Linux From Scratch

Gerard Beekmans Version 2.2, April 3rd, 2000

정하녕 버전 2.2, 2000년 5월 5일

이 문서는 필요한 소프트웨어의 소스코드만을 사용하여 이미 설치되어 있는 리눅스 배포본으로부터 여러분만의 리눅스 시스템을 완전히 새로 만드는 과정에 대해 설명하고 있다.

목 차

제 1 절 소개

1.1 이것들은 다 뭣하는 것인가?

많은 리눅스 배포본들을 써왔지만 나는 그 중 어떤 것에도 완벽히 만족하지 못했다. 부트 스크립트들이 정돈되 있는 방식이나 특정 프로그램들이 기본으로 설정되어 있는 방식 따위가 마음에 들지 않았다. 내가 리눅스 시스템에 완벽히 흡족해 하고자 했을 때 나는 처음부터 나만의 리눅스 시스템을 만들어야 한다는 것을 깨닫게 되었다. 이상적으로는 소스 코드만을 사용하여서. 어떤 종류의 이미 컴파일된 패키지도 없이. 몇몇 기본적인 유틸리티를 설치하는 cdrom이나 부트디스크의 도움도 없이. 여러분은 현재 리눅스 시스템을 사용하고 있을 것이며 여러분만의 리눅스 시스템을 만들기 위해 그것을 사용할 것이다.

예전에는 이 무모한 생각이 매우 어렵고 때론 거의 불가능해 보였다. 대부분의 문제에 대한 이유는 특정 프로그램과 작동과정에 대한 내 지식의 부족이었다. 모든 종류의 의존성 문제와 컴파일 문제 등을 해결한 후에 리눅스 시스템이 손수 만들어졌으며 완벽히 작동하였다. 나는 이 시스템을 LinuxFromScratch를 나타내는 LFS 시스템이라 칭했다.

1.2 새 버전

`http://www.linuxfromscratch.org` 에서 이 문서의 최신 버전을 언제라도 구할 수 있다.

1.3 버전 역사

2.1.5 - March 26th, 2000 2.1.5 - 2000년 3월 26일

아래는 수정 사항의 전체 목록은 아니다. v2.0은 변경사항이 많이 적용된 판이므로 중요한 변화만 언급되었고 그렇지 않은 것은 생략되었다.

- 디렉토리 구조가 수정되었다 - 이제 LFS는 FHS를 준수한다. 비록 100% 는 아닐지라도 근접하고 있다.
- 새로운 Glibc 설치 방법
- 새로운 GCC 설치 방법
- 미리 컴파일된 데비안 패키지가 필요없게 되었다.
- 완전히 개정된 소프트웨어 설치 방법 - 6.1장 이전에서 정적으로 링크된 패키지들이 더이상 필요없게 되었다.
- 소프트웨어 설치시의 많은 버그가 수정되었다.
- util-linux 패키지에서 좀 더 많은 프로그램들을 설치했다.
- Bzip2 프로그램의 설치가 추가되었다.
- $LFS가 무엇인지에 대해 더욱 자세히 설명하였다 - 이를 쓰는 방법과 쓰지 않는 방법.
- 5장에서 9.1장까지의 모든 패키지들의 설치 과정을 간소화하였다.
- 불필요한 별도의 장을 갖는 대신 Glibc와 GCC의 설치를 7장으로 이동하였다.

- 인터넷 서버 관련 장을 수정하였다: Network Daemon 관련 부분과 Network Clients 관련 부분. 인터넷 관련 장은 이 새 두 부분들과 합쳐졌다.
- 13장과 14장(X와 인터넷)을 바꾸었고 X와 Window Maker에 관한 장을 14장으로 통합하였다.
- 우리는 새로운 Man 프로그램을 사용한다. 이는 우리가 이전에 쓰던 man-db 프로그램보다 설정과 사용이 더 쉽다. 두 버전은 거의 같은 일을 한다.
- 13장을 추가하였다: 자원들. 이 장은 여러분이 LFS 시스템을 설치하는 동안이나 설치 후에 읽을 많은 유용한 책들과 HOWTO들을 담고 있다.
- 3장: bzip2 링크 오류 수정
- 7.2.42장: Util-Linux 설치 방법의 간소화
- 3.1장: procps의 위치 변경
- 7.2장: Vim과 Util-Linux의 설치 순서를 바꿈 (Util-Linux를 설치하는 동안 우리는 에디터가 필요하기 때문이다)
- 7.3.33장: procps 설치 오류 수정
- 5.2장: 부트시에 파일이 없다는 경고가 뜨지 않도록 inittab 파일을 제거했다.
- 6장: 커널 설치부분 재작성
- 10.3장: 아파치 부트스크립트의 오류 수정
- 10.3.2장: httpd.conf 파일을 고치는 것에 대한 부분을 제거하였다. 더이상 필요하지 않다. /usr/apache/man을 /usr/share/misc/man.conf에 추가한다는 것을 넣었다.
- 11.1장: 설치 과정을 간소화하기 위해서 잘 동작하는 Makefile 파일이 들어있도록 수정된 mailx 패키지를 제공하였다.
- 11.3.1장: Zlib가 정적이 아니라 동적 라이브러리로 설치되도록 configure에 –share 스위치를 추가하였다.
- 11.6장: Lynx가 Slang 대신 Ncurses 라이브러리에 링크되도록 하였다.
- 12장: 새로운 man-db는 이미 man_db.config 파일에 X11/man 디렉토리를 가지고 있다.

2.2 - April 3rd, 2000 2.2 - 2000년 4월 3일

- 이제 linuxfromscratch.org와 linuxfromscratch.com 도메인이 작동한다. huizen.dds.nl/~glb와 tts.ookhoi.dds.nl에 대한 이전의 모든 링크들이 www.linuxfromscratch.org의 알맞은 링크로 대체되었다.
- 7.3장에서 리부팅 이후에 우리가 소프트웨어의 컴파일을 시작하기 전에 스왑 파티션이 액티브 모드로 된다.

1.4 메일링리스트

여러분이 참가할 수 있는 메일링리스트가 두 개 있다. lfs-discuss와 lfs-announce 리스트. 전자는 제한이 없이 열려진 것이며 이 문서와 관련된 어떤 것들이라도 토론할 수 있다. 후자는 제한이 있는 메일링리스트이다. 누구나 참가할 순 있지만 이 리스트에 글을 쓸 수는 없다 (오직 중재자만이 쓸 수 있다). 이 리스트는 주로 이 문서의 새 버전에 대해 알릴 때 사용된다.

만약 여러분이 lfs-discuss 리스트에 가입한다면 lfs-announce 리스트에도 가입할 필은 없다. lfs-announce 리스트로 가는 모든 글은 lfs-discuss 리스트로도 보내진다.

1.4.1 가입하기

리스트에 가입하기 위해서는 본문에 *subscribe lfs-discuss* 나 *subscribe lfs-announce* 라고 쓴 email을 `majordomo@fist.org`로 보내면 된다.

Majordomo가 여러분에게 확인 요청 email을 보낼 것이다. 이 email은 인증 코드를 담고 있다. 여러분이 이 email을 Majordomo에게 돌려 보내면 (방법은 email 안에 나와있다) 가입이 완료될 것이다.

1.4.2 탈퇴하기

리스트에서 탈퇴하기 위해서는 본문에 *unsubscribe lfs-discuss* 이나 *unsubscribe lfs-announce* 이라 쓴 email을 `majordomo@fist.org` 로 보내면 된다.

1.5 연락처

여러분의 모든 질문은 가능하면 메일링 리스트로 올려주기 바란다. 만약 내게 개인적으로 연락해야 한다면 `gerard@linuxfromscratch.org`로 메일을 보내면 된다.

기타 번역의 오타나 오역에 대해선 `alita@kldp.org`로 메일을 주기 바란다.

1.6 번역자의 변

LFS를 하나 하나 실제로 실행해 가면서 번역을 하고자 했던 계획이 밤 늦게 퇴근하는 직장인이라는 핑계로 수포로 돌아가면서 단순 번역으로 되었습니다.:(그다지 특별하게 까다로운 번역은 없으나 이 v2.2 문서 자체에 버그가 상당량 내포되어 있기 때문에 - 덕분에 lfs-discuss mailing list 에는 무수한 메일들이 쏟아집니다 - 아직은 따라만 해서 완벽할 수 있다는 보장은 못할 것 같습니다. 각종 버그들에 대해선 가능하면 lfs-discuss 를 참조하시길 바라며, 현재 v2.3이 진행중입니다. v2.2 부터는 버전 번호가 kernel 버전 번호 부여 방법과 같이 가고 있으니 v2.4가 안정 버전이 될 것 같습니다.:)

제 2 절 이 문서에서 쓰이는 관례

2.1 $LFS에 대해서

다음을 주의깊게 읽기 바란다: 이 문서 전체에 걸쳐서 여러분은 $LFS 라는 변수를 자주 보게 될 것이다. $LFS는 항상 LFS 시스템을 포함하는 파티션이 마운트되는 디렉토리로 대체되어야 한다. 파티션을 만드는 방법과 마운트할 곳에 대해선 후의 4장에서 자세히 설명될 것이다. 내 경우 LFS 파티션은 /mnt/hda5에 마운트된다. 만약 내가 이 문서를 읽다가 어딘가에서 $LFS를 본다면 /mnt/hda5라고 읽은 것으로 간주할 것이다. 만약 `cp inittab $LFS/etc` 라는 명령을 실행해야 한다는 것을 읽는다면 나는 실제로는 `cp inittab /mnt/hda5/etc` 라고 실행할 것이다.

중요한 것은 여러분이 프롬프트 상에서 실행하는 명령이던 편집하거나 새로 만드는 파일 안이던 여러분이 읽는 모든 곳에서 이 일을 해야 한다는 것이다.

여러분이 원한다면 LFS 라는 환경 변수를 선언할 수 있다. 이런 방식으로 /mnt/hda5 같은 것으로 대체하는 대신 $LFS를 직접 쓸 수 있다. 다음과 같이 하면 된다: export LFS=/mnt/hda5

만약 cp inittab $LFS/etc 라는 문장을 읽는다면 cp inittab $LFS/etc 라고 직접 칠 수 있으며 쉘이 이 명령을 cp inittab /mnt/hda5/etc 로 자동으로 변환해 줄 것이다.

항상 LFS 변수가 설정되어 있도록 하는 것을 잊으면 안된다. 만약 이 변수를 설정하지 않고 명령문 안에서 이를 사용한다면 $LFS는 무시되고 나머지만 실행될 것이다. LFS 변수가 설정되지 않은 상태의 cp inittab $LFS/etc 명령은 inittab 파일을 /etc 디렉토리로 복사하여 여러분의 기존의 시스템의 inittab 을 덮어씌울

것이다. inittab같은 파일은 쉽게 복구될 수 있기 때문에 큰 문제는 아니지만 C 라이브러리의 설치 도중에 실수를 한다면 여러분의 시스템을 심각하게 손상을 입힐 수 있으며 복구하는 방법을 모를 경우 다시 설치 해야할 것이다. 따라서 나는 LFS 변수를 사용하지 말것을 강력히 권유한다. 여러분 스스로가 $LFS를 직접 /mnt/hda5 등으로 대체하는 것이 바람직하다. /mnt/hda5를 입력하는 동안 오타를 입력했을 경우 생길 수 있는 최악의 경우는 "no such file or directory" 라는 에러가 발생하는 것일 뿐 여러분의 시스템을 해치진 않을 것이다. 내가 여러분에게 경고를 하지 않았다고는 말하지 말길 바란다 ;)

2.2 소프트웨어를 받는 방법

이 문서 전반에 걸쳐서 나는 여러분이 다운 받은 모든 패키지들을 $LFS/usr/src 의 서브디렉토리 아래에 저장하고 있다고 가정할 것이다.

나 스스로는 $LFS/usr/src/sources 디렉토리를 사용하였다. sources 안에서 여러분은 0에서 9까지의 디렉토리와 a에서 z까지의 디렉토리를 발견할 것이다. sysvinit-2.78.tar.gz 같은 패키지는 $LFS/usr/src/sources/s/ 아래에 저장된다. bash-3.02.tar.gz 같은 패키지는 $LFS/usr/src/sources/b/ 아래에 저장되며 나머지도 이와 같다. 물로 여러분이 이런 방식을 따를 필요는 없으며 나는 단지 예제를 주었을 뿐이다. 패키지들을 $LFS/usr/src 밖에 보관하고 서브디렉토리로 이동하는 것이 좋다. 이렇게 하면 우리가 패키지를 풀어서 작업을 하는 $LFS/usr/src 디렉토리를 깨끗이 유지할 수 있다.

다음 장은 여러분이 다운받아야 할 모든 패키지들의 목록을 담고 있다. 그러나 아직 우리의 LFS 시스템을 담을 파티션을 만들지 않았으므로 임시로 여러분이 원하는 다른 곳에 보관해야 하며 4장이 끝나면 이것들을 $LFS/usr/src/<somesubdirectory>로 복사하는 것을 잊으면 안된다.

2.3 소프트웨어를 설치하는 방법

패키지를 가지고 실제로 무언가를 하기 전에 그것을 먼저 풀어야(unpack) 한다. 흔히 패키지 파일이 tar로 묶이고 gzip으로 압축된 것을 볼 것이다 (이는 .tar.gz나 .tgz의 확장자로부터 알 수 있다). 나는 ungzip과 untar 방법에 대해 매번 설명하지는 않겠다. 이 단락에서 그 방법에 대해 한 번만 얘기를 할 것이다. 여러분은 .tar.bz2 파일을 받을 수도 있다. 이 파일은 tar로 묶인 후 bzip2 프로그램으로 압축된 것이다. bzip2는 흔히 쓰이는 gzip보다 더 좋은 압축 성능을 가지고 있다. bz2 파일을 사용하기 위해선 bzip2 프로그램이 설치되어 있어야 한다. 전부는 아니지만 대부분의 배포본이 이 프로그램을 포함하고 있으며 따라서 이미 여러분의 시스템에 설치되엉 있을 가능성이 높다. 만약 그렇지 않다면 여러분의 배포본의 설치 툴을 사용하여 설치한다.

- 패키지를 여러분이 보관하고 있던 디렉토리에서 *$LFS/usr/src* 디렉토리로 *복사*하는 것으로 시작한다.
- tar와 gzip으로 묶인 파일을 가지고 있다면 다음을 실행하여 이를 푼다: `tar xvzf filename.tar.gz; rm filename.tar.gz` 혹은 `tar xvzf filename.tgz; rm filename.tgz`
- tar와 bzip2로 묶인 파일을 가지고 있다면 다음을 실행하여 이를 푼다: `tar --use-compress-prog=bzip2 -xvf filename.tar.bz2;rm filename.tar.bz2`
- tar로만 묶인 파일을 가지고 있다면 `tar xvf filename.tar; rm filename.tar`를 실행하여 이를 푼다.

여러분이 파일을 풀고 바로 그 파일이 더이상 필요없기 때문에 지웠다는 것을 주의하라. 이것이 그 파일을 *이동*이 아닌 *복사*를 해야 하는 이유다. 만약 이동을 시키고 지워버렸다면 이 패키지가 또 필요할 경우 다시 다운로드를 해야 할 것이다.

패키지 압축파일이 풀리면 현재 디렉토리 아래에 새로운 디렉토리가 만들어질 것이다 (그리고 이 문서는 여러분이 파일을 $LFS/usr/src 디렉토리 아래에서 풀었다고 가정한다). 설치 과정을 계속하기 전에 새 디렉토리로 이동해야 한다. 위의 모든 과정은 'xxx 압축파일을 푼다' 로 요약된다. 따라서 이러한 것을 읽게

되면 패키지를 $LFS/usr/src로 복사하고 tar 프로그램과 gzip/bzip2를 사용하여 압축을 푼 후 새로 만들어진 디렉토리로 이동한 후에 다음 설치 과정을 읽어야 한다.

패키지를 설치한 후 이를 가지고 두 가지 일을 할 수 있다. 소스가 들어있는 디렉토리를 지우거나 그냥 둘 수 있다. 그냥 두기로 했다면 내겐 좋다. 그러나 이후의 장에서 같은 패키지가 필요하게 된다면 (7.2장 까지의 모든 소프트웨어는 7.3장에서 재설치될 것이다) 이를 다시 쓰기 전에 디렉토리를 먼저 지워야 한다. 그렇지 않으면 이전 설정값들 (여러분의 기존의 리눅스 시스템에는 적용되지만 LFS 시스템으로 컴퓨터를 재시작한 후엔 더이상 적용되지 않는 설정값들) 이 쓰여서 문제가 될 수 있다.단순히 `make clean`을 하는 것이 소스 트리를 완벽히 깨끗하게 한다고 보장할 순 없다. 설정 스크립트는 make clean 과정에 의해 제거되지 않는 다양한 서브디렉토리 안에도 파일을 생성한다.

제 3 절 다운받을 패키지들

아래는 여러분이 이 문서에서 사용하기 위해 다운받아야 할 모든 소프트웨어들의 목록이다. 나는 소프트웨어를 받을 수 있는 사이트와 디렉토리를 표기했지만 최신의 소스 파일을 받는 것은 여러분의 몫이다. 버전 번호는 이 문서가 기반하고 있으며 작동한다고 알려진 버전을 나타낸다. 스스로 해결할 수 없는 문제를 겪게 된다면 이 문서에서 쓰이는 버전을 받는다 (더 새 버전을 받았을 경우에).

3.1 필수 소프트웨어

Sysvinit (2.78) : `ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/`

Bash (2.03) : `ftp://ftp.gnu.org/gnu/bash/`

Linux Kernel (2.2.14) : `ftp://ftp.kernel.org/pub/linux/kernel/`

Binutils (2.9.1) : `ftp://ftp.gnu.org/gnu/binutils/`

Bzip2 (0.9.5d) : `http://sourceware.cygnus.com/bzip2/`

Diff Utils (2.7) : `ftp://ftp.gnu.org/gnu/diffutils/`

File Utils (4.0) : `ftp://ftp.gnu.org/gnu/fileutils/`

GCC (2.95.2) : `ftp://ftp.gnu.org/gnu/gcc/`

Glibc (2.1.3) : `ftp://ftp.gnu.org/gnu/glibc/`

Glibc-crypt (2.1.2) : `ftp://ftp.gwdg.de/pub/linux/glibc/`

Glibc-linuxthreads (2.1.3) : `ftp://ftp.gnu.org/gnu/glibc/`

Grep (2.4) : `ftp://ftp.gnu.org/gnu/grep/`

Gzip (1.2.4) : `ftp://ftp.gnu.org/gnu/gzip/`

Make (3.78.1) : `ftp://ftp.gnu.org/gnu/make/`

Sed (3.02) : `ftp://ftp.gnu.org/gnu/sed/`

Shell Utils (2.0) : `ftp://ftp.gnu.org/gnu/sh-utils/`

Tar (1.13) : `ftp://ftp.gnu.org/gnu/tar/`

Text Utils (2.0) : `ftp://ftp.gnu.org/gnu/textutils/`

Util Linux (2.10f) : `ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/`

Bison (1.28) : `ftp://ftp.gnu.org/gnu/bison/`

Mawk (1.3.3) : `ftp://ftp.whidbey.net/pub/brennan/`

Find Utils (4.1) : `ftp://ftp.gnu.org/gnu/findutils/`

Ncurses (5.0) : `ftp://ftp.gnu.org/gnu/ncurses/`

Less (340) : `ftp://ftp.gnu.org/gnu/less/`

Perl (5.005_03) : `ftp://ftp.gnu.org/gnu/perl/`

M4 (1.4) : `ftp://ftp.gnu.org/gnu/m4/`

Texinfo (4.0) : `ftp://ftp.gnu.org/gnu/texinfo/`

Autoconf (2.13) : `ftp://ftp.gnu.org/gnu/autoconf/`

Automake (1.4) : `ftp://ftp.gnu.org/gnu/automake/`

Flex (2.5.4a) : `ftp://ftp.gnu.org/gnu/flex/`

E2fsprogs (1.18) : `ftp://tsx-11.mit.edu/pub/linux/packages/ext2fs/`

File (3.26) : `http://www.linuxfromscratch.org/download/file-3.26-lfs.tar.gz`

Groff (1.15) : `ftp://ftp.gnu.org/gnu/groff/`

Ld.so (1.9.9) : `ftp://tsx-11.mit.edu/pub/linux/packages/GCC/`

Libtool (1.3.4) : `ftp://ftp.gnu.org/gnu/libtool/`

Linux86 (0.14.3) : `http://www.linuxfromscratch.org/download/linux86-0.14.3-lfs.tar.gz`

Lilo (21) : `ftp://sunsite.unc.edu/pub/Linux/system/boot/lilo/`

Shadow Password Suite (19990827) : `ftp://piast.t19.ds.pwr.wroc.pl/pub/linux/shadow/`

Man (1.5h1) : `ftp://ftp.win.tue.nl/pub/linux-local/utils/man/`

Modutils (2.3.9) : `ftp://ftp.ocs.com.au/pub/modutils/`

Termcap (1.3) : `ftp://ftp.gnu.org/gnu/termcap/`

Procinfo (17) : `ftp://ftp.cistron.nl/pub/people/svm/`

Procps (2.0.6) : `ftp://people.redhat.com/johnsonm/procps/`

Psmisc (19) : `ftp://lrcftp.epfl.ch/pub/linux/local/psmisc/`

Start-stop-daemon (0.4.1) : `http://www.linuxfromscratch.org/download/ssd-0.4.1-lfs.tar.gz`

Sysklogd (1.3.31) : `ftp://sunsite.unc.edu/pub/Linux/system/daemons/`

Vim (5.6) : `ftp://ftp.vim.org/pub/editors/vim/unix/`

3.2 선택 가능한 소프트웨어

이하 모든 소프트웨어는 9장과 그 이후 장에서 사용되며 반드시 필요한 것은 아니다. 특정 패키지를 설치할 것인지는 여러분 스스로 결정해야 한다. 예를 들어 LFS 시스템을 온라인에 연결할 계획이 없다면 email 이나 telnet, ftp, www 등의 유틸리티들을 설치하길 원치 않을 것이다.

Netkit-base (0.17) : `ftp://ftp.uk.linux.org/pub/linux/Networking/netkit-devel/`

Net-tools (1.54) : `http://www.tazenda.demon.co.uk/phil/net-tools/`

Procmail (3.14) : `ftp://ftp.procmail.org/pub/procmail/`

Sendmail (8.9.3) : `ftp://ftp.sendmail.org/pub/sendmail/`

Mailx (8.1.1) : `http://www.linuxfromscratch.org/download/mailx-8.1.1-fixed.tar.gz`

Mutt (1.0i) : `ftp://ftp.mutt.org/pub/mutt/`

Fetchmail (5.2.0) : `http://www.tuxedo.org/\~{}esr/fetchmail/`

Netkit-telnet (0.17) : `ftp://ftp.uk.linux.org/pub/linux/Networking/netkit-devel/`

Proftpd (1.2.0pre9) : `ftp://ftp.tos.net/pub/proftpd/`

Netkit-ftp (0.17) : `ftp://ftp.uk.linux.org/pub/linux/Networking/netkit-devel/`

Apache (1.3.11) : `http://www.apache.org/dist/`

Zlib Library (1.1.3) : `http://www.cdrom.com/pub/infozip/zlib/`

Lynx (2.8.2) : `http://www.slcc.edu/lynx/release/`

PPP (2.3.11) : `ftp://cs.anu.edu.au/pub/software/ppp/`

Xfree86 (3.3.5) : `ftp://ftp.xfree86.org/pub/XFree86/`

libPropList (0.9.1) : `ftp://ftp.windowmaker.org/pub/libs/`

libXpm (4.7) : `ftp://sunsite.unc.edu/pub/Linux/libs/X/`

libpng (1.0.3) : `http://www.cdrom.com/pub/png/`

libtiff (3.4) : `ftp://ftp.sgi.com/graphics/tiff/`

libjpeg (6b) : `http://www.ijg.org/`

libungif (4.1.0) : `ftp://prtr-13.ucsc.edu/pub/libungif/`

WindowMaker (0.61.1) : `ftp://ftp.windowmaker.org/pub/release/`

제 4 절 새 시스템의 준비

4.1 일을 해나가는 방법

우리는 Debian 이나 SuSe, Slackware, Mandrake, RedHat 등의 이미 설치되어 있는 리눅스 배포본을 사용하여 LFS 시스템을 만들 것이다. 여러분은 어떤 류의 부트 디스켓도 필요치 않다. 우리는 기존의 리눅스 시스템을 사용할 것이다 (우리가 컴파일러나 링커, 텍스트 에디터 등의 툴들이 필요하기 때문이다).

만약 여러분이 아직 리눅스를 설치하지 않았다면 이 문서를 당장은 사용할 수 없다. 우선 리눅스 배포본을 설치할 것은 권한다. 어떤 것을 설치하던지 관계는 없다. 또한 최신 버전일 필요도 없지만 그렇다고 너무 오래된 것이어선 안된다. 일년 이내의 것이라면 괜찮을 것이다. 여러분의 기존 시스템이 glibc-2.0 이나 그 이후의 것을 쓴다면 별 문제가 없다. libc5 는 몇몇 문제를 야기할 수 있으며 내가 더이상 그런 머신을 쓰지 않기 때문에 이 문서에서도 지원되지 않는다.

4.2 새 파티션의 생성

새 리눅스 시스템을 만들기 전에 우리는 비어있는 리눅스 파티션이 필요하다. 나는 적어도 500MB 이상의 크기를 갖는 파티션을 권장한다. 부가 기능들 (email 이나 네트웍, 인터넷, X 윈도우 시스템 등) 이 없는 기본적인 시스템은 약 250MB 로도 가능하다. 이미 여러분이 리눅스 Native 파티션이 있다면 이 부분은 뛰어넘어도 된다.

적당한 하드디스크 (새 파티션을 프라이머리 마스터 IDE 디스크 위에 만들고 싶은 경우 /dev/hda 처럼) 를 옵션으로 하여 `fdisk` 프로그램 (혹은 여러분이 선호하는 fdisk 같은 프로그램) 을 실행한다. 리눅스 Native 파티션을 만들고 파티션 테이블을 쓴 후 fdisk 프로그램을 나온다. 파티션 테이블을 갱신시키기 위해 시스템을 리부팅 하라는 메시지를 받았다면 계속하기 전에 지금 리부팅하기 바란다. 여러분의 새 파티션 이름을 기억해 두도록 한다. (나와 같은 경우엔) hda5 같은 것일 수 있다. 이 새로 생성된 파티션은 이 문서에서 *LFS 파티션* 이라고 불려질 것이다.

4.3 새 파티션 위에 ext2 파일 시스템 만들기

일단 파티션이 만들어졌으면 그 파티션 위에 새로이 ext2 파일 시스템을 만들어야 한다. ext2 파일 시스템을 만들기 위해 우리는 `mke2fs` 명령을 사용할 것이다. 새 파티션을 유일한 옵션으로 주고 실행하면 파일 시스템이 생성된다. 만약 여러분의 파티션이 hda5라면 `mke2fs /dev/hda5` 같은 명령을 실행하면 된다.

4.4 새 파티션 마운트하기

ext2 파일 시스템을 만들고나면 사용할 수 있게 된 것이다. 이 파일 시스템을 사용하기 위해 (읽거나 쓰기 위해) 우리가 해야 할 것은 단지 그것을 마운트하는 것이다. 여러분이 이 파일 시스템을 /mnt/hda5 아래에 마운트한다면 /mnt/hda5 디렉토리로 가서 이 파티션을 사용할 수 있으며 이후 원하는 것을 무엇이던 할 수 잇다. 어떤 서브디렉토리를 여러분이 선택하는가는 중요치 않다 (심지어 여러분은 마운트 지점으로 /mnt 디렉토리를 사용할 수도 있다). 그러나 좋은 습관은 파티션과 같은 이름을 갖는 디렉토리를 만드는 것이다. 내 경우에 LFS 파티션은 hda5라 불리며 따라서 나는 이 파티션을 /mnt/hda5에 마운트 하였다.

- 아직 없다면 /mnt 디렉토리를 만든다.
- /mnt/xxx 디렉토리를 만든다. xxx는 LFS 파티션의 이름으로 대체된다.
- mount /dev/xxx /mnt/xxx 명령을 실행하여 LFS 파티션을 마운트한다. xxx는 LFS 파티션의 이름으로 대체된다.

이 디렉토리(/mnt/xxx)는 여러분이 이전에 봤던 $LFS 이다. 따라서 만약 어디선가 "cp inittab $LFS/etc" 라는 것을 본다면 실제로는 "cp inittab /mnt/xxx/etc" 라고 쳐야 하며 xxx는 여러분의 파티션 이름으로 바뀐다.

4.5 디렉토리의 생성

`http://www.pathname.com/fhs/` 에서 찾을 수 있는 FHS 표준에 따라서 LFS 파티션 위에 디렉토리 트리를 만든다. 아래의 명령들을 실행하면 필요한 디렉토리들이 생성될 것이다.

```
cd $LFS
mkdir bin boot dev etc home lib mnt proc root sbin tmp usr var
cd $LFS/usr
mkdir bin include lib sbin share src
ln -s share/man man
ln -s share/doc doc
ln -s . local
ln -s ../etc etc
ln -s ../var var
cd $LFS/usr/share
mkdir dict doc info locale man nls misc terminfo zoneinfo
cd $LFS/usr/share/man
mkdir man1 man2 man3 man4 man5 man6 man7 man8
cd $LFS/var
mkdir lock log run spool tmp
```

이제 디렉토리가 생성되었으면 3장에서 다운로드 받았던 소스 파일들을 $LFS/usr/src (이 서브디렉토리를 스스로 만들어야 한다) 아래의 임의의 서브디렉토리로 복사해야 한다.

4.6 /dev 디렉토리의 복사

mknod 명령을 사용하여 $LFS/dev 디렉토리 안에 있어야 할 파일들을 개별적으로 만들 수 있지만 시간이 많이 걸린다. 나는 현재의 /dev 디렉토리를 $LFS 파티션으로 단순히 복사하는 것을 택했다. 기존의 권한과 심볼릭 링크, 소유권을 유지하기 위하며 디렉토리를 복사하기 위해선 다음의 명령을 사용한다:

```
cp -av /dev $LFS
chown root.root $LFS/dev/*
```

제 5 절 LFS 시스템을 부팅 가능하도록 만들기

5.1 Sysvinit의 설치

일반적인 환경 하에선 커널이 로드되어 여러 시스템 구성 요소들을 초기화한 후 시스템의 부팅 프로세스를 마무리할 `init` 라 불리는 프로그램을 로드한다. 전부 다는 아닐 지라도 거의 대부분의 리눅스 시스템에 들어있는 패키지는 Sysvinit이며 이것이 우리가 LFS 시스템에 설치하려는 프로그램이다.

- Sysvinit의 압출파일을 푼다.
- src 디렉토리로 이동한다.
- `Makefile` 파일을 수정한다.
- 이 파일의 *all:* 규칙 이전의 어느 곳에라도 *ROOT = $LFS* 라는 줄을 첨가한다.
- 이 파일의 마지막 네 줄의 모든 /dev 앞에 각각 *$(ROOT)* 를 붙인다.

마지막 네 줄에 $(ROOT) 부분을 첨가한 후엔 아래와 같이 보일 것이다:

```
@if [! -p $(ROOT)/dev/initctl ]; then \
echo "Creating $(ROOT)/dev/initctl"; \
rm -f $(ROOT)/dev/initctl; \
mknod -m 600 $(ROOT)/dev/initctl p; fi
```

- 다음을 실행하여 패키지를 설치한다:

```
make -e LDFLAGS=-static; make install
```

5.2 Sysvinit 설정

Sysvinit가 동작하기 위해서 우리는 설정 파일을 만들어야 한다. 다음을 포함하도록 `$LFS/etc/inittab` 파일을 만든다:

```
# Begin /etc/inittab

id:2:initdefault:
```

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

1:2345:respawn:/sbin/sulogin

# End /etc/inittab
```

5.3 passwd 와 group 파일의 생성

여러분이 inittab 파일에서 보는 것처럼 우리가 시스템을 부팅하면 init 가 sulogin 프로그램을 실행시키며 sulogin은 여러분에게 root의 패스워드를 물어볼 것이다. 이는 우리가 LFS 시스템 상에 적어도 passwd 파일은 가지고 있어야 한다는 것을 의미한다.

- 다음을 포함하도록 $LFS/etc/passwd 파일을 생성한다: *root:s394ul1Bkvmq2:0:0:root:/root:/bin/bash*
- 다음을 포함하도록 $LFS/etc/group 파일을 생성한다: *root::0:*

위의 인코딩된 패스워드 문자열은 *lfs123* 이다.

여러분의 LFS 시스템에 로그온 할 때 root의 패스워드를 물어온다면 *lfs123*을 입력하면 된다.

5.4 Bash 쉘의 설치

sulogin 이 root 패스워드를 물어오고 여러분이 패스워드를 입력했다면 쉘이 시작되야 한다. 흔히 bash 쉘이 시작된다. 그러나 아직 설치된 라이브러리가 전혀 없기 때문에 우리는 Sysvinit 에서와 같이 bash 를 정적으로 링크해야 한다.

- Bash의 압축파일을 푼다.
- 다음을 실행하여 Bash를 설치한다:

```
./configure --enable-static-link
make; make -e prefix=$LFS/usr install
mv $LFS/usr/bin/bash $LFS/bin
cd $LFS/bin; ln -s bash sh
```

5.5 LILO에 엔트리 추가

이 파티션으로부터 부팅이 가능하기 위해선 우리는 `/etc/lilo.conf` 파일을 수정해야 한다. 다음 줄들을 lilo.conf 에 추가한다:

```
image=<currently used image>
   label=<label>
   root=$LFS
   read-only
```

<currently used image>를 여러분이 기존의 리눅스 시스템을 부팅하기 위해 사용하는 이미지 파일로 바꾼다. <label>는 여러분이 원하는 무엇이든 될 수 있다. 나는 "lfs"라 칭했다. 여러분이 <label>에 넣는 것이 부팅할 시스템을 고르기 위해 LILO 프롬프트에서 여러분이 입력하는 것이다.

부트 로더를 갱신하기 위해 `lilo` 프로그램을 실행한다.

5.6 시스템 검사

이 장을 끝마친 후 우리는 LFS 로 재부팅 하여 시스템을 검사할 수 있으며 그 시스템으로 로그론 할 수 있는지를 볼 수 있다. 여러분이 리부팅한 후 LFS 시스템을 시작하기 위해선 lilo.conf 파일에 삽입했던 label 을 LILO프롬프트에서 입력한다. 그리고 root 패스워드를 입력하면 이제 bash 프롬프트를 얻게 될 것이다. 여러분은 shutdown 같은 프로그램으로 시스템을 shutdown 시킬 수 없을 것이다. 이 프로그램이 있긴 하지만 이를 실행하려 하면 다음과 같은 에러가 날 것이다: "You don't exist. Go away." 이 에러의 의미는 시스템이 password 파일을 찾을 수 없다는 것이다. 비록 shutdown 프로그램이 필요로 하는 라이브러리에 정적으로 링크되어 있지만 GNU C 라이브러리의 일부인 NSS (Name Server Switch) 라이브러리에 여전히 의존하고 있으며 이는 이후 장에서 설치될 것이다. 이 NSS 라이브러리는 (이 경우엔) passwd 파일을 찾을 수 있는 장소에 대한 정보를 넘겨준다.

지금은 시스템을 리부팅하기 위해 `reboot -f` 명령을 쓸 수 있다. 이 명령은 shutdown 프로그램을 사용하는 종료 과정을 생략하고 곧바로 리부팅을 한다. 파일 시스템이 읽기 전용으로 마운트되어 있기 때문에 어떤 방법으로든 이 리부팅 방식이 우리 시스템에 해를 끼치진 않는다 (시스템을 다음번에 마운트할 때, 이전에 깨끗하게 언마운트 되지 않았으며 파일 시스템을 본래대로 유지하기 위해선 e2fsck를 실행해야 한다는 경고문을 보게될 것이지만 말이다).

제 6 절 커널 설치

6.1 ftp.kernel.org에 대한 주의사항

앞의 2장에서 나는 여러분이 새 커널을 ftp://ftp.kernel.org/ 에서 다운받을 수 있다고 얘기했다. 그러나 이 사이트는 매우 붐비며 이 사이트의 운영자들은 여러분이 커널을 여러분에게 가까운 곳에서 받기를 권유한다. 여러분은 ftp://ftp.<country code>.kernel.org/로 가서 미러 사이트에 접근할 수 있다 (ftp.ca.kernel.org 처럼).

6.2 커널 설정

- 현재의 /usr/src/linux 디렉토리를 `mv /usr/src/linux /usr/src/linux-old` 같은 명령을 실행하여 다른 것으로 이름을 바꾼다. (/usr/src/linux는 실제 디렉토리가 아닌 심볼릭 링크일 수 있다. 어느 경우든 이름을 변경한다).
- /usr/include/linux와 /usr/include/asm을 `rm -r /usr/include/linux /usr/include/asm` 을 실행하여 삭제한다.
- 커널 압축파일을 `/usr/src/` 디렉토리 안에 푼다 (이 과정은 새로 /usr/src/linux 디렉토리를 만든다).
- /usr/include/linux와 /usr/include/asm을 다음을 실행하여 심볼릭 링크로 만든다:

```
cd /usr/include
ln -s ../src/linux/include/linux linux
ln -s ../src/linux/include/asm-<cpu> asm
```

/usr/src/linux/include 디렉토리를 찾아서 어떤 asm-* 디렉토리가 존재하는지 본다. 그리곤 여러분의 플랫폼에 맞는 것을 고른다. 만약 Intel 플랫폼을 사용한다면 ln -s ../src/linux/include/asm-i386 asm 을 실행한다.

- 커널을 설정할 방법을 선택하고 (설정 방법에 대한 더 자세한 사항은 README 파일을 참조한다) 아직은 어떤 것오 모듈로 설정하지 않도록 한다. 이는 우리가 당분간은 커널 모듈을 로드하기 위해 필요한 소프트웨어를 가지고 있지 않기 때문이다.

- 커널 설정이 끝난 후 `make dep` 을 실행한다.
- `make bzImage` 를 실행하여 커널을 컴파일한다.
- `arch/<cpu>/boot/bzImage` 파일을 `/boot` 디렉토리 (혹은 여러분의 리눅스 시스템이 커널 이미지 등의 파일을 저장하는 관습에 따라선 다른 곳으로) 복사한다.
- 선택사항으로 `/boot/bzImage` 파일을 `/boot/lfskernel`과 같은 다른 것으로 이름을 바꿀 수 있다.
- `cp -av /usr/src/linux $LFS/usr/src` 를 실행하여 전체 커널 소스 트리를 LFS 파티션으로 복사한다.
- 다음을 실행하여 $LFS/usr/include/linux 와 $LFS/usr/include/asm을 심볼릭 링크로 생성한다.

```
cd $LFS/usr/include
ln -s ../src/linux/include/linux linux
ln -s ../src/linux/include/asm asm
```

6.3 LILO의 갱신

- `/etc/lilo.conf` 파일을 열어 LFS 부분으로 간다.
- 이미지 이름을 `lfskernel` (혹은 여러분이 바꾼 bzImage 파일의 새 이름) 으로 변경한다.
- 부트 로더를 갱신하기 위해 `lilo`를 실행한다.

6.4 시스템의 검사

시스템을 리부팅하여 LFS 시스템을 시작한다. 새로 설치된 커널이 비정상적인 동작 (크래쉬 같은) 을 하는지를 검사한다.

제 7 절 기본적인 시스템 소프트웨어의 설치

이 장에서 우리는 기본적인 리눅스 시스템에 필요한 모든 소프트웨어를 설치할 것이다. 이 장이 끝나면 여러분은 완전하게 동작하는 리눅스 시스템을 갖게 된다. 나머지 장들은 네트워킹 설정이나 인터넷 서버 + 클라이언트 (telnet, ftp, http, email), 인터넷 설정, X 윈도우 시스템 같은 선택적인 주제들에 대해 다룰 것이다. 여러분은 재량껏 그 장들을 뛰어넘어도 된다. 예를 들어 LFS 시스템을 네트워에 연결할 계획이 없다면 인터넷을 설정하는 것은 거의 필요치 않을 것이다.

모든 기본적인 시스템 소프트웨어를 설치하기 전에 미리 설치되어 있어야 할 많은 패키지들이 있다. 전형적인 설정 스크립트들은 rm 이나 grep, sed, mv, cat, cp, diff 같은 프로그램들을 필요로 한다. 여러분은 압축 파일을 tar 와 gzip 으로 묶인 파일을 풀어야 하며 오브젝트 파일로 컴파일한 후 프로그램을 링크시며야 한다. 이 모든 (몇개를 더 포함해서) 프고르매들은 다른 어떤 것을 설치할 수 있기 전에 사용할 수 있어야 한다. 이 프로그램들은 정적으로 링크될 것이다. 그 이유는 여러분의 기존의 리눅스 시스템이 LFS 시스템에 설치하려는 것과 다른 버전의 C 라이브러리를 가지고 있을 수 있기 때문이다. 이 장에서 설치하는 프로그램들은 기존의 리눅스 시스템의 C 라이브러리에 링크될 것이다. 이것은 이 프로그램들을 LFS 시스템에서 실행하는 경우 라이브러리 충돌을 일으킬 수 있다. 따라서 우리는 이 프로그램들을 정적으로 링크해야 한다. 기본적인 시스템 소프트웨어들을 설치하는 동안 우리는 정적으로 링크된 소프트웨어들을 다시 설치하여 LFS 시스템의 C 라이브러리에 동적으로 링크되도록 할 것이다.

7.1 디버깅 심볼에 대하여

모든 프로그램이나 바이너리는 기본적으로 디버깅 심볼을 포함하여 컴파일된다. 이는 여러분이 프로그램이나 라이브러리를 디버거를 통해 실행할 수 있으며 디버거의 출력물이 좀 더 사용자에게 편하게 된다는 것을 뜻한다. 이 디버깅 심볼들은 또한 프로그램이나 라이브러리의 크기를 매우 크게 한다. 이 문서는 디버깅 심볼 없이 소프트웨어를 설치하진 않는다 (많은 독자들이 디버깅 소프트웨어를 사용하는지 않하는지를 내가 알지 못하기 때문에). 대신에 여러분은 원할경우 `strip` 프로그램으로 그 심볼들을 직접 제거할 수 있다.

바이너리 (a.out 이나 ELF 바이너리여야 한다) 로부터 디버깅 심볼을 제거하기 위해선 `strip --strip-debug filename` 을 실행한다. 여러 파일들에서 디버깅 심볼을 제거하고 싶다면 와일드 카드 문자를 쓰면 된다 (`strip --strip-debug $LFS/usr/bin/*` 와 같이).

여러분이 디버깅 심볼이 큰 차이를 만드는지 알고 싶다면 아래에 통계가 있다:

- 디버깅 심볼을 포함하여 정적으로 링크된 Bash 바이너리: 2.3MB
- 디버깅 심볼 없이 정적으로 링크된 Bash 바이너리: 645KB
- 디버깅 심볼을 포함하여 동적으로 링크된 Bash 바이너리: 1.2MB
- 디버깅 심볼 없이 동적으로 링크된 Bash 바이너리: 478KB
- 디버깅 심볼을 포함한 $LFS/usr/lib (glibc 와 gcc 파일들): 87MB
- 디버깅 심볼이 없는 $LFS/usr/lib (glibc 와 gcc 파일들): 16MB

사용하는 컴파일러와 동적으로 프로그램을 링크시키기 위해 쓰이는 C 라이브러리의 버전에 따라 크기가 다를 수 있으나 디버깅 심볼을 포함한 것과 그렇지 않은 것을 비교한다면 여러분의 결과는 비슷할 것이다. 이 장을 끝내고 모든 LFS 바이너리들과 라이브러리들에서 디버깅 심볼을 제거한 후 나는 102MB 를 약간 넘은 디스크 공간을 다시 얻었다. 매우 차이가 크다. 그 차이는 모든 것을 설치하게 되는 이 문서의 마지막에 같은 일을 했을 때 더욱 커질 것이다.

7.2 기본적인 시스템 소프트웨어 설치를 위한 LFS 시스템의 준비

7.2.1 Binutils의 설치

- binutils의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
make -e LDFLAGS=-all-static
make -e prefix=$LFS/usr install
```

7.2.2 Bzip2의 설치

- Bzip2의 압축파일을 푼다.
- 텍스트 에디터에서 Makefile 파일을 수정한다.
- $(CC) $(CFLAGS) -o 로 시작하는 줄을 찾는다.
- 그 부분을 $(CC) $(CFLAGS) $(LDFLAGS) -o 로 수정한다.

- 아래를 실행하여 패키지를 설치한다:

```
make -e LDFLAGS=-static
make -e PREFIX=$LFS/usr install
cd $LFS/usr/bin
mv bunzip2 bzip2 $LFS/bin
```

7.2.3 Diffutils의 설치

- diffutils의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
```

이 패키지는 특정 플랫폼에서 정적 링크에 문제를 일으킨다고 알려져 있다. 여러분이 이 패키지를 컴파일하는데 문제를 겪는다면 수정된 패키지를 `http://www.linuxfromscratch.org/download/diffutils-2.7-fixed.tar.gz` 에서 구할 수 있다.

7.2.4 Fileutils의 설치

- fileutils의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure --disable-nls
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
cd $LFS/usr/bin
mv chgrp chmod chown cp dd df ln ls mkdir mknod mv rm rmdir sync $LFS/bin
```

7.2.5 필요한 경우 기존의 시스템에 GCC 설치하기

Glibc-2.1.3을 컴파일하기 위해서는 gcc-2.95.2가 설치되어 있어야 한다. 2.8 이후의 어떤 버전이라도 가능하지만 2.95.2를 추천한다. 많은 glibc-2.1 기반의 시스템은 gcc-2.7.2.3이 설치되어 있으며 이 컴파일러로는 glibc-2.1을 컴파일할 수 없다. 따라서 우리는 기존의 시스템에도 gcc-2.95.2를 설치할 것이다. 그러나 기존의 컴파일러를 덮어씌우진 않을 것이다. gcc를 여러분의 시스템에 설치하기 전에 이것이 필요한지 확실히 결정해야 한다. 기존의 것의 버전 번호가 2.8 이상인지 보기 위해서 `gcc --version` 를 실행한다. 만약 그렇지 않다면 여러분은 gcc-2.95.2를 설치해야 한다. 후에 glibc를 컴파일하는데 어려움을 겪는다면 어쨌든 gcc-2.95.2를 설치하길 원하게 될 것이다.

- GCC 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
mkdir $LFS/usr/src/gcc-build; cd $LFS/usr/src/gcc-build
../gcc-2.95.2/configure --prefix=/usr/gcc2952 \
--with-local-prefix=/usr/gcc2952 --with-gxx-include-dir=/usr/gcc2952/include/g++ \
--enable-shared --enable-languages=c,c++
make bootstrap; make install
```

7.2.6 LFS 시스템에 GCC를 설치하기

- GCC 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
mkdir $LFS/usr/src/gcc-build;cd $LFS/usr/src/gcc-build
../gcc-2.95.2/configure --enable-languages=c --disable-nls
make -e LDFLAGS=-static bootstrap
make -e prefix=$LFS/usr local_prefix=$LFS/usr install
```

필요한 심볼릭 링크들의 생성 모든 프로그램들이 컴파일러와 프리-프로세서를 찾을 수 있도록 하기 위해서 몇가지 심볼릭 링크가 필요하다. 몇몇 프로그램은 cc 프로그램을 실행하고 다른 것들은 gcc 프로그램을 실행하며, 몇몇 프로그램은 /lib (LFS 시스템에서의 /usr/lib) 에 cpp 프로그램이 있길 예상하지만 다른 것들은 /usr/bin 아래에 있길 예상한다.

- 아래를 실행하여 심볼릭 링크들을 만든다:

```
cd $LFS/lib; ln -s ../usr/lib/gcc-lib/<host>/2.95.2/cpp cpp
cd $LFS/usr/lib; ln -s gcc-lib/<host>/2.95.2/cpp cpp
cd $LFS/usr/bin; ln -s gcc cc
```

<host>를 gcc-2.95.2 파일들이 설치된 디렉토리 (내 경우엔 i686-unknown-linux)로 대체한다. 여러분은 아마 두 개의 서로 다른 디렉토리를 발견할 것이다.

7.2.7 Glibc의 설치

glibc-crypt 패키지에 대한 주의사항:

```
-*-*-*-*-*-
프랑스, 러시아 미국 등의 일부 국가들이 암호화 소프트웨어의 사용과 배포를
매우 엄격히 규제하고 있기 때문에 이 add-on은 GNU C 라이브러리의 주 배포판
안에 포함되지 않는다. 더 자세한 사항은 메뉴얼의 "Legal Problems" 부분을
읽기 바란다.

특별히, 미국은 이 소프트웨어를 라이센스 없이 미국 밖으로 유출하는 것을 -
인터넷을 통하는 것을 포함하여 - 허가하지 않는다. 따라서 여러분이 미국 밖에
있다면 이 add-on을 FSF의 공식 FTP 사이트인 ftp.gnu.org로부터 받지 말길
바란다. 이 소프트웨어는 완전히 미국 밖에서 개발되었다.
-*-*-*-*-*-
```

"이 소프트웨어"는 ftp://ftp.gwdg.de/pub/linux/glibc/에 있는 glibc-crypt 패키지를 가리킨다. 이 법규는 단지 미국에 살지 않는 사람들에게만 적용된다. DES 소프트웨어를 가지고 들어오는 것이 규제되지는 않으므로 여러분이 미국 내에 살고 있다면 위의 독일 사이트로부터 이 소프트웨어를 구할 수 있다.

- Glibc의 압축파일을 푼다.
- glibc-crypt와 glibc-linuxthreads 압축파일을 압축이 풀린 glibc 디렉토리로 복사한다.
- 그곳에서 glibc-crypt와 glibc-linuxthreads의 압축을 푼다. 그러나 그 디렉토리로는 이동하지 않는다. 단시 tar와 gzip으로 압축만을 푼다.
- 아래를 포함하는 `configparms` 파일을 새로 만든다:

```
# Begin configparms
slibdir=/lib
sysconfdir=/etc
# End configparms
```

- 여러분의 기존 시스템이 이미 glibc를 컴파일하기에 적당한 버전의 gcc를 가지고 있다면 아래를 실행하여 패키지를 설치한다:

```
mkdir $LFS/usr/src/glibc-build;cd $LFS/usr/src/glibc-build
../glibc-2.1.3/configure --enable-add-ons
make; make install_root=$LFS install
```

- 적당한 버전의 gcc가 없어서 gcc-2.95.2를 기존 시스템에 설치했다면 아래를 실행하여 패키지를 설치한다:

```
mkdir $LFS/usr/src/glibc-build; cd $LFS/usr/src/glibc-build
CC=/usr/gcc2952/bin/gcc ../glibc-2.1.3/configure --enable-add-ons
make; make install_root=$LFS install
```

이전의 NSS 라이브러리 파일들의 복사 여러분의 기존 리눅스 시스템이 libc-2.0.x를 사용한다면 NSS 라이브러리 파일들을 LFS 파티션으로 복사해야 한다. 정적으로 링크된 프로그램들 중 일부는 여전히 NSS 라이브러리에 의존하며 username, userid 그리고 groupid를 찾는 프로그램들이 특히 그렇다. `ls -l libc.so.*`를 실행하여 기존의 리눅스 시스템이 어떤 버전의 C 라이브러리를 사용하는지 볼 수 있다.

결과가 /lib/libc.so.6 -> libc-2.0.7.so 같은 것이라면 시스템은 glibc-2.0을 사용하는 것이다.

결과가 /lib/libc.so.6 -> libc-2.1.2.so 같은 것이라면 시스템은 glibc-2.1을 사용하는 것이다.

여러분의 시스템이 libc-2.0.x.so 파일 (x는 7 같은 마이크로 버전 번호) 을 가지고 있다면 `cp -av /lib/*nss* $LFS/lib` 를 실행하여 NSS 라이브러리 파일들을 복사한다.

7.2.8 grep의 설치

- grep의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure --disable-nls
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
```

이 패키지는 특정 플랫폼에서 정적 링크 에러를 낸다고 알려져 있다. 여러분이 이 패키지를 컴파일할 때 문제를 겪는다면 수정된 패키지를 `http://www.linuxfromscratch.org/download/grep-2.4-fixed.tar.gz`에서 구할 수 있다.

7.2.9 gzip의 설치

- gzip의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
cd $LFS/usr/bin
mv gunzip gzip $LFS/bin
```

이 패키지는 특정 플랫폼에서 컴파일 문제를 일으킨다고 알려져 있다. 여러분이 이 패키지를 컴파일할 때 문제를 겪는다면 수정된 패키지를 `http://www.linuxfromscratch.org/download/gzip-1.2.4-fixed.tar.gz`에서 구할 수 있다.

7.2.10 Make의 설치

- Make의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
```

7.2.11 Sed의 설치

- sed의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
mv $LFS/usr/bin/sed $LFS/bin
```

이 패키지는 특정 플랫폼에서 정적 링크 문제를 일으킨다고 알려져 있다. 여러분이 이 패키지를 컴파일할 때 문제를 겪는다면 수정된 패키지를 `http://www.linuxfromscratch.org/download/sed-3.02-fixed.tar.gz`에서 구할 수 있다.

7.2.12 Sh-utils의 설치

- sh-utils의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure --disable-nls
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
cd $LFS/usr/bin
mv date echo false pwd stty su true uname hostname $LFS/bin
```

7.2.13 Tar의 설치

- tar의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure --disable-nls
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
mv $LFS/usr/bin/tar $LFS/bin
```

7.2.14 Textutils의 설치

- textutils의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure --disable-nls
make -e LDFLAGS=-static
make -e prefix=$LFS/usr install
mv $LFS/usr/bin/cat $LFS/bin
```

7.2.15 Util-linux의 설치

- util-linux의 압축파일을 푼다.
- 아래를 실행하여 패키지를 설치한다:

```
./configure
cd lib;make
cd ../mount;make -e LDFLAGS=-static
cp mount umount $LFS/bin
cp swapon $LFS/sbin
```

7.3 기본적인 시스템 소프트웨어의 설치

모든 소프트웨어들의 설치는 매구 직관적이며 여러분은 각 패키지에 대한 일반적인 설치 방법을 설명하고 특정 패키지가 다른 설치 방법을 쓴다면 그 방법을 설명하는 것이 훨씬 쉽고 짧다고 생각할 것이다. 이 문제에 대해 나도 여러분의 의견에 찬성하지만 나는 모든 패키지에 완벽한 과정을 설명하는 것을 선택했다. 이것은 단순히 가능할 지도 모를 에러나 혼돈을 막기 위함이다. 이 문서를 더 진행해 가기 전에 여러분은 시스템을 재시작하여 LFS 시스템으로 부팅해야 한다. 그러나 그러기 전에 어떤 파티션이 스왑 파티션으로 사용되는지 알아야 한다. 이 정보는 일반적으로 /etc/fstab 파일에서 찾을 수 있다. */dev/hda6 none swap sw 0 0* 과 비슷한 줄을 이 파일에 있는지 찾아본다.

그 줄의 4번째 필드는 만드시 'sw'여야 한다. 그 줄은 스왑 파티션을 나타낸다. 여러분이 기억해야할 것은 그 위치이다 (내 경우엔 /dev/hda6이지만 여러분의 시스템에선 다를 수 있다). 어떤 파티션이 스왑 파티션인지 알았으면 컴퓨터를 리부팅하여 여기부터 계속해 나가도 된다.

7.3.1 파티션을 재마운트하고 스왑 활성화하기

소프트웨어를 설치하기 전에 우리는 파티션을 읽고쓰기 모드로 다시 마운트해야 한다. 또한 규모가 큰 컴파일 과정 (gcc의 컴파일 처럼) 동안 메모리 부족 문제가 발생하지 않도록 스왑 파티션을 활성화해야 한다.

```
mount -o remount,rw / /
/sbin/swapon <swap device>
```

7.3.2 GCC의 설치

- GCC의 압축 파일을 풀고 아래를 실행하여 설치한다:

```
mkdir $LFS/usr/src/gcc-build;cd $LFS/usr/src/gcc-build
../gcc-2.95.2/configure --with-gxx-include-dir=/usr/include/g++ \
--enable-shared --enable-languages=c,c++
make bootstrap; make install
```

7.3.3 Bison의 설치

- bison의 압축파일을 출고 아래를 실행하여 설치한다:

```
./configure --datadir=/usr/share/bison
make; make install
```

7.3.4 Mawk의 설치

- mawk의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd /usr/bin; ln -s mawk awk
```

7.3.5 Findutils의 설치

- findutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

이 패키지는 컴파일 문제를 일으킨다고 알려져 있다. 여러분이 이 패키지를 컴파일할 때 문제를 겪는다면 수정된 패키지를 `http://www.linuxfromscratch.org/download/findutils-4.1-fixed.tar.gz`에서 구할 수 있다.

7.3.6 Termcap의 설치

- Termcap의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.7 Ncurses의 설치

- ncurses의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure --with-shared
make; make install
```

7.3.8 Less의 설치

- Less의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/bin/less /bin
```

7.3.9 Perl의 설치

- Perl의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./Configure
make; make install
```

우리는 'make test' 과정을 생략한다는 것을 주의한다. 이는 이 순간에 아직 시스템이 perl 테스트를 실행할 준비가 되지 않았기 때문이다. 지금은 우리는 perl 컴파일이 잘 되었다고 믿을 것이다.

7.3.10 M4의 설치

- M4의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.11 Texinfo의 설치

- Texinfo의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.12 Autoconf의 설치

- Autoconf의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.13 Automake의 설치

- Automake의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make install
```

7.3.14 Installing Bash Bash의 설치

- Bash의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/bin/bash /bin
```

7.3.15 Installing Flex Flex의 설치

- Flex의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.16 Binutils의 설치

- Binutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.17 Bzip2의 설치

- Bzip2의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
cd /usr/bin; mv bunzip2 bzip2 /bin
```

7.3.18 Diffutils의 설치

- Diffutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.19 E2fsprogs의 설치

- E2fsprogs의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/sbin/mklost+found /sbin
```

7.3.20 File의 설치

- File의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.21 Fileutils의 설치

- Fileutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd /usr/bin
mv chgrp chmod chown cp dd df ln ls mkdir mknod mv rm rmdir sync /bin
```

7.3.22 Grep의 설치

- Grep의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.23 Groff의 설치

- Groff의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.24 Gzip의 설치

- Gzip의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd /usr/bin; mv z* gunzip gzip /bin
```

7.3.25 Ld.so의 설치

- Ld.so의 압축파일을 풀고 아래를 실행하여 설치한다:

```
cd util; make ldd ldconfig
cp ldd /bin; cp ldconfig /sbin
rm /usr/bin/ldd
```

7.3.26 Libtool의 설치

- Libtool의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.27 Linux86의 설치

- Linux86의 압축파일을 풀고 아래를 실행하여 설치한다:

```
cd as
make; make install
cd ../ld
make ld86; make install
```

7.3.28 Lilo의 설치

- Lilo의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
```

7.3.29 Make의 설치

- Make의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.30 Sh-Utils의 설치

- Sh-utils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd /usr/bin
mv date echo false pwd stty su true uname hostname /bin
```

7.3.31 Shadow Password Suite의 설치

- Shadow의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd etc
cp limits login.access login.defs.linux shells suauth /etc
mv /etc/login.defs.linux /etc/login.defs
cd /usr/sbin
mv chpasswd dpasswd groupadd groupdel groupmod logoutd mkpasswd \
newusers useradd userdel usermod grpck pwck vipw grpconv grpunconv \
pwconv pwunconv /sbin
```

7.3.32 Man의 설치

- Man의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure -default
make all; make install
```

7.3.33 Modutils의 설치

- Modutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.34 Procinfo의 설치

- Procinfo의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
```

7.3.35 Procps의 설치

- Procps의 압축파일을 풀고 아래를 실행하여 설치한다:

```
gcc -O3 -Wall -Wno-unused -c watch.c
make; make -e XSCPT="" install
mv /usr/bin/kill /bin
```

7.3.36 Psmisc의 설치

- Psmisc의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
```

7.3.37 Sed의 설치

- Sed의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/bin/sed /bin
```

7.3.38 start-stop-daemon의 설치

- start-stop-daemon의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make start-stop-daemon
cp start-stop-daemon /sbin
cp start-stop-daemon.8 /usr/share/man/man8
```

7.3.39 Sysklogd의 설치

- Sysklogd의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
```

7.3.40 Sysvinit의 설치

- Sysvinit의 압축파일을 풀고 아래를 실행하여 설치한다:

```
cd src
make; make install
```

7.3.41 Tar의 설치

- Tar의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/bin/tar /bin
```

7.3.42 Textutils의 설치

- Textutils의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
mv /usr/bin/cat /bin
```

7.3.43 Vim의 설치

- Vim-rt와 Vim-src의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

7.3.44 Util-linux의 설치

- Util-linux 패키지를 푼다.
- MCONFIG 파일을 열어서 아래의 변수들을 찾아 다음처럼 고친다:

```
HAVE_PASSWD=yes
HAVE_SLN=yes
HAVE_TSORT=yes
```

- 아래를 실행하여 패키지를 설치한다:

```
groupadd -g 5 tty
./configure
make; make install
```

7.4 예전의 NSS 라이브러리 파일들의 제거

여러분이 기존의 리눅스 시스템에서 LFS 시스템으로 NSS 라이브러리 파일들을 복사했었다면 (기존 시스템이 glibc-2.0을 쓰기 때문에) 이제 아래를 실행하여 그 파일들을 제거한다:

```
rm /lib/libnss*.so.1 /lib/libnss*2.0*
```

7.5 소프트웨어의 설정

이제 모든 소트으웨어들이 설치되었고 몇몇 프로그램이 정상적으로 돌도록 하기 위해 우리가 해야할 것은 설정 파일들을 만드는 것 뿐이다.

7.5.1 Glib의 설정

/etc/nsswitch.conf 파일을 만들어야 한다. 이 파일이 없거나 잘못된 경우 glibc가 기본 값을 사용하긴 하지만 이 기본값은 이후 장에서 다룰 네트워킹과 잘 동작하지 않는다. 또한 우리의 타임존도 설정되어야 한다.

- 아래를 포함하도록 새 /etc/nsswitch.conf 파일을 만든다:

```
# Begin /etc/nsswitch.conf
passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

netgroup: db files
# End /etc/nsswitch.conf
```

- tzselect 스크립트를 실행하여 여러분의 타임존에 맞도록 질문에 답한다.
- 이 작업이 끝나면 프로그램은 여러분에게 필요한 파일의 위치를 알려줄 것이다.
- ln -s /usr/share/zoneinfo/<tzselect's output> /etc/localtime 을 실행하여 localtime을 심볼릭 링크로 생성한다.

/etc/localtime or ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime tzselect의 결과는 "EST5EDT"나 "Canada/Eastern" 같은 것이다. 그런 정보로 여러분이 만드는 심볼릭 링크는 ln -s /usr/share/zoneinfo/EST5EDT 이거나 ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime 이다.

7.5.2 LILO의 설정

lilo의 설정 파일을 처음부터 만들지는 않고 기존의 리눅스 시스템의 파일을 사용할 것이다. 이 파일은 머신에 따라서 틀리며 따라서 내가 여기서 만들 수는 없다. 여러분은 기존의 리눅스 시스템에서 쓰던 것과 똑같은 lilo 옵션들을 쓰길 원할 것이므로 기존 시스템의 것과 똑같이 파일을 만든다.

- /mnt/original 디렉토리의 생성
- mount /dev/xxx /mnt/original 을 실행하여 기존의 리눅스 시스템을 이 곳에 마운트한다 (/dev/xxx를 여러분의 파티션 위치로 변경한다).
- 아래를 실행하여 lilo 설정 파일과 lilo가 사용하는 커널 이미지들을 복사한다:

```
cp /mnt/original/etc/lilo.conf /etc
cp /mnt/original/boot/* /boot
```

기존의 리눅스 시스템이 커널 이미지들은 (어느 것도) /mnt/original/boot 안에 가지고 있지 않다면 그 파일들의 위치를 /etc/lilo.conf 파일에서 확인한 후 그 파일들을 /etc/lilo.conf가 찾을 수 있는 곳으로 복사한다. 혹은 여러분은 그 파일들을 /boot로 복사한 후 /etc/lilo.conf 파일을 수정하여 LFS 시스템에서 커널 이미지들이 있는 새 경로를 찾도록 할 수 있다. 어떤 것도 잘 동작하며 어떻게 할 것인가는 여러분에게 달렸다.

7.5.3 Sysklogd의 설정

- mkdir /var/log 를 실행하여 /var/log 디렉토리를 만든다.
- 아래를 포함하도록 /etc/syslog.conf 파일을 만든다.

```
# 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
```

7.5.4 Shadow Password Suite의 설정

이 패키지는 사용자의 암호를 바꾸고 새로운 사용자/그룹을 추가하고 사용자/그룹을 삭제하는 등의 일을 하기 위핸 유틸리티들을 포함하고 있다. 'password shadowing'이 무엇인지를 설명하진 않겠다. 여러분은 doc/HOWTO 파일 안에서 이에 대한 것을 모두 읽어볼 수 있다. 여러분이 명심할 것이 하나 있는데,

shadow 지원 기능을 사용하기로 했다면 암호를 확인할 필요가 있는 프로그램들 (예를 들어 xdm이나 ftp 데몬, pop3d 등) 이 'shadow-compliant(섀도우 호완)'이어야 한는 것이다. 다시말해 그 프로그램들이 섀도우 패스워드와 잘 작동할 수 있어야 한다.

섀도우 패스워드를 사용하지 않기로 결정했더라도 (doc/HOWTO 문서를 읽은 후에) 여러분은 여전히 패키지를 사용하는데 그것은 이 패키지 안의 유틸리티들으 섀도우 패스워드를 사용하지 않는 시스템에서도 사용되기 때문이다. 이에 대한 모든 내용을 HOWTO에서 볼 수 있다. 또한 여러분이 원하는 때에 섀도우 패스워드의 사용 여부를 변경할 수 있다는 것을 명심한다.

지금이 doc/HOWTO 파일의 #5 부분을 읽을 매우 좋은 시기이다. 여러분은 섀도우 기능이 작동하는지 여부를 검사하는 방볍과 기능을 정지시키는 방볍을 알 수 있다. 만약 작동하지 않고 있으며 검사를 하지 않았었다면 여러분의 모든 콘솔에서 로그아웃 한 후 더이상 시스템을 사용할 수 없게 될 것이다. 이는 더이상 로긴을 할 수 없기 때문이다. 이 문제를 쉽게 해결하는 방볍은 커널에 init=/sbin/sulogin 이라는 인자를 넘겨준 후 util-linux 압축파일을 풀고 login-utils 디렉토리로 이동하여 login 프로그램을 만든 후 /bin/login을 그것으로 바꿔 버리는 것이다. 일이 절망적으로 잘못되진 않겠지만 (적어도 Linux 상에서는) 알맞게 테스트를 하고 매뉴얼을 읽음으로써 어려움을 피할 수 있다 ;)

7.5.5 Sysvinit의 설정

/etc/inittab 파일을 아래처럼 수정한 후 여러분은 이전처럼 로긴할 수 있다 (agetty와 login 프로그램을 사용하여). Sulogin은 일반적인 로긴에선 더이상 쓰이지 않을 것이다.

- /etc/inittab 파일을 수정하여 아래를 포함하도록 한다.

```
# Begin /etc/inittab

id:2:initdefault:

si::sysinit:/etc/init.d/rcS

su:S:wait:/sbin/sulogin

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

ft:6:respawn:/sbin/sulogin

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

1:2345:respawn:/sbin/agetty /dev/tty1 9600
2:2345:respawn:/sbin/agetty /dev/tty2 9600
3:2345:respawn:/sbin/agetty /dev/tty3 9600
4:2345:respawn:/sbin/agetty /dev/tty4 9600
5:2345:respawn:/sbin/agetty /dev/tty5 9600
6:2345:respawn:/sbin/agetty /dev/tty6 9600

# End /etc/inittab
```

7.5.6 /var/run/utmp 파일의 생성

login이나 shutdown 등의 프로그램들은 /var/run/utmp 파일에 무언가 쓰기를 원한다. 이 파일은 현재 로긴해 있는 사람들에 대한 정보를 담고 있다. 또한 컴퓨터가 마지막으로 셧다운 된 때에 대한 정보도 담고 있다.

- touch /var/run/utmp 를 실행하여 /var/run/utmp 파일을 만든다.
- chmod 644 /var/run/utmp 를 실행하여 알맞게 파일 퍼미션을 조절한다.

제 8 절 시스템 부트 스크립트의 생성

이 부트 스크립트들은 시스템이 부팅될 때 시작된다. 이 스크립트들은 루트 파일 시스템을 읽고 쓰기 모드로 마운트하고, 스왑을 활성화하며 몇몇 시스템 설정을 하고 우리 시스템이 필요한 다양한 데몬들을 시작하는 역할을 한다.

8.1 디렉토리와 마스터 파일의 준비

이 장에서 여러분은 다시 Sysvinit 패키지가 필요하다.

- 아래를 실행하여 필요한 디렉토리들을 생성한다:

```
cd /etc
mkdir rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d init.d rcS.d
```

- 풀어놓은 Sysvinit의 소스 디렉토리로 이동한다.
- `Debian/etc/init.d/rc` 파일을 `/etc/init.d`로 복사한다.
- /etc/init.d 디렉토리로 이동한다.
- 아래를 포함하도록 `rcS` 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/rcS

runlevel=S
prevlevel=N
umask 022
export runlevel prevlevel

trap ":" INT QUIT TSTP

for i in /etc/rcS.d/S??*
do
   [ ! -f  "$i" ] && continue;
   $i start
```

```
done

# End /etc/init.d/rcS
```

8.2 reboot 스크립트 만들기

- 아래를 포함하도록 reboot 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/reboot

echo -n "System reboot in progress..."

/sbin/reboot -d -f -i

# End /etc/init.d/reboot
```

8.3 halt 스크립트 생성

- 아래를 포함하도록 halt 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/halt

/sbin/halt -d -f -i -p

# End /etc/init.d/halt
```

8.4 mountfs 스크립트의 생성

- 아래를 포함하도록 mountfs 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/mountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Remounting root file system in read-write mode..."
```

```
/bin/mount -n -o remount,rw /
check_status

> /etc/mtab
/bin/mount -f -o remount,rw /

echo -n "Mounting proc file system..."
/bin/mount proc
check_status

# End /etc/init.d/mountfs
```

8.5 umountfs 스크립트의 생성

- 아래를 포함하도록 umountfs 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/umountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo "Deactivating swap..."
/bin/swapoff -av
check_status

echo -n "Unmounting file systems..."
/bin/umount -a -r
check_status

# End /etc/init.d/umountfs
```

8.6 sendsignals 스크립트의 생성

- 아래를 포함하도록 sendsignals 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/sendsignals

check_status()
{
```

```
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}
echo -n "Sending all processes the TERM signal..."
/sbin/killall5 -15
check_status

echo -n "Sending all processes the KILL signal..."
/sbin/killall5 -9
check_status
```

8.7 checkroot 부트 스크립트의 생성

- 아래를 포함하도록 /etc/init.d/checkroot 파일을 만든다:

```
#!/bin/sh
# Begin /etc/init.d/checkroot

echo "Activating swap..."
/sbin/swapon -av

if [ -f /fastboot ]
then
  echo "Fast boot, no file system check"
else
  /bin/mount -n -o remount,ro /
  if [ $? = 0 ]
  then
    if [ -f /forcecheck ]
    then
      force="-f"
    else
      force=""
    fi

    echo "Checking root file system..."
    /sbin/fsck $force -a /

    if [ $? -gt 1 ]
    then
      echo
      echo "fsck failed. Please repair your file system manually by"
      echo "running fsck without the -a option"

      echo "Please note that the file system is currently mounted in"
```

```
      echo "read-only mode."
      echo
      echo "I will start sulogin now. CTRL+D will reboot your system."
      /sbin/sulogin
      /reboot -f
    fi
  else
    echo "Cannot check root file system because it is not mounted in"
    echo "read-only mode."
  fi
fi

# End /etc/init.d/checkroot
```

8.8 Sysklogd 부트 스크립트의 생성

- 아래를 포함하도록 /etc/init.d/sysklogd 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/sysklogd

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo -n "Starting system log daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/syslogd -- -m 0
    check_status

    echo -n "Starting kernel log daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/klogd
    check_status
    ;;

  stop)
    echo -n "Stopping kernel log daemon..."
    start-stop-daemon -K -q -o -p  /var/run/klogd.pid
    check_status

    echo -n "Stopping system log daemon..."
    start-stop-daemon -K -q -o -p /var/run/syslogd.pid
```

```
    check_status
    ;;

  reload)
    echo -n "Reloading system load daemon configuration file..."
    start-stop-daemon -K -q -o -s 1 -p /var/run/syslogd.pid
    check_status
    ;;

  restart)
    echo -n "Stopping kernel log daemon..."
    start-stop-daemon -K -q -o -p /var/run/klogd.pid
    check_status

    echo -n "Stopping system log daemon..."
    start-stop-daemon -K -q -o -p /var/run/syslogd.pid
    check_status

    sleep 1

    echo -n "Starting system log daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/syslogd -- -m 0
    check_status

    echo -n "Starting kernel log daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/klogd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|reload|restart}"
    exit 1
    ;;
esac

# End /etc/init.d/sysklogd
```

8.9 심볼릭 링크와 권한 설정

- 아래를 실행하여 파일 권한과 심볼릭 링크들을 알맞게 설정한다:

```
chmod 755 rcS reboot halt mountfs umountfs sendsignals checkroot sysklogd
cd ../rc0.d
ln -s ../init.d/sysklogd K90sysklogd
ln -s ../init.d/sendsignals S80sendsignals
ln -s ../init.d/umountfs S90umountfs
ln -s ../init.d/halt S99halt
cd ../rc6.d
ln -s ../init.d/sysklogd K90sysklogd
```

```
ln -s ../init.d/sendsignals S80sendsignals
ln -s ../init.d/umountfs S90umountfs
ln -s ../init.d/reboot S99reboot
cd ../rcS.d
ln -s ../init.d/checkroot S05checkroot
ln -s ../init.d/mountfs S10mountfs
cd /etc/rc2.d
ln -s ../init.d/sysklogd S03sysklogd
```

8.10 /etc/fstab 파일의 생성

- 아래를 포함하도록 `/etc/fstab` 파일을 만든다:

```
/dev/<LFS-partition designation> / ext2 defaults 0 1
/dev/<swap-partition designation> none swap sw 0 0
proc /proc proc defaults 0 0
```

제 9 절 기본적인 네트워킹의 설정

9.1 Netkit-base 설치하기

- Netkit-base 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
cd etc.sample; cp services protocols /etc
mv /usr/bin/ping /bin
```

9.2 Net-tools의 설치

- Net-tools의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
mv /usr/bin/netstat /bin
cd /usr/sbin; mv ifconfig route /sbin
```

9.2.1 /etc/init.d/localnet 부트스크립트 만들기

- 아래를 포함하도록 `/etc/init.d/localnet` 파일을 새로 만든다.

```
#!/bin/sh
# Begin /etc/init.d/localnet

check_status()
```

```
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Setting up loopback device..."
/sbin/ifconfig lo 127.0.0.1
check_status

echo -n "Setting up hostname..."
/bin/hostname --file /etc/hostname
check_status

# End /etc/init.d/localnet
```

9.2.2 권한과 심볼릭 링크의 설정

- chmod 755 /etc/init.d/localnet 을 실행하여 권한을 알맞게 설정한다.
- cd /etc/rcS.d; ln -s ../init.d/localnet S03localnet 을 실행하여 적당한 심볼릭 링크를 생성한다.

9.2.3 /etc/hostname 파일 만들기

/etc/hostname 파일을 새로 만들어 호스트 이름을 그 안에 집어넣는다. 이 리음은 FQDN(Fully Qualified Domain Name)은 아니다. 여러분이 네트웍 상에서 여러분의 컴퓨터를 부르고 싶을 때 쓸 이름이다.

9.2.4 /etc/hosts 파일의 생성

만약 네트웍 카드를 설정하고 싶다면 /etc/hosts 파일 안에서 쓸 IP 주소와 FQDN, 가능하다면 알리아스 들을 결정해야 한다. 아래는 한 예제이다:

```
<myip> myhost.mydomain.org aliases
```

IP 주소가 사설 망의 IP 주소 범위 안에 있도록 해라. 가능한 값은 다음과 같다:

```
Class Networks
A     10.0.0.0
B     172.16.0.0 through 172.31.0.0
C     192.168.0.0 through 192.168.255.0
```

가능한 IP 주소는 192.168.1.1일 수 있다. 이 IP에 대한 FQDN은 me.linuxfromscratch.org일 수 있다.

만약 여러분이 네트워 카드를 사용하지 않을 예정일 지라도 여전히 FQDN이 필요하다. 이는 Sendmail 같은 프로그램들이 정상적으로 작동하는 데 필요하다 (실제로 Sendmail은 FQDN을 결정하지 못하면 작동하지 않는다).

아래는 여러분이 네트워 카드를 설정하지 않는 경우의 /etc/hosts 파일이다:

```
# Begin /etc/hosts (no network card version)
127.0.0.1 me.lfs.org <contents of /etc/hostname> localhost
# End /etc/hosts (no network card version)
```

아래는 네트워 카드를 설정하는 경우의 /etc/hosts 파일이다:

```
# Begin /etc/hosts (network card version)
127.0.0.1 localhost
192.168.1.1 me.lfs.org <contents of /etc/hostname>
# End /etc/hosts (network card version)
```

물론, 192.168.1.1과 me.lfs.org을 여러분의 기호에 맞게(혹은 여러분이 네트워/시스템 관리자로부터 IP 주소를 할당받았고 이 머신을 그 네트워에 연결할 계획이라면 거기에 맞게) 바꾼다.

9.2.5 /etc/init.d/ethnet 파일의 생성

이 부분은 여러분이 네트워 카드를 설정하는 경우에만 해당된다. 그렇지 않다면 이 부분을 뛰어넘어도 된다.

아래를 포함하도록 /etc/init.d/ethnet 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/ethnet

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

/sbin/ifconfig eth0 <ipaddress>
check_status

# End /etc/init.d/ethnet
```

9.2.6 /etc/init.d/ethnet의 권한과 심볼릭 링크를 설정한다.

- `chmod 755 ethnet` 을 실행하여 알맞게 권한 설정을 한다.
- `cd ../rc2.d; ln -s ../init.d/ethnet S10ethnet` 을 실행하여 심볼릭 링크를 만든다.

9.2.7 네트웍 설정의 시험

- `/etc/init.d/localnet`을 실행하여 방금 만든 localnet 스크립트를 시작시킨다.
- ethnet 스크립트를 만들었다면 `/etc/init.d/ethnet`을 실행하여 이를 시작시킨다.
- 아래를 실행하여 /etc/hosts가 제대로 설정 되었는지 검사한다:

```
ping <your FQDN>
ping <what you choose for hostname>
ping localhost
ping 127.0.0.1
ping 192.168.1.1 (only when you configured your network card)
```

이 다섯개의 ping 명령들이 모두 실패 없이 작동해야 한다. 그렇다면 기본적인 네트웍이 잘 작동하는 것이다.

9.3 시스템 검사

이제 모든 소프트웨어가 설치되었고 부트스크립트들이 만들어졌으며 지역 네트웍도 설정되었으므로 여러분의 컴퓨터를 리부팅하여 이 새 스크립트들이 실제로 작동하는지를 보기 위해 검사를 해 볼 시간이다. 우선은 /etc/init.d 디렉토리에서 직접 실행하여 가장 흔한 문제들 (오타나 잘못된 경로 등)을 고치길 바랄 것이다. 이 스크립트들이 직접 실행시켰을 때 잘 작동한다면 시스템의 시작과 종료 시에도 정상적으로 작동해야 한다. 이를 시험하는 방법은 오직 하나이다. 지금 shutdown -r 명령으로 시스템을 종료한 후 LFS로 재부팅 하여라. 재부팅 이후에 여러분은 기존의 리눅스 시스템에서와 같은 보통의 로긴 프롬프트를 보게 될 것이다 (여러분이 XDM이나 기타 다른 Display Manager (XDM의 KDE 판인 KDE 같은)를 사용하지 않는다면).

이 시점에 여러분의 기본적인 LFS 시스템은 사용할 준비가 되었다. 지금부터 나올 것들은 모두 선택사항이며 여러분의 판단에 따라 패키지들을 뛰어넘을 수 있다. 그러나 여러분이 패키지들의 설치를 뛰어넘는다면 (특히 라이브러리들) 다른 패키지들의 의존성을 해칠 수 있다는 것을 명심해야 한다.

제 10 절 네트웍 데몬의 설치

10.1 SMTP 설정하기

10.1.1 group과 user의 생성

아래를 실행하여 Sendmail이 필요로 하는 group들을 만든다:

```
groupadd -g 1 bin
groupadd -g 2 kmem
groupadd -g 3 mail
useradd -u 1 -g bin -d /bin -s /bin/sh bin
```

10.1.2 디렉토리 생성

Sendmail은 밖으로 나가는 메일을 /var/spool/mqueue 디렉토리에 넣어둔다. 들어오는 메일은 Sendmail에 의해 Procmail로 돌려지므로 우리는 /var/mail 이라는 디렉토리도 필요하다. 이 디렉토리들을 생성하고 알맞은 권한을 설정한다:

```
mkdir /var/spool
mkdir /var/mail
cd /var/spool; ln -s ../mail mail
chmod 700 /var/spool/mqueue
chmod 775 /var/mail
chgrp mail /var/mail
chmod 1777 /tmp
```

10.1.3 Sendmail의 설치

- Sendmail 압축파일을 풀고 아래를 실행하여 설치한다:

```
cd src
./Build; ./Build install
```

10.1.4 Sendmail의 설정

Sendmail을 설정하는 것은 앞의 것들처럼 쉽지는 않다. Sendmail을 설정하는 동안 여러분이 고려해야 할 사항이 많으며 내가 그 모든 것을 설명할 수는 없다. 이것이 우리가 지금은 기본적이고 표준적인 설정을 하는 이유이다. Sendmail을 여러분의 기호에 맞도록 고치고 싶다면 할 수 있지만 이 문서는 그 작업에 적합하지는 않다. 여러분이 특정한 기능을 사용하려 한다면 이미 존재하는 /etc/sendmail.cf (혹은 /etc/mail/sendmail.cf) 파일을 사용할 수 있다.

- Go to the cf directory cf 디렉토리로 이동한다.
- 아래를 포함하도록 `cf/lfs.mc` 파일을 새로 만든다:

```
OSTYPE(LFS)
FEATURE(nouucp)
define(`LOCAL_MAILER_PATH', /usr/bin/procmail)
MAILER(local)
MAILER(smtp)
```

- `touch ostype/LFS.m4` 를 실행하여 빈 `ostype/LFS.m4` 파일을 만든다.
- `m4 m4/cf.m4 cf/lfs.mc > cf/lfs.cf` 를 실행하여 lfs.mc 파일을 컴파일한다.
- cf/lfs.cf 를 `/etc/sendmail.cf` 로 복사한다.
- `touch /etc/aliases` 를 실행하여 빈 /etc/aliases 파일을 만든다.
- `sendmail -v -bi` 를 실행하여 이 (빈) alias 데이타베이스를 초기화한다.

10.1.5 Procmail의 설치

- Procmail 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install; make install-suid
```

10.1.6 /etc/init.d/sendmail 부트스크립트의 작성

- 아래를 포함하도록 /etc/init.d/sendmail 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/sendmail

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo -n "Starting Sendmail..."
    start-stop-daemon -S -q -o -x /usr/sbin/sendmail -- -bd
    check_status
    ;;

  stop)
    echo -n "Stopping Sendmail..."
    start-stop-daemon -K -q -o -p /var/run/sendmail.pid
    check_status
    ;;

  reload)
    echo -n "Reloading Sendmail configuration file..."
    start-stop-daemon -K -q -s 1 -p /var/run/sendmail.pid
    check_status
    ;;

  restart)
    echo -n "Stopping Sendmail..."
    start-stop-daemon -K -q -o -p /var/run/sendmail.pid
    check_status

    sleep 1

    echo -n "Starting Sendmail..."
    start-stop-daemon -S -q -o -x /usr/sbin/sendmail -- -bd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|reload|restart}"
    exit 1
```

```
    ;;

esac

# End /etc/init.d/sendmail
```

10.1.7 권한과 심볼릭 링크들을 설정한다.

- `chmod 755 /etc/init.d/sendmail` 을 실행하여 알맞은 권한을 설정한다.
- 아래를 실행하여 알맞은 심볼릭 링크들을 만든다:

```
cd /etc/init.d/rc2.d; ln -s ../init.d/sendmail S20sendmail
cd ../rc0.d; ln -s ../init.d/sendmail K20sendmail
cd ../rc6.d; ln -s ../init.d/sendmail K20sendmail
```

10.2 FTP의 설정

10.2.1 user와 group의 생성

- 아래를 실행하여 필요한 group들을 만든다:

```
groupadd -g 65534 nogroup
groupadd -g 4 ftp
```

- 아래를 실행하여 필요한 user들을 만든다:

```
useradd -u 65534 -g nogroup -d /home nobody
useradd -u 4 -g ftp -s /bin/sh -m ftp
```

10.2.2 Proftpd의 설치

- Proftpd의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

10.2.3 /etc/init.d/proftpd 부트스크립트의 작성

- 아래를 포함하도록 `/etc/init.d/proftpd` 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/proftpd

check_status()
```

```
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo -n "Starting Pro FTP daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/proftpd
    check_status
    ;;

  stop)
    echo -n "Stopping Pro FTP daemon..."
    start-stop-daemon -K -q -o -x /usr/sbin/proftpd
    check_status
    ;;

  restart)
    echo -n "Stopping Pro FTP daemon..."
    start-stop-daemon -K -q -o -x /usr/sbin/proftpd
    check_status

    sleep 1

    echo -n "Starting Pro FTP daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/proftpd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|restart}"
    ;;

esac

# End /etc/init.d/proftpd
```

10.2.4 권한과 심볼릭 링크의 설정

- `chmod 755 /etc/init.d/proftpd` 를 실행하여 알맞은 권한을 설정한다.
- 아래를 실행하여 필요한 심볼릭 링크들을 만든다:

```
cd /etc/rc2.d; ln -s ../init.d/proftpd S40proftpd
cd ../rc0.d; ln -s ../init.d/proftpd K40proftpd
```

```
cd ../rc6.d; ln -s ../init.d/proftpd K40proftpd
```

10.3 HTTP의 설정

10.3.1 Apache의 설치

- Apache의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

10.3.2 Apache의 설정

설정할 것이 많지는 않다. 우리가 해야 할 것은 단지 /usr/apache/man 경로를 /usr/share/misc/man.conf에 추가하는 것 뿐이다.

- /usr/share/misc/man.conf 파일을 수정한다.
- MANPATH로 시작하는 다른 줄 아래에 *MANPATH /usr/apache/man* 이라는 줄을 추가한다.

10.3.3 /etc/init.d/apache 부트스크립트의 생성

- 아래를 포함하도록 `/etc/init.d/apache` 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/apache

case "$1" in
  start)
    echo -n "Starting Apache HTTP daemon..."
    /usr/apache/bin/apachectl start
    ;;

  stop)
    echo -n "Stopping Apache HTTP daemon..."
    /usr/apache/bin/apachectl stop
    ;;

  restart)
    echo -n "Restarting Apache HTTP daemon..."
    /usr/apache/bin/apachectl restart
    ;;

  force-restart)
    echo -n "Stopping Apache HTTP daemon..."
    /usr/apache/bin/apachectl stop

    sleep 1
```

```
    echo -n "Starting Apache HTTP daemon..."
    /usr/apache/bin/apachectl start
    ;;

  *)
    echo "Usage: $0 {start|stop|restart|force-restart}"
    ;;

esac

# End /etc/init.d/apache
```

10.3.4 권한과 심볼릭 링크의 설정

- `chmod 755 /etc/init.d/apache` 를 실행하여 알맞은 권한을 설정한다.
- 아래를 실행하여 필요한 심볼릭 링크들을 만든다:

```
cd /etc/rc2.d; ln -s ../init.d/apache S50apache
cd ../rc0.d; ln -s ../init.d/apache K50apache
cd ../rc6.d; ln -s ../init.d/apache K50apache
```

10.4 Telnet의 설정

10.4.1 telnet 데몬과 클라이언트의 설치

- Netkit-telnet의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

10.4.2 /etc/inetd.conf 설정파일을 만든다.

- 아래를 포함하도록 `/etc/inetd.conf` 파일을 새로 만든다:

```
# Begin /etc/inetd.conf

telnet stream tcp nowait root /usr/sbin/in.telnetd

# End /etc/inetd.conf
```

10.4.3 /etc/init.d/inetd 부트스크립트의 생성

- 아래를 포함하도록 `/etc/init.d/inetd` 파일을 새로 만든다:

```
#!/bin/sh
# Begin /etc/init.d/inetd

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo -n "Starting Internet Server daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/inetd
    check_status
    ;;

  stop)
    echo -n "Stopping Internet Server daemon..."
    start-stop-daemon -K -q -o -p /var/run/inetd.pid
    check_status
    ;;

  reload)
    echo -n "Reloading Internet Server configuration file..."
    start-stop-daemon -K -q -s 1 -p /var/run/inetd.pid
    check_status
    ;;

  restart)
    echo -n "Stopping Internet Server daemon..."
    start-stop-daemon -K -q -o -p /var/run/inetd.pid
    check_status

    sleep 1

    echo -n "Starting Internet Server daemon..."
    start-stop-daemon -S -q -o -x /usr/sbin/inetd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|reload|restart}"
    ;;

esac
```

```
# End /etc/init.d/inetd
```

10.4.4 권한과 심볼릭 링크의 설정

- `chmod 755 /etc/init.d/inetd` 를 실행하여 알맞은 권한을 설정한다.
- 아래를 실행하여 필요한 심볼릭 링크들을 만든다:

```
cd /etc/rc2.d; ln -s ../init.d/inetd S30inetd
cd ../rc0.d; ln -s ../init.d/inetd K30inetd
cd ../rc6.d; ln -s ../init.d/inetd K30 inetd
```

10.5 PPP의 설정

10.5.1 커널 설정하기

여러분이 인터넷에 연결되기 전에 커널이 ppp를 지원해야 한다. 이는 커널 안에 ppp 지원을 직접 넣어서 컴파일하거나 ppp 드라이버를 필요할 때 로드할 수 있는 모듈로 만듦으로써 가능하다. 어떤 방법은 선택하든 필요하다면 지금 커널을 재 컴파일하여 작업을 하라. 이미 여러분의 LFS 커널이 ppp를 지원한다면 커널을 재 설정할 필은 없다.

10.5.2 group의 생성

- `groupadd -g7 daemon` 을 실행하여 damon group을 만든다.

10.5.3 PPP의 설치

- PPP의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

10.5.4 /etc/resolv.conf 생성

- 아래를 포함하도록 `/etc/resolv.conf` 파일을 새로 만든다:

```
# Begin /etc/resolv.conf

nameserver <IP address of your ISP's primary DNS server>
nameserver <IP address of your ISP's secundary DNS server>

# End /etc/resolv.conf
```

10.5.5 /etc/ppp/peers/provider 파일 만들기

- /etc/ppp/peers 디렉토리를 만든다.
- 아래를 포함하도록 /etc/ppp/peers/provider 파일을 새로 만든다:

```
# Begin /etc/ppp/peers/provider

noauth
connect "/usr/sbin/chat -v -f /etc/chatscripts/provider"
/dev/ttyS1
115200
defaultroute
noipdefault

# End /etc/ppp/peers/provider
```

10.5.6 /etc/chatscripts/provider 만들기

- /etc/chatscripts 디렉토리를 만든다.
- 아래를 포함하도록 /etc/chatscripts/provider 파일을 새로 만든다:

```
# Begin /etc/chatscripts/provider

ABORT BUSY
ABORT "NO CARRIER"
ABORT VOICE
ABORT "NO DIALTONE"
ABORT "NO ANSWER"
"" ATZ
OK ATDT <ISP's phonenumber>
TIMEOUT 35
CONNECT ''
TIMEOUT 10
ogin: \q<username>
TIMEOUT 10
assword: \q<mysecretpassword>

# End /etc/chatscripts/provider
```

10.5.7 암호 인증에서 주의할 사항

예제 스크립트들에서 보듯이 (이것들은 내가 X를 쓰지 않을 때 실제로 사용하는 스크립트들이다) 위에서 나는 내 ISP로 pap이나 chap을 통한 인증 대신 chatscripts를 이용하여 접속하였다. 내 ISP가 pap를 지원하긴 하지만 나는 이 장/단점을 모두 가진 약간 다른 방식으로 하기로 결정하였다. 내 경우에 장점이 단점보다 많다. 이 방식으로 나는 내 접속 과정에 대한 더 많은 제어권을 가질 수 있고 언제 어떤 일이 일어나는지를 더 자세히 알 수 있었다.

예를 들어 내가 접속할 때 대부분 나는 `tail -f /var/log/syslog` 를 실행하는 윈도우를 열어 놓아서 사용자 명이나 암호가 보내지는 시기 같은 것들을 직접 관찰할 수 있다.

제 11 절 네트웍 클라이언트의 설치

11.1 Email 클라이언트 설치하기

11.1.1 Mailx의 설치

- Mailx의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make; make install
```

11.1.2 Mutt의 설치

내가 가장 선호하는 email 클라이언트가 Mutt이라서 우리는 Mutt을 설치할 것이다. Mutt이 아닌 여러분이 좋아하는 다른 클라이언트를 설치해도 무방하다.

만약 여러분이 쓰려는 클라이언트가 X 윈도우용이라면 (Netscape Mail 같은) 우리가 X를 설치할 때까지 가만히 앉아서 기다려야 한다.

- Mutt의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

11.1.3 Fetchmail의 설치

- Fetchmail의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

11.1.4 email 시스템의 검사

이제 email 시스템을 검사할 때이다.

- `/usr/sbin/sendmail -bd` 를 실행하여 Sendmail을 실행한다 (여러분은 완전한 경로를 사용하여 sendmail을 실행하여야 한다. 그렇지 않으면 kill -1 <sendmail pid> 을 실행하여 sendmail이 리로드 되도록 할 수 없다.
- `echo "this is an email test" | mail -s test root` 를 실행하여 여러분 자신에게 메일을 보낸다.
- `mail` 프로그램을 실행시켰을 때 여러분의 메일을 봐야 정상이다.
- `useradd -m testuser; passwd testuser` 를 실행하여 새 사용자를 만든다.
- `echo "test mail to testuser" | mail -s test testuser` 를 실행하여 testuser에게 메일을 보낸다.
- testuser로 로긴한 후 (메일 프로그램을 이용하여) 이메일이 왔는지를 검사하고 testuser에게 메일을 보냈던 것과 같은 방법으로 root에게 메일을 보낸다.

모든게 잘 동작했다면 여러분은 로컬 이메일에 대해선 정상적으로 작동하는 이메일 시스템을 가지게 된 것이다. 아직은 인터넷에도 쓸 수 있을 필요 없다. userdel -r testuser 를 실행하여 testuser를 지울 수 있다.

11.2 FTP 클라이언트의 설치

11.2.1 Netkit-ftp의 설치

- Netkit-ftp의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

11.2.2 FTP 시스템의 검사

- `/etc/init.d/proftpd start` 를 실행하여 Pro ftp 데몬을 시작시킨다.
- `ftp localhost` 를 실행하여 로컬 호스트로의 ftp 연결을 연다.
- anonymous 사용자로 로긴한 후 로그아웃한다.

11.3 HTTP 클라이언트의 설치

11.3.1 Zlib의 설치

Zlib는 PKware의 zip이나 unzip 같은 프로그램들에서 사용되는 압축 라이브러리이다. 특정 파일들을 압축하기 위해서 Lynx가 이 라이브러리를 쓸 수 있다.

- Zlib의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure --shared
make; make install
```

11.3.2 Lynx의 설치

- Lynx의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure --libdir=/etc --with-zlib
make; make install
make install-help; make install-doc
```

11.3.3 HTTP 시스템의 검사

- `/etc/init.d/apache start` 를 실행하여 Apache http 데몬을 시작시킨다.
- `lynx http://localhost` 를 실행하여 로컬 호스트로의 http 연결을 연다.
- lynx를 끝낸다.

11.4 텔넷 클라이언트의 설치

앞 장에서 텔넷 데몬을 설치할 때 이미 텔넷 클라이언트가 깔렸다.

11.4.1 텔넷 시스템의 검사

- /etc/init.d/inetd start를 실행하여 인터넷 서버 데몬을 실행한다 (telnetd를 포함하여).
- telnet localhost 를 실행하여 로컬 호스트로의 텔넷 연결을 연다.
- 로긴한 후 호그아웃한다.

11.5 PPP 클라이언트의 설치

11.5.1 연결 스크립트 작성하기

- 아래를 포함하도록 /usr/bin/pan 파일을 새로 만든다.

```
#!/bin/sh
# Begin /usr/bin/pon

/usr/sbin/pppd call provider

# End /usr/bin/pon
```

11.5.2 연결 종료 스크립트 작성하기

- 아래를 포함하도록 /usr/bin/poff 파일을 새로 만든다.

```
#!/bin/sh
# Begin /usr/bin/poff

set -- `cat /var/run/ppp*.pid`

case $# in
  0)
    kill -15 `ps axw|grep "pppd call [[allnum:]]+"|grep -v grep|awk '{print $1}'`
    exit 0
    ;;
  1)
    kill -15 $1
    exit 0
    ;;
esac

# End /usr/bin/poff
```

11.5.3 PPP 시스템의 검사

- `pan`을 실행하여 인터넷에 연결한다.
- http://www.linuxfromscratch.org 같은 사이트에 접속해 본다.
- `poff`를 실행하여 인터넷으로의 연결을 끊는다.

제 12 절 X 윈도우 시스템의 설치

12.1 X의 설치

- X의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make World
make install; make install.man
```

컴파일 과정동안 여러분은 "makedepend" 스크립트가 stddef.h와 stdarg.h, float.h 해더 파일을 찾을 수 없다는 에러를 만날 지도 모른다. 컴파일 자체는 컴파일 에러 없이 잘 되므로 그 스크립트가 컴파일러만큼 똑똑하지 않은 것이다. 더군다나, 일부 심볼릭 링크들을 만드는 것이 문제를 해결하진 않는다. 다만 더 많은 문제를 일으킬 뿐이다.

따라서 여러분이 만나게 될 많은 makedepend 에러들을 그냥 무시하는게 좋다. "pointer targets in passing arg x of somefunction differ in signedness" 같은 에러도 있다. 그 파일들을 원한다면 수정할 수도 있지만 나는 그러지 않았다.

12.2 /etc/ld.so.conf 만들기

아래를 포함하도록 `/etc/ld.so.conf` 파일을 새로 만든다:

```
# Begin /etc/ld.so.conf

/lib
/usr/lib
/usr/X11R6/lib

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

- `ldconfig`을 실행하여 동적 로더 캐쉬를 갱신한다.

12.3 /usr/include/X11 심볼릭 링크의 생성

- 전처리기가 X11/*.h 파일들을 찾을 수 있도록 (소스 코드의 #include 문장에서 볼 수 있다) 다음처럼 심볼릭 링크를 만든다: `ln -s /usr/X11R6/include/X11 /usr/include/X11`

12.4 Creating the /usr/X11 symlink /usr/X11 심볼릭 링크의 생성

종종 소프트웨어가 파일들을 /usr/X11로 복사하기 때문에 여러분이 X의 어떤 릴리즈를 쓰고 있는지 알 필요가 없다. 이 심볼릭 링크는 X의 설치 과정에서 생성되지 않으므로 우리가 직접 만들어야 한다.

- `ln -s /usr/X11R6 /usr/X11` 를 실행하여 /usr/X11 심볼릭 링크를 만든다.

12.5 /usr/X11/bin을 $PATH 환경 변수에 추가하기

/usr/X11/bin 경로를 $PATH 환경 변수에 추가하는 방법엔 몇 가지가 있다. 그 중 하나는 아래와 같다:

- *export PATH=$PATH:/usr/X11/bin* 을 포함하도록 `/root/.bashrc` 파일을 새로 만든다.

수정 사항을 적용하기 위해선 다시 로긴해야 한다. 또는 직접 `export PATH=$PATH:/usr/X11/bin` 을 실행하여 경로를 갱신할 수도 있다.

12.6 X의 설정

- `xf86config`을 실행하여 X 서버를 설정한다.

xf86config에 의해 만들어지는 XF86Config 파일이 만족스럽지 못하다면 기존의 리눅스 시스템에서 쓰고 있던 XF86Config 파일을 /etc로 복사하는 것도 좋은 방법이다. xf86config 프로그램이 지원하지 않는 특별한 수정 사항이 필요한 경우엔 이렇게 해야 할 것이다. 또한 직접 XF86Config 파일을 수정할 수도 있다. 이 방법은 여러분이 수정해야 할 것을 정확히 기억하지 못한다면 매우 시간 소모적인 작업이 될 것이다.

12.7 X의 테스트

이제 X가 알맞게 설정이 되어서 처음으로 시험을 해 볼 시점이다.

- `startx`를 실행하여 X 서버를 시작시킨다.

X 서버가 시작되고 세 개의 xterm이 스크린에 떠야 한다. 만약 이렇다면 X가 잘 도는 것이다.

12.8 Window Maker의 설치

나는 윈도우 메니저로 Window Maker를 설치하기로 결정했다. 이는 내가 꽤 오랫동안 WindowMaker를 사용해 왔고 이에 매우 만족하고 있기 때문이다. 역시나 여러분은 내가 하는 데로 똑같이 할 필요흔 없다. 여러분이 원하는 것을 설치해도 된다. 알다시피 여러분은 여러 개의 윈도우 메니저들을 동시에 설치하고 $HOME/.xinitrc (혹은 xdm을 쓰는 경운 $HOME/.xsession) 파일 안에서 지정을 하여 어떤 것을 실행할지 선택할 수 있다.

12.9 Window Maker 설치를 위한 시스템의 준비

12.9.1 libPropList의 설치

- libPropList의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

12.9.2 libXpm의 설치

- libXpm의 압축파일을 풀고 아래를 실행하여 설치한다:

```
xmkmf; make Makefiles; make includes; make depend
cd lib; make; make install
cd ..; make; make install
```

Makefile들 중 하나의 버그때문에 약간 다른 설치 방법이 필요하다. 이 패키지의 설치는 아직 설치되지 않은 lib 디렉토리의 파일들에 의존하는데 lib 디렉토리 안에서 그 파일들을 찾지 않는다. 따라서 우리는 그 파일들을 실제 패키지의 컴파일 이전에 먼저 설치해야 한다.

12.9.3 libpng의 설치

- libpng의 압축파일을 풀고 아래를 실행하여 설치한다:

```
make -f scripts/makefile.lnx; make -f scripts/makefile.lnx install
```

12.9.4 libtiff의 설치

- libtiff의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

12.9.5 libjpeg의 설치

- libjepg의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure --enable-shared --enable-static
make; make install
```

12.9.6 libungif의 설치

- libungif의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

12.9.7 WindowMaker의 설치

- WindowMaker의 압축파일을 풀고 아래를 실행하여 설치한다:

```
./configure
make; make install
```

12.10 동적 로더 캐쉬의 갱신

- `ldconfig`을 실행하여 동적 로더 캐쉬를 갱신한다.

12.11 WindowMaker의 설정

WindowMaker를 쓰려는 사용자는 실제 사용 전에 wmaker.inst 스크립트를 실행해야 한다. 이 스크립트는 필요한 파일들을 사용자의 홈 디렉토리에 복사한 후 $HOME/.xinitrc 파일을 수정한다 (만약 그 파일이 없다면 생성한다).

- `wmaker.inst`를 실행하여 직접 WindowMaker를 설정한다.

12.12 WindowMaker의 검사

- `startx`를 실행하여 X 서버를 시작한 후 WindowMaker 윈도우 메니저가 제대로 시작하는지 확인한다.

제 13 절 자료들

여러분이 다운로드 받거나 살만한 유용한 책, 하우투, 여타 다른 문서들의 목록이 아래에 있다. 이 목록은 시작하기 위한 일부분이다. 우리가 더 유용한 문서나 책들을 알게 되어 이 목록이 늘어나기를 희망한다.

13.1 책

- Sendmail published by O'Reilly. ISBN: 1-56592-222-0
- Linux Network Administrator's Guide published by O'Reilly. ISBN: 1-56592-087-2
- Running Linux published by O'Reilly. ISBN: 1-56592-151-8

13.2 하우투 문서들

- ISP-Hookup-HOWTO at `http://ww.linuxdoc.org`
- Linux Network Administrator's Guide online at `http://www.linuxdoc.org`

13.3 기타

- 다양한 메뉴얼과 info 페이지들이 패키지들에 딸려있다.

제 14 절 맺음말

여러분은 Linux From Scratch HOWTO의 끝에 도달했다. 나는 이 경험이 여러분이 리눅스를 더 잘 알게 되는 데에 도움이 되었기를 바란다. 여러분이 이 문서에서 다루어져야 한다고 생각하는 것 (오류 수정이나 여러분이 중요하다 생각하지만 이 문서에 빠져있는 기타 소프트웨어들) 이 있다면 우리에게 알려달라. 여러분의 도움, 제안과 함께 이 HWOTO는 더욱 개선될 수 있다.

역자주: 조만간 v2.4가 release될 것 같습니다. 5월 4일자로(5일이던가? –;) v2.3.2가 release 되었는데 상당히 stable 하다고 합니다. v2.2의 bug들도 상당량 fix되었다는군요. v2.4 부터는 그대로 써먹을 수 있을 정도로

안정적이 되었으면 좋겠군요:) 더불어... v2.3 부터 LinuxDoc DTD가 아닌 DocBook DTD를 사용하는지라 번역 작업을 어찌 해야할 지도 난감..:(

제 15 절 Copyright & Licensing Information

의미를 그대로 보존하고 효력 상실을 막기 위해 Copyright와 License 부분은 번역하지 않습니다.