Building Linux from Scratch

Sarthak Ahuja(2012088)

Chaitanya Kumar(2012031)

Danish Goel(2012032)

"Linux From Scratch" is a project that provides step-by step instructions for building a customized Linux system, entirely from source.

Source: http://www.linuxfromscratch.org/lfs/

Methodology:

NOTE: This is not a complete guide, but only a generic step by step process. Since the guide (LFS 7.4 http://www.linuxfromscratch.org/lfs/downloads/7.4/LFS-BOOK-7.4.pdf) already contains the precise procedure for the same, commands have been omitted in the following.

The whole task was divided into sub tasks as such:

1. Preparing your system for the build- partitioning, formatting, creating a file system, mounting the new partition
2. Acquiring required packages and patches
3. Working on the partition-Mounting the partition, creating a tools directory, build directory for compiling all packages, adding the user, setting permissions
4. Constructing a temporary system-installation of packages ("the build-process") and consequently cleaning up after every package, as well as after all packages, to save on space
5. Building the LFS system-preparing the Virtual Kernel file Systems-setting up initial devices and mounting the virtual kernel file system, switching to the chroot command, creating directories and symbolic links, installing packages and cleaning up
6. Setting up and installing the System Bootscripts, implementing run levels, network interface cards
7. Making the LFS system bootable-using GRUB to set up the boot process

Steps followed:

Preparing the system for build:

As per the project recommendations, the host system requirements were checked using the given bash file. (version-check.sh)

A new partition was created with a size of 10GB (as recommended), and a file system was mounted onto it, using the mkfs command, and a swap space partition using mkswap command. Finally the partition was mounted, and appropriate directories were created /mnt/lfs ; /sources;

Necessary packages were downloaded to the /mnt/lfs/sources directory.

As root, /mnt/lfs/tools directory and a symlink were created to allow using tools from the host in the subsequent stages of compilation as well.

The group "lfs" was added to the system using groupadd command, and "lfs" user was added to the group, with bash as the default shell for this user, password, and permission of the /mnt/lfs/tools directory. Besides, the ".bash_profile" was created to set up the login environment.

For the build process, the following were the generic steps followed:

All sources and patches were placed in /mnt/lfs/sources/, and not in /mnt/lfs/tools/

Each package was extracted in the sources directory, and then after changing to the extracted (newly created) directory, the book's instructions were followed.

After following the book's set of commands for building the package, the extracted source directory (newly created) and <package-name>-build directories created, if any, were deleted.

After building of all packages (about 30), more space was freed by using the strip command.

The ownership of the /mnt/lfs/tools/ directory was changed from lfs to root, using the chown command.

Building the LFS System

Installation of basic system software was done, strictly adhering to the book guidelines, without using any optimizations, which would have led to a quicker process, but also brought in the risk of leading to compilation difficulties and problems.

To set up the virtual kernel file system (file systems that communicate through the host kernel, are therefore virtual, and occupy no disk space and reside in memory), directories for the mounting process were created. Further, essential device nodes were created such as console and null.

[A device node is basically a file. Like most other UNIX variants, Linux too views devices as files. This makes things easier as the system can use the same functions to access both hardware devices and files. For instance, /dev/fd0 is a floppy drive in Linux, and /dev/lpt0 is the Printer. (It is possible to have two device nodes pointing to the same device-by means of different names, but same device types- and is done for convenience, such as the swap device.)]

After the above, further compilation was done in the "chroot" mode. This allows one to run a command or interactive shell from another directory, and treats that directory as root. The major advantage here is that it allows you to continue using the host system while LFS is being built.

[NOTE: After the execution of the chroot command, the bash prompt shall say "I have no name!" which is expected as the /etc/passwd/ file hasn't yet been created.

After entering into chroot mode, necessary directories, files and symlinks were created, along with

creation of the /etc/passwd/ file, and initialization of log files that keep track of who was logged into the system and when.

Linux-3.10.10 API headers, Man Pages-3.53, Glibc and locales and several other packages-Ncurses, Shadow, mpfr, Coreutils, Perl, GRUB, IPRoute2, etc., were installed along with the necessary configurations and tests. Along with installation of shadow, the root password was set.

For cleanup after this, chroot mode needs to be logged out first, and then logged in again, so that while “stripping”, no in use binaries get removed.

Then, prior to installation of system bootscripts, device nodes were created and configured, skipping any network configurations at the moment.

Linux systems in general traditionally use a static device creation method, whereby all device nodes are created under /dev directory, irrespective of whether the corresponding hardware device actually exists or not. This is done through a MAKEDEV script, which contains several calls to the mknod program, with relevant major and minor device numbers for every possible device that might exist in the world.

The Udev package installed previously is more efficient. It creates only those devices that are detected by the kernel at boot time, and are stored on a devtmpfs file system-a virtual file system residing only on main memory. Custom symlinks were created to CD-ROMs and USB ports, taken into consideration the choice between “by-id” mode and “by-path” mode.

For the same type of devices such as a webcam and a TV tuner-both of which connect through USB, sometimes their device nodes may get interchanged, because the order in which the devices get discovered is not predictable. To resolve such an issue specific unique attributes of the device (vendor and product IDs) are used in creating symlinks. Consequently, /dev/video0 and /dev/video1 still randomly refer to the webcam and TV tuner, but their sysmlinks always point to the correct device.

Then, LFS Bootscripts (20130821) were installed, which start/stop the lfs system at bootup/shutdown. Linux uses the facility SysVinit that is based on *run-levels.* The default run level is 3, and the following is a description of each run level:

0: halt the computer

1: single-user mode

2: multi-user mode without networking

3: multi-user mode with networking

4: reserved for customization, otherwise same as 3

5: same as 4, used for GUI login

6: reboot the computer

The /etc/initab file was created, which is read by the first program that is run during kernel initialization-init.

The /etc/sysconfig/network file was created and a hostname entered (the name of the computer),

along with configuration of the setclock script (for system time).

The /etc/sysconfig/rc.site file was configured. The script invokes syslogd program with the –m 0 option which turns off the timestamp mark that syslogd write to the log every 20 minutes.

For the boot and shutdown script customizations, the file /etc/sysconfig/rc.site was modified. Since only a single partition and a single Ethernet card was being used, the variable OMIT_UDEV_SETTLE was modified to y, i.e., OMIT_UDEV_SETTLE=y.

On similar lines, as the /var directory was not separately mounted, the variable was set as: OMIT_UDEV_RETRY_SETTLE=y. Further, the file system checks are silent by default. To the user this may appear as a delay. These checks were enabled by setting the variable VERBOSE_FSCK=y. To quicken the boot process further, the time spent to delete the files present in the /tmp directory was skipped by setting the variable SKIPTMPCLEAN=y.

Similarly the shutdown time was reduced by cutting down on the time spent by init in killing all such processes that did not shutdown by their own scripts. The variable KILLDELAY=0 thereby skipping the delay for the sendsignals script.

For the login aspects of the Operating System, the base /etc/profile was used to set up the environment variables necessary for native language support. The character maps, native language and other related configurations were made.

The inputrc file handles keyboard mapping for specific situations. This file is the start-up file used by Readline —the input-related library — used by Bash and most other shells. The file was written for a generic global system.

After all the above configurations and compilations, the following steps were carried out to make the LFS system bootable. These included creating the fstab file, building a kernel for the new lfs system, and installing the GRUB boot loader so that the LFS system can be selected for booting at startup.

The etc/fstab file was created and values of the partition for lfs, swap and file system(ext4) were typed in. Building of the kernel involved configuration, compiling and installation. The kernel used as recommended in the book was Linux 3.10.10. After the make mrproper command ensured a clean linux kernel, the command make LANG=<host_LANG_value> LC_ALL= menuconfig was used to establish the locale settings to be the one used on the host. Consequently, the kernel image was compiled and modules installed using make and make modules_install respectively. Additional configurations such as copying the path to the kernel image , copying the System.map file- that maps the function entry points of every function in the kernel API, as well as the addresses of the kernel data structures for the running kernel, and the installation of the documentation for the Linux kernel.

The configuration file /etc/modprobe.d/usb.conf for directing the order of the Linux modules in the kernel was created. This isn't always needed, as modules mostly are loaded automatically. GRUB was then set up to make LFS bootable.

Following the instructions in Chapter 9, the release-file was created, though it is an optional idea.

Precautions and Sources of Error:

The partitioning process needs to be carried out carefully and the space allocated must be taken care of-the unit taken by cfdisk is in MBs and not GBs.

The command echo $LFS must be run for every terminal session to check whether the environment variable is set or not (an environment variable setting lasts only as long as the current terminal session does-it expires with the closing of the terminal).

During the creation of the lfs user, the book doesn't mention adding lfs to the sudo group. That part however, is necessary and needs to be done.

During the Build Process, care must be taken that the packages exist in a directory that is accessible in the chroot mode-such as the $LFS/sources directory. After having extracted the package and having carried out all the instructions in the book, the deletion of the extracted directory and the any other directories created during the process is of utmost importance.

Changes to the grub.cfg file must be carried out after complete understanding of the GRUB naming conventions, and the file itself. Any haste or error could render the system in an unbootable state.

Learning outcomes:

Having completed the compilation of Linux From Scratch project after several trials, we have developed a fair idea of how a Linux based operating system is compiled, what its components are, and how they fit in together, besides the customizations that one can bring in, when compiling one's own operating system.

The project enabled us to realize how UNIX based and Windows operating systems differ in their internal structures, such as the way they treat their devices-Windows has mnemonic familiar, identifiable names for devices, while Linux doesn't (*/dev/fd0 is a floppy drive in Linux*). However, this gives an advantage to the system, for the system can then use the same functions to access both hardware and files.

We thank our instructor for having given us an opportunity to work on such a project, so that we could grow in this domain.