Building a Secure Linux Server from Scratch

Bruce Dubbs

Building a Secure Linux Server from Scratch
by Bruce Dubbs

Abstract

This paper provides a step-by-step guide to creating a secure Domain Name System (DNS) server. It starts with a minimal Linux distribution and constructs all of the system programs, applications, scripts, and supporting files, including the Linux kernel, from source code. It introduces and guides the reader through additions to the system including networking, logging, security applications, and the server software itself. Finally, a discussion of testing and system maintenance is presented. The techniques presented in this paper can be generalized to support a variety of secure servers for an organization.

Table of Contents

Acknowledgements

I would like to thank several people for making this paper possible. First I would like to thank Gerard Beekmans and the Linux From Scratch Team for creating a method of taking Open Source Software and showing how to put it together into an integrated whole. Secondly, I would like to thank Steve Kolars, Nate Durr, Othniel Graichen, and Bill Maltby for reviewing this paper and making many excellent suggestions for improvement. Finally, I would like to thank my wife, Joanna, for putting up with me working many late nights while I built and tested software and wrote this paper.

Preface

When securing a server, most administrators start with a commercial distribution and try to modify the configuration to eliminate security problem areas. The problem is that most distributions have many packages installed that are unnecessary on a server. For instance, the RedHat 9 distribution loads a minimum of 115 packages. Knowing what these packages are and the security implementations of each is very difficult.

This paper takes a different approach. It starts by building a base system from "scratch" using the techniques from the *Linux From Scratch* [Beek] project. To that base, the administration and security tools required to manage the system are added. Finally, the server applications are installed.

After the system is built however, configuration is not finished. Even though a small number of packages have been installed, some files need to be removed for security reasons. After that, final configuration tasks are required before deploying the server.

When deployed, the job of maintaining security is never complete. The administrator must continue to be vigilant and enter an ongoing cycle of security tasks. This cycle consists of four phases: Planning, Implementing, Monitoring, and Analyzing security and performance issues for the life of the system.

To demonstrate the principles described above, this paper will provide a step by step guide to implementing a Domain Name System (DNS) server for a medium size organization. From this description, an administrator can use most of the techniques described to build many types of servers by removing the DNS software and adding a few applications to the secured base system.

There are some prerequsites for developing this type of system. A moderate amount of UNIX system administrator skills including familiarity with building software from source distributions is needed. Beyond that, the only other skill needed is to be able to precisely follow instructions. In some cases, deviations made by a knowledgeable administrator are appropriate, however changes to the procedures given are not recommended for the first build.

Chapter 1. Introduction

When building a secure server, most experienced administrators say to remove any applications that are not needed. They also recommend that servers be dedicated to a single function to avoid leaving a system open to multiple types of attacks. In some cases, economics dictate that server functions be combined, but that is a tradeoff that needs to be evaluated depending on the individual circumstances of the organization deploying a server.

When building a new server, administrators usually start with a commercially produced distribution. The problem is that most general purpose Unix™ [1] and Linux™ [2] products are delivered with many more applications than are required for a dedicated server. For instance, a RedHat 9 Linux distribution installs a minimum of 115 packages. To that configuration, the specific server application and other support packages have to be added.

This paper will show how to build a Linux server completely from source code–from scratch. This approach has several advantages. First it provides great control over exactly what packages are installed in the system. The administrator decides every aspect of the system including executables, configuration files, and scripts. Second, the system produced is a very compact system. In this example, a complete, secure server is built that is less than 250 megabytes–without optimizing for size. Finally, and probably most importantly, building everything from source code provides security. The administrator has a completely auditable system. There are no 'special' distribution modifications. Fixes can be applied immediately without waiting for a vendor to integrate a patch into their system, produce a binary, and advertise the availability.

1.1. Overview

This paper starts by providing a detailed analysis of the hardware that is going to be used in Chapter 2. The example server that is being created is a Domain Name System (DNS) server. An analysis of the risks such a server must face is the subject of Chapter 3.

The first actual task will be to build a slightly modified Linux From Scratch [Beek] (LFS) system in Chapter 4. The purpose of a basic LFS system is to create a foundation from which all other applications, including a follow on LFS system, can be built.

Caution

Chapter 4 starts by downloading all the sources for the LFS system. It is an important security issue to ensure that our new system is not compromised during the build process. For that reason, the system should be disconnected from the Internet and all files imported from CD-ROM. This caution also applies to each subsequent chapter that requires additional source code.

Additionally, many of the package files presented have integrity mechanisms such as MD5 sums or gnupg signatures associated with them. These have been checked, when available, for the versions of software listed in this paper, however, each administrator should do this individually each time a package is downloaded from the Internet.

After the base system is installed, several sets of packages are added. Chapter 5 adds some administrative tools and initialization scripts to make things easier for the Administrator. Chapter 6 adds logging tools and other tools needed to manage the logs. Chapter 7 provides a set of security applications essential to running a secure server. The last thing added is the DNS server software itself in Chapter 8.

Note: Commands presented in the above chapters are quite specific for the configuration described. If modifying the instructions, several specifics such as devices, parameters, and in some cases, filenames will have to be changed.

Note: Although the system described in this paper has been built and is in current operation, all host names and IP addresses have been changed to generic values.

After the software is installed, Chapter 9 describes removing files that are used for software construction, but should not be left on a production server. Chapter 10 then presents tasks that must be done to properly test the system prior to placing it into production. Finally, Chapter 11 describes software maintenance procedures and the continuing process of planning$\longrightarrow$ implementation$\longrightarrow$ monitoring$\longrightarrow$ analyzing that must occur on every system to provide continuing effective and secure operation.

Notes

1. UNIX is a registered trademark of The Open Group.
2. Linux is a registered trademark of Linus Torvalds.

Chapter 2. Description of the System

2.1. Physical Environment

The first task when setting up a new server is to make a survey of the physical environment. It is important to determine what electrical power is required and what cooling is necessary. The physical network connectivity also needs to be determined.

In the case of our example system, the system will reside in a computer rack inside a locked server room with controlled access. The rack itself is capable of being locked and has an integral Uninterruptible Power Supply (UPS) with adequate capacity available. Cooling is via building facilities, however there is a dedicated air conditioning vent directed at the computer rack.

2.2. Computer Hardware

Determining the exact computer hardware configuration for a server is especially important when building your software from scratch. This information is needed for configuring the hardware drivers required for the operating system to interact with the hardware and to ensure we have an appropriate means of installing the software.

For this paper, we will use a minimally configured Dell PowerEdge 650 server in a 1U rack case. The details of the configuration are as follows:

- One 2.4 GHz Pentium Processor
- 512 MB RAM
- 40GB IDE Hard Disk – Seagate ST340014A
- IDE interface for Hard Disk – CMD680 Chipset
- Intel Pro/1000 Gigabit Ethernet Card (82546EB) with two ports
- CD-ROM
- IDE interface for CD-ROM – ServerWorks CSB6 Chipset
- Floppy Disk Drive – IDE interface
- ATI Rage XL Video Controller
- PS/2 Mouse Connector
- PS/2 Keyboard Connector
- USB Connectors
- Serial Port

The external devices: monitor, keyboard, and mouse, are provided via an 8-way Keyboard, Mouse, Video (KVM) switch in the computer rack.

Chapter 3. Risk Analysis

3.1. Threats

The server we are building is a Domain Name System (DNS) server that is designed to connect to the Internet. This server is a critical part of an overall network that provides network services to a department in an educational institution. The network provides electronic mail and web access to students, faculty, and staff. The web access includes distance learning resources for the entire department. The network also provides internal access to systems providing file and print services, web development, database development, and general application support to students and faculty.

When evaluating this environment, we see that potential security problems can arise from external threats or internal threats from different classes of users. The most hostile threat is the Internet where various sophisticated elements desire to break into systems for many reasons. On the other hand, the inside threat from students should not be minimized. The areas of instruction include various skills surrounding computer programming and administration, including network security skills. These developing skills and student knowledge of the systems being used can make the local systems an attractive target.

Factors that mitigate the threats also exist. There are no financial records or transactions on this network. Neither are there permanent student records nor faculty or staff personnel data.

3.2. Analysis

The overall threat to this network is moderate. There is no economic incentive to break into these systems, but because the systems have available storage and bandwidth they could have value to a cracker as an intermediate location to store files or launch other attacks. Additionally students trying to embarrass the institution or individual faculty members is a concern. Given these considerations the potential for highly skilled individuals to attempt penetration of these systems is high.

On the other hand, the nature of the data is not particularly volatle or sensitive and recovery from a compromise can be accomplished with a regular backup procedure. This makes the potential for long term damage relatively low.

In response to this judgment, we will configure our DNS server to take advantage of many of the security protections available today, but will not use some of the more exotic techniques such as using Security-Enhanced Linux [Secu] to provide mandatory access controls to the access data. For this situation the normal Unix permissions, in combination with the other security mechanisms available, provide adequate protection.

The above comments notwithstanding, we still need to provide for a defense in depth. To do that, we will provide for physical security, system security, and network security.

Physical security will be provided, as stated earlier, by placing the system in a controlled access room which prevents unauthorized physical access to the system, including electrical and network access.

Network security will be provided by placing the system in a logically partitioned subnetwork protected by a firewall. This subnetwork is normally referred to as a "demilitarized zone" or DMZ. Access to the server will go through the firewall for all services as depicted in Figure 1. As you can see, the servers in the DMZ have direct contact, through the firewall, with the high threat groups identified above. Our DNS system will provide name services to all users.

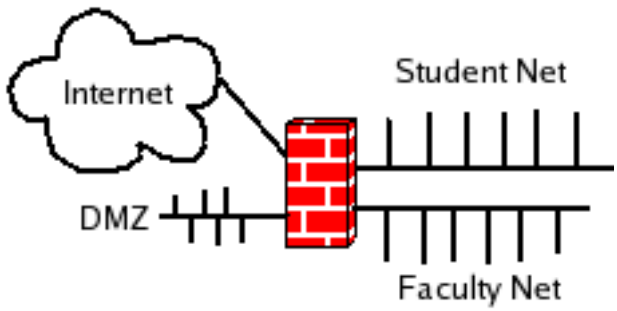

Figure 3-1. Placement of a DNS Server in a Demilitarized Zone

System security is addressed by the choice and configuration of applications on the system. This is the topic of the remainder of this paper.

Chapter 4. Building a Basic LFS System

4.1. Installing a Host System

In order to build a basic LFS system, we must start from an existing distribution. In this paper, we will use RedHat 9.0, but configure it to stay out of the way of our new system. We start by inserting the RedHat CD #1 into the system and booting. The system boots and asks several basic questions. The first significant activity is partitioning the hard disk.

4.1.1. Partitioning The hard Disk

The Dell PowerEdge 650 comes with a diagnostic partition. This partition is FAT and is located as /dev/hda1. For future maintenance, it is best to leave this partition in place. For security reasons, we will be mounting different partitions with different options. Therefore we will be setting up the partition table according to Table 4-1.

Table 4-1. Partition Table

Partition	Size	Contents
hda1	32MB	Dell Partition
hda2	100MB	/boot
hda3	500MB	swap
hda4		extended
hda5	2GB	/
hda6	500MB	/var
hda7	100MB	/var/named
hda8	500MB	/tmp
hda9	4GB	/home
hda10	4GB	/

All partitions except the Dell partition and the swap partition should be formatted as ext3.

During the initial RedHat installation, the system should be told to use /dev/hda10 for the root directory, /dev/hda2 for the /boot directory, and /dev/hda9 as the home directory. The others should be left empty. This is designed allow us to build our final system software in its final location.

> **Note:** The /dev/hda5 partition is much larger than is needed for the final system, but is set to 2GB to facilitate the build process.

4.1.2. Selecting Software from the Host Distribution

The next significant step is to select the proper software to install. It is not necessary to load a minimal set of packages, however the gcc compiler and support tools must be installed. A graphical user interface is optional and takes extra time to load. The specific packages needed are:

- binutils
- bzip2
- cpp
- ftp
- gcc
- gcc-c++
- glibc-devel
- glibc-kernheaders
- gnupg
- libstdc++
- libstdc++-devel
- make
- perl

After the software loads, reboot into the host distribution and log in as a regular user. The next step will be to download the source code for our basic LFS system.

4.2. Getting the Code

The source code for the packages needed to build the basic system is located at several locations on the Internet. To facilitate getting this code a `wget` script is located in Appendix A. You can download all the files by executing the command:

```
wget --input-file lfs-files
```

> **Note:** The wget script in the appendix differs from the LFS Book in a few ways. First, some packages are not downloaded because they are not needed on our server: ed, modutils, and gcc-2.95.3. Also one package has been upgraded to a more current version due to security fixes that have been released: linux-kernel-2.4.25.

Included in the code above are patches that have been created to fix various problems in the distributions. These address problems of interoperability with our

compiler and other tools or make minor changes to integrate the package properly with our system. An explanation of each patch is in the LFS book [Beek].

4.3. Building the Tools and Core Applications

4.3.1. Setting up the LFS Partition

The first step in building the system is to set up the target partition. We will do that with the following commands:

```
mkfs -t ext3 /dev/hda5
mkdir -p /mnt/lfs
mount /dev/hda5 /mnt/lfs
```

In the scripts that build the system, we use a specific environment variable, LFS, to designate the destination build directory. We specify this with:

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

To make this persistent, you may also want to put this command in the `/etc/profile` shell configuration file.

The general strategy of building an LFS system is to create a set of tools from our sources in a temporary directory within the LFS partition. We then enter a `chroot` environment to separate us from the host system and then build the final system with only our known software tools. To set this up, we create a directory in the new partition and a symbolic link from the host system's root directory to this tools directory.

```
mkdir -p $LFS/tools
ln -snf $LFS/tools /tools
```

4.3.2. Preparing to run the scripts

To avoid accidently corrupting our host system, we need to build our new system as a special non-privileged user. We will use the username lfs and assign ownership of the LFS partition to that user. We will also set up the environment especially for this user:

```
useradd -s /bin/bash -m lfs
passwd lfs
chown lfs $LFS/tools

cat > /home/lfs/.bash_profile << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:$PATH
```

```
PS1='[\u@\h \w]\\$'
export LFS LC_ALL PATH PS1
unset CC CXX CPP LD_LIBRARY_PATH LD_PRELOAD
EOF
```

At this point we need to get the scripts that are used to build the tools and core applications. These can be viewed in Appendix B, but they are available for download from http://www.linuxfromscratch.org/~bdubbs/lfs-scripts.tar.bz2.

The scripts make some assumptions about the location of subordinate scripts and the location of the source code. Place the uncompressed scripts in the lfs user's home directory and the compressed packages and patches in a subdirectory named `lfsfiles`:

```
cd /home/lfs
tar -jxvf lfs-scripts.tar.bz2
mkdir -p lfsfiles
mv {/download/location/*} lfsfiles
```

As mentioned earlier, building a secure LFS system requires some minor modifications to the standard build. We are not going to need `module utilities` or the `ed` packages. We will also be using the latest version of the Linux kernel available in the 2.4 series. As of this writing it is version 2.4.25. Finally, we are going to use `GCC 3.3.1` to compile the kernel.

4.3.3. Running the Basic LFS scripts

The main script must be run as *root*, because it runs the build of the final code in a `chroot` environment. Execute:

```
/home/lfs/make-LFS
```

The build takes a little over three hours on the hardware described in Chapter 2. The time will vary depending on processor and disk speed and memory available.

The commands used to build the system are explained in detail in the *Linux From Scratch* [Beek] book, but the scripts used here deviate from the book for security reasons as explained below.

4.3.3.1. Building the kernel

The last item of the build process is to make the kernel. This process is not particularly hard, but it does require a bit of care when selecting the compilation options. First, although modules are relatively secure, we are going to avoid any possible issue by creating a monolithic kernel and not use modules at all. We will also avoid compiling into the kernel any items not needed. This includes the RS-232 serial port and the USB connections. The default configuration should be selected except for:

- Loadable module support - off
- Processor type and features
  - Symmetric multi-processing support - off
- General setup
  - ISA Bus Support - off
  - Support for hot-pluggable devices - off
  - Kernel support for a.out binaries - off
  - Kernel support for MISC binaries - off
- Plug and Play configuration - off
- Networking options
  - Network packet filtering (replaces ipchains) - ON
  - IP: multicasting - off
  - IP: Netfilter Configuration
    - Connection tracking - on
    - IP Tables Support - All ON
- ATA/IDE/MFM/RLL
  - ServerWorks OSB4/CSB5/CSB6 chipsets support - ON
  - Silicon Image chipset support - ON
- SCSI support - off
- Network device support
  - Ethernet (1000 Mbit)
    - Intel(R) PRO/1000 Gigabit Ethernet support- ON
    - All others - off
- Character devices
  - Enhanced Real Time Clock Support - ON
  - /dev/agpgart (AGP Support) - off
- Multimedia devices - off
- File systems
  - Kernel automounter - off
  - Ext3 journaling file system support - ON
  - Network File Systems
    - NFS file system support - off
    - NFS server support - off
- Sound Support- off
- USB Support- off

Modifications that need to be made to this kernel configuration for different hardware are generally related to the hard disk and the network devices. Different IDE or SCSI drive interfaces and different network interfaces will require selection of the appropriate drivers for those devices. A different CPU or other main board components will also require different settings.

4.4. Making the New System Bootable

After the build is complete, check `/tools/src/build.log` for errors. This file is quite long (about 35 MB) and searches for the term "Error" or "***" will find any problems. In the build of the main system, the scripts run checks for many of the builds. You can look for these areas by searching for " ===". In a normal build, gcc will report three "unexpected failures", but this is normal.

After checking everything, you need to reenter the `chroot` environment with:

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

Set the root password with:

```
passwd
```

Leave the chroot environment with **`exit`**.

Now move kernel and System map, along with the grub boot files, from the LFS partition to the `/boot` partition:

```
mv $LFS/boot/linux* /boot
mv $LFS/boot/System.map* /boot
cp $LFS/boot/grub/* /boot/grub/
```

Edit `/boot/grub/grub.conf` to add an entry for our new kernel. Add the following lines:

```
title LFS 5.0
root (hd0,4)
kernel  /boot/linux-2.4.25-20040322 root=/dev/hda5
```

Change the filename of the kernel as appropriate to the date the kernel was built.

Now reenter the `chroot` environment as above and install the new `grub` files in the boot region of the hard disk.

At this point it is wise to create a `grub` boot disk. To do this, put a blank floppy into the disk drive and run the following commands:

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Now run the `grub` shell:

```
grub
```

At the grub prompt, enter:

```
grub> root (hd0,1)
```

```
grub> setup (hd0)
grub> exit
```

Finally, reboot to the LFS system.

4.5. Security Issues Addressed in the Build Scripts

The build scripts have several changes from the stock LFS system that address security. Most of these are in the `chapter6.sh` in Section B.2.1.

First of all the `/etc/passwd` and `/etc/group` files are very short. The `passwd` file has only one entry, root. The `group` file has only six entries and these are to satisfy group ownership of devices in the `/dev` directory.

Second, the devices that are created are only a minimal set needed for this server. There are no SCSI, loopback, ram disk, audio, parallel, or ppp devices. Also, the only IDE devices are `hda` for the hard disk and `hde` for the CD-ROM drive. If this system used different block devices, additional or different devices would have to be created.

Third, the `/etc/hosts` file only has one entry. This points to the local system name and localhost. It will be updated later.

Fourth, the `/etc/fstab` file has been created for the specific configuration of this system. Several partitions have the nodev and noexec options set.

Fifth, as mentioned earlier, the `ed` and the `modutils` packages are not built because they are not used in this system. The `gcc-2.95.3` compiler was not built because we chose to build the kernel with the more up to date `gcc-3.3.1` compiler.

In the `make6-sysvinit` build script Section B.2.46, we create the `/etc/inittab` file. Here we remove the options for entering run levels 2, 4, and 5 because they have no meaning for our system. We instead map run level 2 to run level 1 and run levels 4 and 5 to run level 3. We also reduce the number of virtual consoles to two. We elected to leave the Control-Alt-Delete function in the file because of the relatively secure physical environment of this system, but some administrators may want to remove that line too.

Chapter 5. Basic Administration Issues

5.1. Initialization Files

The base LFS system provides a very austere environment. To make administration of the system a bit more manageable, there are several things that we should do. Additionally, there are some security related issues that are best handled here.

The first item to consider is to create appropriate `bash` initialization scripts: `/etc/profile`, `/etc/bashrc`, `/root/.bash_profile`, `/root/.bashrc`, and `/root/.bash_logout`. The scripts found in *Beyond Linux From Scratch* [BLFS] provide a good starting point, however we will customize them explicitly for our system.

5.1.1. `/etc/profile`

```
# Start /etc/profile
# Setup some environment variables.
HISTSIZE=1000
HISTIGNORE="&:[bf]g:exit"
PS1="[\u@\h \w]\\$ "

# Setup the INPUTRC environment variable.
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ] ; then
    INPUTRC=/etc/inputrc
fi

# Setup for /bin/ls to support color, the alias is in /etc/bashrc.
if [ -f "/etc/dircolors" ] ; then
    eval $(dircolors -b /etc/dircolors)

    if [ -f "$HOME/.dircolors" ] ; then
        eval $(dircolors -b $HOME/.dircolors)
    fi
fi

export PATH HISTSIZE HISTIGNORE PS1 LS_COLORS INPUTRC

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

USER="‘id -un‘"
LOGNAME=$USER
MAIL="/var/mail/$USER"
HOSTNAME=‘/bin/hostname‘
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8

export USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC LANG

# Automatically log the user off if there is no input in 10 minutes
```

```
export TMOUT=600

# End /etc/profile
```

5.1.2. `/etc/dircolors`

We create the colorization initialization file by running the following command:

```
dircolors -p > /etc/dircolors
```

5.1.3. `/etc/bashrc`

```
# /etc/bashrc
umask 022
alias ls='ls --color=auto'
alias ll='ls -l'
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i'
```

5.1.4. Individual User `bash` Scripts

The individual `bash` initialization files can be made quite simple for our dedicated server:

5.1.4.1. `/root/.bash_profile`

```
# ~/.bash_profile
if [ -f "$HOME/.bashrc" ] ; then
    source $HOME/.bashrc
fi
```

5.1.4.2. `/root/.bashrc`

```
# ~/.bashrc
source /etc/bashrc
```

5.1.4.3. `/root/.bash_logout`

```
# ~/.bash_logout
# Clear the screen upon logout
clear
```

5.1.5. `/root/.vimrc`

We also want to set up some reasonable `vim` defaults for the administrators with the `/root/.vimrc` file.

```
" Begin ~/.vimrc
set nocompatible
set bs=2
set expandtab
set tabstop=4
set backup
set ruler
set laststatus=2
" End /root/.vimrc
```

5.1.6. The `/etc/skel` Directory

Since the root user's initialization scripts are generic, we should also put them in the `/etc/skel` directory so an account of a new administrator will also use them.

```
mkdir -p /etc/skel
cp /root/.* /etc/skel/
```

5.1.7. `/etc/inputrc`

The `/etc/inputrc` file is useful for effective command line editing.

```
set meta-flag on
set input-meta on
set convert-meta off
set output-meta on

# Only use one tab to get completion alternatives
set show-all-if-ambiguous on

# Completed names which are symbolic links to
# directories have a slash appended.
set mark-symlinked-directories on

$if mode=emacs
# for linux console and RH/Debian xterm
"\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
"\e[5C": forward-word
"\e[5D": backward-word
"\e\e[C": forward-word
```

```
"\e\e[D": backward-word
$endif
```

5.1.8. Creating a Login Warning Banner

When addressing system use policy, it is important for legal reasons to have a warning banner on a system. We do this by placing the appropriate message in the `/etc/issue` file. This warning should be approved by the organization management after being reviewed by the organization's legal staff.

```
Unauthorized use prohibited.

Your use of this network constitutes consent to monitoring and
disclosure of the fruits of the monitoring.  You have no
reasonable expectation of privacy on this network.
```

5.1.9. Ensuring Appropriate Random Number Generation

When using cryptographic techniques, one important factor is the availability of random numbers. In Linux, this is done through the devices `/dev/random` and `/dev/urandom`. These devices rely on a set of numbers called an entropy pool. If a system is started in similar ways, this pool will not provide the necessary randomness needed. To correct this problem, the following script (slightly modified) from *Beyond Linux From Scratch* [BLFS] should be placed in `/etc/rc.d/init.d/random`.

```
#!/bin/bash
# Begin $rc_base/init.d/random

# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org
# Random script elements by Larry Lawrence

source /etc/sysconfig/rc
source $rc_functions

case "$1" in
   start)
        echo "Initializing kernel random number generator..."
        if [ -f /var/tmp/random-seed ]; then
           cat /var/tmp/random-seed >/dev/urandom
        fi
        dd if=/dev/urandom of=/var/tmp/random-seed count=1 &>/dev/null
        evaluate_retval
        ;;

    stop)
        echo "Saving random seed..."
        dd if=/dev/urandom of=/var/tmp/random-seed count=1 &>/dev/null
```

```
        evaluate_retval
        ;;

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

# End $rc_base/init.d/random
```

To automate this script, run the following commands:

```
chmod 755 /etc/rc.d/init.d/random
cd /etc/rc.d/init.d
ln -sf ../init.d/random ../rc0.d/K45random
ln -sf ../init.d/random ../rc3.d/S25random
ln -sf ../init.d/random ../rc6.d/K45random
```

5.2. Setting up Networking

To ensure that networking is initialized properly, there are several files that need to be reviewed and updated for the specific implementation. These files set the IP addresses and name of the system and allow proper domain name resolution.

5.2.1. `/etc/sysconfig/network`

The `/etc/sysconfig/network` file provides the definitions for the system host name and the default gateway. The values in this file should be updated for the specific system being implemented.

```
#/etc/sysconfig/network
HOSTNAME=ns1
GATEWAY=192.168.0.1
GATEWAY_IF=eth0
```

5.2.2. `/etc/sysconfig/network-devices/ifconfig.eth{0,1}`

The initialization of the network interface cards in the system are controlled by files in the `/etc/sysconfig/network-devices/` directory. There is one for each interface. In this example, there are two files, `ifconfig.eth0` and `ifconfig.eth1`. The parameters in these files need to be adjusted for each server implementation.

```
#ifconfig.eth0
ONBOOT=yes
IP=192.168.0.2
NETMASK=255.255.255.0
```

```
BROADCAST=192.168.0.255


#ifconfig.eth1
ONBOOT=yes
IP=192.168.0.20
NETMASK=255.255.255.0
BROADCAST=192.168.0.255
```

5.2.3. `/etc/nsswitch.conf`

This file controls, among other things, the order of hostname resolution. Since the definitive addresses are in the DNS system, we will look there first. There is one line in this file that needs to be changed as follows:

```
hosts: dns files
```

5.2.4. `/etc/hosts`

The `/etc/hosts` file provides IP addresses for local lookups when the DNS system is not available. The only entries required here relate to the local system. Note that although there are two network interfaces for this system, only one address is necessary in this file.

```
# Begin /etc/hosts

127.0.0.1   localhost

192.168.0.2 ns1.example.edu pear

# End /etc/hosts
```

5.2.5. `/etc/resolv.conf`

The `/etc/resolv.conf` file tells local applications how to resolve host names when using DNS. We specify the default domain and the IP addresses of the nameservers. Since this system is a nameserver, we look at that application first. If it is not available, we look at a secondary nameserver.

```
domain example.edu
nameserver 127.0.0.1
nameserver 200.16.0.2
```

5.3. Useful Programs

There are a couple of useful utilities that administrators may use that can be installed here. They are not required for proper functioning of the system, but are recommended.

> **Note:** At this point we have not yet connected the system to the Internet. Files mentioned should be downloaded to another system and transferred to the server system via CD-ROM. Source files should be placed in `/usr/src`.

5.3.1. `which`

The `which` command shows the full path of commands. The source can be downloaded from

```
ftp://ftp.gnu.org/gnu/which/which-2.16.tar.gz
```

The application is installed by:

```
tar -zxvf which-2.16.tar.gz
cd which-2.16
./configure --prefix=/usr
make
make install
cd ..
```

5.3.2. `traceroute`

The `traceroute` command is used to display the route and timing of packets as they proceed to a destination host. It is a tool used for troubleshooting network connectivity. The source can be downloaded from:

```
ftp://ftp.ee.lbl.gov/traceroute-1.4a12.tar.gz
```

The application is installed by:

```
tar -zxvf traceroute-1.4a12.tar.gz
cd  traceroute-1.4a12
sed -i 's/-o bin/-o root/' Makefile.in
./configure --prefix=/usr
make
make install
cd ..
```

Chapter 6. Adding Logging Tools

In order to keep a network server running securely, the administrator needs to log important activities and review those logs. To accomplish this goal, there are several packages that need to be added to our system. Again, the source code for these files should be downloaded and transferred to the system via CD-ROM. The packages should be placed in `/usr/src`.

6.1. `syslog-ng`

The `syslog-ng` program is a daemon that takes the place of the traditional `syslog` daemon. This program allows the administrator the flexibility to use fine-grained message filtering. It also allows sending messages to a separate logging server using TCP instead of the traditional UDP protocol. This allows a reliable transmission stream and greatly reduces the possibility of lost messages when sent to an external server.

6.1.1. Building

The source files consist of two packages, a library and the application itself. It is not necessary to install the library in our final system but we do need to create the static library in a temporary location. After the library is built, we can then build `syslog-ng` and intall it. We do all of this with the following procedure.

First, download the sources

```
http://www.balabit.com/downloads/syslog-ng/libol/0.3/libol-0.3.13.tar.gz
http://www.balabit.com/downloads/syslog-ng/1.6/src/syslog-ng-1.6.2.tar.gz
```

Compile the program as follows:

```
tar -zxvf libol-0.3.13.tar.gz
cd libol-0.3.13
./configure --prefix=/usr
make
cd ..

tar -zxvf syslog-ng-1.6.2.tar.gz
cd syslog-ng-1.6.2
./configure --prefix=/usr --sysconfdir=/etc --with-libol=../libol-0.3.13
make
make install
```

6.1.2. Configuration

The configuration file is `/etc/syslog-ng/syslog-ng.conf`. For our name-server, we use the following configuration:

```
# syslog-ng.conf
```

```
options { stats(3600); };

source local { pipe("/proc/kmsg"); unix-stream("/dev/log"); internal(); };

destination sys       { file("/var/log/messages"
                        template("$DATE $FULLHOST $PROGRAM: $MESSAGE\n")); };
destination kern      { file("/var/log/kern.log"
                        template("$DATE $FULLHOST $PROGRAM: $MESSAGE\n")); };
destination mail      { file("/var/log/maillog");       };
destination auth      { file("/var/log/authlog");       };
destination daemon    { file("/var/log/daemon.log");    };
destination cron      { file("/var/log/cron");          };
destination netfilter { file("/var/log/netfilter.log"); };

destination loghost   { tcp("192.168.0.7" port(514));   };

filter f_auth       { facility(auth, authpriv);       };
filter f_daemon     { facility(daemon);               };
filter f_kern       { facility(kern);                 };
filter f_mail       { facility(mail);                 };
filter f_cron       { facility(cron);                 };
filter f_emerg      { level(emerg);                   };
filter f_netfilter  { facility(kern) and level(info); };

log { source(local); filter(f_auth);         destination(auth);      };
log { source(local);                         destination(sys);       };
log { source(local);                         destination(loghost);   };
log { source(local); filter(f_daemon);       destination(daemon);    };
log { source(local); filter(f_kern);         destination(kern);      };
log { source(local); filter(f_mail);         destination(mail);      };
log { source(local); filter(f_cron);         destination(cron);      };
log { source(local); filter(f_netfilter);    destination(netfilter); };
```

The *stats* parameter outputs a message every hour that tells the success of message output. This message provides a direct indication of the operation of the program.

The single source *local* defines all the input streams into the program.

The destination parameters *sys* and *kern* need to have a specific output template because the messages from the kernel are not automatically labeled.

The destination parameter *loghost* provides for output to the log server.

All messages are sent to the loghost server.

The other required script is the startup script. This script, named `syslog-ng` replaces the existing `syslog` script in `/etc/rc.d/init.d/`.

```
#!/bin/bash
# Begin $rc_base/init.d/syslog-ng - syslog-ng and klogd loaders

source /etc/sysconfig/rc
source $rc_functions

case "$1" in
```

```
    start)
        echo "Starting kernel log daemon..."
        loadproc klogd

        echo "Starting system log daemon..."
        loadproc syslog-ng
        ;;

    stop)
        echo "Stopping kernel log daemon..."
        killproc klogd

        echo "Stopping system log daemon..."
        killproc syslog-ng
        ;;

    reload)
        echo "Reloading system log daemon config file..."
        kill -HUP ‘cat /var/run/syslog-ng.pid‘
        ;;

    restart)
        $0 stop
        sleep 1
        $0 start
        ;;

    status)
        statusproc klogd
        statusproc syslogng
        ;;

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

# End $rc_base/init.d/sysklog-ng
```

Finally, we set up the symbolic links.

```
cd /etc/rc.d/init.d
rm sysklog
rm /etc/rc.d/rc{0136}.d/*sysklog
ln -sf ../init.d/syslog-ng ../rc0.d/K49syslog-ng
ln -sf ../init.d/syslog-ng ../rc3.d/S10syslog-ng
ln -sf ../init.d/syslog-ng ../rc6.d/K49syslog-ng
```

6.2. logrotate

In order to keep the logs from growing without bound, we need to manage them. The logrotate application will keep several generations of files according to its configuration file.

6.2.1. Building

First, download the sources

```
http://ftp.debian.org/debian/pool/main/p/popt/popt_1.7.orig.tar.gz
http://ftp.debian.org/debian/pool/main/p/popt/popt_1.7-4.diff.gz
http://ftp.debian.org/debian/pool/main/l/logrotate/logrotate_3.6.5.orig.tar.gz
http://ftp.debian.org/debian/pool/main/l/logrotate/logrotate_3.6.5-2.diff.gz
```

Compile the program and its required library as follows:

```
gunzip popt_1.7-4.diff.gz
tar -zxvf popt_1.7.orig.tar.gz
cd popt-1.7
patch -Np1 -i ../popt_1.7-4.diff
./configure --prefix=/usr
make
make install

gunzip logrotate_3.6.5-2.diff.gz
tar -zxvf logrotate_3.6.5.orig.tar.gz
cd logrotate-3.6.5
patch -Np1 -i ../logrotate_3.6.5-2.diff
make
make install
cd ..
```

6.2.2. Configuration

There is only one configuration file for logrotate, /etc/logrotate.conf. Create this file as follows:

```
# logrotate.conf
create

/var/log/lastlog /var/log/wtmp
{
  monthly
  create 0644 root root
  rotate 1
}

/var/log/daemon.log /var/log/kern.log /var/log/messages
{
  weekly
  rotate 6
```

```
  sharedscripts
  postrotate
     /bin/kill -HUP ‘cat /var/run/syslog-ng.pid‘
  endscript
}

/var/log/authlog /var/log/maillog /var/log/cron
{
  monthly
  rotate 3
  sharedscripts
  postrotate
    /bin/kill -HUP ‘cat /var/run/syslog-ng.pid‘
  endscript
}

/var/log/netfilter.log
{
  daily
  rotate 14
  postrotate
    /bin/kill -HUP ‘cat /var/run/syslog-ng.pid‘
  endscript
}
```

In this script, we keep two months worth of login records. Since logins will be relatively infrequent, we only need to rotate them monthly. The main logs, `daemon.log`, `kern.log`, and `messages.log` are kept for six weeks and rotated weekly. `auth.log` keeps login records; `maillog` keeps logs for the `sendmail` application; and `messages` keeps the a record of all the messages. These logs are kept for three months but rotated monthly because of relatively low input rates. The `netfilter.log` keeps a separate record of kernel messages, especially those generated by the netfilter component of the kernel. This file is rotated daily and files are kept for two weeks.

Since this program must be run daily, we will also need to set up a `cron` daemon to do that. We will use `fcron`.

6.3. `sendmail`

Before we build `fcron`, we need to set up a mail transfer agent to send the output to the system administrator. To do that, we will use `sendmail` for consistency with the other servers in our network.

6.3.1. Building

When building sendmail, we need to build two prerequisite programs — `procmail` to enable local mail delivery if necessary, and the Berkeley Database Libraries. Building `procmail` is a relatively simple procedure and no configuration is needed for our DNS server.

Download the source file:

```
ftp://ftp.procmail.net/pub/procmail/procmail-3.22.tar.gz
```

Compile the program as follows:

```
tar -zxvf procmail-3.22.tar.gz
cd procmail-3.22
make BASENAME=/usr install
make install-suid
cd ..
```

Building `Berkeley DB` is also relatively easy.

Download the source file:

```
http://www.sleepycat.com/update/snapshot/db-4.1.25.tar.gz
```

Compile the program as follows:

```
tar -zxvf db-4.1.25.tar.gz
cd db-4.1.25/build_unix
../dist/configure --prefix=/usr --enable-compat185 --enable-cxx
make
make docdir=/usr/share/doc/db-4.1.25 install
cd ..
```

Now we build `sendmail` itself. This can be a daunting task. There are literally hundreds of parameters that can be set in the building and configuration of `sendmail`. For a comprehensive treatment, you should see *Sendmail, Third Edition* [Cost].

In our server, we really only want to be able to use email to send log information to the system administrator. It would seem that this would not require a daemon. However, in `sendmail` version 8.12 there was a separation of the message submission function and the message transport function so the program does not run SUID root. Therefore, when `sendmail` receives a message, it cannot transmit it directly, but must instead pass it to a daemon for transmission to the destination.

To build `sendmail`, do the following.

Download the source file:

```
ftp://ftp.sendmail.org/pub/sendmail/sendmail.8.12.11.tar.gz
```

Compile the program as follows:

```
groupadd -g 20 smmsp
groupadd -g 21 mail
useradd -g smmsp -G mail -u 20 -s /bin/false smmsp
chmod 1777 /tmp
chmod 1777 /var/mail
mkdir -p /var/spool/mqueue
mkdir -p /etc/mail
```

```
tar -zxvf sendmail.8.12.11.tar.gz
cd sendmail-8.12.11

cat > devtools/Site/site.config.m4 << "EOF"
define('confMANGRP','root')
define('confMANOWN','root')
define('confSBINGRP','root')
define('confUBINGRP','root')
define('confUBINOWN','root')
EOF

cd sendmail
sh Build
cd ../cf/cf
```

Explanations of these commands can be found in [BLFS].

6.3.2. Configuration

The `sendmail` configuration file is quite long. Fortunately, we don't have to make it directly. We will create a `sendmail.mc` file and use the `m4` macro processor to build our `sendmail.cf` file. Place this file in the `cf/cf/`directory.

> **Note:** In the file below, the quote marks are different for the left and right. The `m4` macro processor requires starting a quote with an acute mark and ending with a single quote mark.

```
divert(-1)
#
#  This is a custom  configuration file for building a secure DNS server.
#  It has support for local and SMTP mail only.
#
divert(0)dnl
VERSIONID('sendmail.mc - Restricted MTA for DNS Server - 2/19/2004')
OSTYPE(linux)dnl
DOMAIN(generic)dnl
DAEMON_OPTIONS('NAME=NoMTA, Family=inet, Addr=127.0.0.1')dnl
FEATURE('no_default_msa')dnl
FEATURE('always_add_domain')dnl
FEATURE('nouucp','reject')dnl
define('confEBINDIR', '/usr/sbin')
define('PROCMAIL_MAILER_PATH', '/usr/bin/procmail')
FEATURE(local_procmail)
MAILER(local)dnl
MAILER(smtp)dnl
```

In the above configuration, there are several security entries. The *DAEMON_OPTIONS* instruction tells the program to only listen to port 25 on the loopback address. The *no_default_msa* feature tells the program not to listen to the submission port (587) at all. The *nouucp* feature rejects any uucp

mail addresses. Finally, the *always_add_domain* feature will add the domain name to the sender's address.

Now we finish the install.

```
sh Build sendmail.mc
sh Build install-cf

cd ../..
sh Build install

echo ‘hostname‘ > /etc/mail/local-host-names

cat > /etc/mail/aliases << "EOF"
postmaster: root
MAILER-DAEMON: root
EOF

newaliases -v

cp cf/cf/{submit,sendmail}.mc /etc/mail
cp cf/cf/{submit,sendmail}.cf /etc/mail

cd /etc/mail

cat > access << "EOF"
localhost   RELAY
127.0.0.1   RELAY
EOF

makemap hash access < access
```

The last thing to do is to set up `sendmail` for automatic startup. The following file should be placed in `/etc/rc.d/init.d/sendmail`.

```
#!/bin/bash
# Begin $rc_base/init.d/sendmail

# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org

source /etc/sysconfig/rc
source $rc_functions

case "$1" in

    start)
        echo "Starting sendmail..."
        loadproc /usr/sbin/sendmail -bd -q1h
        ;;

    stop)
        echo "Stopping Sendmail..."
        killproc sendmail
```

```
        ;;

    status)
        statusproc sendmail
        ;;

    restart)
        kill -HUP ‘head -n1 /var/run/sendmail.pid‘
        ;;

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

# End $rc_base/init.d/sendmail
```

Make the script executable and create the symlinks.

```
chmod 755 /etc/rc.d/init.d/sendmail

cd /etc/rc.d/init.d
ln -sf ../init.d/sendmail ../rc0.d/K20sendmail
ln -sf ../init.d/sendmail ../rc3.d/K20sendmail
ln -sf ../init.d/sendmail ../rc3.d/S35sendmail
ln -sf ../init.d/sendmail ../rc6.d/K20sendmail
```

6.4. `nail`

Since we have spent a lot of time on `sendmail`, we might as well add a simple command line mail user agent here.

6.4.1. Building

Download the source

```
http://nail.berlios.de/archive/nail-10.7.tar.bz2
```

Install the program with:

```
tar -jxvf nail-10.7.tar.bz2
pushd nail-10.7
./configure --prefix=/usr
make
make install
cd /usr/bin
ln -sf nail mail
popd
```

A configuration file, `/etc/nail.rc`, is installed as a part of the build, but no changes are required.

6.5. `fcron`

The `fcron` application is a replacement for Vixie Cron.

6.5.1. Building

Download the source

```
http://fcron.free.fr/fcron-2.9.4.src.tar.gz
```

Install the program with:

```
groupadd -g 18 fcron
useradd -g fcron -s /bin/false -u 18 fcron

tar -zxvf fcron-2.9.4.src.tar.gz
cd fcron-2.9.4
./configure --with-answer-all=no
make
make install
cd ..
```

6.5.2. Configuration

We need to create the startup script as `/etc/rc.d/init.d/fcron`.

```
#!/bin/bash
# Begin $rc_base/init.d/fcron

# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org

source /etc/sysconfig/rc
source $rc_functions

case "$1" in
    start)
      echo "Starting fcron..."
      loadproc fcron
      ;;

    stop)
      echo "Stopping fcron..."
      killproc fcron
      ;;
```

```
    restart)
      $0 stop
      sleep 1
      $0 start
      ;;

    status)
      statusproc fcron
      ;;

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

# End $rc_base/init.d/fcron
```

Now make the script executable and set up the symlinks.

```
cd /etc/rc.d/init.d
chmod 755 fcron
ln -sf ../init.d/fcron ../rc0.d/K08fcron
ln -sf ../init.d/fcron ../rc3.d/K08fcron
ln -sf ../init.d/fcron ../rc3.d/S40fcron
ln -sf ../init.d/fcron ../rc6.d/K08fcron
```

At this point we want to start `fcron` and set up the system table to run `logrotate`.

```
/etc/rc.d/int.d/fcron start
fcrontab -e systab
```

Edit the system table using `vi` commands to run `logrotate` daily.

```
#m h d m wd cmd line
5 3 * * *  /usr/sbin/logrotate /etc/logrotate.conf
```

6.6. `logwatch`

Now that the logs are being updated and rotated properly, it is a good idea to automate the process of examining the logs. The `logwatch` application is a series of `perl` scripts that does this for us.

6.6.1. Building

Download the source

```
ftp://ftp.kaybee.org/pub/old/linux/logwatch-5.0.tar.gz
```

Since the application is a set of `perl` scripts, there is no compilation. We just need to extract the files from the tar file and put them in `/etc`.

```
tar -zxvf logwatch-5.0.tar.gz
cd  logwatch-5.0
mkdir -p /etc/log.d
cp -R conf    /etc/log.d/
cp -R lib     /etc/log.d/
cp -R scripts /etc/log.d/
```

6.6.2. Configuration

Before we can use `logwatch`, there are a couple of minor changes we need to make to the configuration file, `/etc/log.d/conf/logwatch.conf`. We don't have the `MkTemp` program and the configuration file needs to have the location of `mail` updated.

```
sed -i -e "s:/bin/mail:/usr/bin/mail:" \
       -e "s:UseMkTemp = Yes:UseMkTemp = No:" \
       /etc/log.d/conf/logwatch.conf
```

Last, we update the `fcron` system table with **`fcrontab -e systab`** and add the following line:

```
0 2 * * *  /etc/log.d/scripts/logwatch.pl
```

6.7. `ntpd`

The accuracy of timestamps on the logs is very important for several reasons. First, accurate time is necessary when trying to reconcile events on multiple systems. It is needed for documentation of hostile activity for legal documentation. It is also required to make sure `fcron` jobs are executed at the right time.

The most widely used means for coordinating system clocks is the Network Time Protocol. We will use the standard daemon `ntpd` to implement this capability.

6.7.1. Building

Download the source

```
ftp://ftp.udel.edu/pub/ntp/ntp4/ntp-4.2.0.tar.gz
```

Create the executable with:

```
tar -zxvf ntp-4.2.0.tar.gz
cd ntp-4.2.0
./configure --prefix=/usr --bindir=/usr/sbin --sysconfdir=/etc
make
```

```
make install
cd ..
```

6.7.2. Configuration

The configuration file is `/etc/ntp.conf`. To get the system to give accurate time, there should be at least three sources of time. The locations of the time servers are at [Mill2]. As the page says, don't use a NTP primary time service for a single host.

The configuration file below is set up as a simple `ntp` client, however there are commented lines that, if enabled, will make this server into a Stratum 2 public time server by opening up the access slightly. You would also need to remove the first *`restrict`* line and change the IP addresses and peer servers appropriately.

```
server clepsydra.dec.com
server navobs1.usnogps.navy.mil
server tick.uh.edu

#peer tick
#peer tock

restrict default ignore
#restrict default nomodify nopeer noquery
#restrict 192.168.1.0 mask 255.255.255.0
restrict 127.0.0.1

driftfile /var/cache/ntp.drift
pidfile   /var/run/ntp.pid
```

There are also two methods of controlling access to the server for remote administration using cryptographic techniques. These authentication methods are not necessary for our system, but the best reference for the procedures is located at [Mill].

The last step is to make `ntpd` start automatically at startup. The file is `/etc/rc.d/init.d/ntpd`.

```
#!/bin/bash
# Begin $rc_base/init.d/ntpd
source /etc/sysconfig/rc
source $rc_functions
case "$1" in
    start)
       echo "Starting ntpd..."
       loadproc ntpd
       ;;
    stop)
       echo "Stopping ntpd..."
       killproc ntpd
       ;;
    restart)
```

```
        $0 stop
        sleep 1
        $0 start
        ;;
    status)
        statusproc ntpd
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
        ;;
esac
# End $rc_base/init.d/ntpd
```

Make the script bootable and create the symlinks:

```
cd /etc/rc.d/init.d
chmod 755 ntpd
ln -sf ../init.d/ntpd ../rc0.d/K46ntpd
ln -sf ../init.d/ntpd ../rc3.d/K46ntpd
ln -sf ../init.d/ntpd ../rc3.d/S26ntpd
ln -sf ../init.d/ntpd ../rc6.d/K46ntpd
```

Chapter 7. Adding Security Applications

When securing a system, it is always wise to install multiple layers of security. This provides a 'defense in depth' such that an error or vulnerability in one area is not catastrophic. To properly secure our DNS server, there are several applications needed to provide these levels of protection.

It is significant to note here that there are some security packages that could be installed but we choose *not* to install them. Examples include `tcpwrappers`, `sudo`, and a cryptographic filesystem. Since the only accounts on this system are administrators and the data is not sensitive, these packages are unnecessary.

7.1. Pluggable Authentication Modules

Pluggable Authentication Modules, or `PAM`, provides a variety of library modules to give administrators different means of authentication, authorization, and logging of critical programs. In order to use `PAM`, the application must have the capability programmed into the code. Here we will construct the libraries and then recompile the `shadow` package to enable `PAM` support for the most critical programs needing authentication, including `login`, `passwd`, and `su`.

7.1.1. Building

We start by downloading the source code for `PAM`, an optional `cracklib` utility, a list of words to use with `cracklib`, and a patch file for the `shadow` package.

```
http://www.crypticide.org/users/alecm/security/cracklib,2.7.tar.gz
ftp://ftp.debian.org/debian/pool/main/s/scowl/scowl_5.orig.tar.gz
http://www.linuxfromscratch.org/patches/blfs/cvs/
    Linux-PAM-0.77-linkage-1.patch
http://www.linuxfromscratch.org/patches/blfs/cvs/shadow-4.0.4.1-pam-1.patch
ftp://ftp.kernel.org/pub/linux/libs/pam/pre/library/Linux-PAM-0.77.tar.bz2
```

The `cracklib` utility provides a means for creating a database of words to check against passwords as they are being set by the `passwd` program. It requires a word list. First we put the words in `/usr/dict/words`.

```
mkdir -p /usr/dict
tar -zxvf scowl_5.orig.tar.gz
cat scowl-5/final/* > /usr/dict/words
```

Now create the dictionary. This process puts the word database files into `/usr/local/lib/pw_dict.{hwm,pdw,pwi}`.

```
tar -zxvf cracklib,2.7.tar.gz
cd  cracklib,2.7
make all
make install
cp cracklib/libcrack.a /usr/lib
cd ..
```

We can now build PAM.

```
tar -jxvf Linux-PAM-0.77.tar.bz2
cd Linux-PAM-0.77

patch -Np1 -i ../Linux-PAM-0.77-linkage-1.patch
./configure --enable-static-libpam --with-mailspool=/var/mail \
            --enable-read-both-confs --sysconfdir=/etc
make
make install
mv /lib/libpam.a /lib/libpam_misc.a /lib/libpamc.a /usr/lib
ln -sf ../../lib/libpam.so.0.77 /usr/lib/libpam.so
ln -sf ../../lib/libpam_misc.so.0.77 /usr/lib/libpam_misc.so
ln -sf ../../lib/libpamc.so.0.77 /usr/lib/libpamc.so
cd ..
```

Now rebuild the shadow package.

```
cp /tools/src/shadow-4.0.3.tar.bz2 .
tar -jxvf shadow-4.0.3.tar.bz2
cd shadow-4.0.3
patch -Np1 -i ../shadow-4.0.4.1-pam-1.patch
LIBS="-lpam -lpam_misc" ./configure --libdir=/usr/lib \
   --enable-shared --with-libpam
echo '#define HAVE_SETLOCALE 1' >> config.h
make
make install
mv /bin/sg /usr/bin
mv /bin/vigr /usr/sbin
rm /bin/groups
mv /usr/lib/lib{misc,shadow}.so.0* /lib
ln -sf ../../lib/libshadow.so.0 /usr/lib/libshadow.so
ln -sf ../../lib/libmisc.so.0 /usr/lib/libmisc.so
cd ..
```

7.1.2. Configuration

The PAM implementation consists of several sets of files. The loadable modules are located in the directory /lib/security. The PAM configuration files are spread out over several locations. There is a main file, /etc/pam.conf, as well as two directories, /etc/pam.d and /etc/security. Since we compiled the modules with the directive *--enable-read-both-confs*, we can put application configuration files in /etc/pam.d and omit /etc/pam.conf. The /etc/security directory contains configuration files for individual modules.

Details of the configuration files are found in the *Linux-PAM System Administrators' Guide* [Morg].

The application configuration files have the same names as the application. If there is no specific configuration file, a PAM enabled application will use the configuration file other. This file is set up to deny everything and log information about an authentication or a password change.

```
# Begin /etc/pam.d/other

auth        required        pam_deny.so
auth        required        pam_warn.so
account     required        pam_deny.so
session     required        pam_deny.so
password    required        pam_deny.so
password    required        pam_warn.so

# End /etc/pam.d/other
```

The `login` configuration file does several things for us. It provides authentication, account, and session services each time a user logs in. The specific details of the functions can be found in the Administrators' Guide.

```
# Begin /etc/pam.d/login

auth        requisite      pam_securetty.so
auth        requisite      pam_nologin.so
auth        required       pam_env.so
auth        required       pam_unix.so

account     required       pam_access.so
account     required       pam_unix.so

session     required       pam_motd.so
session     required       pam_limits.so
session     optional       pam_mail.so     dir=/var/mail empty
session     optional       pam_lastlog.so
session     required       pam_unix.so

# End /etc/pam.d/login
```

The `passwd` configuration file deserves special consideration. Here we use the `pam_cracklib.so` module to check for passwords that are in the dictionary database. The total character count for a password is 10 characters, but each punctuation character counts as two. We also only give the user two tries.

```
# Begin /etc/pam.d/passwd

password    required       pam_cracklib.so     \
     retry=2 minlen=10 ocredit=2 dcredit=1 ucredit=1 lcredit=1

password    sufficient     pam_unix.so         use_authtok md5 shadow
password    required       pam_deny.so

# End /etc/pam.d/passwd
```

The other files are similar and are listed here for reference.

```
# Begin /etc/pam.d/chage
auth        sufficient      pam_rootok.so
auth        required        pam_unix.so
account     required        pam_unix.so
```

```
session     required        pam_unix.so
password    required        pam_permit.so
# End /etc/pam.d/chage


# Begin /etc/pam.d/fcron
# Warning : fcron has no way to prompt user for a password !
auth        required        pam_rootok.so
account     required        pam_unix.so
session     required        pam_permit.so
# End /etc/pam.d/fcron


# Begin /etc/pam.d/fcrontab
### There is no reason for anyone other than root
### to run crontab on this system
auth        required        pam_rootok.so
account     required        pam_permit.so
session     required        pam_permit.so
# End /etc/pam.d/fcrontab


# Begin /etc/pam.d/shadow
auth        sufficient      pam_rootok.so
auth        required        pam_unix.so
account     required        pam_unix.so
session     required        pam_unix.so
password    required        pam_permit.so
# End /etc/pam.d/shadow


# Begin /etc/pam.d/su
auth        sufficient      pam_rootok.so
auth        required        pam_unix.so
account     required        pam_unix.so
session     required        pam_unix.so
# End /etc/pam.d/su


# Begin /etc/pam.d/useradd
auth        sufficient      pam_rootok.so
auth        required        pam_unix.so
account     required        pam_unix.so
session     required        pam_unix.so
password    required        pam_permit.so
# End /etc/pam.d/useradd
```

The configuration files in `/etc/security` are installed during the build phase and no changes are needed.

One last configuration file is `/etc/securetty`. This file lists terminals which are valid for user login. Since we only allowed two consoles earlier when we created `/etc/inittab`, we change this file to make it consistent.

```
# /etc/securetty: list of terminals on which root is allowed to login.
```

```
# See securetty(5) and login(1).
tty1
tty2
```

7.2. openssl

The primary way administrators will access the system for updates and maintenance is via an encrypted remote login. To set this up, the `openssl` application must be installed to provide the required cryptographic libraries.

The definitive reference for `openssl` is *Network Security with OpenSSL* [Vieg].

7.2.1. Building

The `openssl` application has had some security updates recently. Be sure to get the most recent file.

```
ftp://ftp.openssl.org/source/openssl-0.9.7d.tar.gz
```

Compile and install the application with the following:

```
tar -zxvf openssl-0.9.7d.tar.gz
cd openssl-0.9.7d
sed 's/^passwd/openssl-passwd/' doc/apps/passwd.pod \
    > doc/apps/openssl-passwd.pod
rm doc/apps/passwd.pod
mv doc/crypto/{,openssl_}threads.pod
./config --openssldir=/etc/ssl --prefix=/usr shared
make MANDIR=/usr/share/man
make MANDIR=/usr/share/man install
cp -r certs /etc/ssl
rmdir /etc/ssl/lib
chmod 755 /usr/lib/pkgconfig
cd ..
```

7.2.2. Configuration

The `openssl` package provides a variety of security related functions, including the capability to generate and sign Public Key Certificates. To do this, there is a configuration file `/etc/ssl/openssl.conf`. We do not need this capability on a DNS server and do not need to modify the default file.

7.3. openssh

Probably the most used application on a server for remote access is `ssh`. This is one of the critical pieces of software that make distributed administration possible. It allows a remote login or file transfer over an encrypted tunnel.

The preeminent reference for `openssh` is *OpenSSH, The Secure Shell: The Definitive Guide* [Barr]. Another excellent reference for `openssh` is *Linux Systems Security* [Mann]. The latter book has some very nice examples and tables of configuration file entries and their function.

7.3.1. Building

Again, the first step is to download the file:

```
ftp://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-3.8p1.tar.gz
```

Continue the install with:

```
tar -zxvf openssh-3.8p1.tar.gz
cd openssh-3.8p1
mkdir -p /var/empty
chown root:root /var/empty
groupadd -g 25 sshd
useradd -c 'sshd privsep' -d /var/empty -g sshd -u 25 -s /bin/false sshd

./configure --prefix=/usr --sysconfdir=/etc/ssh \
    --libexecdir=/usr/sbin --with-md5-passwords
make
make install
cd ..
```

At the end of the install, a host key needs to be generated. When asked, do *not* specify a password. This will disable automatic startup of `sshd`.

```
ssh-keygen -t dsa -f /etc/ssh/ssh_host_key -N ""
```

7.3.2. Configuration

There are two system configuration files for `openssh`: `/etc/ssh/sshd_config` and `/etc/ssh/ssh_config`. We are not interested in the `ssh` client, but only in the server. Indeed, later we will remove the `ssh` client altogether. We configure `sshd` as follows:

```
#/etc/ssh/sshd_config
Port 22
Protocol 2
ListenAddress 192.168.0.2
```

```
HostKey /etc/ssh/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 1h
ServerKeyBits 768

# Logging
SyslogFacility AUTHPRIV
LogLevel INFO

# Authentication:

LoginGraceTime 1m
PermitRootLogin no
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys

RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreRhosts yes

PasswordAuthentication yes
PermitEmptyPasswords no

ChallengeResponseAuthentication yes

# Set this to 'yes' to enable PAM authentication (via
# challenge-response) and session processing. Depending
# on your PAM configuration, this may bypass the setting
# of 'PasswordAuthentication'
#UsePAM yes

AllowTcpForwarding no
GatewayPorts no
X11Forwarding no
PrintMotd yes
PrintLastLog yes
KeepAlive yes
#UseLogin no
UsePrivilegeSeparation yes
PermitUserEnvironment no
#Compression yes
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS yes
PidFile /var/run/sshd.pid
#MaxStartups 10

# no default banner path
Banner /etc/issue

# override default of no subsystems
```

```
Subsystem       sftp    /usr/sbin/sftp-server
```

There are several security issues addressed in the file above. First, ssh1 is disabled due to weaknesses in the protocol. Second, there is only one IP address enabled for connection. Third, and very important, *RhostsRSAAuthentication* and *HostbasedAuthentication* are disabled. The only allowed forms of authentication are via a password or by a dsa key. Fourth, the ability to log in directly as root is disabled.

We choose to *not* enable PAM because it disables the sshd password authentication. If PAM is desired, we would have to add *--with-pam* to the **./configure** instruction in the build and create an /etc/pam.d/sshd configuration file.

Other security issues include specifying the use of /etc/issue as a banner and disallowing forwarding through *GatewayPorts*.

After configuration, the application needs to be enabled on startup by creating the /etc/rc.d/init.d/sshd:

```
#!/bin/bash
# Begin $rc_base/init.d/sshd

# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org

source /etc/sysconfig/rc
source $rc_functions

case "$1" in
    start)
       echo "Starting SSH Server..."
       loadproc /usr/sbin/sshd
       ;;

    stop)
       echo "Stopping SSH Server..."
       killproc /usr/sbin/sshd
       ;;

    reload)
       echo "Reloading SSH Server..."
       reloadproc /usr/sbin/sshd
       ;;

    restart)
       $0 stop
       sleep 1
       $0 start
       ;;

    status)
       statusproc /usr/sbin/sshd
       ;;

    *)
```

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

# End $rc_base/init.d/sshd
```

Make the script executable and create the symlinks.

```
cd /etc/rc.d/init.d
chmod 755 sshd
ln -sf ../init.d/sshd ../rc0.d/K30sshd
ln -sf ../init.d/sshd ../rc3.d/K30sshd
ln -sf ../init.d/sshd ../rc3.d/S30sshd
ln -sf ../init.d/sshd ../rc6.d/K30sshd
```

7.4. gnupg

The Gnu Privacy Guard application provides a method of using cryprographic techniques to encrypt and digitally sign files. It is an open source version of Phil Zimmerman's *Pretty Good Privacy* [Garf]. The main use on a server is to allow the administrator to digitally sign messages as they are sent from `cron` jobs.

7.4.1. Building

Start by getting the source code at:

```
ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-1.2.4.tar.bz2
```

Construct the executable with:

```
tar -jxvf gnupg-1.2.4.tar.bz2
cd gnupg-1.2.4
./configure --prefix=/usr --libexecdir=/usr/lib
make
make install
```

Some authors recommend setting the SUID bit for the executable, **gpg**, because the program complains about the possibility of the system swapping out sensitive data if run as a non-privileged user. Since the program is only going to be run as root on this server, setting the SUID bit is not required.

7.4.2. Configuration

The main configuration requirement for a server is to generate a key pair for signing a document. To do this, run as root:

```
gpg --gen-key
```

The program will generate a key pair. This process will ask several questions. The key can be set to not expire. Do *not* set a pass phrase. Normally this is not a good idea, but we need to do this so signing a document can be automated.

Export the public key with:

```
gpg --export --armor > server-pub-key.asc
```

The public key can then be transferred to the administrator's system for checking the validity of email sent from the server.

Documents can now be signed. For instance, the `logwatch` output could be signed with the following command:

```
/etc/log.d/scripts/logwatch.pl | gpg --clearsign --armor --batch
```

7.5. `lsof`

The `lsof` is a security tool. It can be used for system administration and forensics. `lsof` combines many of the capabilities of `netstat` and `ps` as well as providing information not easily available about open files, including devices and network connections.

There are a lot of options available for `lsof`. Detailed use of the program is beyond the scope of this paper, but the program comes with a detailed man page.

7.5.1. Building

Download the source code:

```
ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/lsof_4.71.tar.bz2
```

Build the application:

```
tar -jxvf lsof_4.71.tar.bz2
cd lsof_4.71
tar -xvf lsof_4.71_src.tar
cd lsof_4.71_src
./Configure linux
make
cp lsof /sbin
cp lsof.8 /usr/share/man/man8/
chmod 0700 /sbin/lsof
```

Since `lsof` is so powerful, we choose to set permissions to limit it to the root user.

7.5.2. Configuration

There is no configuration required for `lsof`.

7.6. `iptables`

The Linux kernel has the ability to act as a firewall. This portion of the kernel is called the `netfilter`. The `iptables` application is the user interface to to set up the firewall in the kernel.

7.6.1. Building

The source code is available at:

```
http://www.iptables.org/files/iptables-1.2.9.tar.bz2
```

Building the executable is quite straightforward.

```
tar -jxvf iptables-1.2.9.tar.bz2
cd iptables-1.2.9
make PREFIX=/usr LIBDIR=/lib BINDIR=/sbin
make PREFIX=/usr LIBDIR=/lib BINDIR=/sbin install
cd ..
```

7.6.2. Configuration

The `iptables` script to establish the firewall is located in Appendix C.

The script is divided into four main parts. The first part sets the policy for all network traffic to *DROP* and flushes any existing rules. The second part sets kernel flags in the `/proc` pseudo directory to control some fundamental problems, including responses to ICMP echo request broadcasts, ICMP redirection, and source routed packets.

The third part of the script is the most interesting part. Here are the rules that allow just the packets we want. We start by accepting all packets on the loopback interface and any packet that has established a valid connection. A separate chain that limits connection requests helps prevent some denial of service attacks. All fragments are also dropped.

After the general rules are created, specific packets for our server are allowed. We allow general UDP DNS packets, but only allow TCP DNS packets to or from our secondary name server. We have chosen to allow NTP time queries on this server so we allow those inbound as well as outbound. SMTP is allowed, but only outbound to our mail server. If the mail server requests IDENT information, we reject it with a TCP reset. Finally, SSH is allowed inbound, but only from a trusted internal subnet.

ICMP is also controlled by allowing only certain message types: destination unreachable, time exceeded, and pings. Some administrators may not want to allow pings, but it is generally appropriate to allow them on publicly accessible servers.

In summary, the firewall only allows:

- DNS udp packets
- DNS tcp packets to and from the secondary nameserver
- SSH tcp packets from a trusted internal network
- SMTP tcp packets to a trusted mail server
- NTP udp packets
- Echo request and reply icmp packets
- Time exceeded icmp packets
- Destination unreachable icmp packets

The final part of the script has to do with logging. There are several types of messages, notably NETBIOS related, that we drop without logging. We classify all other packets and log them for analysis and comparison to other systems.

A general reference for `iptables` is the *Packet Filtering HOWTO* [Russ]. An additional in depth treatment of `iptables` can be found in *Linux Firewalls* [Zieg].

We also need to put the startup script into `/etc/rc.d/init.d/iptables`.

```
#!/bin/bash
# Begin $rc_base/init.d/iptables

# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org

source /etc/sysconfig/rc
source $rc_functions

case "$1" in
  start)
     echo "Starting iptables..."
     loadproc /etc/rc.d/rc.iptables
     ;;

  stop)
     echo "Stopping iptables..."
     /usr/sbin/iptables -P INPUT ACCEPT
     /usr/sbin/iptables -P OUTPUT ACCEPT
     /usr/sbin/iptables -P FORWARD DROP
     /usr/sbin/iptables -F
     /usr/sbin/iptables -X
     ;;

  *)
```

```
     echo "Usage: $0 {start|stop}"
     exit 1
     ;;
esac

# End $rc_base/init.d/iptables
```

We enable the startup script and set the symlinks with:

```
cd /etc/rc.d/init.d
chmod 755 sshd
ln -sf ../init.d/sshd ../rc3.d/S30sshd
```

There is no need to open up the firewall when changing run levels. If required, it can be done manually with **`/etc/rc.d/init.d/iptables stop`**.

7.7. `tripwire`

`Tripwire` is an intrusion detection package. It creates a snapshot of the system files and periodically checks the current system against the baseline. It uses several techniques such as cryptographic hashes and filesystem parameters to determine if a file has been changed on the system.

`Tripwire` is a commercial application, however, most of the capabilities of the system are available through an open source version.

7.7.1. Building

The open source version of `tripwire` has not been updated for the `gcc` 3.0 series of compilers. To get the system to compile properly, a patch is needed as well as the source.

```
http://www.linuxfromscratch.org/patches/blfs/cvs/
              tripwire-2.3.1-2-gcc3-build-fixes.patch
ftp://ftp.fu-berlin.de/unix/security/tripwire/tripwire-2.3.1-2.tar.gz
```

The build is done with the following commands:

```
tar -zxvf tripwire-2.3.1-2.tar.gz
cd tripwire-2.3.1-2

patch -Np1 -i ../tripwire-2.3.1-2-gcc3-build-fixes.patch
make -C src release
cp install/install.{sh,cfg} .

./install.sh
cp /etc/tripwire/tw.cfg /usr/sbin
cp policy/*.txt /usr/share/doc/tripwire
cd ..
```

7.7.2. Configuration

Configuration of `tripwire` is accomplished by editing the policy file, `/etc/tripwire/twpol.txt`. This file specifies which files and directories on the system should be checked and what checks need to be made. Because all the applications of the system are not yet installed, we are going to postpone setting up the policy file until all the programs and configuration files are in place.

Chapter 8. Adding the Server Application

The Domain Name System (DNS) is one of the essential parts of the Internet that makes network communications possible. Its fundamental purpose is to translate host names with some semantic meaning to a numerical IP address for use by machines.

Because of the critical nature of this service, it has become one of the most highly targeted applications on the Internet. The most widely used implementation of DNS is `bind`. It is also listed perennially as the SANS top vulnerability for Unix systems [Twen]. There are two main reasons for this vulnerability. First, the latest security upgrades are not installed. Second, poor configuration allows certain types of attacks such as cache poisoning.

To address the first problem, we are going to use the most up to date version of `bind`, 9.2.2. A search of the `bind` entries in the *Common Vulnerabilities and Exposures* [CVE] database show no entries for Version 9. Additionally, this version adds several security capabilities that provides additional ways to secure our server.

8.1. Building

The source code is located at:

```
ftp://ftp.isc.org/isc/bind9/9.2.2/bind-9.2.2.tar.gz
```

Compilation is quite straightforward.

```
tar -zxvf bind-9.2.2.tar.gz
cd bind-9.2.2
./configure --prefix=/usr --sysconfdir=/etc \
            --localstatedir=/var --disable-ipv6
make
make install
cd ..
```

8.2. Configuration

The best source of information about configuring `bind` is *DNS and BIND* [Albi]. This book documents three versions of `bind`: 4, 8 and 9. In the book they state that DNS is basically a distributed database that allows local control of parts of the database. Setting up the local portion of this database is the critical task in creating an effective, secure DNS server.

The first step is to create a special user and group, both labeled *named*:

```
groupadd -g 30 named
useradd -g named -u 30 -s /bin/false named
```

The application executable is also called `named`. We are going to run it in a `chroot` environment on a separate partition. This partition is `/var/named`. Doing this allows us to mount `/var` with the parameters *noexec* and *nodev* so no devices or executables may be used on that partition, but `/var/named` can be mounted allowing the devices needed for the `chroot` environment. In this step the subdirectories and devices needed within the `chroot` environment are created.

```
cd /var/named
mkdir -p dev etc var/run
mknod /home/named/dev/null c 1 3
mknod /home/named/dev/random c 1 8
chmod 666 dev/null
chmod 444 dev/random
cp /etc/localtime /var/named/etc
```

The next step is to create `/etc/named.conf`.

```
acl "internal" { 192.68.0/24; 192.168.1/24; 192.168.2/24; 127/8; };
acl "external-slaves"   { 200.0.0.2; };
acl "internal-slaves"   { none; };

options
{
  directory "/var/named";
  pid-file  "/var/run/named.pid";
  version   "Administratively withheld";
  allow-recursion { "internal"; };
  max-ncache-ttl 3600;
  listen-on { 127.0.0.1; };
  query-source address 192.168.0.2 port 53;
};

controls
{
  inet 127.0.0.1 port 953 allow { 127.0.0.1; } keys { "rndc-key"; };
};

include "/etc/rndc.key";


view internal
{
  match-clients { "internal"; };
  allow-transfer { "internal-slaves"; };
  zone "example.edu" IN
  {
    type master;
    file "db.example.edu";
  };

  zone "0.168.192.in-addr.arpa" IN
  {
    type master;
```

```
    file "db.192.168.0";
  };

  zone "1.168.192.in-addr.arpa" IN
  {
    type master;
    file "db.192.168.1";
  };

  zone "2.168.192.in-addr.arpa" IN
  {
    type master;
    file "db.192.168.2";
  };

  zone "." IN
  {
    type hint;
    file "named.root";
  };

  zone "0.0.127.in-addr.arpa" IN
  {
    type master;
    file "db.127.0.0";
  };
};

view external
{
  match-clients { any; };
  allow-transfer { "external-slaves"; };
  zone "example.edu" IN
  {
    type master;
    file "db.example.edu.external";
  };

  zone "192-223.1.1.200.in-addr.arpa" IN
  {
    type master;
    file "db.200.1.1.192-223";
  };

  zone "." IN
  {
    type hint;
    file "named.root";
  };

  zone "0.0.127.in-addr.arpa" IN
  {
    type master;
    file "db.127.0.0";
  };
```

```
};
```

Lets go over this file in some detail. The initial step is to create three access control lists. The first specifies the internal networks that connect to the server. The second and third are lists of secondary DNS servers that we will allow to transfer zone files.

The *options* section specifies global parameters. Since the configuration file is read before the program does a `chroot`, the directory where the program looks for zone files and places its *pid* file is relative to the root directory.

Security settings follow. First, the version is withheld from users to deny any attackers the specific knowledge of the program that is running. Second, the program is allowed to only provide recursive DNS lookups for the systems on the internal networks. This procedure makes cache poisoning quite difficult. The next statement mitigates 'not found' denial of service responses to the recursive side of the system. No matter what the time to live setting of a server's response, the maximum time negative responses will be left in cache is one hour. The next statement limits control of the server via the `rndc` program to the local system. Finally, the system is told to use a single address and port 53 to make recursive DNS queries so that the firewall can be tightened to a single address/port combination.

The controls section explicitly sets the port used by the `rndc` and specifies which cryptographic key is required for server–`rndc` communication. The next statement loads the key.

The next two parts specify the split view function of the server. In one view, the addresses provided are private addresses appropriate for the internal network. In the other view, the public addresses for a subnet (200.1.1.192/27) are provided.

The internal view specifies the forward and reverse zone files for the local network with the private addresses. It also allows recursion and zone transfers to any internal secondary name servers. The external view, which is the default, only provides public addresses to external queries. It also allows zone transfers only to designated external secondary servers.

The zone files that the system uses are listed in Appendix D. They are placed in `/var/named/`. These are standard zone files that have no specific security settings.

The next file to consider is `/etc/rndc.key`. This file is shared between `rndc`, which is the interface program for controlling the daemon, and `named` to provide a common security key. The file looks like:

```
key rndc_key
{
    algorithm "hmac-md5";
    secret "tN1RqHxbGsD5Ix1Psryckg==";
};
```

Now create the `/etc/rndc.conf` file.

```
include "/etc/rndc.key";
options
{
    default-server localhost;
    default-key    rndc_key;
};
```

The secret key is generated by the `dnssec-keygen` program.

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST ns1.example.edu
```

The *`-a HMAC-MD5`* option provides the name of the algorithm used; the *`-b 128`* option specifies the length of the key; the *`-n HOST`* option specifies the type of key to generate. The last argument is the name of the key.

The output is two files, in this case `Kns1.example.edu.+157+02305` and `Kns1.example.edu.+157+02305.private`. The first file is the only one of interest here and contains:

```
ns1.example.edu. IN KEY 512 3 157 tN1RqHxbGsD5Ix1Psryckg==
```

The last element in the entry should be used for the secret key.

Copy the `/etc/named.conf` and `/etc/rndc.key` files where `named` can read them inside of the `chroot` environment and set the ownership of the files.

```
cp /etc/named.conf /etc/rndc.key /var/named/etc/
chown -R named.named /var/named
```

Finally, set up `named` to start automatically at boot. Notice that the startup script uses the *`-u named`* and *`-t /var/named`* parameters to direct the program to switch to the `chroot` environment after startup.

```
#!/bin/bash
# Begin $rc_base/init.d/bind
# Based on sysklogd script from LFS-3.1 and earlier.
# Rewritten by Gerard Beekmans  - gerard@linuxfromscratch.org
source /etc/sysconfig/rc
source $rc_functions
case "$1" in
   start)
      echo "Starting named..."
      loadproc /usr/sbin/named -u named -t /var/named -c /etc/named.conf
      ;;
   stop)
       echo "Stopping named..."
       killproc /usr/sbin/named
       ;;
   restart)
       $0 stop
       sleep 1
       $0 start
       ;;
```

```
   reload)
       echo "Reloading named..."
       /usr/sbin/rndc -c /etc/rndc.conf reload
       ;;

   status)
       statusproc /usr/sbin/named
       ;;
   *)
       echo "Usage: $0 {start|stop|restart|status}"
       exit 1
       ;;
esac
# End $rc_base/init.d/bind


cd /etc/rc.d/init.d
chmod 754 bind
ln -s  ../init.d/bind ../rc0.d/K49bind
ln -s  ../init.d/bind ../rc1.d/K49bind
ln -s  ../init.d/bind ../rc3.d/K49bind
ln -s  ../init.d/bind ../rc3.d/S22bind
ln -s  ../init.d/bind ../rc6.d/K49bind
```

Chapter 9. Removing Unneeded Packages and Files

All the programs are now installed, however there are some programs that we need to remove from a production server. These files include development code such as compilers and linkers, static libraries, and other build utilities. Also included here is `ssh` because there is no need for anyone to to use this program *from* the server.

An alternative to deleting the files below is to move them to `/home` and then place the files in a `tar` file. That file can then be moved to other media before deleting.

```
rm -rf /tools
rm -rf /usr/include
rm -rf /usr/ssl
rm -rf /usr/src
rm -rf /usr/share/man/man3
rm -f  /lib/*a
rm -f  /lib/cpp
rm -f  /usr/lib/*a
rm -f  /bin/install
rm -f  /usr/bin/{aclocal,aclocal-1.7,ar,as,auto*}
rm -f  /usr/bin/{c++*,cc,cpp,flex*,g++,gcc*,i686*}
rm -f  /usr/bin/{install*,ld,libtool*,obj*,patch,pod*}
rm -f  /usr/bin/{ranlib,read*,ssh,ssh-*,telnet,tex*}
rm -f  /usr/bin/{vimtutor,xtrace,yacc,bison,slogin}
```

The total size of all filesystems after removing these files is approximately 250 megabytes. There are other files that could be deleted here. For instance there is really no reason to have man pages or info files on the system and other unnecessary files could be removed. However, even though disk space could be saved, removing these files would not significantly enhance security.

9.1. Final `tripwire` Configuration

Now that the system is cleaned up, the final configuration of `tripwire` can be done and set up for automated checks that can be run by the `fcron` daemon. The main configuration file is `/etc/tripwire/twpol.text`. The policy file for this server is located in Appendix E. This file has been adjusted to take into account only the files on the system.

Initialize `tripwire` with the following steps:

```
cd /etc/tripwire
./twinstall.sh
```

The initialization will create a site key and a local key which are stored in `/etc/tripwire/`. The site key protects the configuration and policy files. The local key protects the `tripwire` database and local report files. These keys are

protected by passwords. Be sure to remember them. It is also important to use good passwords for these keys. This will protect them from an attacker that gets access to the system so that they cannot be altered.

Next run:

```
tripwire --init
```

This makes a baseline of the system that is the foundation for future checks. At this time it is a good idea to make sure the ownership and permissions of `twpol.txt` allow only root to read or update the file.

Checks can be made with:

```
tripwire --check
```

To automate the check put the line

```
9 4 * * *  /usr/sbin/tripwire --check | gpg --armor --clearsign --batch
```

into the `fcron` system table with **fcrontab -e systab**. The output will be mailed to the root user. Create a `/root/.forward` file to have `sendmail` forward the output to the appropriate administrator on the network mail server.

Reports are also saved in `/var/lib/tripwire/report/`. The `tripwire` database can be updated with:

```
tripwire --update --twrfile /var/lib/tripwire/report/<reportname>.twr
```

Finally, if the policy file needs to be updated, edit the `twpol.txt` file and run:

```
twadmin --create-polfile -S site.key /etc/tripwire/twpol.txt
rm /var/lib/tripwire/*.twd
tripwire --init
```

Chapter 10. Checking the Configuration

After the system is built, test it before deployment. The first thing to do is to connect the system to an internal network and check that all the functions (name resolution, network time, email, logging, log rotation, and remote login) work properly.

One check that is important to make is to run **netstat -an --inet** or **lsof -i**. This will display processes that are listening or connected to network ports. All the connections or listening applications should be identified as one of our network applications.

After the functionality is confirmed, test for security. There are several tests that can be made, but there are only two that are really needed for this server: `nmap` and `nessus`. `nmap` is a security scanner that checks for open services from a remote system. `nessus` is a detailed remote security scanner that attempts to use *numerous* known security vulnerabilities to penetrate a system.

`nmap` is available at http://www.insecure.org/. `nessus` is available at http://www.nessus.org/. Installation is left as an exercise for the reader. These applications should *not* be installed on the DNS server.

When making scans, there are several locations from which scans should be made.

- From a system on the same subnet.
- From a system with a public address.
- From a system on the internal network.
- From a designated secondary nameserver.
- From the mail server.

The `nmap` commands are:

```
nmap -sT -I -O -PI -PT -T4 -v  <ip address>
nmap -sU -O -PI -PT -T4 -v <ip address>
```

These commands make TCP and UDP scans of the target system. The only open ports should be those expected for the network or system of the probing system.

A `nessus` scan is done through a graphical interface. When making a scan, the first time through, enabling all plugins, including dangerous ones, is appropriate, but to save time, scans from subsequent systems should concentrate on known open services. The results should indicate that the known services are indeed running. The name server should not be doing recursive queries from the outside network, but should be detected from inside networks. If you chose to enable a public time server, the NTP protocol should be detected, but information about the system, such as operating system version, from the NTP queries should only be available when scanning from the peer time servers.

At this point, the system is ready to be connected to the Internet. If desired, the system may be ported to another system. To do this, start by creating a `tar` file of each partition on the system such as:

```
tar --one-file-system --same-permissions -cjvf root.tar.bz2 /
```

A new system can be built using a system like Knoppix [Knop] to partition the disk, copy the `tar` files and extract them. The new system can be made bootable by running the version of `grub` from the extracted files (not the Knoppix version).

Chapter 11. Ongoing Maintenance of the System

After deployment, the server needs to be maintained. System administration is an ongoing cycle of planning⟶ implementation⟶ monitoring⟶ analyzing. In this paper we have planned the deployment of the server, implemented the system from source, and set up monitoring capabilities. The ongoing tasks that remain start with watching the output of the log files and monitoring the Internet for new developments. There are several web sites that provide ongoing security updates. The most prominent are the CERT Coordination Center [CERT] at Carnegie-Mellon University, the SANS Internet Storm Center [ISC], and the Bugtraq Database [Bugt]. There are also mailing lists such as the SANS NetworkBits or the SANS Consensus Security Alert. Sign up details for these newsletters can be found at the SANS Web site [SANS].

Information on new and existing vulnerabilities can be found in the Common Vulnerabilities and Exposures database [CVE].

Planning for a new package must consider issues such as the nature of the change, including the severity and time sensitivity of the software that needs to be updated. The timing of the change can have as much impact on the overall network as the upgraded software itself.

If a vulnerability is found, the system needs to be updated. Even though everything has been built from scratch, updates are not hard. The new package needs to be downloaded to a system that is compatible with the target hardware that has compilation capabilities for testing. The term compatible basically means that the test system has the same processor type and basic libraries as the server.

After the system has been compiled and tested, a new compilation should be made from a clean extraction of the source code. Before installation, mark the time by touching a file and then complete the installation.

```
touch new-progs
make install
```

The new executable files and libraries can be collected with:

```
find /bin /sbin /lib /usr -newer new-progs |
    xargs tar -jcvf new-progs.tar.bz2
```

The file can then be copied to the server via `scp` and installed, as root, with:

```
cd /
tar -jxvf /home/admin/new-progs.tar.bz2
```

Any daemons that are affected by the changes then need to be restarted with:

```
/etc/rc.d/init.d/<service> restart
```

The only time the system needs to be rebooted should be when a new kernel is installed.

In addition to updating packages, other ongoing maintenance tasks include making backups and updating passwords on a regular basis. Because there are only administrators allowed on the system, updating passwords can be done manually on a regular schedule as defined by the organizational password policy. Password complexity is enforced by the PAM rules. Further checking via a password checking program such as *John the Ripper* [John] could be accomplished, but probably is not needed for this type of server.

Backups for this type of system are very easy. The only data that really needs to be backed up are the DNS configuration and zone files. These are all located in the directory `/var/named/` and are quite small. They also change somewhat infrequently. A backup can be made on a time schedule according to the organization backup policy with the following command:

```
tar -jcvf named-data.tar.bz2 /var/named/
```

The resulting file can be copied to a backup device such as a floppy disk or electronically copied to another system via `scp`.

References

Albitz, Paul and Cricket Liu. *DNS and Bind, Fourth Edition.* Sebastopol, CA: O'Reilly, 2001.

Barrett, Daniel J. and Richard Silverman and Richard Silverman. *OpenSSH, The Secure Shell.* Sebastopol, CA: O'Reilly, 2001.

Beekmans, Gerard. *Linux From Scratch 5.0.* http://www.linuxfromscratch.org.

Beyond Linux From Scratch 5.0. Ed. Larry Lawrence. http://beyond.linuxfromscratch.org.

Bugtraq Database. Security Focus. http://www.securityfocus.com/bid/.

CERT Coordination Center. Carnegie Mellon Software Engineering Institute. http://www.cert.org.

Common Vulnerabilities and Exposures. Mitre Corporation. http://www.cve.mitre.org/.

Costales, Bryan with Eric Allman. *Sendmail, Third Edition.* Sebastopol, CA: O'Reilly, 2003.

Garfinkel, Simson. *PGP: Pretty Good Privacy.* Sebastopol, CA: O'Reilly, 1995.

John the Ripper password cracker. http://www.openwall.com/john/.

Mann, Scott and Ellen L. Mitchell. *Linux Systems Security An Administrator's Guide to Open Source Security Tools.* Upper Saddle River, NJ: Prentice Hall, 2000.

Internet Storm Center. SANS Institute. http://isc.sans.org.

Knoppix Home Page. http://www.knopper.net/knoppix/index-en.html.

Mills, David. *Authentication Options.* http://www.eecis.udel.edu/~mills/ntp/html/authopt.html.

---. *Public NTP Time Servers.* http://www.eecis.udel.edu/~mills/ntp/servers.html.

Morgan, Andrew G. *Linux-PAM System Administrators' Guide.* http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html.

Russell, Rusty. *Packet Filtering HOWTO.* http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html.

SANS Computer Security Newsletters and Digests. SANS Institute. http://www.sans.org/newsletters/.

Security Enhanced Linux. National Security Agency–Central Security Service. http://www.nsa.gov/selinux/.

The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts Consensus. SANS Institute. http:// www.sans.org/top20/.

Viega, John, Matt Messier, and Pravir Chandra. *Network Security with OpenSSL.* Sebastopol, CA: O'Reilly, 2002.

Ziegler, Robert L. amd Carl B. Constantine. *Linux Firewalls, Second Edition.* Indianapolis: New Riders, 2002.

Appendix A. Linux From Scratch `wget` Script

```
ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.57.tar.bz2
ftp://ftp.gnu.org/gnu/automake/automake-1.7.6.tar.bz2
ftp://ftp.gnu.org/gnu/bash/bash-2.05b.tar.gz
ftp://ftp.gnu.org/gnu/binutils/binutils-2.14.tar.bz2
ftp://ftp.gnu.org/gnu/bison/bison-1.875.tar.bz2
ftp://sources.redhat.com/pub/bzip2/v102/bzip2-1.0.2.tar.gz
ftp://ftp.gnu.org/gnu/coreutils/coreutils-5.0.tar.bz2
ftp://ftp.gnu.org/pub/gnu/dejagnu/dejagnu-1.4.3.tar.gz
ftp://ftp.gnu.org/gnu/diffutils/diffutils-2.8.1.tar.gz
http://unc.dl.sourceforge.net/sourceforge/e2fsprogs/
       e2fsprogs-1.34.tar.gz
http://freshmeat.net/redir/expect/2476/url_tgz/expect.tar.gz
ftp://ftp.astron.com/pub/file/file-4.07.tar.gz
ftp://alpha.gnu.org/gnu/findutils/findutils-4.1.20.tar.gz
ftp://ftp.gnu.org/gnu/non-gnu/flex/flex-2.5.4a.tar.gz
ftp://ftp.gnu.org/gnu/gawk/gawk-3.1.3.tar.bz2
ftp://ftp.gnu.org/gnu/gcc/gcc-3.3.1/gcc-core-3.3.1.tar.bz2
ftp://ftp.gnu.org/gnu/gcc/gcc-3.3.1/gcc-g++-3.3.1.tar.bz2
ftp://ftp.gnu.org/gnu/gcc/gcc-3.3.1/gcc-testsuite-3.3.1.tar.bz2
ftp://ftp.gnu.org/pub/gnu/gettext/gettext-0.12.1.tar.gz
ftp://sources.redhat.com/pub/glibc/releases/glibc-2.3.2.tar.bz2
ftp://sources.redhat.com/pub/glibc/releases/
      glibc-linuxthreads-2.3.2.tar.bz2
http://ftp.uos.ac.kr/p/GNU/gnu/grep/grep-2.5.1.tar.bz2
http://ftp.uos.ac.kr/p/GNU/gnu/groff/groff-1.19.tar.gz
ftp://alpha.gnu.org/pub/gnu/grub/grub-0.93.tar.gz
http://ftp.uos.ac.kr/p/GNU/alpha/gzip/gzip-1.3.5.tar.gz
ftp://ftp.gnu.org/gnu/inetutils/inetutils-1.4.2.tar.gz
ftp://ftp.win.tue.nl/pub/home/aeb/linux-local/utils/kbd/
      kbd-1.08.tar.gz
http://freshmeat.net/redir/less/5583/url_tgz/less-381.tar.gz
http://downloads.linuxfromscratch.org/lfs-bootscripts-1.12.tar.bz2
http://www.linuxfromscratch.org/~winkie/downloads/lfs-utils/
       lfs-utils-0.3.tar.bz2
http://ftp.uos.ac.kr/p/GNU/gnu/libtool/libtool-1.5.tar.gz
http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.25.tar.bz2
ftp://ftp.seindal.dk/gnu/m4-1.4.tar.gz
ftp://ftp.gnu.org/gnu/make/make-3.80.tar.bz2
http://downloads.linuxfromscratch.org/MAKEDEV-1.7.bz2
ftp://ftp.kernel.org/pub/linux/utils/man/man-1.5m2.tar.bz2
ftp://ftp.kernel.org/pub/linux/docs/manpages/man-pages-1.60.tar.bz2
ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.3.tar.gz
http://www.tazenda.demon.co.uk/phil/net-tools/net-tools-1.60.tar.bz2
ftp://ftp.gnu.org/pub/gnu/patch/patch-2.5.4.tar.gz
http://csociety-ftp.ecn.purdue.edu/pub/gentoo/distfiles/
       perl-5.8.0.tar.gz
ftp://ftp.cistron.nl/pub/people/svm/procinfo-18.tar.gz
http://procps.sf.net/procps-3.1.11.tar.gz
http://unc.dl.sourceforge.net/sourceforge/psmisc/psmisc-21.3.tar.gz
http://ftp.uos.ac.kr/p/GNU/gnu/sed/sed-4.0.7.tar.gz
ftp://ftp.pld.org.pl/software/shadow/old/shadow-4.0.3.tar.bz2
http://www.ibiblio.org/pub/Linux/system/daemons/sysklogd-1.4.1.tar.gz
```

```
ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.85.tar.gz
http://ftp.uos.ac.kr/p/GNU/alpha/tar/tar-1.13.25.tar.gz
http://umn.dl.sourceforge.net/sourceforge/tcl/tcl8.4.4-src.tar.gz
ftp://ftp.gnu.org/gnu/texinfo/texinfo-4.6.tar.gz
http://ftp.cwi.nl/aeb/util-linux/util-linux-2.12.tar.gz
ftp://ftp.vim.org/pub/vim/unix/vim-6.2.tar.bz2
http://www.gzip.org/zlib/zlib-1.1.4.tar.bz2
http://downloads.linuxfromscratch.org/bash-2.05b-2.patch
http://downloads.linuxfromscratch.org/bison-1.875-attribute.patch
http://downloads.linuxfromscratch.org/coreutils-5.0-hostname-2.patch
http://downloads.linuxfromscratch.org/coreutils-5.0-uname.patch
http://downloads.linuxfromscratch.org/expect-5.39.0.patch
http://downloads.linuxfromscratch.org/gawk-3.1.3.patch
http://downloads.linuxfromscratch.org/gcc-3.3.1-no_fixincludes-2.patch
http://downloads.linuxfromscratch.org/gcc-3.3.1-specs-2.patch
http://downloads.linuxfromscratch.org/gcc-3.3.1-suppress-libiberty.patch
http://downloads.linuxfromscratch.org/glibc-2.3.2-sscanf-1.patch
http://downloads.linuxfromscratch.org/grub-0.93-gcc33-1.patch
http://downloads.linuxfromscratch.org/kbd-1.08.patch
http://downloads.linuxfromscratch.org/man-1.5m2-80cols.patch
http://downloads.linuxfromscratch.org/man-1.5m2-manpath.patch
http://downloads.linuxfromscratch.org/man-1.5m2-pager.patch
http://downloads.linuxfromscratch.org/ncurses-5.3-etip-2.patch
http://downloads.linuxfromscratch.org/ncurses-5.3-vsscanf.patch
http://downloads.linuxfromscratch.org/
       net-tools-1.60-miitool-gcc33-1.patch
http://downloads.linuxfromscratch.org/perl-5.8.0-libc-3.patch
http://downloads.linuxfromscratch.org/procps-3.1.11.patch
http://downloads.linuxfromscratch.org/shadow-4.0.3-newgrp-fix.patch
http://downloads.linuxfromscratch.org/zlib-1.1.4-vsnprintf.patch
```

Note: Several lines in the above file have been wrapped to fit the page. The file can be downloaded directly from http://www.linuxfromscratch.org/~bdubbs/lfs-file.wget.

Appendix B. LFS Build Scripts

B.1. LFS Tools Scripts

B.1.1. make-LFS

```
#!/bin/sh

LFS=/mnt/lfs
SCRIPTDIR=‘pwd‘
PACKAGES=$SCRIPTDIR/lfsfiles
TIMEFMT=’%1R Elapsed Time - ’
export TZ=CST6CDT

#SKIP=yes
SKIP=no
export CHECK=no
#export CHECK=check

unset CFLAGS CXXFLAGS

# Make sure $LFS is mounted
mounted=‘mount|grep $LFS‘

if test "${mounted:-x}" = x; then
  echo $LFS must be mounted
  exit 1
fi

if [ ‘whoami‘ = root ]; then
####
if [ x$SKIP = "xno" ]; then
####
  dirs=$LFS/*
  for dir in $dirs; do
    if [ $dir != $LFS/lost+found ]; then rm -rf $dir; fi
  done
  mkdir $LFS/tools
  chown -R lfs $LFS/tools
####
fi
####
fi


# Now start over as user lfs
if [ ‘whoami‘ != lfs ]; then
  su - lfs -c $0
  rc=$?; if [ $rc -ne 0 ]; then echo "Failure in Chapter 5"; exit 1; fi;
  # After completing below, we need to be root again to chroot
  # Continue with chapter 6 script
  mkdir -p /tools/src/scripts
```

```
  # Copy all scripts with a 6 in the name
  cp -f $SCRIPTDIR/*6* /tools/src/scripts/
  /usr/sbin/chroot $LFS /tools/bin/env -i SKIP=$SKIP \
     CHECK=$CHECK HOME=/root /tools/src/scripts/chapter6.sh
  rc=$?; if [ $rc -ne 0 ]; then echo "Failure in Chapter 6"; exit 1; fi;

  echo Done
  exit 0
fi

# When we get here, we must be lfs
# Source user lfs' .bash_profile
source ~/.bash_profile

# Create $LFS/tools/src for building packages and copy packages there
pushd $LFS/tools
mkdir -p src
cd src
####
if [ x$SKIP = "xno" ]; then
####

cp $PACKAGES/* .

$SCRIPTDIR/make-binutils-pass1 || exit 1
$SCRIPTDIR/make-gcc            || exit 1
$SCRIPTDIR/make-kernel-headers || exit 1
$SCRIPTDIR/make-glibc          || exit 1
$SCRIPTDIR/lock-glibc          || exit 1
$SCRIPTDIR/make-tcl            || exit 1
$SCRIPTDIR/make-expect         || exit 1
$SCRIPTDIR/make-dejagnu        || exit 1
$SCRIPTDIR/make-gcc-pass2      || exit 1
$SCRIPTDIR/make-binutils-pass2 || exit 1
$SCRIPTDIR/make-gawk           || exit 1
$SCRIPTDIR/make-coreutils      || exit 1
$SCRIPTDIR/make-bzip2          || exit 1
$SCRIPTDIR/make-gzip           || exit 1
$SCRIPTDIR/make-diffutils      || exit 1
$SCRIPTDIR/make-findutils      || exit 1
$SCRIPTDIR/make-make           || exit 1
$SCRIPTDIR/make-grep           || exit 1
$SCRIPTDIR/make-sed            || exit 1
$SCRIPTDIR/make-gettext        || exit 1
$SCRIPTDIR/make-ncurses        || exit 1
$SCRIPTDIR/make-patch          || exit 1
$SCRIPTDIR/make-tar            || exit 1
$SCRIPTDIR/make-texinfo        || exit 1
$SCRIPTDIR/make-bash           || exit 1
$SCRIPTDIR/make-util-linux     || exit 1
$SCRIPTDIR/make-perl           || exit 1

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

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

# Now we have to change back to root

####
fi
####

exit 0
```

B.1.2. make-binutils-pass1

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

######################
# Installing binutils-pass-1

BINUTILS=binutils-2.14
PROGRAM=$BINUTILS.tar.bz2
TAROPTS=-jxf

TITLE="$PROGRAM - Pass 1"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar $TAROPTS $PROGRAM

cd $BINUTILS
{ time \
  {
    echo Making $TITLE
    date
    mkdir ../binutils-build &&
    cd ../binutils-build &&
    ../binutils-2.14/configure --prefix=/tools --disable-nls &&
    make configure-host &&
    make LDFLAGS="-all-static" &&
    make install &&
    make -C ld clean &&
    make -C ld LDFLAGS="-all-static" LIB_PATH=/tools/lib
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM` size" | tee -a ../build.log
after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
```

```
     | tee -a ../build.log

cd ..

#######################

exit 0
```

B.1.3. make-gcc

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

GNUCC=gcc-3.3.1
GNUCCCORE=gcc-core-3.3.1
PROGRAM=$GNUCCCORE.tar.bz2
TAROPTS=-jxf

TITLE="$GNUCC-Pass 1"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar $TAROPTS $PROGRAM
cd $GNUCC
{ time \
  {
    echo Making $TITLE
    date
    mkdir ../gcc-build &&
    cd ../gcc-build &&
    ../gcc-3.3.1/configure --prefix=/tools --with-local-prefix=/tools \
      --disable-nls --enable-shared --enable-languages=c &&
    make BOOT_LDFLAGS="-static" bootstrap &&
    make install &&
    ln -sf gcc /tools/bin/cc
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM` size" | tee -a ../build.log
after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $GNUCC
rm -rf gcc-build

exit 0
```

B.1.4. make-kernel-headers

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

KERNEL=linux-2.4.25

TITLE="$KERNEL-headers"
TIMEFORMAT="$TIMEFMT $TITLE"
PROGRAM=$KERNEL.tar.bz2
TAROPTS=-jxf

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar $TAROPTS $PROGRAM
cd $KERNEL
{ time \
  {
    echo Making $TITLE
    date
    make mrproper &&
    make include/linux/version.h &&
    make symlinks &&
    mkdir /tools/include/asm &&
    cp include/asm/* /tools/include/asm &&
    cp -R include/asm-generic /tools/include &&
    cp -R include/linux /tools/include &&
    touch /tools/include/linux/autoconf.h
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM` size" | tee -a ../build.log
after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $KERNEL

exit 0
```

B.1.5. make-glibc

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

GLIBC=glibc-2.3.2
LINUXTHREADS=glibc-linuxthreads-2.3.2
PROGRAM=$GLIBC.tar.bz2
TAROPTS=-jxf

TITLE="$GLIBC"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar $TAROPTS $PROGRAM
cd $GLIBC
tar $TAROPTS ../$LINUXTHREADS.tar.bz2
{ time \
  {
    echo Making $TITLE
    date
    mkdir /tools/etc &&
    touch /tools/etc/ld.so.conf &&
    patch -Np1 -i ../glibc-2.3.2-sscanf-1.patch &&
    mkdir ../glibc-build &&
    cd ../glibc-build &&

    ../glibc-2.3.2/configure --prefix=/tools --disable-profile \
    --enable-add-ons --with-headers=/tools/include \
    --with-binutils=/tools/bin  --without-gd &&

    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    mkdir -p /tools/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 es_MX -f ISO-8859-1 es_MX &&
    localedef -i fr_FR -f ISO-8859-1 fr_FR &&
    localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro &&
    localedef -i it_IT -f ISO-8859-1 it_IT &&
    localedef -i ja_JP -f EUC-JP ja_JP
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM` size" | tee -a ../build.log
echo "`du -k ../$LINUXTHREADS.tar.bz2` size" | tee -a ../build.log
after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
```

```
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $GLIBC
rm -rf glibc-build

exit 0
```

B.1.6. lock-glibc

```
#!/bin/bash

########################
# Locking in glibc

TIMEFMT='%1R Elapsed Time - '
TITLE="Locking-in-glibc"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

cd binutils-build
{ time \
  {
    echo Making $TITLE
    date
    make -C ld install &&
    SPECFILE=/tools/lib/gcc-lib/*/*/specs &&
    sed -e 's@/lib/ld-linux.so.2@/tools/lib/ld-linux.so.2@g' \
            $SPECFILE > tempspecfile &&
    mv -f tempspecfile $SPECFILE &&
    unset SPECFILE &&
    rm -f /tools/lib/gcc-lib/*/*/include/{pthread.h,bits/sigthread.h}
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $BINUTILS
rm -rf binutils-build

echo 'main(){}' > dummy.c
echo "gcc dummy.c" | tee -a build.log
gcc dummy.c 2>&1 | tee -a build.log
```

```
rc=$?; if [ $rc -ne 0 ]; then echo "Failure in Toolchain check"; exit 1; fi;

readelf -l a.out | grep ': /tools' | tee -a build.log

rm dummy.c a.out

exit 0
```

B.1.7. make-tcl

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing tcl

TCL=tcl8.4.4
PROGRAM=$TCL-src.tar.gz
TAROPTS=-zxf


TITLE="$TCL"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar $TAROPTS $PROGRAM
cd $TCL
{ time \
  {
    echo Making $TITLE
    date
    cd unix &&
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then TZ=UTC make test; fi &&
    make install &&
    ln -sf tclsh8.4 /tools/bin/tclsh
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM` size" | tee -a ../build.log
after2=`df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ../..
```

```
######################

exit 0
```

B.1.8. make-expect

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

######################
# Installing tcl

EXPECT=expect-5.40
EXPECTDIR=$EXPECT
PROGRAM=expect.tar.gz
TAROPTS=-zxf

TITLE="$EXPECT"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar $TAROPTS $PROGRAM
cd $EXPECTDIR
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../expect-5.39.0.patch &&
    ./configure --prefix=/tools --with-tcl=/tools/lib --with-x=no &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make test; fi &&
    make SCRIPTS="" install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $EXPECTDIR
rm -rf tcl8.4.4
######################

exit 0
```

B.1.9. make-dejagnu

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing tcl

PROGRAM=dejagnu-1.4.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $PROGRAM
#######################

exit 0
```

B.1.10. make-gcc-pass2

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing gcc-pass-2

PROGRAM=gcc-3.3.1
```

```
TITLE="$PROGRAM - Pass 2"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf gcc-core-3.3.1.tar.bz2
tar -jxf gcc-g++-3.3.1.tar.bz2
tar -jxf gcc-testsuite-3.3.1.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../gcc-3.3.1-no_fixincludes-2.patch &&
    patch -Np1 -i ../gcc-3.3.1-specs-2.patch &&
    mkdir ../gcc-build &&
    cd ../gcc-build &&

    ../gcc-3.3.1/configure --prefix=/tools \
        --with-local-prefix=/tools \
        --enable-clocale=gnu \
        --enable-shared \
        --enable-threads=posix \
        --enable-__cxa_atexit \
        --enable-languages=c,c++ &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make -k check; fi &&
    ../gcc-3.3.1/contrib/test_summary &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../gcc-core-3.3.1.tar.bz2‘ size" | tee -a ../build.log
echo "‘du -k ../gcc-g++-3.3.1.tar.bz2‘ size" | tee -a ../build.log
echo "‘du -k ../gcc-testsuite-3.3.1.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $PROGRAM
rm -rf gcc-build
#########################

exit 0
```

B.1.11. make-binutils-pass2

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing binutils-pass-2

PROGRAM=binutils-2.14

TITLE="$PROGRAM - Pass 2"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    mkdir ../binutils-build &&
    cd ../binutils-build &&
    ../binutils-2.14/configure --prefix=/tools \
        --enable-shared --with-lib-path=/tools/lib &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    make -C ld clean &&
    make -C ld LIB_PATH=/usr/lib:/lib
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log


cd ..

#######################

exit 0
```

B.1.12. make-coreutils

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing coreutils

PROGRAM=coreutils-5.0

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make RUN_EXPENSIVE_TESTS=yes check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.13. make-bzip2

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing gawk
```

```
PROGRAM=bzip2-1.0.2

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make PREFIX=/tools install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $PROGRAM
#########################

exit 0
```

B.1.14. make-gzip

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#########################
# Installing gzip

PROGRAM=gzip-1.3.5

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
```

```
    date
    ./configure --prefix=/tools  &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

rm -rf $PROGRAM
########################

exit 0
```

B.1.15. make-diffutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing diffutils

PROGRAM=diffutils-2.8.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;
```

```
echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.16. make-findutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing findutils

PROGRAM=findutils-4.1.20

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################
```

```
exit 0
```

B.1.17. make-make

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing coreutils

PROGRAM=make-3.80

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.18. make-grep

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing coreutils

PROGRAM=grep-2.5.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools \
       --disable-perl-regexp --with-included-regex &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.1.19. make-sed

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing sed
```

```
PROGRAM=sed-4.0.7

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.20. make-gettext

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing gettext

PROGRAM=gettext-0.12.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz
```

```
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make
    rc=$?
    if [ x$CHECK = "xcheck" -a $rc -eq 0 ]; then make -k check; fi
    if [ $rc -eq 0 ]; then make install; fi
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.21. make-ncurses

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing ncurses

PROGRAM=ncurses-5.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../ncurses-5.3-etip-2.patch &&
    patch -Np1 -i ../ncurses-5.3-vsscanf.patch &&
```

```
    ./configure --prefix=/tools --with-shared --without-debug \
        --without-ada --enable-overwrite &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log


cd ..
rm -rf $PROGRAM
#########################

exit 0
```

B.1.22. make-patch

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#########################
# Installing patch

PROGRAM=patch-2.5.4

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    CPPFLAGS=-D_GNU_SOURCE ./configure --prefix=/tools &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;
```

```
echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.23. make-tar

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing coreutils

PROGRAM=tar-1.13.25

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################
```

```
exit 0
```

B.1.24. make-texinfo

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing texinfo

PROGRAM=texinfo-4.6

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.1.25. make-bash

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing bash

PROGRAM=bash-2.05b

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../bash-2.05b-2.patch
    ./configure --prefix=/tools &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make tests; fi &&
    make install &&
    ln -s bash /tools/bin/sh
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.26. make-util-linux

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
```

```
# Installing util-linux

PROGRAM=util-linux-2.12

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    cp configure configure.backup &&
    sed "s@/usr/include@/tools/include@g" configure.backup > configure &&
    ./configure &&
    make -C lib &&
    make -C mount mount umount &&
    make -C text-utils more &&
    cp mount/{,u}mount text-utils/more /tools/bin
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.1.27. make-perl

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing perl

PROGRAM=perl-5.8.0

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"
```

```
before2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../perl-5.8.0-libc-3.patch &&
    chmod u+w hints/linux.sh &&
    echo ’static_ext="IO re Fcntl"’ >> hints/linux.sh &&

    ./configure.gnu --prefix=/tools &&
    make  perl utilities &&
    cp perl pod/pod2man /tools/bin &&
    mkdir -p /tools/lib/perl5/5.8.0 &&
    cp -R lib/* /tools/lib/perl5/5.8.0
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k $LFS | grep $LFS | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2. Main System Scripts

B.2.1. chapter6.sh

```
#!/tools/bin/bash

echo "Starting Chapter 6"

export PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin
export TZ=CST6CDT

set +h

export CHECK=check
```

```
TIMEFMT='%1R Elapsed Time - '

if [ x$SKIP = "x" ]; then
  echo "The SKIP environment variable must be defined as 'yes' or 'no'"
  exit 1
fi

SCRIPTDIR=/tools/src/scripts

cd /tools/src

####
if [ x$SKIP = "xno" ]; then
####

# Change ownershp of everything to root
chown -R 0:0 /tools

# Create the directories
mkdir -p /{bin,boot,dev/{pts,shm},etc/opt,home,lib,mnt,proc}
mkdir -p /{root,sbin,tmp,usr/local,var,opt}
for dirname in /usr /usr/local
    do
    mkdir $dirname/{bin,etc,include,lib,sbin,share,src}
    ln -s share/{man,doc,info} $dirname
    mkdir $dirname/share/{dict,doc,info,locale,man}
    mkdir $dirname/share/{nls,misc,terminfo,zoneinfo}
    mkdir $dirname/share/man/man{1,2,3,4,5,6,7,8}
done
mkdir /var/{lock,log,mail,run,spool}
mkdir -p /var/{tmp,opt,cache,lib/misc,local}
mkdir /opt/{bin,doc,include,info}
mkdir -p /opt/{lib,man/man{1,2,3,4,5,6,7,8}}

chmod 0750 /root
chmod 1777 /tmp /var/tmp

####
fi
####

# Mount proc (not sure this is needed for script)
mount proc /proc -t proc
mount devpts /dev/pts -t devpts

####
if [ x$SKIP = "xno" ]; then
####

# Make essential symlinks
ln -s /tools/bin/{bash,pwd,cat,stty} /bin
ln -s /tools/bin/perl /usr/bin
ln -s /tools/bin/libgcc_s.so.1 /usr/lib
ln -s bash /bin/sh
```

```
# Password and group files

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

cat > /etc/group << "EOF"
root:x:0:
kmem:x:3
tty:x:4
floppy:x:7
disk:x:8
dialout:x:10
EOF

TITLE="Making Devices"
TIMEFORMAT="$TIMEFMT $TITLE"
before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

# Make devices
bzcat MAKEDEV-1.7.bz2 > /dev/MAKEDEV
chmod 754 /dev/MAKEDEV

time { \
  {
    echo $TITLE
    date
    pushd /dev
    ./MAKEDEV -v std
    ./MAKEDEV -v fd
    ./MAKEDEV -v ptmx
    ./MAKEDEV -v console
    ./MAKEDEV -v fd0
    ./MAKEDEV -v hda hde
    ./MAKEDEV -v rtc
    popd
  }
} 2>&1 | tee -a build.log

after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a build.log

###################
$SCRIPTDIR/make6-kernel-headers  || \
                          { echo "mk6-kernel-headers failed";exit 1;}
$SCRIPTDIR/make6-man-pages       || { echo "make6-man-pages failed"; exit 1; }
$SCRIPTDIR/make6-glibc           || { echo "make6-glibc failed"; exit 1; }
$SCRIPTDIR/make6-readj-toolchain || \
                          { echo "mk6-readj-toolchain failed";exit 1;}
$SCRIPTDIR/make6-binutils        || { echo "make6-bunutils failed"; exit 1; }
$SCRIPTDIR/make6-gcc             || { echo "make6-gcc failed"; exit 1; }
$SCRIPTDIR/make6-coreutils       || { echo "make6-coreutils failed"; exit 1; }
$SCRIPTDIR/make6-zlib            || { echo "make6-zlib failed"; exit 1; }
$SCRIPTDIR/make6-lfs-utils       || { echo "make6-lfs-utils failed"; exit 1; }
```

```
$SCRIPTDIR/make6-findutils       || { echo "make6-findutils failed"; exit 1; }
$SCRIPTDIR/make6-gawk            || { echo "make6-gawk failed"; exit 1; }
$SCRIPTDIR/make6-ncurses         || { echo "make6-ncurses failed"; exit 1; }
$SCRIPTDIR/make6-vim             || { echo "make6-vim failed"; exit 1; }
$SCRIPTDIR/make6-m4              || { echo "make6-m4 failed"; exit 1; }
$SCRIPTDIR/make6-bison           || { echo "make6-bison failed"; exit 1; }
$SCRIPTDIR/make6-less            || { echo "make6-less failed"; exit 1; }
$SCRIPTDIR/make6-groff           || { echo "make6-groff failed"; exit 1; }
$SCRIPTDIR/make6-sed             || { echo "make6-sed failed"; exit 1; }
$SCRIPTDIR/make6-flex            || { echo "make6-flex failed"; exit 1; }
$SCRIPTDIR/make6-gettext         || { echo "make6-gettext failed"; exit 1; }
$SCRIPTDIR/make6-nettools        || { echo "make6-nettools failed"; exit 1; }
$SCRIPTDIR/make6-inetutils       || { echo "make6-inetutils failed"; exit 1; }
####################

# Set up basic networking

echo "127.0.0.1 $(hostname) localhost" > /etc/hosts

tar -jxf lfs-utils-0.3.tar.bz2
cd lfs-utils-0.3
cp -f etc/{services,protocols} /etc
cd ..
rm -rf lfs-utils-0.3

####################
$SCRIPTDIR/make6-perl       || { echo "make6-perl failed" ; exit 1; }
$SCRIPTDIR/make6-texinfo    || { echo "make6-texinfo failed" ; exit 1; }
$SCRIPTDIR/make6-autoconf   || { echo "make6-autoconf failed" ; exit 1; }
$SCRIPTDIR/make6-automake   || { echo "make6-automake failed" ; exit 1; }
$SCRIPTDIR/make6-bash       || { echo "make6-bash failed" ; exit 1; }
$SCRIPTDIR/make6-file       || { echo "make6-file failed" ; exit 1; }
$SCRIPTDIR/make6-libtool    || { echo "make6-libtool failed" ; exit 1; }
$SCRIPTDIR/make6-bzip2      || { echo "make6-bzip2 failed" ; exit 1; }
$SCRIPTDIR/make6-kbd        || { echo "make6-kbd failed" ; exit 1; }
$SCRIPTDIR/make6-diffutils  || { echo "make6-diffutils failed" ; exit 1; }
$SCRIPTDIR/make6-e2fsprogs  || { echo "make6-e2fsprogs failed" ; exit 1; }
$SCRIPTDIR/make6-grep       || { echo "make6-grep failed" ; exit 1; }
$SCRIPTDIR/make6-grub       || { echo "make6-grub failed" ; exit 1; }
$SCRIPTDIR/make6-gzip       || { echo "make6-gzip failed" ; exit 1; }
$SCRIPTDIR/make6-man        || { echo "make6-man failed" ; exit 1; }
$SCRIPTDIR/make6-make       || { echo "make6-make failed" ; exit 1; }
$SCRIPTDIR/make6-patch      || { echo "make6-patch failed" ; exit 1; }
$SCRIPTDIR/make6-procinfo   || { echo "make6-procinfo failed" ; exit 1; }
$SCRIPTDIR/make6-procps     || { echo "make6-procps failed" ; exit 1; }
$SCRIPTDIR/make6-psmisc     || { echo "make6-psmisc failed" ; exit 1; }
$SCRIPTDIR/make6-shadow     || { echo "make6-shadow failed" ; exit 1; }
$SCRIPTDIR/make6-sysklogd   || { echo "make6-sysklogd failed" ; exit 1; }
$SCRIPTDIR/make6-sysvinit   || { echo "make6-sysvinit failed" ; exit 1; }
$SCRIPTDIR/make6-tar        || { echo "make6-tar failed" ; exit 1; }
$SCRIPTDIR/make6-util-linux || { echo "make6-util-linux failed" ; exit 1; }
####################

####
fi
```

```
####
export PATH=/bin:/usr/bin:/sbin:/usr/sbin

###
if [ x$SKIP = "xno" ]; then
###

$SCRIPTDIR/make6-bootscripts || { echo "make6-bootscripts" ; exit 1; }

########

#ln -s path/to/keymap /usr/share/kbd/keymaps/defkeymap.map.gz
#loadkeys -m /usr/share/kbd/keymaps/defkeymap.map.gz > \
#    /usr/src/linux-2.4.22/drivers/char/defkeymap.c

# Configuring the setclock script
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock
EOF

echo "HOSTNAME=lfs" > /etc/sysconfig/network

# Creating the /etc/hosts file
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1   localhost

# End /etc/hosts (network card version)
EOF

# Configuring the network script

cat >> /etc/sysconfig/network << "EOF"
GATEWAY=192.168.0.1
GATEWAY_IF=eth0
EOF

cat > /etc/sysconfig/network-devices/ifconfig.eth0 << "EOF"
ONBOOT=yes
IP=192.168.0.2
NETMASK=255.255.255.0
BROADCAST=192.168.0.255
EOF

cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# filesystem   mount-point fs-type    options             dump fsck-order

/dev/hda2      /boot       ext3       noexec,nodev,auto     1    2
```

```
/dev/hda5      /           ext3       defaults              1    1
/dev/hda6      /var        ext3       defaults,noexec,nodev 1    2
/dev/hda7      /var/named  ext3       defaults,noexec       1    2
/dev/hda8      /tmp        ext3       defaults,noexec,nodev 1    2
/dev/hda9      /home       ext3       defaults,nodev        1    2

/dev/hda3      swap        swap       defaults              0    0

pts            /dev/pts    devpts     gid=4,mode=620        0    0
proc           /proc       proc       defaults              0    0
shm            /dev/shm    tmpfs      defaults              0    0

/dev/cdrom     /mnt/cdrom  udf,iso9660 noauto,owner,ro      0    0
/dev/fd0       /mnt/floppy auto        noauto,owner         0    0

# End /etc/fstab
EOF

find /{,usr/,usr/local/}{bin,sbin,lib} -type f -exec /usr/bin/strip \
    --strip-debug '{}' ';'
echo 'date' > /etc/lfs-build-date

###
fi
###

$SCRIPTDIR/make6-kernel || { echo "make6-kernel failed" ; exit 1; }

umount /dev/pts
umount /proc

echo "Copy kernel and System.map to /boot"
echo "Set up /boot/grub/menu.lst"
echo "Reboot"

exit 0
```

B.2.2. make6-kernel-headers

```
#!/tools/bin/bash

TIMEFMT='%1R Elapsed Time - '
KERNEL=linux-2.4.25

TITLE="$KERNEL-headers"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $KERNEL.tar.bz2
cd $KERNEL
{ time \
```

```
  {
    echo Making $TITLE
    date
    make mrproper &&
    make include/linux/version.h &&
    make symlinks &&
    cp -HR include/asm /usr/include &&
    cp -R include/asm-generic /usr/include &&
    cp -R include/linux /usr/include &&
    touch /usr/include/linux/autoconf.h
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$KERNEL.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $TITLE" \
     | tee -a ../build.log

cd ..
rm -rf $KERNEL
```

B.2.3. make6-man-pages

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing man pages

PROGRAM=man-pages-1.60

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;
```

```
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#########################

exit 0
```

B.2.4. make6-glibc

```
#!/tools/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

PROGRAM=glibc-2.3.2
LINUXTHREADS=glibc-linuxthreads-2.3.2

TITLE=$PROGRAM
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2
cd $PROGRAM
tar -jxf ../$LINUXTHREADS.tar.bz2
CHECK=check
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../glibc-2.3.2-sscanf-1.patch &&
    mkdir ../glibc-build &&
    cd ../glibc-build &&

    ../glibc-2.3.2/configure --prefix=/usr --disable-profile \
       --enable-add-ons --libexecdir=/usr/bin                \
       --with-headers=/usr/include  &&

    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    mkdir -p /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 es_MX -f ISO-8859-1 es_MX &&
    localedef -i fr_FR -f ISO-8859-1 fr_FR &&
    localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro &&
```

```
    localedef -i it_IT -f ISO-8859-1 it_IT &&
    localedef -i ja_JP -f EUC-JP ja_JP &&
    make -C ../glibc-2.3.2/linuxthreads/man &&
    make -C ../glibc-2.3.2/linuxthreads/man install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
echo "‘du -k ../$LINUXTHREADS.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..

# Configure glibc

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

passwd: files
group: files
shadow: files

publickey: 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
EOF

# Set the time zone
cp --remove-destination /usr/share/zoneinfo/America/Chicago /etc/localtime

# Configure dynamic loader
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf

/usr/local/lib
/opt/lib

# End /etc/ld.so.conf
EOF

# Remove the source and build files
```

```
rm -rf $PROGRAM
rm -rf glibc-build

########################

exit 0
```

B.2.5. make6-readj-toolchain

```
#!/bin/bash

# Re-adjust the toolchain

TITLE="Readjusting Toolchain"
TIMEFMT='%1R Elapsed Time - '
TIMEFORMAT="$TIMEFMT $TITLE"

BINUTILS=binutils-2.14
before2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

time { \
  {
    echo $TITLE
    date
    pushd binutils-build &&
    make -C ld INSTALL=/tools/bin/install install &&
    popd &&

    rm -rf binutils-build &&
    rm -rf $BINUTILS &&

    SPECFILE=/tools/lib/gcc-lib/*/*/specs &&
    sed -e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g' \
        $SPECFILE > newspecfile  &&
    mv newspecfile $SPECFILE  &&
    unset SPECFILE
  }
} 2>&1 | tee -a build.log   # Note - not '../build.log' here

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

after2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $TITLE" \
     | tee -a build.log

# Toolchain Check
echo 'main(){}' > dummy.c
echo "gcc dummy.c" | tee -a build.log
gcc dummy.c 2>&1 | tee -a build.log

if [ $PIPESTATUS -ne 0 ]; then
    echo "Failure in Toolchain check";
```

```
    exit 1;
fi;

readelf -l a.out | grep ': /lib' | tee -a build.log
rm dummy.c a.out
```

B.2.6. make6-binutils

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing binutil
PROGRAM=binutils-2.14

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar -jxf $PROGRAM.tar.bz2

CHECK=check

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    mkdir ../binutils-build &&
    cd ../binutils-build &&

    ../$PROGRAM/configure --prefix=/usr --enable-shared &&
    make toolsdir=/usr &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make tooldir=/usr install &&
    cp ../$PROGRAM/include/libiberty.h /usr/include
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM.tar.bz2` size" | tee -a ../build.log
after2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf binutils-build
```

```
rm -rf $PROGRAM
#######################

exit 0
```

B.2.7. make6-gcc

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing gcc

PROGRAM=gcc-3.3.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf gcc-core-3.3.1.tar.bz2
tar -jxf gcc-g++-3.3.1.tar.bz2
tar -jxf gcc-testsuite-3.3.1.tar.bz2

CHECK=check

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../gcc-3.3.1-no_fixincludes-2.patch &&
    patch -Np1 -i ../gcc-3.3.1-suppress-libiberty.patch &&
    mkdir ../gcc-build &&
    cd ../gcc-build &&

    ../gcc-3.3.1/configure --prefix=/usr --enable-shared \
        --enable-threads=posix --enable-__cxa_atexit     \
        --enable-clocale=gnu --enable-languages=c,c++ &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make -k check; fi &&
    make install &&
    ln -s ../usr/bin/cpp /lib &&
    ln -s gcc /usr/bin/cc
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../gcc-core-3.3.1.tar.bz2' size" | tee -a ../build.log
echo "'du -k ../gcc-g++-3.3.1.tar.bz2' size" | tee -a ../build.log
```

```
echo "‘du -k ../gcc-testsuite-3.3.1.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf gcc-build
rm -rf $PROGRAM
#######################

echo ’main(){}’ > dummy.c
echo "gcc dummy.c" | tee -a build.log
gcc dummy.c 2>&1 | tee -a build.log

if [ $PIPESTATUS -ne 0 ]; then
   echo "Failure in Chapter 6 GCC  check";
   exit 1;
fi;

readelf -l a.out | grep ’: /lib’ | tee -a build.log
rm dummy.c a.out

exit 0
```

B.2.8. make6-coreutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing zlib

PROGRAM=coreutils-5.0

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    set +h

    patch -Np1 -i ../coreutils-5.0-uname.patch &&
    patch -Np1 -i ../coreutils-5.0-hostname-2.patch &&
```

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

    make install-root &&

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

    if [ x$CHECK = "xcheck" ]; then make check-root; fi &&
    if [ x$CHECK = "xcheck" ];
    then
       su -c "make RUN_EXPENSIVE_TESTS=yes check";
    fi &&
    sed -i.bak '/dummy/d' /etc/passwd /etc/group &&

    make install &&

    mv /usr/bin/{basename,cat,chgrp,chmod,chown,cp,dd,df} /bin &&
    mv /usr/bin/{dir,dircolors,du,date,echo,false,head} /bin &&
    mv /usr/bin/{install,ln,ls,mkdir,mkfifo,mknod,mv,pwd} /bin &&
    mv /usr/bin/{rm,rmdir,shred,sync,sleep,stty,su,test} /bin &&
    mv /usr/bin/{touch,true,uname,vdir} /bin &&
    mv /usr/bin/chroot /usr/sbin &&

    ln -s test /bin/[  &&
    ln -s ../../bin/install /usr/bin
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.9. make6-zlib

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing zlib
```

```
PROGRAM=zlib-1.1.4

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../zlib-1.1.4-vsnprintf.patch &&

    ./configure --prefix=/usr --shared  &&
    make &&
    make install &&

    make clean &&
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make test; fi &&
    make install &&

    mv /usr/lib/libz.so.* /lib &&
    ln -sf ../../lib/libz.so.1 /usr/lib/libz.so &&
    cp zlib.3 /usr/share/man/man3
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.10. make6-lfs-utils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
```

```
# Installing lfs-utils

PROGRAM=lfs-utils-0.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.11. make6-findutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing findutils

PROGRAM=findutils-4.1.20

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz
```

```
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --libexecdir=/usr/bin  &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.12. make6-gawk

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing gawk

PROGRAM=gawk-3.1.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../gawk-3.1.3.patch &&

    ./configure --prefix=/usr --libexecdir=/usr/bin  &&
    make &&
```

```
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.13. make6-ncurses

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing ncurses

PROGRAM=ncurses-5.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../ncurses-5.3-etip-2.patch &&
    patch -Np1 -i ../ncurses-5.3-vsscanf.patch &&

    ./configure --prefix=/usr --with-shared --without-debug &&
    make &&
    make install &&

    chmod 755 /usr/lib/*.5.3 &&
    chmod 644 /usr/lib/libncurses++.a &&
    mv /usr/lib/libncurses.so.5* /lib &&
    ln -sf ../../lib/libncurses.so.5 /usr/lib/libncurses.so &&
```

```
    ln -sf libncurses.so /usr/lib/libcurses.so
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.14. make6-vim

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing vim

PROGRAM=vim-6.2
SRCDIR=vim62

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    echo ’#define SYS_VIMRC_FILE "/etc/vimrc"’ >> src/feature.h &&
    echo ’#define SYS_GVIMRC_FILE "/etc/gvimrc"’ >> src/feature.h &&
    ./configure --prefix=/usr  &&
    make &&
    make install &&

    ln -sf vim /usr/bin/vi &&

    cat > /root/.vimrc << "EOF"
" Begin /root/.vimrc
```

```
set nocompatible
set bs=2

" End /root/.vimrc
EOF

  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#########################

exit 0
```

B.2.15. make6-m4

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#########################
# Installing m4

PROGRAM=m4-1.4
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr  &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log
```

```
if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
########################

exit 0
```

B.2.16. make-bison

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing bison

PROGRAM=bison-1.875
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../bison-1.875-attribute.patch &&
    ./configure --prefix=/usr  &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log
```

```
cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.17. make6-less

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing less

PROGRAM=less-381
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz
cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --bindir=/bin --sysconfdir=/etc &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.18. make6-groff

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing groff

PROGRAM=groff-1.19
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    PAGE=letter ./configure --prefix=/usr  &&
    make &&
    make install &&
    ln -s soelim /usr/bin/zsoelim &&
    ln -s eqn /usr/bin/geqn &&
    ln -s tbl /usr/bin/gtbl
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.19. make6-sed

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’
```

```
#######################
# Installing sed

PROGRAM=sed-4.0.7
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --bindir=/bin &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.20. make6-flex

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing flex

PROGRAM=flex-2.5.4a
SRCDIR=flex-2.5.4

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"
```

```
before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr  &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make bigcheck; fi &&
    make install &&
    ln -s libfl.a /usr/lib/libl.a &&
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
    chmod 755 /usr/bin/lex
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.21. make6-gettext

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing gettext

PROGRAM=gettext-0.12.1
SRCDIR=$PROGRAM
```

```
TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr  &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.22. make6-nettools

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing net-tools

PROGRAM=net-tools-1.60
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $SRCDIR
```

```
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../net-tools-1.60-miitool-gcc33-1.patch &&
    yes "" | make config  &&
    make &&
    make update
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#########################

exit 0
```

B.2.23. make6-inetutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing inetutils

PROGRAM=inetutils-1.4.2
SRCDIR=$PROGRAM

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $SRCDIR
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --disable-syslogd \
        --libexecdir=/usr/sbin --disable-logger \
         --sysconfdir=/etc --localstatedir=/var \
         --disable-whois --disable-servers &&
```

```
    make &&
    make install &&
    mv /usr/bin/ping /bin
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $SRCDIR
#######################

exit 0
```

B.2.24. make6-perl

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing perl

PROGRAM=perl-5.8.0

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure.gnu --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make test; echo ""; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
```

```
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.25. make6-texinfo

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing texinfo

PROGRAM=texinfo-4.6

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    make TEXMF=/usr/share/texmf install-tex
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################
```

```
exit 0
```

B.2.26. make6-autoconf

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing autoconf

PROGRAM=autoconf-2.57

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.27. make6-automake

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing automake

PROGRAM=automake-1.7.6

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3`

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    ln -s automake-1.7 /usr/share/automake
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "`du -k ../$PROGRAM.tar.bz2` size" | tee -a ../build.log
after2=`df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3`
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.28. make6-bash

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

########################
# Installing bash
```

```
PROGRAM=bash-2.05b

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../bash-2.05b-2.patch
    ./configure --prefix=/usr --bindir=/bin &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make tests; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.29. make6-file

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing file

PROGRAM=file-4.07


TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘
```

```
tar -zxf $PROGRAM.tar.gz || exit 1

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --datadir=/usr/share/misc &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
########################

exit 0
```

B.2.30. make6-libtool

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

########################
# Installing libtool

PROGRAM=libtool-1.5

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
```

```
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.31. make6-bzib2

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing bzip2

PROGRAM=bzip2-1.0.2

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make -f Makefile-libbz2_so &&
    make clean &&
    make &&
    make install &&
    cp bzip2-shared /bin/bzip2 &&
    cp -a libbz2.so* /lib &&
    ln -s ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so &&
    rm /usr/bin/{bunzip2,bzcat,bzip2} &&
    mv /usr/bin/{bzip2recover,bzless,bzmore} /bin &&
    ln -s bzip2 /bin/bunzip2 &&
    ln -s bzip2 /bin/bzcat
  }
} 2>&1 | tee -a ../build.log
```

```
if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.32. make6-diffutils

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing diffutils

PROGRAM=diffutils-2.8.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
```

```
#######################

exit 0
```

B.2.33. make6-kbd

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing kbd

PROGRAM=kbd-1.08

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../kbd-1.08.patch &&
    ./configure &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.34. make6-e2fsprogs

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing e2fsprogs

PROGRAM=e2fsprogs-1.34

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    mkdir ../e2fsprogs-build &&
    cd ../e2fsprogs-build &&
    ../$PROGRAM/configure --prefix=/usr --with-root-prefix="" \
       --enable-elf-shlibs &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install &&
    make install-libs
  }
} 2>&1 | tee -a ../build.log

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf e2fsprogs-build
rm -rf $PROGRAM
#######################

exit 0
```

B.2.35. make6-grep

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '
```

```
#######################
# Installing grep

PROGRAM=grep-2.5.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --bindir=/bin --with-included-regex &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.36. make6-grub

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing grub

PROGRAM=grub-0.93

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘
```

```
tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../grub-0.93-gcc33-1.patch &&
    ./configure --prefix=/usr &&
    make &&
    make install &&
    mkdir -p /boot/grub &&
    #cp /usr/share/grub/i386-pc/stage{1,2} /boot/grub
    cp /usr/share/grub/i386-pc/* /boot/grub
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.37. make6-gzip

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing gzip

PROGRAM=gzip-1.3.5

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
```

```
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    cp gzexe.in{,.backup} &&
    sed 's%"BINDIR"%/bin%' gzexe.in.backup > gzexe.in &&
    make &&
    make install &&
    mv /usr/bin/gzip /bin &&
    rm /usr/bin/{gunzip,zcat} &&
    ln -s gzip /bin/gunzip &&
    ln -s gzip /bin/zcat &&
    ln -s gunzip /bin/uncompress
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.38. make6-man

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing man

PROGRAM=man-1.5m2

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../man-1.5m2-manpath.patch &&
```

```
    patch -Np1 -i ../man-1.5m2-pager.patch &&
    patch -Np1 -i ../man-1.5m2-80cols.patch &&
    ./configure -default -confdir=/etc &&

    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.39. make6-make

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing make

PROGRAM=make-3.80

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log
```

```
if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.40. make6-patch

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing patch

PROGRAM=patch-2.5.4

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    CPPFLAGS=-D_GNU_SOURCE ./configure --prefix=/usr &&
    make &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################
```

```
exit 0
```

B.2.41. make6-procinfo

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing procinfo

PROGRAM=procinfo-18

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make LDLIBS=-lncurses &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.42. make6-procps

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing procps

PROGRAM=procps-3.1.11

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    patch -Np1 -i ../$PROGRAM.patch &&
    make &&
    make install &&
    rm /lib/libproc.so
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.43. make6-psmisc

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing psmisc
```

```
PROGRAM=psmisc-21.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --exec-prefix=/ &&
    make &&
    make install &&
    ln -s killall /bin/pidof
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#########################

exit 0
```

B.2.44. make6-shadow

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#########################
# Installing shadow

PROGRAM=shadow-4.0.3

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2
```

```
cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    touch /var/run/utmp /var/log/{btmp,lastlog,wtmp} &&
    chmod 644 /var/run/utmp /var/log/{btmp,lastlog,wtmp} &&
    touch /usr/bin/passwd &&

    patch -Np1 -i ../shadow-4.0.3-newgrp-fix.patch &&

    ./configure --prefix=/usr --libdir=/usr/lib --enable-shared  &&
    make &&
    make install &&

    cp etc/{limits,login.access} /etc &&
    sed -e 's%/var/spool/mail%/var/mail%' \
        -e 's%#MD5_CRYPT_ENAB.no%MD5_CRYPT_ENAB yes%' \
        etc/login.defs.linux > /etc/login.defs &&

    ln -s vipw /usr/sbin/vigr &&
    rm /bin/vipw &&
    mv /bin/sg /usr/bin &&
    mv /usr/lib/lib{shadow,misc}.so.0* /lib &&
    ln -sf ../../lib/libshadow.so.0 /usr/lib/libshadow.so &&
    ln -sf ../../lib/libmisc.so.0 /usr/lib/libmisc.so &&
    rm /bin/groups &&

    /usr/sbin/pwconv &&
    /usr/sbin/grpconv
  }
} 2>&1 | tee -a ../build.log

echo "'du -k ../$PROGRAM.tar.bz2' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.45. make6-sysklogd

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing sysklogd
```

```
PROGRAM=sysklogd-1.4.1

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make &&
    make install &&
    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
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.46. make6-sysvinit

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’
```

```
######################
# Installing sysvinit

PROGRAM=sysvinit-2.85

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    cp src/init.c{,.backup} &&
    sed ’s/Sending processes/Sending processes started by init/g’ \
        src/init.c.backup > src/init.c &&

    make -C src &&
    make -C src install &&
    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:S12:wait:/etc/rc.d/init.d/rc 1
l3:345:wait:/etc/rc.d/init.d/rc 3
l6:6:wait:/etc/rc.d/init.d/rc 6

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

su:S0126:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600

# End /etc/inittab
EOF
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
```

```
rm -rf $PROGRAM
#######################

exit 0
```

B.2.47. make6-tar

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing tar

PROGRAM=tar-1.13.25

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    ./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/bin &&
    make &&
    if [ x$CHECK = "xcheck" ]; then make check; fi &&
    make install
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "'du -k ../$PROGRAM.tar.gz' size" | tee -a ../build.log
after2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3'
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.48. make6-util-linux

```
#!/bin/bash

TIMEFMT='%1R Elapsed Time - '

#######################
# Installing sysvinit

PROGRAM=sysvinit-2.85

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

before2='df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d' ' -f3'

tar -zxf $PROGRAM.tar.gz

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    cp src/init.c{,.backup} &&
    sed 's/Sending processes/Sending processes started by init/g' \
        src/init.c.backup > src/init.c &&

    make -C src &&
    make -C src install &&
    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:S12:wait:/etc/rc.d/init.d/rc 1
l3:345:wait:/etc/rc.d/init.d/rc 3
l6:6:wait:/etc/rc.d/init.d/rc 6

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

su:S0126:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600

# End /etc/inittab
EOF
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;
```

```
echo "‘du -k ../$PROGRAM.tar.gz‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log

cd ..
rm -rf $PROGRAM
#######################

exit 0
```

B.2.49. make6-kernel

```
#!/bin/bash

TIMEFMT=’%1R Elapsed Time - ’

#######################
# Installing kernel

VERSION=2.4.25
PROGRAM=linux-$VERSION

TITLE="$PROGRAM"
TIMEFORMAT="$TIMEFMT $TITLE"

DATE=‘date +-%Y%m%d‘

before2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d’ ’ -f3‘

tar -jxf $PROGRAM.tar.bz2

cd $PROGRAM
{ time \
  {
    echo Making $TITLE
    date
    make mrproper &&
    make EXTRAVERSION=$DATE menuconfig &&
    make EXTRAVERSION=$DATE dep &&
    make EXTRAVERSION=$DATE bzImage &&
    cp arch/i386/boot/bzImage /boot/linux-${VERSION}${DATE} &&
    cp System.map /boot/System.map-${VERSION}${DATE}
  }
} 2>&1 | tee -a ../build.log

if [ $PIPESTATUS -ne 0 ]; then exit 1; fi;

echo "‘du -k ../$PROGRAM.tar.bz2‘ size" | tee -a ../build.log
after2=‘df -k / | grep / | sed -e "s/ \{2,\}/ /g" | cut -d" " -f3‘
echo "$(($after2-$before2)) kilobytes / build size - $PROGRAM" \
     | tee -a ../build.log
```

```
cd ..
rm -rf $PROGRAM
#########################

exit 0
```

Appendix C. Initialization Script for `iptables`

```
#!/bin/bash

IPTABLES=/usr/sbin/iptables

# Set up a default DROP policy for the built-in chains.
# If we modify and re-run the script mid-session then (because we
# have a default DROP policy), what happens is that there is a small
# time period when packets are denied until the new rules are back
# in place. There is no period, however small, when packets we
# don't want are allowed.
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP

# We want to remove all rules and pre-exisiting user defined
# chains and zero the counters before we implement new rules.
$IPTABLES -F
$IPTABLES -X
$IPTABLES -Z
$IPTABLES -t nat -F
$IPTABLES -t mangle -F

## ============================================================
## Some definitions:

BROADCAST="192.168.0.255"

LOOPBACK="127.0.0.0/8"
CLASS_A="10.0.0.0/8"
CLASS_B="172.16.0.0/12"
CLASS_C="192.168.0.0/16"
CLASS_D_MULTICAST="224.0.0.0/4"
CLASS_E_RESERVED_NET="240.0.0.0/5"

P_PORTS="0:1023"
UP_PORTS="1024:65535"
TR_SRC_PORTS="32769:65535"
TR_DEST_PORTS="33434:33523"

IFACE=eth+

## ============================================================
## Kernel flags
# To dynamically change kernel parameters and variables on the fly you need
# CONFIG_SYSCTL defined in your kernel.

# Disable response to ping.
#/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

# Disable response to broadcasts.
# You don't want yourself becoming a Smurf amplifier.
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

```
# Don’t accept source routed packets. Attackers can use source
# routing to generate traffic pretending to be from inside your
# network, but which is routed back along the path from which it
# came, namely outside, so attackers can compromise your
# network. Source routing is rarely used for legitimate purposes.
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route

# Disable ICMP redirect acceptance. ICMP redirects can be used to
# alter your routing tables, possibly to a bad end.
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects

# Enable bad error message protection.
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# Log spoofed packets, source routed packets, redirect packets.
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians

# Make sure that IP forwarding is turned off. We only want this
# for a multi-homed host used as a router.
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward

## ============================================================
# RULES

## LOOPBACK
# Allow unlimited traffic on the loopback interface.
$IPTABLES -A INPUT  -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

## Let the package through early if we are already established
$IPTABLES -A INPUT  -i $IFACE -m state \
          --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -m state \
          --state ESTABLISHED,RELATED -j ACCEPT

# SYN-FLOODING PROTECTION
# This rule maximizes the rate of incoming connections. In order to
# do this we divert tcp packets with the SYN bit set off to a
# user-defined chain. Up to limit-burst connections can arrive in
# 1/limit seconds ..... in this case 4 connections in one second.
# After this, one of the burst is regained every second and connections
# are allowed again. The default limit is 3/hour. The default limit
# burst is 5.

$IPTABLES -N syn-flood
$IPTABLES -A syn-flood -m limit --limit 1/s --limit-burst 4 -j RETURN
$IPTABLES -A syn-flood -j DROP

$IPTABLES -A INPUT -i $IFACE -p tcp --syn -j syn-flood

## Make sure NEW tcp connections are SYN packets
$IPTABLES -A INPUT -i $IFACE -p tcp ! --syn -m state --state NEW -j DROP

## FRAGMENTS
```

```
# Sending lots of non-first fragments was what allowed Jolt2
# to effectively "drown" Firewall-1. Fragments can be overlapped,
# and the subsequent interpretation of such fragments is very
# OS-dependent (see this paper for details).

# Don't trust any fragments.
# Log fragments just to see if we get any, and deny them too.
$IPTABLES -A INPUT -i $IFACE -f -j LOG \
          --log-prefix "IPTABLES FRAGMENTS: " --log-level 6
$IPTABLES -A INPUT -i $IFACE -f -j DROP

## SPOOFING
# Most of this anti-spoofing stuff is theoretically not really
# necessary with the flags we have set in the kernel above
# but you never know there isn't a bug somewhere in your IP stack.

# Refuse packets claiming to be from a Class A private network.
$IPTABLES -A INPUT  -i $IFACE -s $CLASS_A -j DROP

# Refuse packets claiming to be from a Class B private network.
$IPTABLES -A INPUT  -i $IFACE -s $CLASS_B -j DROP

# We are on a Class C network.

# Refuse Class D multicast addresses. Multicast is illegal as a
# source address. We also drop legal multicast destination so we
# don't log them later.
$IPTABLES -A INPUT -i $IFACE -s $CLASS_D_MULTICAST -j DROP
$IPTABLES -A INPUT -i $IFACE -d $CLASS_D_MULTICAST -j DROP
# Refuse Class E reserved IP addresses.
$IPTABLES -A INPUT -i $IFACE -s $CLASS_E_RESERVED_NET -j DROP

# Refuse packets claiming to be to the loopback interface.
# Refusing packets claiming to be to the loopback interface
# protects against source quench, whereby a machine can be told
# to slow itself down by an icmp source quench to the loopback.
$IPTABLES -A INPUT  -i $IFACE -d $LOOPBACK -j DROP

## DNS
# NOTE: DNS uses tcp for zone transfers, for transfers greater
# than 512 bytes (possible, but unusual -- so we deny it),
$IPTABLES -A INPUT  -i $IFACE -p udp --dport 53 -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p udp            -j ACCEPT

NAMESERVER_1=200.0.0.2
$IPTABLES -A INPUT  -i $IFACE -p tcp -s $NAMESERVER_1 \
          --sport 53 -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp -s $NAMESERVER_1 \
          --dport 53-j ACCEPT

## SSH
# Allow ssh inbound only
$IPTABLES -A INPUT  -i $IFACE -p tcp --dport 22 \
          -m state --state NEW -j ACCEPT
```

```
## NTP
$IPTABLES -A INPUT  -i $IFACE -p udp --dport 123 \
          -m state --state NEW  -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p udp --dport 123 \
          -m state --state NEW  -j ACCEPT

## SMTP
MAILSERVER=192.168.0.10
# Allow smtp outbound to our mail server.
$IPTABLES -A OUTPUT -o $IFACE -p tcp -d $MAILSERVER --dport 25 \
          -m state --state NEW -j ACCEPT

## AUTH server
# Reject ident probes with a tcp reset.
# Needed for a mail host that won't accept or delays
# mail if we just drop its ident probe.
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 113 \
          -j REJECT --reject-with tcp-reset

# ICMP
# We accept some icmp requests including pings
$IPTABLES -A INPUT  -i $IFACE -p icmp --icmp-type destination-unreachable \
          -j ACCEPT
$IPTABLES -A INPUT  -i $IFACE -p icmp --icmp-type time-exceeded          \
          -j ACCEPT
$IPTABLES -A INPUT  -i $IFACE -p icmp --icmp-type echo-request           \
          -j ACCEPT
$IPTABLES -A INPUT  -i $IFACE -p icmp --icmp-type echo-reply             \
          -j ACCEPT

# We always allow icmp out.
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j ACCEPT

# Drop some stuff without logging

## NETBIOS
NETBIOS=137:139
$IPTABLES -A INPUT -i $IFACE -p udp --dport $NETBIOS -j DROP

# Other stuff from MS we don't want to log
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 445 -j DROP
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 135 -j DROP
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 1433 -j DROP

# After smb, we can now refuse broadcast address packets.
$IPTABLES -A INPUT -i $IFACE -d $BROADCAST -j LOG --log-level 6
$IPTABLES -A INPUT -i $IFACE -d $BROADCAST -j DROP

## LOGGING
# You don't have to split up your logging like we do below,
# but this way we can grep for things in the logs more easily.

# Any udp not already allowed is logged and then dropped.
$IPTABLES -A INPUT  -i $IFACE -p udp -j LOG \
          --log-prefix "IPTABLES UDP-IN: " --log-level 6
```

```
$IPTABLES -A INPUT  -i $IFACE -p udp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p udp -j LOG \
          --log-prefix "IPTABLES UDP-OUT: " --log-level 6
$IPTABLES -A OUTPUT -o $IFACE -p udp -j DROP

# Any icmp not already allowed is logged and then dropped.
$IPTABLES -A INPUT  -i $IFACE -p icmp -j LOG \
          --log-prefix "IPTABLES ICMP-IN: " --log-level 6
$IPTABLES -A INPUT  -i $IFACE -p icmp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j LOG \
          --log-prefix "IPTABLES ICMP-OUT: " --log-level 6
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j DROP

# Any tcp not already allowed is logged and then dropped.
$IPTABLES -A INPUT  -i $IFACE -p tcp -j LOG \
          --log-prefix "IPTABLES TCP-IN: " --log-level 6
$IPTABLES -A INPUT  -i $IFACE -p tcp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p tcp -j LOG \
          --log-prefix "IPTABLES TCP-OUT: " --log-level 6
$IPTABLES -A OUTPUT -o $IFACE -p tcp -j DROP

# Anything else not already allowed is logged and then dropped.
# It will be dropped by the default policy anyway ........
# but let's be paranoid.
$IPTABLES -A INPUT  -i $IFACE -j LOG \
          --log-prefix "IPTABLES PROTOCOL-X-IN: " --log-level 6
$IPTABLES -A INPUT  -i $IFACE -j DROP
$IPTABLES -A OUTPUT -o $IFACE -j LOG \
          --log-prefix "IPTABLES PROTOCOL-X-OUT: " --log-level 6
$IPTABLES -A OUTPUT -o $IFACE -j DROP
```

Appendix D. DNS Zone Files

D.1. Root Zone

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.INTERNIC.NET
;       -OR-                    RS.INTERNIC.NET
;
;       last update:    Jan 29, 2004
;       related version of root zone:   2004012900
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     192.228.79.201
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
```

```
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; operated by VeriSign, Inc.
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     192.58.128.30
;
; operated by RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; operated by ICANN
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

D.2. Localhost Reverse Zone

```
$TTL 3h
@  IN SOA example.edu. dnsadmin.example.edu. (
                2004022502   ; serial
                3h           ; refresh
                1h           ; retry
                1w           ; expire
                1h )         ; Negative caching TTL

   IN NS ns1.example.edu.

1  IN PTR localhost.
```

D.3. External Forward Zone

```
$TTL 3h

@           IN SOA example.edu. dnsadmin.example.edu. (
               2004022501      ; Serial
               3h              ; Refresh
               1h              ; Retry
               1w              ; Expire
               1h )            ; Negative caching

            IN NS   ns1.example.edu.
            IN MX 0 mail.example.edu.

localhost   IN A     127.0.0.1

ns1         IN A     200.1.1.194
tick        IN CNAME ns1.example.edu.

www         IN A     200.1.1.195
tock        IN CNAME www.example.edu.

mail        IN A     200.1.1.196
time        IN CNAME mail.example.edu.
```

D.4. External Reverse Zone

```
$TTL 1d

@  IN SOA example.edu. dnsadmin.example.edu. (
      2004022601      ; Serial
      3h              ; Refresh
      1h              ; Retry
      1w              ; Expire
      1h )            ; Negative caching

   IN NS ns1.example.edu.

194     IN PTR ns1.example.edu.
195     IN PTR www.example.edu.
196     IN PTR mail.example.edu.

$GENERATE 200-222 $ PTR user$.example.edu.
```

D.5. Internal Forward Zone

```
$TTL 3h

@           IN SOA example.edu. dnsadmin.example.edu. (
               2004022503      ; Serial
               3h              ; Refresh
               1h              ; Retry
               1w              ; Expire
               1h )            ; Negative caching

            IN NS   ns1.example.edu.
            IN MX 0 mail.example.edu.

;localhost
localhost   IN A     127.0.0.1

;dmz -- net 0
ns1         IN A     192.168.0.3
            IN A     192.168.0.13
tick        IN CNAME ns1.example.edu.

www         IN A     192.168.0.4
            IN A     192.168.0.14
tock        IN CNAME www.example.edu.

mail        IN A     192.168.0.5
            IN A     192.168.0.15
time        IN CNAME mail.example.edu.

;faculty -- net 1
;   Blade 5
banana      IN A     192.168.1.7
            IN A     192.168.1.17
syslog      IN CNAME banana.example.edu.

;  PE 650 2
date        IN A     192.168.1.3
            IN A     192.168.1.13
ns2         IN CNAME date.example.edu.

;  Opteron
fig         IN A     192.168.1.20
            IN A     192.168.1.120

;  PE 4400
orange      IN A     192.168.1.6
            IN A     192.168.1.16

george      IN A     192.168.1.103
ralph       IN A     192.168.1.101

;student  -- net 2
;  Blade 1
ds1         IN A     192.168.2.30
```

```
            IN A     192.168.2.130

;  Blade 2
ds2         IN A     192.168.2.31
            IN A     192.168.2.131

;  Blade 3
pear        IN A     192.168.2.32
            IN A     192.168.2.132

;  Blade 6
apple       IN A     192.168.2.4
            IN A     192.168.2.14

;  PE 650 1
grape       IN A     192.168.2.8
            IN A     192.168.2.18
oracle      IN CNAME grape.example.edu.
```

D.6. Internal 192.168.0 Reverse Zone

```
$TTL 3h

@  IN SOA example.edu. dnsadmin.example.edu. (
      2004022501      ; Serial
      3h              ; Refresh
      1h              ; Retry
      1w              ; Expire
      1h )            ; Negative caching

   IN NS ns1.example.edu.

1     IN PTR ns1.example.edu.
13    IN PTR ns1.example.edu.

4     IN PTR www.example.edu.
14    IN PTR www.example.edu.

5     IN PTR mail.example.edu.
15    IN PTR mail.example.edu.
```

D.7. Internal 192.168.1 Reverse Zone

```
$TTL 3h

@  IN SOA example.edu. dnsadmin.example.edu. (
      2004022502      ; Serial
      3h              ; Refresh
      1h              ; Retry
      1w              ; Expire
      1h )            ; Negative caching

   IN NS ns1.example.edu.

3    IN PTR date.example.edu.
13   IN PTR date.example.edu.

20   IN PTR fig.example.edu.
120  IN PTR fig.example.edu.

6    IN PTR orange.example.edu.
16   IN PTR orange.example.edu.

101  IN PTR ralph.example.edu.
103  IN PTR george.example.edu.
```

D.8. Internal 192.168.2 Reverse Zone

```
$TTL 3h

@  IN SOA example.edu. dnsadmin.example.edu. (
      2004022502      ; Serial
      3h              ; Refresh
      1h              ; Retry
      1w              ; Expire
      1h )            ; Negative caching

   IN NS ns1.example.edu.

30      IN PTR ds1.example.edu.
130     IN PTR ds1.example.edu.

31      IN PTR ds2.example.edu.
131     IN PTR ds2.example.edu.

40      IN PTR apple.example.edu.
14      IN PTR apple.example.edu.

8       IN PTR grape.example.edu.
18      IN PTR grape.example.edu.
```

Appendix E. Policy File for `tripwire`

```
  ##########################################################################
 #                                                                          ##
##########################################################################  #
#                                                                          # #
# Global Variable Definitions                                              # #
#                                                                          # #
# These are defined at install time by the installation script.            # #
# You may manually edit these if you are using this file directly          # #
# and not from the installation script itself.                             # #
#                                                                          ##
##########################################################################

@@section GLOBAL
TWDOCS="/usr/share/doc/tripwire";
TWBIN="/usr/sbin";
TWPOL="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME=phobos;

@@section FS
SEC_CRIT      = $(IgnoreNone)-SHa ;  # Critical files that cannot change
SEC_SUID      = $(IgnoreNone)-SHa ;  # Binaries with the SUID or SGID flags set
SEC_BIN       = $(ReadOnly) ;        # Binaries that should not change
SEC_CONFIG    = $(Dynamic) ;         # Config files that are changed
                                     #   infrequently but accessed often
SEC_LOG       = $(Growing) ;         # Files that grow, but that should
                                     #   never change ownership
SEC_INVARIANT = +tpug ;              # Directories that should never
                                     #   change permission or ownership
SIG_LOW       = 33 ;                 # Non-critical files that are of
                                     #   minimal security impact
SIG_MED       = 66 ;                 # Non-critical files that are of
                                     #   significant security impact
SIG_HI        = 100 ;                # Critical files that are significant
                                     #   points of vulnerability


# Tripwire Binaries
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI)
)
{
  $(TWBIN)/siggen                    -> $(SEC_BIN) ;
  $(TWBIN)/tripwire                  -> $(SEC_BIN) ;
  $(TWBIN)/twadmin                   -> $(SEC_BIN) ;
  $(TWBIN)/twprint                   -> $(SEC_BIN) ;
}
```

```
# Tripwire Data Files - Configuration Files, Policy Files, Keys, Reports,
#                       Databases
(
  rulename = "Tripwire Data Files",
  severity = $(SIG_HI)
)
{
  # NOTE: We remove the inode attribute because when Tripwire creates a
  # backup, it does so by renaming the old file and creating a new one
  # (which will have a new inode number).  Inode is left turned on for
  # keys, which shouldn't ever change.

  # NOTE: The first integrity check triggers this rule and each integrity
  # check afterward triggers this rule until a database update is run,
  # since the database file does not exist before that point.

  $(TWDB)                                -> $(SEC_CONFIG) -i ;
  $(TWPOL)/tw.pol                        -> $(SEC_BIN) -i ;
  $(TWPOL)/tw.cfg                        -> $(SEC_BIN) -i ;
  $(TWLKEY)/$(HOSTNAME)-local.key        -> $(SEC_BIN) ;
  $(TWSKEY)/site.key                     -> $(SEC_BIN) ;

  #don't scan the individual reports
  $(TWREPORT)                            -> $(SEC_CONFIG) (recurse=0) ;
}


# Tripwire HQ Connector Binaries
#(
#  rulename = "Tripwire HQ Connector Binaries",
#  severity = $(SIG_HI)
#)
#{
#  $(TWBIN)/hqagent                      -> $(SEC_BIN) ;
#}
#
# Tripwire HQ Connector - Configuration Files, Keys, and Logs

  ############################################################################
 #                                                                          ##
################################################################################ #
#                                                                            # #
# Note: File locations here are different than in a stock HQ Connector       # #
# installation.  This is because Tripwire 2.3 uses a different path          # #
# structure than Tripwire 2.2.1.                                             # #
#                                                                            # #
# You may need to update your HQ Agent configuration file (or this           # #
# policy file) to correct the paths.  We have attempted to support the       # #
# FHS standard here by placing the HQ Agent files similarly to the way       # #
# Tripwire 2.3 places them.                                                  # #
#                                                                            ##
################################################################################

#(
#  rulename = "Tripwire HQ Connector Data Files",
```

```
#  severity = $(SIG_HI)
#)
#{
#   ##########################################################################
#  ##########################################################################
#  # NOTE: Removing the inode attribute because when Tripwire creates a ##
#  # backup it does so by renaming the old file and creating a new one  ##
#  # (which will have a new inode number).  Leaving inode turned on for ##
#  # keys which shouldn't ever change.                                    ##
#  ##########################################################################
#
#  $(TWBIN)/agent.cfg                   -> $(SEC_BIN) -i ;
#  $(TWLKEY)/authentication.key         -> $(SEC_BIN) ;
#  $(TWDB)/tasks.dat                    -> $(SEC_CONFIG) ;
#  $(TWDB)/schedule.dat                 -> $(SEC_CONFIG) ;
#
#  # Uncomment if you have agent logging enabled.
#  #/var/log/tripwire/agent.log      -> $(SEC_LOG) ;
#}



# Commonly accessed directories that should remain static with regards to
# owner and group
(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
  /                                   -> $(SEC_INVARIANT) (recurse = 0) ;
  /home                               -> $(SEC_INVARIANT) (recurse = 0) ;
  /etc                                -> $(SEC_INVARIANT) (recurse = 0) ;
}
  ################################################
 #                                               ##
################################################ #
#                                               # #
# File System and Disk Administration Programs # #
#                                               ##
################################################

(
  rulename = "File System and Disk Administration Programs",
  severity = $(SIG_HI)
)
{
  /sbin/badblocks                     -> $(SEC_CRIT) ;
  /sbin/e2fsck                        -> $(SEC_CRIT) ;
  /sbin/debugfs                       -> $(SEC_CRIT) ;
  /sbin/dumpe2fs                      -> $(SEC_CRIT) ;
  /sbin/e2label                       -> $(SEC_CRIT) ;
  /sbin/fdisk                         -> $(SEC_CRIT) ;
  /sbin/fsck                          -> $(SEC_CRIT) ;
  /sbin/fsck.ext2                     -> $(SEC_CRIT) ;
  /sbin/fsck.ext3                     -> $(SEC_CRIT) ;
```

```
  /sbin/hdparm                          -> $(SEC_CRIT) ;
  /sbin/mke2fs                          -> $(SEC_CRIT) ;
  /sbin/mkfs                            -> $(SEC_CRIT) ;
  /sbin/mkfs.ext2                       -> $(SEC_CRIT) ;
  /sbin/mkfs.ext3                       -> $(SEC_CRIT) ;
  /sbin/mkswap                          -> $(SEC_CRIT) ;
  /sbin/parted                          -> $(SEC_CRIT) ;
  /sbin/resize2fs                       -> $(SEC_CRIT) ;
  /sbin/sfdisk                          -> $(SEC_CRIT) ;
  /sbin/tune2fs                         -> $(SEC_CRIT) ;
  /bin/mount                            -> $(SEC_CRIT) ;
  /bin/umount                           -> $(SEC_CRIT) ;
  /bin/touch                            -> $(SEC_CRIT) ;
  /bin/mkdir                            -> $(SEC_CRIT) ;
  /bin/mknod                            -> $(SEC_CRIT) ;
  /bin/mktemp                           -> $(SEC_CRIT) ;
  /bin/rm                               -> $(SEC_CRIT) ;
  /bin/rmdir                            -> $(SEC_CRIT) ;
  /bin/chgrp                            -> $(SEC_CRIT) ;
  /bin/chmod                            -> $(SEC_CRIT) ;
  /bin/chown                            -> $(SEC_CRIT) ;
  /bin/cp                               -> $(SEC_CRIT) ;
}

  ###################################
 #                                  ##
################################### #
#                                  # #
# Kernel Administration Programs # #
#                                  ##
###################################

(
  rulename = "Kernel Administration Programs",
  severity = $(SIG_HI)
)
{
  /sbin/ctrlaltdel                      -> $(SEC_CRIT) ;
  /usr/sbin/klogd                       -> $(SEC_CRIT) ;
  /sbin/ldconfig                        -> $(SEC_CRIT) ;
  /sbin/sysctl                          -> $(SEC_CRIT) ;
}

  ######################
 #                     ##
###################### #
#                    # #
# Networking Programs # #
#                    ##
######################

(
  rulename = "Networking Programs",
  severity = $(SIG_HI)
)
```

```
{
  /sbin/arp                                -> $(SEC_CRIT) ;
  /sbin/agetty                             -> $(SEC_CRIT) ;
  /sbin/ifconfig                           -> $(SEC_CRIT) ;
  /usr/sbin/iptables                       -> $(SEC_CRIT) ;
  /usr/sbin/iptables-restore               -> $(SEC_CRIT) ;
  /usr/sbin/iptables-save                  -> $(SEC_CRIT) ;
  /sbin/rarp                               -> $(SEC_CRIT) ;
  /sbin/route                              -> $(SEC_CRIT) ;
  /bin/ping                                -> $(SEC_CRIT) ;
}

  ###################################
 #                                  ##
################################### #
#                                  # #
# System Administration Programs # #
#                                  ##
###################################

(
  rulename = "System Administration Programs",
  severity = $(SIG_HI)
)
{
  /sbin/halt                               -> $(SEC_CRIT) ;
  /sbin/init                               -> $(SEC_CRIT) ;
  /sbin/killall5                           -> $(SEC_CRIT) ;
  /sbin/shutdown                           -> $(SEC_CRIT) ;
  /sbin/sulogin                            -> $(SEC_CRIT) ;
  /sbin/swapon                             -> $(SEC_CRIT) ;
  /usr/sbin/syslog-ng                      -> $(SEC_CRIT) ;
  /sbin/unix_chkpwd                        -> $(SEC_CRIT) ;
  /bin/pwd                                 -> $(SEC_CRIT) ;
  /bin/uname                               -> $(SEC_CRIT) ;
}

  ########################################
 #                                       ##
######################################## #
#                                      # #
# Hardware and Device Control Programs # #
#                                      ##
########################################
(
  rulename = "Hardware and Device Control Programs",
  severity = $(SIG_HI)
)
{
  /sbin/hwclock                            -> $(SEC_CRIT) ;
  /sbin/losetup                            -> $(SEC_CRIT) ;
}

  ###############################
 #                             ##
```

```
############################### #
#                             # #
# System Information Programs # #
#                             ##
###############################
(
  rulename = "System Information Programs",
  severity = $(SIG_HI)
)
{
  /sbin/kernelversion                -> $(SEC_CRIT) ;
  /sbin/runlevel                     -> $(SEC_CRIT) ;
}

  ###################################
 #                                  ##
################################### #
#                                 # #
# Application Information Programs # #
#                                 ##
###################################

(
  rulename = "Application Information Programs",
  severity = $(SIG_HI)
)
{
  /sbin/genksyms                     -> $(SEC_CRIT) ;
  /sbin/sln                          -> $(SEC_CRIT) ;
}

  ##########################
 #                        ##
########################## #
#                       # #
# Shell Related Programs # #
#                       ##
##########################
#(
#  rulename = "Shell Releated Programs",
#  severity = $(SIG_HI)
#)
#{
#  /sbin/getkey                        -> $(SEC_CRIT) ;
#  /sbin/sash                          -> $(SEC_CRIT) ;
#}


  ###############
 #              ##
############### #
#              # #
# OS Utilities # #
#              ##
###############
```

```
(
  rulename = "Operating System Utilities",
  severity = $(SIG_HI)
)
{
  /bin/cat                                -> $(SEC_CRIT) ;
  /bin/date                               -> $(SEC_CRIT) ;
  /bin/dd                                 -> $(SEC_CRIT) ;
  /bin/df                                 -> $(SEC_CRIT) ;
  /bin/echo                               -> $(SEC_CRIT) ;
  /bin/egrep                              -> $(SEC_CRIT) ;
  /bin/false                              -> $(SEC_CRIT) ;
  /bin/fgrep                              -> $(SEC_CRIT) ;
  /bin/fuser                              -> $(SEC_CRIT) ;
  /usr/bin/gawk                           -> $(SEC_CRIT) ;
  /bin/grep                               -> $(SEC_CRIT) ;
  /bin/true                               -> $(SEC_CRIT) ;
  /bin/arch                               -> $(SEC_CRIT) ;
  /bin/basename                           -> $(SEC_CRIT) ;
  /bin/dmesg                              -> $(SEC_CRIT) ;
  /bin/gunzip                             -> $(SEC_CRIT) ;
  /bin/gzip                               -> $(SEC_CRIT) ;
  /bin/hostname                           -> $(SEC_CRIT) ;
  /bin/kill                               -> $(SEC_CRIT) ;
  /bin/killall                            -> $(SEC_CRIT) ;
  /bin/ln                                 -> $(SEC_CRIT) ;
  /bin/loadkeys                           -> $(SEC_CRIT) ;
  /bin/login                              -> $(SEC_CRIT) ;
  /bin/ls                                 -> $(SEC_CRIT) ;
  /usr/bin/mail                           -> $(SEC_CRIT) ;
  /bin/more                               -> $(SEC_CRIT) ;
  /bin/mv                                 -> $(SEC_CRIT) ;
  /bin/netstat                            -> $(SEC_CRIT) ;
  /usr/bin/nice                           -> $(SEC_CRIT) ;
  /bin/ps                                 -> $(SEC_CRIT) ;
  /bin/pstree                             -> $(SEC_CRIT) ;
  /bin/sed                                -> $(SEC_CRIT) ;
  /bin/sleep                              -> $(SEC_CRIT) ;
  /usr//bin/sort                          -> $(SEC_CRIT) ;
  /bin/su                                 -> $(SEC_CRIT) ;
  /bin/sync                               -> $(SEC_CRIT) ;
  /bin/tar                                -> $(SEC_CRIT) ;
  /usr/bin/vi                             -> $(SEC_CRIT) ;
  /bin/zcat                               -> $(SEC_CRIT) ;
}

  ################################
 #                              ##
################################ #
#                              # #
# Critical Utility Sym-Links # #
#                              ##
################################
(
  rulename = "Critical Utility Sym-Links",
```

```
  severity = $(SIG_HI)
)
{
  /bin/pidof                          -> $(SEC_CRIT) ;
  /sbin/poweroff                      -> $(SEC_CRIT) ;
  /sbin/swapoff                       -> $(SEC_CRIT) ;
  /sbin/reboot                        -> $(SEC_CRIT) ;
  /sbin/telinit                       -> $(SEC_CRIT) ;
  /usr/bin/awk                        -> $(SEC_CRIT) ;
  /bin/dnsdomainname                  -> $(SEC_CRIT) ;
  /bin/domainname                     -> $(SEC_CRIT) ;
  /bin/nisdomainname                  -> $(SEC_CRIT) ;
  /bin/red                            -> $(SEC_CRIT) ;
  /usr/bin/view                       -> $(SEC_CRIT) ;
  /bin/ypdomainname                   -> $(SEC_CRIT) ;
}


  #########################
 #                       ##
######################### #
#                      # #
# Temporary directories # #
#                      ##
#########################
(
  rulename = "Temporary directories",
  recurse = false,
  severity = $(SIG_LOW)
)
{
  /var/tmp                            -> $(SEC_INVARIANT) ;
  /tmp                                -> $(SEC_INVARIANT) ;
}

  ###############
 #             ##
############### #
#             # #
# Local files # #
#             ##
###############
(
  rulename = "User binaries",
  severity = $(SIG_MED)
)
{
  /sbin                               -> $(SEC_BIN) (recurse = 1) ;
  /usr/local/bin                      -> $(SEC_BIN) (recurse = 1) ;
  /usr/sbin                           -> $(SEC_BIN) (recurse = 1) ;
  /usr/bin                            -> $(SEC_BIN) (recurse = 1) ;
}

(
  rulename = "Shell Binaries",
```

```
  severity = $(SIG_HI)
)
{
  /bin/sh                                -> $(SEC_BIN) ;
  /bin/bash                              -> $(SEC_BIN) ;
}

(
  rulename = "Security Control",
  severity = $(SIG_HI)
)
{
  /etc/group                             -> $(SEC_CRIT) ;
  /etc/security/                         -> $(SEC_CRIT) ;
  /var/spool/fcron/                      -> $(Dynamic) ;
}

#(
#  rulename = "Boot Scripts",
#  severity = $(SIG_HI)
#)
#{
#  /etc/rc                                 -> $(SEC_CONFIG) ;
#  /etc/rc.bsdnet                          -> $(SEC_CONFIG) ;
#  /etc/rc.dt                              -> $(SEC_CONFIG) ;
#  /etc/rc.net                             -> $(SEC_CONFIG) ;
#  /etc/rc.net.serial                      -> $(SEC_CONFIG) ;
#  /etc/rc.nfs                             -> $(SEC_CONFIG) ;
#  /etc/rc.powerfail                       -> $(SEC_CONFIG) ;
#  /etc/rc.tcpip                           -> $(SEC_CONFIG) ;
#  /etc/trcfmt.Z                           -> $(SEC_CONFIG) ;
#}

(
  rulename = "Login Scripts",
  severity = $(SIG_HI)
)
{
  /etc/profile                           -> $(SEC_CONFIG) ;
  /etc/inputrc                           -> $(SEC_CONFIG) ;
}

# Libraries
(
  rulename = "Libraries",
  severity = $(SIG_MED)
)
{
  /usr/lib                               -> $(SEC_BIN) ;
}


  ########################################################
 #                                                      ##
########################################################## #
```

```
#                                                     # #
# Critical System Boot Files                          # #
# These files are critical to a correct system boot. # #
#                                                     ##
#########################################################

(
  rulename = "Critical system boot files",
  severity = $(SIG_HI)
)
{
     /boot                              -> $(SEC_CRIT) ;
     /usr/sbin/grub                     -> $(SEC_CRIT) ;
     /boot/System.map-2.4.25-20040227   -> $(SEC_CRIT) ;
     /boot/linux-2.4.25-20040227        -> $(SEC_CRIT) ;
     /boot/grub/grub.conf               -> $(SEC_CRIT) ;
}
   ###################################################
  ###################################################
  # These files change every time the system boots ##
  ###################################################
(
  rulename = "System boot changes",
  severity = $(SIG_HI)
)
{
      # Logrotate changes inode
     /dev/log                           -> $(SEC_CONFIG) -i ;
     /dev/tty1                          -> $(SEC_CONFIG) -u ;

      # User ID may change on console login/logout.
     /dev/console                       -> $(SEC_CONFIG) -u ;
     /dev/tty2                          -> $(SEC_CONFIG) ; # tty devices
     /dev/urandom                       -> $(SEC_CONFIG) ;
     /dev/initctl                       -> $(SEC_CONFIG) ;
     /var/run                           -> $(SEC_CONFIG) ; # daemon PIDs
     /var/log                           -> $(SEC_CONFIG) ;
     /etc/issue                         -> $(SEC_CONFIG) ;
     /etc/.pwd.lock                     -> $(SEC_CONFIG) ;

     # Inode number changes on any mount/unmount
     /etc/mtab                          -> $(SEC_CONFIG) -i ;
}

# These files change the behavior of the root account
(
  rulename = "Root config files",
  severity = 100
)
{
     # Catch all additions to /root
     /root                              -> $(SEC_CRIT) ;
     /root/.bashrc                      -> $(SEC_CONFIG) ;
     /root/.bash_profile                -> $(SEC_CONFIG) ;
     /root/.bash_logout                 -> $(SEC_CONFIG) ;
```

```
     /root/.bash_history               -> $(SEC_CONFIG) ;

     # Vim changes .viminfo
     !/root/.viminfo  ;
}

  ################################
 #                               ##
################################ #
#                              # #
# Critical configuration files # #
#                              ##
################################
(
  rulename = "Critical configuration files",
  severity = $(SIG_HI)
)
{
     /etc/fcron.conf                   -> $(SEC_BIN) ;
     /etc/fcron.allow                  -> $(SEC_BIN) ;
     /etc/fcron.deny                   -> $(SEC_BIN) ;
     /etc/fstab                        -> $(SEC_BIN) ;
     /etc/group-                       -> $(SEC_BIN) ;
     /etc/protocols                    -> $(SEC_BIN) ;
     /etc/services                     -> $(SEC_BIN) ;
     /etc/rc.d                         -> $(SEC_BIN) ;
     #/etc/motd                        -> $(SEC_BIN) ;
     /etc/named.conf                   -> $(SEC_BIN) ;
     /etc/passwd                       -> $(SEC_CONFIG) ;
     /etc/passwd-                      -> $(SEC_CONFIG) ;
     /etc/sysconfig                    -> $(SEC_BIN) ;
     /etc/nsswitch.conf                -> $(SEC_BIN) ;
     /etc/hosts                        -> $(SEC_CONFIG) ;
     /etc/inittab                      -> $(SEC_CONFIG) ;
     /etc/resolv.conf                  -> $(SEC_CONFIG) ;
     /etc/syslog-ng/syslog-ng.conf     -> $(SEC_CONFIG) ;

}

  ####################
 #                  ##
#################### #
#                  # #
# Critical devices # #
#                  ##
####################
(
  rulename = "Critical devices",
  severity = $(SIG_HI),
  recurse = false
)
{
     /dev/kmem                         -> $(Device) ;
     /dev/mem                          -> $(Device) ;
     /dev/null                         -> $(Device) ;
```

```
    /dev/zero                       -> $(Device) ;
    /proc/devices                   -> $(Device) ;
    /proc/net                       -> $(Device) ;
    /proc/sys                       -> $(Device) ;
    /proc/cpuinfo                   -> $(Device) ;
    /proc/mounts                    -> $(Device) ;
    /proc/dma                       -> $(Device) ;
    /proc/filesystems               -> $(Device) ;
    /proc/pci                       -> $(Device) ;
    /proc/interrupts                -> $(Device) ;
    /proc/ioports                   -> $(Device) ;
    /proc/kcore                     -> $(Device) ;
    /proc/self                      -> $(Device) ;
    /proc/kmsg                      -> $(Device) ;
    /proc/stat                      -> $(Device) ;
    /proc/loadavg                   -> $(Device) ;
    /proc/uptime                    -> $(Device) ;
    /proc/locks                     -> $(Device) ;
    /proc/version                   -> $(Device) ;
    /proc/meminfo                   -> $(Device) ;
    /proc/cmdline                   -> $(Device) ;
    /proc/misc                      -> $(Device) ;
}

# Rest of critical system binaries
(
  rulename = "OS executables and libraries",
  severity = $(SIG_HI)
)
{
    /bin                            -> $(SEC_BIN) ;
    /lib                            -> $(SEC_BIN) ;
}
```