
fsck.minix	Esegue un controllo di consistenza in un file system Minix nel dispositivo selezionato
getopt	Esegue l'analisi delle opzioni fornite nella riga di comando
hexdump	Scarica il file specificato in esadecimale o in un altro formato
hwclock	Legge o imposta l'orologio hardware di sistema, anche chiamato RTC (Real-Time Clock) o orologio BIOS (Basic Input-Output System)
ipcrm	Rimuove la risorsa Inter-Process Communication (IPC) specificata

ipcs	Fornisce informazioni di stato sull'IPC
isosize	Fornisce la dimensione di un file system iso9660
line	Copia una singola linea
logger	Inserisce il messaggio specificato nel log del sistema
look	Mostra le linee che iniziano con la stringa specificata
losetup	Imposta e controlla i dispositivi loop
mcookie	Genera dei magic cookie (numeri casuali esadecimali a 128 bit) per **xauth**
mkfs	Crea un file system in un dispositivo (di solito una partizione di un disco rigido)
mkfs.bfs	Crea un file system bfs di SCO (Santa Cruz Operations)
mkfs.cramfs	Crea un file system cramfs
mkfs.minix	Crea un file system Minix
mkswap	Inizializza il dispositivo o file specificato per essere utilizzato come area di swap
more	Un filtro per impaginare il testo una schermata per volta
mount	Collega il file system del dispositivo specificato ad una directory nell'albero del file system
namei	Mostra i collegamenti simbolici nei percosi specificati
pg	Mostra un file di testo una schermata alla volta
pivot_root	Rende il file system specificato il nuovo file system radice per il processo corrente
ramsize	Imposta la dimensione del disco RAM in un'immagine avviabile
raw	Usato per legare un dispositivo a caratteri di Linux ad un dispositivo a blocchi
rdev	Consulta e definisce il dispositivo root, tra le altre cose, in una immagine avviabile
readprofile	Legge le informazioni di profilo del kernel
rename	Rinomina i file specificati, rimpiazzando la stringa fornita con un'altra
renice	Altera la priorità dei processi in esecuzione
rev	Inverte le linee del file specificato
rootflags	Imposta il flag di root in un'immagine avviabile
script	Fa un typescript di una sessione a terminale
setfdprm	Imposta i parametri forniti dall'utente per un disco floppy
setsid	Avvia il programma specificato in una nuova sessione
setterm	Definisce gli attributi del terminale
sfdisk	Un manipolatore di tabelle delle partizioni
swapoff	Disattiva i dispositivi e i file per la paginazione e lo swapping
swapon	Abilita i dispositivi e i file per la paginazione e lo swapping ed elenca dispositivi e file correntemente in uso

tailf	Traccia l'incremento di un file di log. Visualizza le ultime 10 righe di un file di log, poi continua a visualizzare ogni nuova entrata nel file di log come vengono create
tunelp	Regola i parametri della stampante
ul	Un filtro per la traduzione di underscore in sequenze di escape che indicano la sottolineatura per il terminale in uso
umount	Disconnette un file system dall'albero dei file system
vidmode	Imposta la modalità video in una immagine avviabile
whereis	Riporta la locazione di binario, sorgente e pagina man del comando specificato
write	Invia un messaggio all'utente specificato *se* questo utente non ha disabilitato la ricevuta di tali messaggi

6.57. Vim-7.0

Il pacchetto Vim contiene un potente editor di testi.

Tempo di costruzione approssimativo: 0.4 SBU
Spazio necessario su disco: 47.4 MB

Alternative a Vim

Se si preferisce un altro editor (come Emacs, Joe, o Nano) a Vim, fare riferimento a *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* per le istruzioni di installazione consigliate.

6.57.1. Installazione di Vim

Prima scompattare entrambi gli archivi `vim-7.0.tar.bz2` e (opzionalmente) `vim-7.0-lang.tar.gz` nella stessa directory. Poi, applicare la patch a Vim con le diverse correzioni da upstream di sviluppatori a partire dal rilascio iniziale di Vim-7.0:

```
patch -Np1 -i ../vim-7.0-fixes-7.patch
```

Questa versione di Vim installa le man page tradotte e le pone dentro le directory dove non saranno trovate dal Man-DB. Applicare la patch a Vim così che installi le proprie man page dentro una directory trovabile e per ultimo permetta a Man-DB di trascodificare la pagina nel formato desiderato al momento dell'esecuzione:

```
patch -Np1 -i ../vim-7.0-mandir-1.patch
```

C'è un problema introdotto da una delle patch upstream che crea un problema al download degli spellfile via HTTP. Fintanto che gli sviluppatori non lo hanno aggiornato, la patch seguente corregge il problema:

```
patch -Np1 -i ../vim-7.0-spellfile-1.patch
```

In fine, cambiare le locazioni di default dei file di configurazione `vimrc` in `/etc`.

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Preparare Vim per la compilazione:

```
./configure --prefix=/usr --enable-multibyte
```

Significato delle opzioni di configurazione:

`--enable-multibyte`
Questo switch abilita il suporto per editare file con la codifica dei caratteri multibyte. Ciò è necessario se si usa una localizzazione con un insieme di caratteri multibyte. Questo switch è anche d'aiuto per poter editare file di testo inizialmente creati in distribuzioni Linux come Fedora Core che usa UTF-8 come set caratteri di default.

Compilare il pacchetto:

```
make
```

Per verificare il risultato digitare: **make test**. Tuttavia questa suite di test invia molti dati binari sullo schermo, il che può provocare problemi alle impostazioni del terminale attivo. Questo può essere risolto redirigendo l'output verso un file log.

Installare il pacchetto:

```
make install
```

Con la localizzazione UTF-8,il programma **vimtutor** prova a convertire i tutorial da ISO-8859-1 a UTF-8. Da quando alcuni tutorial non sono in ISO-8859-1, il loro testo è perciò reso illeggibile. Se si decomprime l'archivio `vim-7.0-lang.tar.gz` e si ha intenzione di usare una localizzazione basata su UTF-8, rimuovere i tutorial non-ISO-8859-1. Al suo posto sarà usato un tutorial Inglese.

```
rm -f /usr/share/vim/vim70/tutor/tutor.{gr,pl,ru,sk}
rm -f /usr/share/vim/vim70/tutor/tutor.??.*
```

Molti utenti sono abituati ad utilizzare **vi**, invece di **vim**. Per consentire loro di eseguire **vim** quando abitualmente si digita **vi**, creare un collegamento simbolico per entrambi i binari e la pagina man nei linguaggi forniti:

```
ln -sv vim /usr/bin/vi
for L in "" fr it pl ru; do
    ln -sv vim.1 /usr/share/man/$L/man1/vi.1
done
```

Di default, la documentazione di Vim è installata in `/usr/share/vim`. Il seguente symlink permette alla documentazione di essere acceduta via `/usr/share/doc/vim-7.0`, rendendola compatibile con la collocazione della documentazione per gli altri pacchetti:

```
ln -sv ../vim/vim70/doc /usr/share/doc/vim-7.0
```

Se si ha intenzione di installare un sistema X Window sul proprio sistema LFS, si potrebbe voler ricompilare Vim dopo aver installato X. Vim è fornito di una piacevole versione GUI dell'editor che richiede X ed una manciata di altre librerie per essere installata. Per maggiori informazioni su questo processo, fare riferimento alla documentazione di Vim e alla pagina di installazione di Vim nel libro BLFS su *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.57.2. Configurazione di Vim

Per default **vim** viene eseguito in modalità incompatibile con vi. Questo può risultare nuovo ad utenti che in passato hanno usato altri editor. La definizione «nocompatible» è inclusa in seguito per evidenziare il fatto che viene utilizzato un nuovo ambiente. Ricorda anche a coloro che vogliono cambiare in modalità «compatible» che essa deve essere la prima impostazione nel file di configurazione. Questo è necessario, poiché cambia le altre configurazioni, e le sovrapposizioni devono venire dopo questa impostazione. Si crei un file di configurazione di default di **vim** eseguendo quanto segue:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif
```

```
" End /etc/vimrc
EOF
```

Il parametro *`set nocompatible`* fa in modo che **vim** si comporti in un modo molto più utile (quello predefinito) della modalità compatibile con vi. Eliminare il «no» se si vuole il vecchio ambiente **vi**. Il parametro *`set backspace=2`* consente di effettuare il backspace in presenza di interruzioni di riga, indentazioni automatiche e all'inizio dell'inserimento. L'istruzione *`syntax on`* abilita la colorazione su base semantica di **vim**. Infine, lo statement *if* con *`set background=dark`* corregge le assunzioni di **vim** sul colore di sfondo di certi emulatori terminale. Questo da alla evidenziazione semantica un miglior schema colore da usare sugli sfondi neri di questi programmi.

Documentazione per altre opzioni disponibili può essere ottenuta eseguendo il seguente comando:

```
vim -c ':options'
```

Nota

Di default, Vim installa solo file spell per la lingua Inglese. Per installare file spell per la propria lingua preferita, scaricare il file `*.spl` e opzionalmente, il file `*.sug` per la propria lingua e la propria codifica dei caratteri da *ftp://ftp.vim.org/pub/vim/runtime/spell/* e salvarli in `/usr/share/vim/vim70/spell/`.

Per usare questi file spell, sono necessarie alcune configurazioni dentro `/etc/vimrc`, es.:

```
set spelllang=en,ru
set spell
```

Per ulteriori informazioni, vedere l'appropriato file README situato all'URL di sopra.

6.57.3. Contenuti di Vim

Programmi installati: efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, vim132, vim2html.pl, vimdiff (link to vim), vimm, vimspell.sh, vimtutor, and xxd

Brevi descrizioni

efm_filter.pl	È un filtro per creare un file errore che possa essere letto da **vim**
efm_perl.pl	Riformatta i messaggi di errore dell'interprete Perl per l'uso col modo «quickfix» di **vim**
ex	Avvia **vim** in ex mode.
less.sh	È uno script che avvia **vim** con less.vim
mve.awk	Processa errori **vim**.
pltags.pl	Crea un file tag per codice perl, per l'uso da parte di **vim**
ref	Verifica lo spelling degli argomenti
rview	È una versione ristretta di **view**; non può essere avviato nessun comando shell e **view** non può venire sospeso
rvim	È una versione ristretta di **vim**: non può essere avviato nessun comando shell e **vim** non può venire sospeso

shtags.pl	Genera un file tag per script Perl
tcltags	Genera un file tag per codice TCL
view	Avvia **vim** in modalità sola lettura
vi	È il link a **vim**
vim	È l'editor
vim132	Avvia **vim** con il terminale in modo 132 colonne
vim2html.pl	Converte documentazione Vim in HypterText Markup Language (HTML)
vimdiff	Edita due o tre versioni di un file **vim** e mostra le differenze
vimm	Abilita il DEC locator input model su un terminale remoto
vimspell.sh	È uno script che scrive un file e genera istruzioni sintattiche necessarie per l'evidenziazione in **vim**. Questo script richiede il vecchio comando Unix **spell**, che non è fornito nè in LFS nè in BLFS
vimtutor	Insegna tasti e comandi base di **vim**
xxd	Fa una visualizzazione esadecimale del file specificato; può anche fare il contrario, così può essere usato per la correzione dei binari

6.58. Simboli di debug

Molti programmi e librerie sono compilati, per default, con i simboli di debug inclusi (con il parametro di **gcc** `-g`). Questo significa che, facendo il debug di un programma o libreria compilato con le informazioni di debug incluse, il debugger può dare non solo indirizzi di memoria, ma anche nomi di routine e variabili.

L'inclusione di questi simboli di debug, tuttavia, aumenta significativamente le dimensioni di un programma o libreria. Per avere un'idea dell'ammontare di spazio occupato da questi simboli si dia un'occhiata ai dati seguenti:

- un binario **bash** con i simboli di debug: 1200 KB
- un binario **bash** senza i simboli di debug: 480 KB
- file di Glibc e GCC (`/lib` e `/usr/lib`)con i simboli di debug: 87 MB
- file di Glibc e GCC senza i simboli di debug: 16 MB

Le dimensioni possono in qualche modo variare in funzione di quale compilatore è stato usato e quale libreria C, ma quando si confrontano programmi con e senza i simboli di debug la differenza di solito sarà di un fattore tra 2 e 5.

Dal momento che la maggior parte delle persone probabilmente non userà mai un debugger col proprio software di sistema, è possibile guadagnare un sacco di spazio disco rimuovendo questi simboli. La prossima sezione mostrerà come eliminare tutti i simboli di debug da tutti i programmi e le librerie. Informazioni su altri modi di ottimizzare il proprio sistema si possono trovare su *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*.

6.59. Eseguire nuovamente lo strip

Se non si è programmatori e non si intende fare nessun debugging sul proprio software di sistema, è possibile ridurne l'ingombro di circa 90 MB rimuovendo i simboli di debug dai binari e dalle librerie. Questo non causa alcun inconveniente, tranne non avere più alcuna possibilità di fare un debug completo del software.

Molte persone che usano il comando mostrato di seguito non hanno alcun problema. È facile, tuttavia, fare un errore di digitazione e rendere il proprio sistema inutilizzabile, così prima di eseguire il comando **strip** è probabilmente una buona idea fare un backup del sistema LFS nella situazione corrente.

Prima di eseguire la compattazione, è necessario porre particolare attenzione ad assicurare che non siano in esecuzione nessuno dei binari che stanno per essere compattati. Se non si è certi di essere entrati in chroot con il comando dato nella Sezione 6.4, «Accesso all'ambiente chroot,», uscire prima da chroot:

```
logout
```

Quindi rientrarvi con:

```
chroot $LFS /tools/bin/env -i \
    HOME=/root TERM=$TERM PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /tools/bin/bash --login
```

Ora è possibile compattare in sicurezza binari e librerie:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
   -exec /tools/bin/strip --strip-debug '{}' ';'
```

Il sistema segnalerà che non riconosce il formato di un grande numero di file. Questi avvisi possono essere tranquillamente ignorati. Questi avvisi significano solo che questi file sono script invece che binari.

Se si hanno seri problemi di spazio sul disco, è possibile utilizzare l'opzione `--strip-all` sui binari in `/{,usr/}{bin,sbin}` per guadagnare molti più megabyte. Non usare questa opzione su librerie in quanto verrebbero distrutte.

6.60. Pulizia

D'ora in avanti quando si rientra nell'ambiente chroot dopo essere usciti, usare il seguente comando chroot modificato:

```
chroot "$LFS" /usr/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /bin/bash --login
```

La ragione per questo è che i programmi in `/tools` non sono più necessari. Perciò si può cancellare la directory `/tools` se lo si desidera.

Nota

La rimozione di `/tools` rimuoverà anche le copie temporanee di Tcl, Expect e DejaGNU, che sono state usate per eseguire i test della toolchain. Se in seguito si ha bisogno di questi programmi, essi dovranno essere ricompilati e reinstallati. Il libro BLFS ha istruzioni per questo (vedere *http://www.linuxfromscratch.org/blfs/*).

Se i file system virtuali del kernel sono stati smontati, sia manualmente sia con un reboot, assicurarsi che siano ri-montati quando si entra di nuovo nel chroot. Questo processo è stato spiegato in Sezione 6.2.2, «Montaggio e popolamento di /dev» e in Sezione 6.2.3, «Montaggio dei file system virtuali del kernel».

Capitolo 7. Impostazione degli script di avvio del sistema

7.1. Introduzione

In questo capitolo si installeranno e imposteranno gli LFS-Bootscripts. Molti di questi script funzioneranno senza la necessità di alcuna modifica, ma alcuni richiedono file di configurazione aggiuntivi, poiché si interfacciano ad informazioni dipendenti dall'hardware.

In questo libro sono stati utilizzati script di avvio stile System-V perché essi sono largamente utilizzati. Per altre opzioni, è disponibile uno script che descrive le impostazioni di avvio stile BSD su *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. La ricerca della parola «depinit» nelle mailing list di LFS, inoltre, offrirà ulteriori scelte.

Se si decide di utilizzare uno stile alternativo per gli init script saltare questo capitolo ed andare al Capitolo 8.

7.2. LFS-Bootscripts-6.2

Il pacchetto LFS-Bootscripts contiene un insieme di script per avviare/arrestare il sistema LFS all'accensione/spegnimento.

Tempo di costruzione approssimativo: meno di 0.1 SBU
Spazio necessario su disco: 0.4 MB

7.2.1. Installazione di LFS-Bootscripts

Installazione del pacchetto:

```
make install
```

7.2.2. Contenuti di LFS-Bootscripts

Script installati:: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template, and udev

Brevi descrizioni

checkfs	Verifica l'integrità dei file system prima che vengano montati (con l'eccezione dei file system di tipo journal e di rete)
cleanfs	Rimuove i file che non devono essere conservati tra un riavvio e l'altro, come quelli in `/var/run/` e `/var/lock/`. Esso ricrea `/var/run/utmp` e rimuove i file `/etc/nologin`, `/fastboot` eventualmente presenti, e i file `/forcefsck`
console	Carica la mappa caratteri corretta della tastiera specificata per la propria tastiera e configura il font per lo schermo
functions	Contiene funzioni comuni, come verifiche di errore e stato, utilizzate da numerosi script di avvio
halt	Arresta il sistema
ifdown	Assiste lo script di rete nell'arresto dei dispositivi di rete
ifup	Assiste lo script di rete nell'avvio dei dispositivi di rete
localnet	Imposta il nome host e la periferica locale di loopback del sistema
mountfs	Monta tutti i file system, tranne quelli segnati come *noauto* o di rete
mountkernfs	Monta i file system virtuali del kernel, come `proc`
network	Imposta le interfacce di rete, come le schede di rete, e imposta il gateway di default (ove applicabile)
rc	È lo script principale di controllo dei run-level. È responsabile dell'esecuzione di tutti gli altri bootscript uno per uno, in una sequenza determinata dal nome dei link simbolici che vengono processati
reboot	Riavvia il sistema
sendsignals	Assicura che ogni processo sia terminato prima che il sistema si riavvii o si arresti

setclock	Reimposta l'orologio del kernel a localtime nel caso che l'orologio hardware non sia impostato a UTC
static	Fornisce la funzionalità necessaria per assegnare un indirizzo Internet Protocol (IP) statico a un'interfaccia di rete
swap	Abilita e disabilita file swap e partizioni
sysklogd	Avvia e arresta i demoni di log del kernel e del sistema
template	È un modello che si può utilizzare per creare i propri script di avvio per altri demoni
udev	Prepara la directory `/dev` e avvia Udev

7.3. Come funziona il processo di avvio con questi script?

Linux utilizza uno speciale servizio di boot chiamato SysVinit, basato sul concetto dei *run-level*. Può essere estremamente differente da un sistema ad un altro, perciò non possiamo assumere che se le cose funzionano in una particolare distribuzione Linux, esse funzioneranno anche in LFS. LFS ha il suo modo di fare le cose, ma rispetta gli standard generalmente accettati.

SysVinit (che chiameremo «init» da adesso in poi) funziona utilizzando uno schema a run-level. Ci sono 7 (da 0 a 6) run-level (attualmente ci sono più run-level, ma sono per casi speciali e generalmente non sono utilizzati. Vedere `init(8)` per maggiori dettagli), ed ognuno di essi corrisponde alle cose che il computer dovrebbe fare quando si avvia. Il run-level predefinito è il 3. Qui sono elencati i differenti livelli con una descrizione di come sono solitamente implementati:

```
0: spegne il computer
1: modalità singolo utente
2: modalità multi-utente senza rete
3: modalità multi-utente con rete
4: riservato per la personalizzazione, altrimenti lo stesso del livello 3
5: come il livello 4, è solitamente utilizzato per il login grafico (come con xdm per X o kdm per KDE)
6: riavvia il computer
```

Il comando utilizzato per cambiare livello è **init *<run-level>*** dove *<run-level>* è il run-level di destinazione. Per esempio, per riavviare il computer, un utente lancerebbe il comando **init 6**. il quale è solo un alias del comando **reboot**. Così come il comando **init 0** è un alias per il comando **halt** .

Ci sono un certo numero di directory in `/etc/rc.d` con nomi del tipo `rc?.d` (dove ? è il numero di run-level) e `rcsysinit.d`, contenenti una serie di link simbolici. Alcuni iniziano con la lettera *K*, gli altri con la lettera *S*, e tutti hanno due numeri dopo la prima lettera. La lettera K specifica di fermare (kill) un servizio, e la S significa far partire un servizio. I numeri determinano l'ordine con cui gli script vengono avviati, da 00 a 99— più basso è il numero prima lo script verrà eseguito. Quando **init** cambia run-level, i servizi specificati vengono fermati e altri vengono avviati, dipendendo dal runlevel scelto.

Gli script reali sono in `/etc/rc.d/init.d`. Essi eseguono il lavoro e tutti i link simbolici puntano ad essi. I link di avvio e di stop dei servizi puntano entrami allo stesso script in `/etc/rc.d/init.d`. Questo perché gli script possono essere chiamati con parametri differenti, come *`start`*, *`stop`*, *`restart`*, *`reload`*, e *`status`*. Quando si incontra un link K, lo script appropriato viene lanciato con l'argomento *`stop`*. Quando si incontra un link S, lo script appropriato viene lanciato con l'argomento *`start`*.

C'è un'eccezione a questa spiegazione. I link che iniziano con la lettera *S* nelle directory `rc0.d` e `rc6.d` non causano l'avvio di nessuno script. Essi vengono chiamati con il parametro *`stop`* per fermare qualche cosa. La logica dietro a questo comportamento è che durante il riavvio o lo spegnimento del sistema non deve essere avviato nulla. Il sistema deve solo essere fermato.

Queste sono le descrizioni di cosa gli argomenti fanno fare agli script:

`start`
: Il servizio viene avviato.

`stop`
: Il servizio viene fermato.

`restart`
: Il servizio viene fermato e poi avviato.

`reload`

La configurazione del servizio viene aggiornata. È utilizzato dopo che il file di configurazione di un servizio viene modificato e il servizio non necessita di essere riavviato.

status
Specifica se il servizio sta funzionando e con quale PID.

Ci si senta liberi di modificare il modo in cui funziona il processo di avvio (dopotutto, è il proprio sistema LFS). I file forniti sono solamente un esempio di come può essere fatto.

7.4. Gestione dei dispositivi e dei moduli in un sistema LFS

Nel Capitolo 6, è stato installato il pacchetto Udev. Prima di scendere nel dettaglio di come funziona, facciamo un breve accenno al precedente metodo per la gestione delle periferiche/dispositivi.

I sistemi Linux usano tradizionalmente un metodo statico di creazione dei dispositivi, in questo modo si creano molti device node nella directory `/dev` (qualche volta letteralmente migliaia di nodi), anche se non esiste in quel momento l'hardware corrispondente. Questo avviene tramite lo script **MAKEDEV**, che contiene un numero di chiamate al programma **mknod**, con i relativi numeri major e minor della periferica, pari ad ogni possibile periferica che possa esistere al mondo.

Usando il metodo udev, vengono creati i device node solo per quelle periferiche che vengono trovate dal kernel. Dato che questi device node vengono creati ad ogni avvio del sistema, vengono memorizzati in un file system `tmpfs` (un file system virtuale che risiede interamente in memoria). Dato che i device node non richiedono molto spazio disco, la memoria usata è irrisoria.

7.4.1. Cronistoria

Nel febbraio 2000, un nuovo filesystem chiamato `devfs` venne intergrato nel kernel 2.3.46 e divenne disponibile con la serie 2.4 dei kernel stabili. Sebbene fosse presente nei sorgenti del kernel, questo metodo di creazione dinamica dei dispositivi non ha mai ricevuto un consenso ed un supporto unanime da parte degli sviluppatori del kernel.

Il problema principale dell'approccio adottato da `devfs` è stato la gestione dell'identificazione delle periferiche, della loro creazione e nomenclatura. D'altra parte, il problema della nomenclatura dei device node era forse il più critico. È generalmente accettato che se i nomi dei dispositivi sono configurabili la politica di come chiamare le periferiche deve essere lasciata all'amministratore di sistema e non essere imposta da un particolare sviluppatore. Il file system `devfs` soffre anche di condizioni che sono inerenti alla propria progettazione e non possono essere corrette senza una sostanziale revisione del kernel. È stato anche contrassegnato come deprecato a causa della mancanza di manutenzione.

Con lo sviluppo dell'albero instabile del kernel della serie 2.5, che ha poi portato alla serie 2.6 dei kernel stabili, è stato introdotto un nuovo filesystem virtuale chiamato `sysfs`. Quello che fa `sysfs` è esportare una vista della configurazione del sistema hardware nello userspace. Con questa rappresentazione visibile dallo userspace, la possibilità di vedere un sostituto di `devfs` è diventata molto più realistica.

7.4.2. Implementazione di Udev

7.4.2.1. Sysfs

Il filesystem `sysfs` è stato menzionato brevemente sopra. Ci si può meravigliare di come `sysfs` conosca le periferiche presenti sul sistema e quale numero di dispositivo usare per loro. I driver che sono stati compilati direttamente nel kernel registrano i propri oggetti nel `sysfs` non appena sono riconosciuti dal kernel. Per i driver compilati come moduli, questo succede non appena il modulo viene caricato. Una volta che il filesystem `sysfs` viene montato (in `/sys`), i dati che i driver compilati nel kernel registrano attraverso `sysfs` divengono disponibili allo userspace dei processi e a **udevd** per la creazione dei device node.

7.4.2.2. Il bootscript Udev

Lo script di avvio **S10udev** si occupa della creazione di questi device node quando Linux viene avviato. Questo script disattiva il gestore di uevent dalla **/sbin/hotplug** predefinita. Questo viene fatto perché il kernel non ha più bisogno di richiamare un binario esterno. Invece **udevd** ascolterà su un socket netlink per gli uevent che il kernel raccoglie. Dopo, il bootscript copia ogni device node statico che esiste in `/lib/udev/devices` in `/dev`. Questo è necessario perché alcuni dispositivi, directory, e symlink sono necessari prima che i processi di gestione dinamica dei dispositivi siano disponibili durante i primi passi dell'avvio di un sistema. La creazione di device node statici in `/lib/udev/devices` è una soluzione semplice anche per i dispositivi che non sono supportati dall'infrastruttura di gestione dinamica dei device node. Il bootscript poi avvia il demone Udev, **udevd**, il quale agirà in base agli uevent che riceve. Infine, il bootscript forza il kernel a replicare gli uevent per ciascun dispositivo che è già stato registrato e poi aspetta **udevd** che li gestisca.

7.4.2.3. Creazione dei Device Node

Per ottenere i corretti numeri major e minor di un dispositivo, Udev fa affidamento sull'informazione fornita da `sysfs` dentro `/sys`. Per esempio, `/sys/class/tty/vcs/dev` contiene la stringa «7:0». Questa stringa è usata da **udevd** per creare un device node con numero major *7* e minor *0*. I nomi e i permessi dei nodi creati sotto la directory `/dev` sono determinati da regole specificate nei file dentro la directory `/etc/udev/rules.d/`. Questi sono numerati in una maniera simile al pacchetto LFS-Bootscripts. Se **udevd** non può trovare una regola per il dispositivo lo crea, esso avrà permessi di default a *660* e il proprietario a *root:root*. La documentazione della sintassi dei file di configurazione delle regole di Udev è in `/usr/share/doc/udev-096/index.html`

7.4.2.4. Caricamento dei Moduli

I driver dei device compilati come moduli possono avere inclusi degli alias. Gli alias sono visibili nell'output del programma **modinfo** e sono di solito associati agli identificativi dei dispositivi specifici del bus, supportati dal modulo. Per esempio, il driver *snd-fm801* supporta dispositivi PCI con vendor ID 0x1319 e device ID 0x0801, e ha un alias «pci:v00001319d00000801sv*sd*bc04sc01i*». Per la maggior parte dei dispositivi, il driver bus esporta l'alias del driver che gestirebbe il dispositivo via `sysfs`. Es., il file `/sys/bus/pci/devices/0000:00:0d.0/modalias` può contenere la stringa «pci:v00001319d00000801sv00001319sd00001319bc04sc01i00». Le regole che LFS installa faranno sì che **udevd** chiami **/sbin/modprobe** con i contenuti della variabile d'ambiente uevent `MODALIAS` (che dovrebbe essere lo stesso del contenuto del file `modalias` dentro sysfs), caricando in questo modo tutti i moduli i cui alias corrispondono con la stringa dopo l'espansione del metacarattere.

In questo esempio, questo significa che, in aggiunta a *snd-fm801*, l'obsoleto (e indesiderato) driver *forte* sarà caricato se è disponibile. Guardare sotto per i modi in cui il caricamento dei driver indesiderati possono essere prevenuti.

Il kernel per se stesso è anche capace di caricare moduli per i protocolli di rete, filesystem e supporto a NLS su richiesta.

7.4.2.5. Gestione delle periferiche dinamiche e hotplug

Quando si collega una periferica, come un lettore MP3 USB (Universal Serial Bus), il kernel riconosce che la periferica è ora collegata e genera un evento hotplug. Questo uevent viene poi gestito da **udevd** come descritto sopra.

7.4.3. Problemi legati al caricamento dei moduli e alla creazione di dispositivi

Ci sono alcuni problemi conosciuti quando si creano automaticamente dei device node.

7.4.3.1. Un modulo del kernel non viene caricato automaticamente

Udev caricherà un modulo solamente se esso ha un alias specifico del bus e il driver del bus abbia propriamente esportato l'alias necessario in `sysfs`. Negli altri casi, si dovrebbe organizzare il caricamento del modulo attraverso altri mezzi. Con Linux-2.6.16.27, Udev è conosciuto caricare driver propriamente scritti per dispositivi INPUT, IDE, PCI, USB, SCSI, SERIO e FireWire.

Per determinare se il driver del dispositivo che si richiede abbia il necessario supporto per Udev, eseguire **modinfo** con il nome del modulo come argomento. Adesso provare a localizzare la directory del dispositivo sotto `/sys/bus` e controllare se lì c'è un file `modalias`.

Se esiste il file `modalias` in `sysfs`, il driver supporta il dispositivo e può comunicare con esso direttamente, ma se non ha l'alias, questo è un bug nel driver. Caricare il driver senza l'aiuto di Udev e aspettare che il problema sia risolto successivamente.

Se non c'è il file `modalias` dentro la relativa directory sotto `/sys/bus`, questo significa che gli sviluppatori del kernel non hanno ancora aggiunto il supporto modalias a questo tipo di bus. Con Linux-2.6.16.27, è questo il caso con i bus ISA. Aspettare che questo problema sia risolto nelle successive versioni del kernel.

Udev non è affatto pensato per caricare driver «wrapper» come *snd-pcm-oss* e driver non-hardware come *loop*.

7.4.3.2. Un modulo del kernel non viene caricato automaticamente, e Udev non è fatto per caricarlo

Se il modulo «wrapper» migliora soltanto la funzionalità fornita da alcuni altri moduli (es., *snd-pcm-oss* migliora la funzionalità di *snd-pcm* rendendo la scheda audio disponibile alle applicazioni OSS), configurare **modprobe** per caricare il wrapper dopo che Udev abbia caricato il modulo wrapped. Per fare questo, aggiungere una riga «install» dentro `/etc/modprobe.conf`. Per esempio:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

Se il modulo in questione non è un wrapper e è utile per se stesso, configurare il bootscript **S05modules** per caricare questo modulo all'avvio del sistema. Per fare questo, aggiungere il nome del modulo al file `/etc/sysconfig/modules` su una riga separata. Questo funziona anche per i moduli wrapper, ma in questi casi sarebbe sottoottimizzato.

7.4.3.3. Udev carica alcuni moduli indesiderati

O non compilare il modulo, o porlo nella blacklist dentro il file `/etc/modprobe.conf` come fatto con il modulo *forte* nell'esempio di seguito:

```
blacklist forte
```

I moduli posti nella Blacklist possono ancora essere caricati manualmente con il comando esplicito **modprobe**.

7.4.3.4. Udev crea un dispositivo non correttamente, o crea un symlink sbagliato

Questo di solito accade se una regola si combina con un dispositivo in modo inaspettato. Per esempio, una regola mal scritta può combinarsi sia con un disco SCSI (come desiderato) e sia con il corrispondente driver generico SCSI (non correttamente) del commerciante. Trovare la regola dannosa e renderla più specifica.

7.4.3.5. Regola Udev non funziona in modo affidabile

Questo può essere un'altra manifestazione del precedente problema. Se no, e la propria regola usa gli attributi di `sysfs`, può essere un problema di sincronizzazione, da correggere nei prossimi kernel. Per adesso, si può lavorare intorno ad esso creando una regola che aspetta l'attributo utilizzato di `sysfs` e lo appende al file `/etc/udev/rules.d/10-wait_for_sysfs.rules`. Pregasi informare la lista LFS Development se si fa così e questo è di aiuto.

7.4.3.6. Udev non crea un dispositivo

Un ulteriore testo suppone che il driver sia compilato staticamente nel kernel o sia già caricato come modulo, e che si sia già controllato che Udev non abbia creato un dispositivo malnominandolo.

Udev non ha informazioni necessarie per creare un device node se un driver del kernel non ha esportato i propri dati in `sysfs`. Questo è più comune con driver di terze parti al di fuori dell'albero del kernel. Creare un device node statico in `/lib/udev/devices` con gli approriati numeri major/minor (vedere il file `devices.txt` dentro la documentazione del kernel o la documentazione fornita dal comerciale del driver di terze parti). Il device node statico sarà copiato in `/dev` dal bootscript **S10udev**.

7.4.3.7. L'ordine di nomenclatura dei dispositivi cambia casualmente dopo il riavvio

Questo è dovuto al fatto che Udev, per progetto, gestisce uevent e carica i moduli in parallelo, e pertanto in un ordine imprevedibile. Questo non sarà mai «corretto». Non si dovrebbe far affidamento sul fatto che i nomi dei dispositivi del kernel diventino stabili. Invece, creare le proprie regole in modo tale che facciano dei symlink con dei nomi stabili basati su alcuni attributi stabili del dispositivo, come un numero seriale o l'output delle varie utility *_id installate da Udev. Vedere Sezione 7.12, «Creazione personalizzata di link simbolici ai dispositivi» e Sezione 7.13, «Configurazione dello script di rete» per degli esempi.

7.4.4. Letture Utili

Documentazione utile aggiuntiva è disponibile ai seguenti siti:

- Un'Implementazione in Userspace di `devfs`
  http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- udev FAQ
  http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ
- Il Filesystem `sysfs`
  http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf

7.5. Configurazione dello script setclock

Lo script **setclock** legge il tempo dall'orologio hardware, conosciuto anche come orologio del BIOS o CMOS (Complementry Metal-Oxide Semiconductor). Se l'orologio hardware è impostato su UTC lo script convertirà l'ora dell'orologio hardware nell'ora locale utilizzando il file `/etc/localtime` (che dice al programma **hwclock** il fuso orario dell'utente). Non esiste modo per determinare se l'orologio hardware è impostato a UTC o meno, perciò è necessario configurarlo manualmente.

Se non ci si ricorda se l'orologio hardware è impostato su UTC, lo si può determinare eseguendo il comando **`hwclock --localtime --show`**. Questo dirà quale è l'ora corrente secondo con l'orologio hardware. Se questo collima con l'ora effettiva, l'orologio hardware è configurato con l'ora locale. Se l'output di **hwclock** non corrisponde all'ora locale è probabile che sia configurato con l'ora UTC. Lo si verifichi aggiungendo o sottraendo l'appropriato numero di ore del proprio fuso orario all'ora mostrata da **hwclock**. Per esempio, se si vive nella fascia MST, conosciuta anche come GMT -0700, aggiungere sette ore all'ora locale.

Cambiare il valore della variabile `UTC` a *`0`* (zero) se l'orologio hardware *non* è impostato su UTC.

Creare un nuovo file `/etc/sysconfig/clock` con il seguente comando:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock
EOF
```

Un buon hint che spiega come gestire il tempo in LFS è disponibile presso *http://www.linuxfromscratch.org/hints/downloads/files/time.txt*. Il documento tratta argomenti come i fusi orari, UTC e la variabile d'ambiente `TZ`.

7.6. Configurazione della console Linux

In questa sezione si descrive come configurare lo script di avvio **console** che imposta la mappa della tastiera (keymap) e i font della console. Se non si useranno caratteri non-ASCII (es., il segno di copyright, la sterlina inglese e il simbolo dell'Euro) e la tastiera è US saltare questa sezione. Senza il file di configurazione lo script di avvio della **console** non farà nulla.

Lo script **console** legge il file `/etc/sysconfig/console` per avere informazioni di configurazione. Decide quale mappa per la tastiera e quale font per lo schermo verrà usato. Vari HOWTO specifici per la lingua possono aiutare con esso, vedere *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Se si è ancora in dubbio, controllare la directory `/usr/share/kbd` per avere mappe di tastiera e font validi. Leggere le pagine di manuale di `loadkeys(1)` e `setfont(8)` per determinare i parametri corretti per questi programmi.

Il file `/etc/sysconfig/console` deve contenere linee nella forma: VARIABLE="value". Vengono riconosciute le seguenti variabili:

KEYMAP
: Questa variabile specifica gli argomenti per il programma **loadkeys**, tipicamente il nome della mappa tastiera da caricare, ad esempio «es». Se questa variabile non è impostata lo script di avvio non eseguirà il programma **loadkeys**, e verrà usata la mappa tastiera predefinita.

KEYMAP_CORRECTIONS
: Questa variabile (raramente usata) specifica gli argomenti per la seconda chiamata al programma **loadkeys**. Questo è utile se la mappa tastiera fornita non è totalmente soddisfacente e deve essere fatto un piccolo aggiustamento. Ad esempio per includere il segno Euro in una mappa tastiera che normalmente non lo ha, impostare questa variabile a «euro2».

FONT
: Questa variabile specifica gli argomenti per il programma **setfont**. Tipicamente questo include il nome del font, «-m», e il nome della mappa caratteri di applicazione da caricare. Ad esempio per caricare il font «lat1-16» assieme alla mappa caratteri di applicazione «8859-1» (come è appropriato negli USA), impostare questa variabile a «lat1-16 -m 8859-1». Se questa variabile non è impostata il bootscript non eseguirà il programma **setfont**, e il font VGA predefinito sarà usato assieme alla mappa caratteri di applicazione predefinita.

UNICODE
: Impostare questa variabile a «1», «yes» o «true» per porre la console in modalità UTF-8. Essa è utile nelle localizzazioni basate su UTF-8, altrimenti è dannosa.

LEGACY_CHARSET
: Per molte configurazioni di tastiera non c'è una mappa tastiera Unicode tra quelle fornite nel pacchetto Kbd. Il bootscript **console** convertirà al volo una mappa tastiera disponibile in UTF-8 se questa variabile è impostata per la codifica della mappa tastiera disponibile non-UTF-8. Notare, tuttavia, che i dead key (ovvero, i tasti che di per s´ stessi non producono un carattere, ma mettono un accento su un carattere prodotto dal tasto successivo; non ci sono dead key nella tastiera US standard) e i tasti composti (es., premendo Ctrl+. A E allo scopo di produrre il carattere Æ) non funzioneranno nel modo UTF-8 senza una patch speciale sul kernel. Questa variabile è utile solo in modalità UTF-8.

BROKEN_COMPOSE
: Impostare questa a «0» se si intende applicare la patch del kernel nel Capitolo 8. Notare che bisogna anche aggiungere il set caratteri corretto per le regole di composizione nella propria mappa tastiera alla variabile FONT dopo l'opzione «-m». Questa variabile è solo nel modo UTF-8.

Il supporto per compilare la mappa tastiera direttamente nel kernel è stato rimosso, poich´ è stato riportato che induce risultati scorretti.

Alcuni esempi:

- Per una impostazione non-Unicode generalmente sono necessarie solo le variabili KEYMAP e FONT. Es., per un'impostazione polacca si userà:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Come accennato in precedenza talvolta è necessario aggiustare leggermente una mappa tastiera fornita. L'esempio seguente aggiunge il simbolo Euro alla mappa tastiera tedesca:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Quello seguente è un esempio abilitato Unicode per quella Bulgara, dove esiste una mappa tastiera fornita e non definisce alcun dead key o regola di composizione:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- A causa dell'uso di un font 512-glyph LatArCyrHeb-16 nell'esempio precedente i colori chiari non sono più disponibili nella console Linux a meno che si usi un framebuffer. Se si vogliono avere colori chiari senza framebuffer, e si può vivere senza caratteri che non appartengono al proprio linguaggio, è ancora possibile usare un font 256-glyph specifico del linguaggio, come illustrato di seguito.

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- L'esempio seguente illustra l'autoconversione della mappa tastiera da ISO-8859-15 a UTF-8 e abilita i dead key in modalità Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
BROKEN_COMPOSE="0"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Per cinese, giapponese, coreano e altri lunguaggi, la console Linux non può essere configurata per visualizzare i caratteri necessari. Gli utenti che hanno bisogno di tali linguaggi devono installare l'X Window System, font che coprono gli insiemi di caratteri necessari, e i metodi di input appropriati (ad es., SCIM supporta un'ampia varietà di linguaggi).

Nota

Il file `/etc/sysconfig/console` controlla solo la localizzazione della console testuale di Linux. Non ha niente a che fare con l'impostazione della configurazione di tastiera appropriata e i font del terminale nell'X Window System, con sessioni ssh o con una console seriale.

7.7. Configurazione dello script sysklogd

Lo script `sysklogd` invoca il programma **syslogd** con l'opzione *`-m 0`*. Questa opzione disattiva la marcatura oraria periodica che **syslogd** scrive nei file di log ogni 20 minuti per default. Se si vuole attivare questa marcatura oraria periodica, editare lo script `sysklogd` e apportare i cambiamenti di conseguenza. Vedere **`man syslogd`** per maggiori informazioni.

7.8. Creazione del file /etc/inputrc

Il file `inputrc` gestisce la mappa tastiera per situazioni specifiche. Questo file è il file di avvio usato da Readline — la libreria per l'input — usato da Bash e molte altre shell.

Molte persone non hanno bisogno di mappe tastiera personalizzate, quindi il comando seguente crea un `/etc/inputrc` globale usato da chiunque acceda. Se in seguito si decide che si ha bisogno di sovrascrivere i default su una base personalizzata per l'utente si può creare un file `.inputrc` nella home directory dell'utente con le mappe modificate.

Per maggiori informazioni su come editare il file `inputrc` vedere **info bash** nella sezione *Readline Init File*. Anche **info readline** è una buona sorgente di informazione.

Di seguito c'è un `inputrc` globale generico con commenti che spiegano cosa fanno le varie opzioni. Notare che i commenti non possono essere sulla stessa linea dei comandi. Creare il file usando il seguente comando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the
# value contained inside the 1st argument to the
# readline specific functions

"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
```

```
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. I file di avvio della shell Bash

Il programma di shell **/bin/bash** (a cui faremo riferimento come «la shell» da ora in poi) usa una collezione di file di avvio per creare un ambiente in cui funzionare. Ogni file ha uno specifico uso e può influenzare login e ambienti interattivi in modi differenti. I file nella directory `/etc` forniscono configurazioni globali. Se esiste un file equivalente nella propria home directory, questo sovrascriverà le definizioni globali.

Una shell login interattiva viene fatta partire dopo un login portato a termine con successo, usando **/bin/login**, leggendo il file `/etc/passwd`. Una shell non-login interattiva viene fatta partire da linea di comando (es., `[prompt]$` **/bin/bash**). Una shell non interattiva di solito è presente quando è in esecuzione uno shell script. Non è interattiva perché sta processando uno script e non aspetta un input dell'utente tra i vari comandi.

Per maggiori informazioni vedere **info bash** nella sezione *Bash Startup Files and Interactive Shells*.

I file `/etc/profile` e `~/.bash_profile` sono letti quando una shell viene invocata come login shell interattiva.

Il file `/etc/profile` di base creato di seguito definisce alcune variabili ambiente necessarie per il supporto del linguaggio nativo. Definendole appropriatamente si ottiene:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

Questo script definisce anche la variabile ambiente `INPUTRC` che permette a Bash e a Readline di usare il file `/etc/inputrc` creato prima.

Sostituire *<ll>* con le due lettere del codice della propria lingua (es. «en») e *<CC>* con le due lettere del codice del proprio paese (es. «GB»). *<charmap>* deve essere rimpiazzato con il charmap canonico per la propria localizzazione scelta. Possono essere presenti anche modifiche opzionali come «@euro».

L'elenco di tutte le localizzazioni supportate da Glibc può essere ottenuto eseguendo il seguente comando:

```
locale -a
```

Le localizzazioni possono avere molti sinonimi, ad es. a «ISO-8859-1» si fa anche riferimento come «iso8859-1» e «iso88591». Alcune applicazioni non possono gestire correttamente i vari sinonimi (es., richiedere che «UTF-8» sia scritto come «UTF-8», e non «utf8»), quindi è più sicuro nella maggioranza dei casi scegliere il nome canonico per una particolare localizzazione. Per determinare il nome canonico eseguire il seguente comando, dove *<locale name>* è l'output dato da **locale -a** per la propria localizzazione preferita («en_GB.iso88591» nel nostro esempio).

```
LC_ALL=<locale name> locale charmap
```

Per la localizzazione «en_GB.iso88591» il comando precedente stamperà:

```
ISO-8859-1
```

Il risultato di questo è un'impostazione finale di «en_GB.ISO-8859-1». È importante che la localizzazione

trovata usando l'euristica precedente sia testata prima di aggiungerla ai file di avvio di Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

I comandi precedenti dovrebbero stampare i nomi di nazione e linguaggio, la codifica caratteri usata dalla localizzazione, la moneta locale e il prefisso da anteporre al numero telefonico per chiamare il paese. Se uno qualunque dei comandi precedenti fallisce con un messaggio simile a quello mostrato sotto ciò significa che la propria localizzazione non è stata installata nel capitolo 6 o non è supportata dalla installazione di default di Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Se ciò succede bisogna installare la localizzazione desiderata usando il comando **localedef**, o considerare la scelta di una localizzazione differente. Le prossime istruzioni suppongono che non ci sia tale messaggio di errore in Glibc.

Anche alcuni pacchetti oltre LFS potrebbero non avere il supporto per la localizzazione scelta. Un esempio è la libreria X (parte del sistema X Window), che emette il seguente messaggo di errore se la localizzazione non corrisponde esattamente a uno dei nomi delle mappe caratteri nei propri file interni:

```
Warning: locale not supported by Xlib, locale set to C
```

In parecchi casi Xlib si aspetta che la mappa caratteri sia elencata in maiuscolo con tratti canonici. Per esempio, "ISO-8859-1" invece che "iso88591". È anche possibile trovare un'appropriata specificazione rimuovendo la mappa caratteri parte delle specifiche locali. Questo può essere verificato eseguendo il comando **locale charmap** in entrambe le localizzazioni. Per esempio uno potrebbe dover cambiare "de_DE.ISO-8859-15@euro" in "de_DE@euro" per avere questa localizzazione riconosciuta da Xlib.

Anche altri pacchetti potrebbero funzionare scorrettamente (ma non necessariamente mostrare messaggi di errore) se il nome della localizzazione non è come si aspettano. In questi casi investigare come altre distribuzioni Linux supportano la propria localizzazione potrebbe fornire utili informazioni.

Una volta che sono state determinate le impostazioni locali corrette creare il file `/etc/profile`:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>
export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

Le localizzazioni «C» (default) e «en_US» (quella raccomandata per utenti inglesi negli Stati Uniti) sono diverse. «C» usa il set di caratteri US-ASCII 7-bit, e tratta i byte con il bit più alto attivo come caratteri invalidi. Questo perché, per esempio, il comando **ls** li sostituisce con il punto interrogativo in quella localizzazione. Inoltre un tentativo di inviare mail con tali caratteri da Mutt o Pine provoca messaggi trasmessi non conformi RFC (il set dei caratteri nella posta in uscita è indicato come «unknown 8-bit»). Così si può usare il locale «C» solo se si è sicuri che non si avrà mai bisogno di caratteri 8-bit.

Localizzazioni basate su UTF-8 non sono ben supportate da molti programmi. Es., il programma **watch** visualizza solo caratteri ASCII in localizzazioni UTF-8 e non ha tale limitazione nelle localizzazioni 8-bit tradizionali come en_US. È in sviluppo la documentazione e, se possibile, la correzione di alcuni problemi, vedere *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html*.

7.10. Configurazione dello script localnet

Parte del lavoro dello script **localnet** è l'impostazione dell'hostname di sistema. Tale impostazione deve essere configurfata nel file `/etc/sysconfig/network`.

Creare il file `/etc/sysconfig/network` e inserire l'hostname eseguendo:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

<lfs> deve essere sostituito dal nome dato al computer. Non si dovrebbe inserire qui l'FQDN (Fully Qualified Domain Name). Questa informazione verrà inserita nel file `/etc/hosts` nella prossima sezione.

7.11. Personalizzazione del file /etc/hosts

Se si deve configurare una scheda di rete è necessario decidere l'indirizzo IP, l'FQDN ed i possibili alias da utilizzare nel file `/etc/hosts`. La sintassi è:

```
IP_address myhost.example.org aliases
```

A meno che il computer non debba essere visibile da Internet (per esempio se si possiede un dominio registrato e un blocco di indirizzi IP assegnati — la maggior parte degli utenti non li possiede) è necessario assicurarsi che l'indirizzo IP scelto appartenga ad un intervallo di rete privato. Gli intervalli validi sono:

```
Private Network Address Range      Normal Prefix
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x può essere qualunque numero nell'intervallo 16-31. y può essere qualunque numero nell'intervallo 0-255.

Un indirizzo IP valido potrebbe essere 192.168.1.1. Un FQDN valido per questo indirizzo IP potrebbe essere lfs.example.org.

Anche se non si utilizza una scheda di rete sarà comunque necessario impostare un FQDN. Ciò serve a certi programmi per funzionare correttamente.

Creare il file `/etc/hosts` eseguendo:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (versione per scheda di rete)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (versione per scheda di rete)
EOF
```

I valori *<192.168.1.1> <HOSTNAME.example.org>* dovranno essere modificati in funzione di necessità o utenti specifici (se l'indirizzo IP è assegnato da un amministratore di rete/sistema e la macchina dovrà essere connessa ad una rete esistente). Il nome(i) alias opzionale può essere omesso.

Se non si deve configurare una scheda di rete creare il file `/etc/hosts` eseguendo:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (versione senza scheda di rete)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (versione senza scheda di rete)
EOF
```

7.12. Creazione personalizzata di link simbolici ai dispositivi

7.12.1. link simbolici ai CD-ROM

Alcuni software che si possono voler installare in seguito (es., vari media player) si aspettano l'esistenza dei symlink /dev/cdrom e /dev/dvd. In oltre, può essere conveniente inserire i riferimenti a questi symlink dentro `/etc/fstab`. Per ciascuno dei propri dispositivi CD-ROM, cercare la directory corrispondente sotto `/sys` (es., questo può essere `/sys/block/hdd`) e eseguire un comando simile al seguente:

```
udevtest /block/hdd
```

Guardare le righe contenenti l'output di vari *_id programs.

Ci sono due approcci nel creare i link simbolici. Il primo è di usare il nome del modello e il numero seriale, il secondo è basato su la locazione del dispositivo nel bus. Se si ha intenzione di usare il primo approccio, creare un file simile al seguente:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_MODEL}=="SAMSUNG_CD-ROM_SC-148F", \
    ENV{ID_REVISION}=="PS05", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_MODEL}=="PHILIPS_CDD5301", \
    ENV{ID_SERIAL}=="5VO1306DM00190", SYMLINK+="cdrom1 dvd"

EOF
```

Nota

Sebbene gli esempi in questo libro funzionino correttamente tenere presente che udev non riconosce la backslash come proseguimento di riga. Se si modificano regole di udev con un editor, assicurarsi di lasciare ogni regola su una riga fisica.

In questo modo, i symlink resteranno correttamente anche se si spostassero i drive in una posizione differente sul bus IDE, ma il symlink `/dev/cdrom` non sarebbe creato se si rimpiazzasse il vecchio CD-ROM SAMSUNG con un drive vuoto.

La chiave SUBSYSTEM=="block" serve per evitare la corrispondenza con dispositivi SCSI generici. Senza questa, in caso di CD-ROM SCSI, il symlink qualche volta punterà al corretto dispositivo `/dev/srX`, e qualche volta a `/dev/sgX`, il quale è sbagliato.

Il secondo approccio propone:

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF

# Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-0:1", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-1:1", SYMLINK+="cdrom1 dvd"

EOF
```

In questo metodo, i symlink rimarrebbero correttamente anche se si sostituisse il drive con modelli diversi, ma li pone nelle vecchie posizioni sul bus IDE. La chiave ENV{ID_TYPE}=="cd" assicura che il symlink scompaia se si inserisse qualcosa di diverso da un CD-ROM in quella posizione nel bus.

Ovviamente, è possibile miscelare i due approcci.

7.12.2. Comportamento con dispositivi duplicati

Come spiegato in Sezione 7.4, «Gestione dei dispositivi e dei moduli in un sistema LFS», l'ordine nel quale i dispositivi con la stessa funzione appaiono in `/dev` è essenzialmente random. Es., se si ha una web camera USB e un sintonizzatore TV, qualche volta `/dev/video0` si riferisce alla camera e `/dev/video1` si riferisce al sintonizzatore, e qualche volta dopo un riavvio l'ordine diventa opposto. Per tutte le classi di hardware ad eccezione delle schede audio e delle schede di rete, questo è correggibile con la creazione personalizzata di regole di udev per symlink persistenti. Nel caso di schede di rete è descritto separatamente in Sezione 7.13, «Configurazione dello script di rete», e la configurazione delle schede audio può essere trovata in *BLFS*.

Per ciascuno dei propri dispositivi che probabilmente hanno questo problema (sebbene il problema non esista nella propria attuale distribuzione Linux), cercare la corrispondente directory sotto `/sys/class` o `/sys/block`. Per dispositivi video, questa può essere `/sys/class/video4linux/video`*X*. Valutare gli attributi che identificano univocamente il dispositivo (di solito, funzionano gli ID dei venditori e dei produttori e/o i numeri seriali):

```
udevinfo -a -p /sys/class/video4linux/video0
```

Poi scrivere le regole che creano i symlink, es.:

```
cat >/etc/udev/rules.d/83-duplicate_devs.rules << EOF

# Persistent symlinks for webcam and tuner
KERNEL=="video*", SYSFS{idProduct}=="1910", SYSFS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", SYSFS{device}=="0x036f", SYSFS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"

EOF
```

Il risultato è che i dispositivi `/dev/video0` e `/dev/video1` ancora riferiscono casualmente al sintonizzatore e alla web camera (e pertanto non dovrebbero essere mai usati direttamente), ma ci sono i symlink `/dev/tvtuner` e `/dev/webcam` che puntano ancora al corretto dispositivo.

Maggiori informazioni sulla scrittura delle regole di Udev può essere trovata in `/usr/share/doc/udev-096/index.html`.

7.13. Configurazione dello script di rete

Questa sezione si applica solamente se è necessario configurare un scheda di rete.

Se non si dispone di schede di rete è consigliabile non creare alcun file di configurazione relativo alle schede di rete. In questo caso è necessario rimuovoere i link simbolici `network` da tutte le directory dei run-level (`/etc/rc.d/rc*.d`).

7.13.1. Creazione di nomi stabili delle interfacce di rete

Le istruzioni in questa sezione sono opzionali se si ha soltanto una scheda di rete.

Con Udev e i driver di rete modulari, la numerazione dell'interfaccia di rete predefinita non permane dopo il riavvio, perché i driver sono caricati in parallelo e, pertanto, in ordine casuale. Per esempio, su un computer che ha due schede di rete di marca Intel e Realtek, la scheda di rete fabbricata da Intel può divenire `eth0` e la scheda Realtek `eth1`. In qualche caso, dopo il riavvio le schede ottengono numerazioni diverse. Per evitare questo, occorre creare regole di Udev che assegnino nomi stabili alle schede di rete basati o sul loro indirizzo MAC o sulla posizione del bus.

Se si ha intenzione di usare l'indirizzo MAC per identificare le proprie schede di rete, cercare gli indirizzi con il seguente comando:

```
grep -H . /sys/class/net/*/address
```

Per ogni scheda di rete (tranne che per l'interfaccia loopback), inventare un nome descrittivo, simile a «realtek», e creare regole di Udev simili alla seguente:

```
cat > /etc/udev/rules.d/26-network.rules << EOF
ACTION=="add", SUBSYSTEM=="net", SYSFS{address}=="00:e0:4c:12:34:56", \
    NAME="realtek"
ACTION=="add", SUBSYSTEM=="net", SYSFS{address}=="00:a0:c9:78:9a:bc", \
    NAME="intel"
EOF
```

Nota

Sebbene gli esempi in questo libro funzionino propriamente, occorre sapere che udev non riconosce la backslash per la prosecuzione di riga. Se si modificano regole di udev con un editor, assicurarsi di lasciare ogni regola su una riga fisica.

Se si ha intenzione di utilizzare come chiave la posizione del bus, creare regole di Udev simili alla seguente:

```
cat > /etc/udev/rules.d/26-network.rules << EOF
ACTION=="add", SUBSYSTEM=="net", BUS=="pci", ID=="0000:00:0c.0", \
    NAME="realtek"
ACTION=="add", SUBSYSTEM=="net", BUS=="pci", ID=="0000:00:0d.0", \
    NAME="intel"
EOF
```

Queste regole rinomineranno sempre le schede di rete come «realtek» e «intel», indipendentemente dalla numerazione originale fornita dal kernel (per esempio: le originali interfacce «eth0» e «eth1» non esisteranno più, a meno che si inseriscano quei nomi come «descrittivi» nella chiave NAME). Nelle regole di Udev usare nomi descrittivi invece di «eth0» nei file di configurazione dell'interfaccia di rete di sopra.

Notare che le regole suddette non funzionano per ogni setup. Per esempio, regole basate su MAC falliscono quando sono usate con bridge o VLAN, perché bridge e VLAN hanno lo stesso indirizzo MAC della scheda di rete. Si rinomini solo l'interfaccia della scheda di rete, non l'interfaccia bridge o VLAN, ma la regola d'esempio le abbina entrambe. Se si usano tali interfacce virtuali, si hanno due soluzioni potenziali. Una è di aggiungere la chiave DRIVER=="?*" dopo SUBSYSTEM=="net" nelle regole basate sul MAC le quali stabiliscono l'abbinamento delle interfacce virtuali. Questo si sa che fallisce con alcune vecchie schede Ethernet, perché esse non hanno la variabile DRIVER nell'uevent e per questo motivo la regola non si abbina con tali schede. Un'altra soluzione è di passare a regole che usino come chiave la posizione del bus.

Il secondo caso di non-funzionamento è con schede wireless che usano i driver MadWifi o HostAP, perché esse creano almeno due interfacce con lo stesso indirizzo MAC e posizione del bus. Per esempio, il driver Madwifi crea sia un interfaccia athX che una wifiX dove X è il numero. Per differenziare queste interfacce, aggiungere una appropriato parametro del KERNEL come KERNEL=="ath*" dopo SUBSYSTEM=="net".

Ci possono essere altri casi dove le regole sopra non funzionano. Attualmente vengono ancora segnalati bug su questo problema alle distribuzioni Linux, e non esiste soluzione che copra ogni caso.

7.13.2. Creazione dei file di configurazione per le intefacce di rete

Quali interfacce vengano attivate e disattivate dagli script di rete dipende dai file e directory nella gerarchia `/etc/sysconfig/network-devices`. Questa directory deve contenere una sottodirectory per ciascuna interfaccia da configurare, nella forma `ifconfig.xyz`, dove «xyz» è il nome di un'interfaccia di rete. In questa directory ci saranno file che definiscono gli attributi di questa interfaccia, come indirizzo(i) IP, maschera della sottorete, e così via

Il seguente comando crea un file `ipv4` di esempio per il dispositivo *eth0*:

```
cd /etc/sysconfig/network-devices &&
mkdir -v ifconfig.eth0 &&
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Naturalmente i valori di queste variabili dovranno essere modificati in ogni file, per una corretta impostazione. Se la variabile `ONBOOT` è impostata a «yes», lo script di rete attiverà la scheda di rete (NIC) durante l'avvio del sistema. Se si imposta a qualcos'altro diverso da «yes», la scheda di rete (NIC) verrà ignorata dallo script e quindi non attivata.

La variabile `SERVICE` definisce il metodo di ottenimento dell'indirizzo IP. Il pacchetto LFS-Bootscripts ha un formato modulare di assegnamento degli IP, e la creazione di altri file nella directory permette altri metodi di assegnamento degli IP. Questo è comunemente utilizzato per il Dynamic Host Configuration Protocol (DHCP), che verrà utilizzato nel libro BLFS.

La variabile GATEWAY deve contenere l'IP del gateway predefinito, se esiste. Altrimenti commentare la variabile.

La variabile PREFIX deve contenere il numero di bit usati nella sottorete. Ogni ottetto in un indirizzo IP consiste di 8 bit. Se la netmask della sottorete è 255.255.255.0 si usano i primi tre ottetti (24 bit) per specificare il numero della rete. Se la netmask è 255.255.255.240 si useranno i primi 28 bit. Prefissi maggiori di 24 bit sono usati comunemente da Internet Service Provider (ISP) di reti DSL e via cavo. In questo esempio (PREFIX=24), la netmask è 255.255.255.0. Aggiustare la variabile PREFIX in base alla sottorete specifica.

7.13.3. Creazione del file /etc/resolv.conf

Se si vuole connettersi a Internet sarà necessario definire le modalità di risoluzione dei nomi DNS (Domain Name Service) per risolvere i nomi di dominio in indirizzi IP. Tale funzionalità può essere realizzata inserendo l'indirizzo IP del proprio DNS, reso disponibile dall'ISP (Internet Service Provider) o dall'amministratore di rete, nel file /etc/resolv.conf. Creare il file eseguendo il comando seguente:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain {[Il nome del vostro dominio]}
nameserver [indirizzo IP del nameserver primario]
nameserver [indirizzo IP del nameserver secondario]


# End /etc/resolv.conf
EOF
```

Naturalmente sostituire *[indirizzo IP del nameserver]* con l'indirizzo IP del DNS più appropriato per la propria configurazione. Spesso ci può essere più di un indirizzo IP nel file di configurazione (è richiesto un server secondario per evitare fallimenti). Se non è necessario o si desidera avere solo un server DNS, rimuovere dal file la seconda linea contenente *nameserver*. L'indirizzo IP può essere un router nella rete locale.

Capitolo 8. Rendere avviabile il sistema LFS

8.1. Introduzione

È il momento di rendere avviabile il sistema LFS. Questo capitolo discute la creazione di un file `fstab`, la costruzione di un kernel per il nuovo sistema LFS, e l'installazione del boot loader GRUB in modo che il sistema LFS possa essere selezionato per avviarsi all'accensione.

8.2. Creazione del file /etc/fstab

Il file `/etc/fstab` è utilizzato da alcuni programmi per determinare dove devono essere montati di default i file system, in quale ordine, e quali devono essere controllati (per verificare gli errori di integrità). Si crei un nuova tabella dei file system come la seguente:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type   options         dump  fsck
#                                                       order

/dev/<xxx>     /            <fff>  defaults        1     1
/dev/<yyy>     swap         swap   pri=1           0     0
proc           /proc        proc   defaults        0     0
sysfs          /sys         sysfs  defaults        0     0
devpts         /dev/pts     devpts gid=4,mode=620  0     0
shm            /dev/shm     tmpfs  defaults        0     0
# End /etc/fstab
EOF
```

Sostituire *<xxx>*, *<yyy>*, e *<fff>* con i valori appropriati per il sistema, per esempio, `hda2`, `hda5`, e `ext3`. Per tutti i dettagli sui sei campi della tabella vedere **man 5 fstab**.

il punto di mount `/dev/shm` per `tmpfs` è incluso per permettere l'abilitazione della memoria POSIX-shared. Il supporto necessario perché tale opzione funzioni deve essere compilato nel kernel (maggiori informazioni su questa modalità nella prossima sezione). Da notare che pochissimi software attualmente utilizzano la memoria POSIX-shared. Pertanto si consideri opzionale il punto di montaggio `/dev/shm`. Per maggiori informazioni consultare `Documentation/filesystems/tmpfs.txt` nell'albero sorgente del kernel.

Filesystem con origine MS-DOS o Windows (es.: vfat, ntfs, smbfs, cifs, iso9660, udf) hanno bisogno dell'opzione di mount «iocharset» perché i caratteri non-ASCII siano interpretati correttamente. Il valore di questa opzione deve essere lo stesso del set caratteri della propria localizzazione, aggiustato in modo che il kernel lo capisca. Questo funzione se la definizione corretta del set caratteri (che si trova sotto File systems -> Native Language Support) è stata compilata nel kernel o costruita come modulo. Per i filesystem vfat e smbfs è necessria anche l'opzione «codepage». Essa deve essere impostata sul numero di codepage usato sotto MS-DOS nel proprio paese. Ad es. per montare i drive flash USB un utente ru_RU.KOI8-R avrebbe bisogno della seguente linea in `/etc/fstab`:

```
/dev/sda1    /media/flash vfat
    noauto,user,quiet,showexec,iocharset=koi8r,codepage=866 0 0
```

La linea corrispondente per utenti ru_RU.UTF-8 è:

```
/dev/sda1    /media/flash vfat
    noauto,user,quiet,showexec,iocharset=utf8,codepage=866 0 0
```

Nota

Nell'ultimo caso il kernel emette il seguente messaggio:

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

Questa raccomandazione negativa deve essere ignorata, poiché tutti gli altri valori dell'opzione «iocharset» darebbero una visualizzazione errata dei nomi di file nelle localizzazioni UTF-8.

È anche possibile specificare la codepage predefinita e i valori iocharset per alcuni filesystem durante la configurazione del kernel. I parametri importanti si chiamano «Default NLS Option» (CONFIG_NLS_DEFAULT), «Default Remote NLS Option» (CONFIG_SMB_NLS_DEFAULT), «Default codepage for FAT» (CONFIG_FAT_DEFAULT_CODEPAGE), and «Default iocharset for FAT» (CONFIG_FAT_DEFAULT_IOCHARSET). Non c'è modo di specificare queste impostazioni per il filesystem ntfs al momento della compilazione del kernel.

8.3. Linux-2.6.16.27

Il pacchetto Linux contiene il kernel Linux.

Tempo di costruzione approssimativo: 1.5 - 3 SBU
Spazio necessario su disco: 310 - 350 MB

8.3.1. Installazione del kernel

La costruzione del kernel comporta alcuni passi: configurazione, compilazione e installazione. Leggere il file README nei sorgenti dell'albero del kernel per metodi alternativi al modo in cui questo libro configura il kernel.

Per default il kernel Linux genera sequenze di byte errate quando nel modo tastiera UTF-8 vengono usati i dead key. Inoltre non è possibile copiare e incollare caratteri non-ASCII quando è attivo il modo UTF-8. Correggere questi problemi con la patch:

```
patch -Np1 -i ../linux-2.6.16.27-utf8_input-1.patch
```

Preparare per la compilazione eseguendo il seguente comando:

```
make mrproper
```

Questo assicura che l'albero del kernel sia assolutamente pulito. Il team del kernel raccomanda che questo comando sia dato prima di ogni compilazione del kernel. Non pensare che l'albero dei sorgenti sia pulito dopo la scompattazione.

Configurare il kernel attraverso un'interfaccia a menu. BLFS ha alcune informazioni riguardanti particolari esigenze di configurazione del kernel di pacchetti al di fuori di LFS presso *http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*:

```
make menuconfig
```

In alternativa **make oldconfig** potrebbe essere più appropriato in alcune situazioni. Vedere il file README per maggiori informazioni.

Se lo si desidera saltare la configurazione del kernel copiando il file config del kernel, `.config`, dal sistema host (supponendo che sia disponibile) alla directory scompattata `linux-2.6.16.27`. Tuttavia non raccomandiamo questa opzione. È spesso meglio esplorare tutti i menu di configurazione e creare la configurazione del kernel da zero.

Compilare l'immagine del kernel e i moduli:

```
make
```

Se si usano i moduli kernel sarà necessario un file `/etc/modprobe.conf`. Informazioni riguardanti moduli e configurazione del kernel si trovano nel capitolo Sezione 7.4, «Gestione dei dispositivi e dei moduli in un sistema LFS» e nella documentazione del kernel nella directory `linux-2.6.16.27/Documentation`. Anche `modprobe.conf(5)` potrebbe essere interessante.

Installare i moduli, se la configurazione del kernel li usa:

```
make modules_install
```

Dopo il completamento della compilazione del kernel sono necessari ulteriori passi per completare l'installazione. Alcuni file devono essere copiati nella directory `/boot`.

Il percorso all'immagine del kernel potrebbe variare in funzione della piattaforma usata. Il seguente comando presuppone un'architettura x86:

```
cp -v arch/i386/boot/bzImage /boot/lfskernel-2.6.16.27
```

`System.map` è un file di simboli per il kernel. Esso mappa i punti di ingresso delle funzioni di ciascuna funzione nelle API del kernel, oltre agli indirizzi delle strutture dati del kernel per il kernel in funzione. Digitare il seguente comando per installare il file map:

```
cp -v System.map /boot/System.map-2.6.16.27
```

Il file di configurazione del kernel, `.config` prodotto nella fase **make menuconfig** precedente contiene tutte le selezioni di configurazione per il kernel che è stato appena compilato. È una buona idea tenere questo file per un riferimento futuro:

```
cp -v .config /boot/config-2.6.16.27
```

Installare la documentazione del kernel Linux:

```
install -d /usr/share/doc/linux-2.6.16.27 &&
cp -r Documentation/* /usr/share/doc/linux-2.6.16.27
```

È importante notare che i file nella directory dei sorgenti del kernel non hanno come proprietario *root*. Qualora un pacchetto sia scompattato come utente *root* (come quando lo facciamo in chroot), i file hanno ID di utente e gruppo di colui che li ha pacchettizzati sul proprio computer. Questo di solito non è un problema per installare qualunque altro pacchetto, perché l'albero dei sorgenti è rimosso dopo l'installazione. Tuttavia l'albero dei sorgenti di Linux è spesso mantenuto per lungo tempo. A causa di questo c'è la possibilità che l'ID usato dall'utente che ha pacchettizzato i file sia assegnato a qualcuno sulla macchina. Questa persone allora avrebbe accesso in scrittura ai sorgenti del kernel.

Se si terranno i sorgenti del kernel eseguire **chown -R 0:0** nella directory `linux-2.6.16.27` per assicurarsi che tutti i file appartengano all'utente *root*.

Avvertimento

Certa documentazione del kernel raccomanda di creare un link simbolico che da `/usr/src/linux` punti alla directory dei sorgenti del kernel. Questo è specifico per kernel precedenti alla serie 2.6, e *non deve* essere creato su un sistema LFS, poiché può causare problemi per i pacchetti che si potrebbe voler costruire dopo aver completato il sistema LFS base.

Inoltre gli header nella directory di sistema `include` devono *sempre* essere quelli sui quali è stata compilata Glibc, ovvero quelli del pacchetto Linux-Libc-Headers, e quindi non devono *mai* essere rimpiazzati dagli header del kernel.

8.3.2. Contenuti di Linux

File installati: config-2.6.16.27, lfskernel-2.6.16.27, e System.map-2.6.16.27

Brevi descrizioni

`config-2.6.16.27`	Contiene tutte le selezioni di configurazione del kernel
`lfskernel-2.6.16.27`	Il motore del sistema Linux. Quando si avvia il computer il kernel è la prima parte del sistema operativo che viene caricata. Esso rileva e inizializza tutti i componenti hardware del computer, quindi rende disponibili questi componenti come albero di file al software e trasforma una singola CPU in una macchina multitasking capace di eseguire pezzi di programmi apparentemente nello stesso tempo
`System.map-2.6.16.27`	Un elenco di indirizzi e simboli; esso mappa i punti di ingresso e gli indirizzi di tutte le funzioni e le strutture dati nel kernel

8.4. Rendere avviabile il sistema LFS

Il nuovo sistema LFS è quasi completo. Una delle ultime cose da fare è assicurarsi che si possa avviare. Le istruzioni sottostanti si applicano esclusivamente ai computer con architettura IA-32 e quindi la maggioranza dei PC. Informazioni sul «boot loading» per altre architetture dovrebbero essere disponibile nelle rispettive locazioni specifiche per queste architetture.

La fase di avvio può essere un argomento molto complesso. Innanzitutto alcuni accorgimenti. È necessario avere familiarità con il boot loader esistente e gli altri sistemi operativi installati nel disco rigido che è necessario mantenere avviabili. È necessario assicurarsi di avere un disco di avvio d'emergenza a disposizione, in maniera da poter avviare il sistema nel caso in cui il computer divenga inutilizzabile.

All'inizio dovremo compilare e installare il boot loader GRUB. La procedura richiede la scrittura di alcuni file speciali di GRUB in specifiche locazioni del disco. Raccomandiamo caldamente di creare un floppy di avvio di GRUB come backup. Inserire un disco vuoto nel drive a lanciare il comando seguente:

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Rimuovere il dischetto e custodirlo in qualche luogo sicuro. Ora si avvii la shell di **grub**:

```
grub
```

GRUB usa una propria struttura di nomi per i drive e le partizioni, nella forma *(hdn,m)*, dove *n* è il numero di disco rigido, e *m* il numero di partizione, entrambi partendo da zero. Per esempio la partizione `hda1` è *(hd0,0)* per GRUB, e `hdb3` è *(hd1,2)*. A differenza di Linux, GRUB non considera i CD-ROM come dischi rigidi. Per esempio se si usa un CD in `hdb`, e un secondo disco rigido in `hdc`, allora il secondo disco rigido resterà *(hd1)*.

Utilizzando tali informazioni, determinare il corretto indicatore per la partizione di root (o per la partizione di boot, se ne viene usata una separata). Per l'esempio successivo, si suppone che la partizione di root (o di boot separata) sia `hda4`.

Indicare a GRUB dove cercare i suoi file `stage{1,2}`. È possibile utilizzare il tasto Tab perché GRUB mostri le possibili alternative:

```
root (hd0,3)
```

Avvertimento

Il comando seguente sovrascriverà il boot loader corrente. Non lanciare il comando se questo non è ciò che si vuole. Per esempio, è possibile si stia utilizzando un boot manager di terze parti per gestire il Master Boot Record (MBR). In questo scenario, sarà probabilmente più sensato installare GRUB nel «settore di avvio» della partizione di LFS: In questo caso il prossimo comando diverrebe **`setup (hd0,3)`**.

Indicare a GRUB di installarsi nel MBR (Master Boot Record) di `hda`:

```
setup (hd0)
```

Se tutto funziona, GRUB riporterà di aver trovato i suoi file in `/boot/grub`. Questo è tutto. Uscire dalla shell di GRUB:

```
quit
```

Creare un file «menu list» che definisca il menu di avvio di GRUB:

```
cat > /boot/grub/menu.lst << "EOF"
# Begin /boot/grub/menu.lst

# By default boot the first menu entry.
default 0

# Allow 30 seconds before booting the default.
timeout 30

# Use prettier colors.
color green/black light-green/black

# The first entry is for LFS.
title LFS 6.2
root (hd0,3)
kernel /boot/lfskernel-2.6.16.27 root=/dev/hda4
EOF
```

Potrebbe essere utile aggiungere una voce per il sistema host. Tale voce potrebbe apparire così:

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

Inoltre, se è necessario eseguire un dual-boot con Windows, dovrà essere aggiunta la seguente voce:

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

Se il comando **info grub** non riporta tutte le informazioni che è necessario conoscere ulteriori informazioni su GRUB si trovano sul sito web su: *http://www.gnu.org/software/grub/*.

L'FHS ha stabilito che il file di GRUB menu.lst deve avere un link simbolico in `/etc/grub/menu.lst`. Per soddisfare questa necessità si dia il seguente comando:

```
mkdir -v /etc/grub &&
ln -sv /boot/grub/menu.lst /etc/grub
```

Capitolo 9. Fine

9.1. Fine

Ben fatto! Il nuovo sistema LFS è installato! Vi auguriamo un grande successo con il vostro nuovissimo sistema Linux personalizzato.

Può essere una buona idea creare un file `/etc/lfs-release`. Avendo questo file è molto facile per voi (e per noi se si sta per chiedere aiuto per qualcosa ad un certo punto) trovare quale versione LFS è stata installata sul sistema. Creare questo file eseguendo:

```
echo 6.2 > /etc/lfs-release
```

9.2. Farsi contare

Si vuole essere contati come utenti LFS ora che è stato finito il libro? Si faccia riferimento a *http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi* e ci si registri come utente LFS inserendo il proprio nome e la prima versione LFS che si è utilizzato.

Ora riavviare in LFS.

9.3. Riavvio del sistema

Ora che tutto il software è stato installato, è il momento di riavviare il proprio computer. Tuttavia bisogna porre attenzione ad alcune cose. Il sistema che è stato creato in questo libro è decisamente minimale, e probabilmento non avrà le funzionalità che servono per poter proseguire. L'installazione di alcuni pacchetti extra dal libro BLFS mentre si è ancora nell'ambiente chroot corrente può metterci in una posizione migliore per continuare quando si riavvia nella propria nuova installazione LFS. Installando un browser web testuale, come Lynx, si può facilmente visualizzare il libro BLFS in un terminale virtuale, mentre si costruiscono i pacchetti in un altro. Il pacchetto GPM permetterà anche di praticare azioni di copia/incolla nei propri terminali virtuali. Infine, se si è in una situazione in cui la configurazione di un IP statico non viene incontro alle proprie necessità di rete, anche l'installazione di pacchetti come Dhcpcd o PPP a questo punto può essere utile.

Ora che abbiamo detto questo, si vada al primo riavvio della propria nuova installazione LFS! Per prima cosa si esca dall'ambiente chroot:

```
logout
```

Quindi smontare i file system virtuali:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Smontare lo stesso file system LFS:

```
umount -v $LFS
```

Se sono state create partizioni multiple smontare le altre partizioni prima di smontare quella principale, in questo modo:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Ora riavviare il proprio sistema con:

```
shutdown -r now
```

Supponendo che il boot loader GRUB sia stato configurato come proposto in precedenza, il menu è impostato per avviare automaticamente *LFS 6.2*.

Quando il riavvio è completo, il sistema LFS è pronto per l'uso e si può iniziare ad aggiungere il proprio software.

9.4. E ora?

Grazie per aver letto il libro LFS. Speriamo che si sia trovato il libro di aiuto e si abbia imparato di più sul processo di creazione del sistema.

Ora che il sistema LFS è installato, ci si potrebbe chiedere «E poi?» Per rispondere a questa domanda abbiamo compilato un elenco di risorse.

- Manutenzione

  Bug e note di sicurezza sono riportate regolarmente per tutti i software. Poiché un sistema LFS è compilato dai sorgenti è vostro compito mantenersi aggiornato con questi rapporti. Ci sono numerose risorse online che tracciano questi rapporti, alcune delle quali sono mostrate di seguito:

  - Freshmeat.net (*http://freshmeat.net/*)

    Freshmeat può notificare (via email) le nuove versioni dei pacchetti installati sul proprio sistema.

  - *CERT* (Computer Emergency Response Team)

    CERT ha una mailing list che pubblica avvisi di sicurezza riguardanti vari sistemi operativi e applicazioni. Informazioni per la sottoscrizione sono disponibili presso *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq è una mailing list full-disclosure sulla sicurezza dei computer. Esso pubblica i nuovi problemi di sicurezza scoperti, e occasionalmente le potenziali correzioni per esso. Informazioni per la sottoscrizione sono disponibili presso *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  Il libro Beyond Linux From Scratch copre le procedure di installazione di un'ampia varietà di software oltre lo scopo del libro LFS. Il progetto BLFS può essere trovato presso *http://www.linuxfromscratch.org/blfs/*.

- LFS Hints

  Gli Hint LFS sono una collezione di piccoli documenti educativi presentati da volontari nella comunità LFS. Gli Hint sono disponibili su *http://www.linuxfromscratch.org/hints/list.html*.

- Mailing list

  Ci sono numerose mailing list di LFS che si possono sottoscrivere se si ha bisogno di aiuto, si vuole essere al corrente degli ultimi sviluppi, si vuole contribuire al progetto, e altro. Vedere Chapter 1 - Mailing Lists per maggiori informazioni.

- The Linux Documentation Project

  Lo scopo del The Linux Documentation Project (TLDP) è di collaborare a tutti gli aspetti della documentazione Linux. Il TLDP ospita una grande collezione di HOWTO, guide, e man pages. Si trova presso *http://www.tldp.org/*.

Parte IV. Appendici

Appendice A. Acronimi e termini

ABI Application Binary Interface

ALFS Automated Linux From Scratch

ALSA Advanced Linux Sound Architecture

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BIOS Basic Input/Output System

BLFS Beyond Linux From Scratch

BSD Berkeley Software Distribution

chroot change root

CMOS Complementary Metal Oxide Semiconductor

COS Class Of Service

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CVS Concurrent Versions System

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

EGA Enhanced Graphics Adapter

ELF Executable and Linkable Format

EOF End of File

EQN equation

EVMS Enterprise Volume Management System

ext2 second extended file system

ext3 third extended file system

FAQ Frequently Asked Questions

FHS Filesystem Hierarchy Standard

FIFO First-In, First Out

FQDN Fully Qualified Domain Name

FTP File Transfer Protocol

GB Gibabytes

GCC GNU Compiler Collection

GID Group Identifier

GMT Greenwich Mean Time
GPG GNU Privacy Guard
HTML Hypertext Markup Language
IDE Integrated Drive Electronics
IEEE Institute of Electrical and Electronic Engineers
IO Input/Output
IP Internet Protocol
IPC Inter-Process Communication
IRC Internet Relay Chat
ISO International Organization for Standardization
ISP Internet Service Provider
KB Kilobytes
LED Light Emitting Diode
LFS Linux From Scratch
LSB Linux Standard Base
MB Megabytes
MBR Master Boot Record
MD5 Message Digest 5
NIC Network Interface Card
NLS Native Language Support
NNTP Network News Transport Protocol
NPTL Native POSIX Threading Library
OSS Open Sound System
PCH Pre-Compiled Headers
PCRE Perl Compatible Regular Expression
PID Process Identifier
PLFS Pure Linux From Scratch
PTY pseudo terminal
QA Quality Assurance
QOS Quality Of Service
RAM Random Access Memory
RPC Remote Procedure Call
RTC Real Time Clock
SBU Standard Build Unit

SCO The Santa Cruz Operation

SGR Select Graphic Rendition

SHA1 Secure-Hash Algorithm 1

SMP Symmetric Multi-Processor

TLDP The Linux Documentation Project

TFTP Trivial File Transfer Protocol

TLS Thread-Local Storage

UID User Identifier

umask user file-creation mask

USB Universal Serial Bus

UTC Coordinated Universal Time

UUID Universally Unique Identifier

VC Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Appendice B. Riconoscimenti

Vogliamo ringraziare le seguenti persone e organizzazioni per i loro contributi al progetto Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Creatore di LFS, leader del progetto LFS
- *Matthew Burgess* <matthew@linuxfromscratch.org> – Leader del progetto LFS, redattore tecnico di LFS, responsabile delle release di LFS
- *Archaic* <archaic@linuxfromscratch.org> – redattore/editore tecnico di LFS, leader del progetto HLFS, editore di BLFS, manutentore del progetto Hints e Patches
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Responsabile degli script di avvio di LFS
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Leader del progetto BLFS
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Responsabile di XML/XSL per LFS/BLFS/HLFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Redattore tecnico di LFS, leader del progetto Patches
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Redattore tecnico di LFS, responsabile del LiveCD di LFS, leader del progetto ALFS
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Responsabile degli script di backend del sito
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Responsabile della Toolchain di LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – Responsabile di Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Redattore del libro BLFS, leader dei progetti Hints e Patches
- Innumerevoli altre persone sulle varie mailing list LFS e BLFS che hanno aiutato a rendere possibile questo libro dando i loro suggerimenti, controllando il libro e inviando segnalazioni di bug, istruzioni e la loro esperienza nell'installazione di diversi pacchetti.

Traduttori

- *Manuel Canales Esparcia* <macana@lfs-es.org> – Progetto spagnolo di traduzione di LFS
- *Johan Lenglet* <johan@linuxfromscratch.org> – Progetto francese di traduzione di LFS
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Progetto portoghese di traduzione di LFS
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Progetto tedesco di traduzione di LFS

Responsabili dei mirror

Mirror nordamericani

- *Scott Kveton* <scott@osuosl.org> – Mirror lfs.oregonstate.edu
- *Mikhail Pastukhov* <miha@xuy.biz> – Mirror lfs.130th.net
- *William Astle* <lost@l-w.net> – Mirror ca.linuxfromscratch.org

- *Jeremy Polen* <jpolen@rackspace.com> – Mirror us2.linuxfromscratch.org
- *Tim Jackson* <tim@idge.net> – Mirror linuxfromscratch.idge.net
- *Jeremy Utley* <jeremy@linux-phreak.net> – Mirror lfs.linux-phreak.net

Mirror sudamericani

- *Andres Meggiotto* <sysop@mesi.com.ar> – mirror lfs.mesi.com.ar
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mirror lfsmirror.lfs-es.org
- *Eduardo B. Fonseca* <ebf@aedsolucoes.com.br> – Mirror br.linuxfromscratch.org

Mirror europei

- *Barna Koczka* <barna@siker.hu> – Mirror hu.linuxfromscratch.org
- *UK Mirror Service* – Mirror linuxfromscratch.mirror.ac.uk
- *Martin Voss* <Martin.Voss@ada.de> – Mirror lfs.linux-matrix.net
- *Guido Passet* <guido@primerelay.net> – Mirror nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – Mirror lfs.pagefault.net
- *Roel Neefs* <lfs-mirror@linuxfromscratch.rave.org> – Mirror linuxfromscratch.rave.org
- *Justin Knierim* <justin@jrknierim.de> – Mirror www.lfs-matrix.de
- *Stephan Brendel* <stevie@stevie20.de> – Mirror lfs.netservice-neuss.de
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – Mirror at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – Mirror se.linuxfromscratch.org
- *Amministratori parigini* <archive@doc.cs.univ-paris8.fr> – Mirror www2.fr.linuxfromscratch.org
- *Alexander Velin* <velin@zadnik.org> – Mirror bg.linuxfromscratch.org
- *Dirk Webster* <dirk@securewebservices.co.uk> – Mirror lfs.securewebservices.co.uk
- *Thomas Skyt* <thomas@sofagang.dk> – Mirror dk.linuxfromscratch.org
- *Simon Nicoll* <sime@dot-sime.com> – Mirror uk.linuxfromscratch.org

Mirror asiatici

- *Pui Yong* <pyng@spam.averse.net> – Mirror sg.linuxfromscratch.org
- *Stuart Harris* <stuart@althalus.me.uk> – Mirror lfs.mirror.intermedia.com.sg

Mirror australiani

- *Jason Andrade* <jason@dstc.edu.au> – Mirror au.linuxfromscratch.org

Vecchi membri del team del progetto

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Book Editor
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Sviluppatore del sito, responsabile delle FAQ
- Alex Groenewoud – redattore tecnico di LFS
- Marc Heerdink
- Mark Hymers
- Seth W. Klein – FAQ maintainer

- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Responsabile del Wiki
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Responsabile del gateway NNTP di LFS
- *Alexander Patrakov* <semzx@newmail.ru> – Redattore tecnico di LFS
- *Greg Schafer* <gschafer@zip.com.au> – redattore tecnico di LFS
- Jesse Tie-Ten-Quee – redattore tecnico di LFS
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Redattore tecnico di LFS, responsabile di Bugzilla, Responsabile degli script di avvio di LFS
- *Zack Winkles* <zwinkles@gmail.com> – LFS Technical Writer

Un grazie davvero speciale ai nostri donatori

- *Dean Benson* <dean@vipersoft.co.uk> per diversi contributi monetari
- *Hagen Herrschaft* <hrx@hrxnet.de> per la donazione di un sistema P4 2.2 GHz, che ora porta il nome di Lorien
- *SEO Company Canada* supporta progetti Open Source e diverse distribuzioni Linux
- *VA Software* che, a nome di *Linux.com*, ha donato una workstation VA Linux 420 (la vecchia StartX SP2)
- Mark Stone per la donazione di Belgarath, il server di linuxfromscratch.org

Appendice C. Dipendenze

Ogni pacchetto costruito in LFS dipende da uno o più altri pacchetti per la sua corretta costruzione e installazione. Alcuni pacchetti partecipano anche a dipendenze circolari, cioè il primo pacchetto dipende dal secondo, che a sua volta dipende dal primo. A causa di queste dipendenze l'ordine nel quale i pacchetti vengono costruiti è molto importante. Lo scopo di questa pagina è di documentare le dipendenze di ciascun pacchetto costruito in LFS.

Per ciascun pacchetto che costruiamo abbiamo elencato tre tipi di dipendenze. La prima elenca quali altri pacchetti devono essere disponibili per poter compilare e installare il pacchetto in questione. La seconda elenca quali pacchetti, in aggiunta a quelli del primo elenco, devono essere disponibili per poter eseguire la suite di test. L'ultimo elenco di dipendenze sono pacchetti che richiedono che questo pacchetto sia costruito e installato nella sua locazione finale per essere costruiti e installati. Nella maggior parte dei casi ciò avviene perché questi pacchetti scrivono i percorsi ai binari nei propri script. Se non costruiti in un certo ordine la conseguenza potrebbe essere che /tools/bin/[binario] sia messo negli script installati nel sistema finale. Questo, ovviamente non è desiderabile.

Autoconf

L'installazione dipende da: Bash, Coreutils, Grep, M4, Make, Perl, Sed e Texinfo

Le suite di test dipendono da: Automake, Diffutils, Findutils, GCC e Libtool

Deve essere installato prima di: Automake

Automake

L'installazione dipende da: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, e Texinfo

Le suite di test dipendono da: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, e Tar. Può usare anche numerosi altri pacchetti che non son installati in LFS.

Deve essere installato prima di: Nessuno

Bash

L'installazione dipende da: Bash, Bison, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, e Texinfo

Le suite di test dipendono da: Diffutils e Gawk

Deve essere installato prima di: Nessuno

Berkeley DB

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Non eseguita. Richiede che TCL sia installata sul sistema finale

Deve essere installato prima di: Nessuno

Binutils

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl, Sed, e Texinfo

Le suite di test dipendono da: DejaGNU ed Expect

Deve essere installato prima di: Nessuno

Bison

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, e Sed

Le suite di test dipendono da: Diffutils e Findutils

Deve essere installato prima di: Flex, Kbd, e Tar

Bzip2

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, e Patch

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Coreutils

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, e Texinfo

Le suite di test dipendono da: Diffutils

Deve essere installato prima di: Bash, Diffutils, Findutils, Man-DB, e Udev

DejaGNU

L'installazione dipende da: Bash, Coreutils, Diffutils, GCC, Grep, Make, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Diffutils

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Expect

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, e Tcl

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

E2fsprogs

L'installazione dipende da: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, Make, Sed, e Texinfo

Le suite di test dipendono da: Diffutils

Deve essere installato prima di: Util-Linux

File

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Zlib

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Findutils

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Le suite di test dipendono da: DejaGNU, Diffutils, ed Expect

Deve essere installato prima di: Nessuno

Flex

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Bison e Gawk

Deve essere installato prima di: IPRoute2, Kbd, e Man-DB

Gawk

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed e Texinfo

Le suite di test dipendono da: Diffutils

Deve essere installato prima di: Nessuno

Gcc

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, Tar, e Texinfo

Le suite di test dipendono da: DejaGNU ed Expect

Deve essere installato prima di: None

Gettext

L'installazione dipende da: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Texinfo

Le suite di test dipendono da: Diffutils, Perl, e Tcl

Deve essere installato prima di: Automake

Glibc

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed, e Texinfo

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Grep

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Diffutils e Gawk

Deve essere installato prima di: Man-DB

Groff

L'installazione dipende da: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Man-DB e Perl

GRUB

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed, e Texinfo

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Gzip

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Man-DB

Iana-Etc

L'installazione dipende da: Coreutils, Gawk, e Make

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Perl

Inetutils

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, e Texinfo

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Tar

IProute2

L'installazione dipende da: Bash, Berkeley DB, Bison, Coreutils, Flex, GCC, Glibc, Make, e Linux-Libc-Headers

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Kbd

L'installazione dipende da: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Less

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Libtool

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Texinfo

Le suite di test dipendono da: Findutils

Deve essere installato prima di: Nessuno

Linux Kernel

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Module-Init-Tools, Ncurses, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

M4

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Diffutils

Deve essere installato prima di: Autoconf e Bison

Man-DB

L'installazione dipende da: Bash, Berkeley DB, Binutils, Bzip2, Coreutils, Flex, GCC, Gettext, Glibc, Grep, Groff, Gzip, Less, Make, e Sed

Le suite di test dipendono da: Non eseguita. Richiede la suite di test di Man-DB

Deve essere installato prima di: Nessuno

Make

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Le suite di test dipendono da: Perl

Deve essere installato prima di: Nessuno

Mktemp

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Patch, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Module-Init-Tools

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, e Zlib

Le suite di test dipendono da: File, Findutils, e Gawk

Deve essere installato prima di: Nessuno

Ncurses

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-Linux, e Vim

Patch

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Perl

L'installazione dipende da: Bash, Berkeley DB, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Groff, Make, e Sed

Le suite di test dipendono da: Iana-Etc e Procps

Deve essere installato prima di: Autoconf

Procps

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Make, e Ncurses

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Psmisc

L'installazione dipende da: Bash, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Readline

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, e Texinfo

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Bash

Sed

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Le suite di test dipendono da: Diffutils e Gawk

Deve essere installato prima di: E2fsprogs, File, Libtool, e Shadow

Shadow

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Sysklogd

L'installazione dipende da: Binutils, Coreutils, GCC, Glibc, Make, e Patch

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Sysvinit

L'installazione dipende da: Binutils, Coreutils, GCC, Glibc, Make, e Sed

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Tar

L'installazione dipende da: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Patch, Sed, e Texinfo

Le suite di test dipendono da: Diffutils, Findutils, e Gawk

Deve essere installato prima di: Nessuno

Tcl

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Texinfo

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, e Sed

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Udev

L'installazione dipende da: Binutils, Coreutils, GCC, Glibc, e Make

Le suite di test dipendono da: Findutils, Perl, e Sed

Deve essere installato prima di: Nessuno

Util-Linux

L'installazione dipende da: Bash, Binutils, Coreutils, E2fprogs, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, Sed, e Zlib

Le suite di test dipendono da: Nessuna suite di test disponibile

Deve essere installato prima di: Nessuno

Vim

L'installazione dipende da: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, e Sed

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: Nessuno

Zlib

L'installazione dipende da: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, e Sed

Le suite di test dipendono da: Nessuno

Deve essere installato prima di: File, Module-Init-Tools, e Util-Linux

Indice

Pacchetti

Programmi

Librerie

Script

Altri