JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 2 : 316~321 Feb. 2017

LFS 리눅스 시스템 구축 과정 분석

이계상*

An Analysis on Building Steps of LFS Linux System

Kyesang Lee*

Department of Information and Communications Engineering, Dong-eui University, Busan 47340, Korea

요 약

최근 리눅스가 IoT, 임베디드, 모바일, 데스크톱 및 서버 산업 등에서 널리 사용됨에 따라, 리눅스 시스템 구조에 대한 이해와 맞춤형 리눅스 시스템 구축 기술의 중요성은 날로 커지고 있다. 이에 비례하여 리눅스 시스템의 모든 구성 요소를 소스 파일로부터 구축할 수 있는 능력이 매우 중요해 졌다. LFS (Linux From Scratch) 사이트[1]는 소스로부터 기본 리눅스 시스템을 구축하는 과정을 안내한다. LFS 구축 과정은 빌드 호스트 준비 단계, 임시 시스템 구축 단계, 최종 시스템 구축 단계와 시스템 설정 및 부트 준비 단계로 구성된다. 하지만 LFS 시스템 구축 과정에 내재한 개념은 초보자가 쉽게 이해하기 힘들다. 본 논문은 LFS 시스템의 구축 단계를 분석 정리하고, 각 단계에서 사용된 핵심 빌드 개념 및 원리를 제시한다. 끝으로, 본 연구의 구축 사례에서 측정한 설치 소요 시간을 보인다.

ABSTRACT

With the recent wide adoption of Linux in the fields of IoT, embedded, mobile, desktop and server industries, the importances of understanding the Linux system architecture as well as customizing the Linux system are increasing very steeply. Accordingly, the capabilities of building every component of Linux system from source code files have been important. The LFS (Linux From Scratch) site[1] guides the steps of building the basic Linux system from source files. The steps consist of build host preparation step, temporary system building step, final system building step, and system configuration and boot preparation step. However, the underlying concepts behind the steps used in building LFS are difficult to understand, particularly to the beginner. This paper analyzes the LFS build steps and reveal the core build concepts and principles used in each step. Additionally, this paper shows the measured package build times obtained from our build experience.

키워드 : 리눅스, 빌드, 소스 파일 패키지, LFS, 크로스 컴파일

Key word : build, cross-compile, Linux, LFS, source file package

Received 30 November 2016, **Revised** 07 December 2016, **Accepted** 19 December 2016

* **Corresponding Author** Kyesang Lee(E-mail:ksl789@gmail.com, Tel:+82-51-890-1691)
Department of Information and Communications Engineering, Dong-eui University, Busan 47340, Korea

Open Access http://doi.org/10.6109/jkiice.2017.21.2.316 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 리눅스는 매우 다양하게 이용되고 있다. 임베디드 장치, 안드로이드 폰, 라즈베리 보드 등의 기본 운영체제로 널리 사용되며, 서버의 운영체제로도 많이 보급되었다[2]. 아울러, 리눅스는 수많은 배포판들의 보급으로 그 설치가 매우 쉬워졌다. 그러나 배포판들의 거의 대부분은 미리 컴파일된 바이너리를 담고 있어 설치는 용이하지만, 리눅스 시스템 구조를 파악하기가 어렵다. 리눅스 시스템 구조를 이해하는 가장 빠른 길 중의 하나는 모든 구성 요소를 소스 파일로부터 직접 컴파일하여 시스템을 구축해 보는 경험을 통해서라 할 수 있다. 이 기술은 특히 리눅스를 이용하여 임베디드 또는 IoT 장치를 맞춤 제작하는 경우에 꼭 필요하다. 하지만, 무의 상태에서 소스 파일로부터 직접 컴파일하여 리눅스 시스템을 구축하는 일은 매우 어렵다. 다행히, 리눅스를 바닥부터 구축하는 경험을 공유하기 위한 사이트가 몇몇 존재하고[3], 그 중 대표적인 사이트가 Linux From Scratch (LFS) [1]이다. LFS 사이트의 지시를 따라 이행하면 간단하지만 완전한 하나의 리눅스 시스템을 비교적 쉽게 구축해 볼 수 있다. 그러나, 초보자가 그 구축 과정을 잘 이해하고, 특히 이면에 숨어 있는 빌드 개념 및 원리를 파악하기는 아직도 쉬운 일이 아니다.

이에 본 연구에서는 동 사이트의 리눅스 빌드 단계를 분석하여 명확히 정리하고, 각 단계에서 사용된 핵심 빌드 개념 및 원리를 중점적으로 제시하고자 한다. 아울러, 본 연구를 통해 측정한 LFS 구축 시간을 보인다.

Ⅱ. LFS 구축 단계 및 준비

이 장은 LFS 시스템에서 채용한 구축 단계를 개관하고, 시스템 구축을 위한 준비 과정을 기술한다.

2.1. LFS 구축 단계

LFS 시스템 구축은 준비 단계, 임시 시스템 구축 단계, 최종 시스템 구축 단계, 시스템 설정 및 부팅 준비 단계로 나뉜다. 준비 단계는 먼저 빌드용 호스트로서 리눅스 시스템을 준비하는 것으로 시작한다. LFS 시스템의 구축 장소가 될 새 디스크를 준비하고, 이 디스크에 소스 파일을 다운 받는다. 임시 시스템이 구축될 디렉토리와 빌드에 사용할 일반 사용자 계정을 생성한다. 준비 단계는 다음 2.2절에서 자세히 기술한다.

임시 시스템은 최종 시스템을 구축하기 위한 최소한의 중간 시스템으로서, 이를 위해 크로스 툴체인 (cross tool chain)을 먼저 설치한다. 크로스 툴체인[4]은 성격상 호스트의 영향을 받지 않음으로 호스트와 격리된 임시 시스템을 구축하는 첫 관문이다. 다음 크로스 툴체인을 사용하여 임시 C 라이브러리를 구축하고, 이어서 이 라이브러리에 링크된 링커와 컴파일러를 구축한다. 이렇게 빌드된 핵심 툴체인은 호스트와는 독립된 임시 시스템의 기반이 된다. 이 툴체인을 이용하여 임시 시스템에 필요한 최소한의 나머지 유틸리티 패키지들을 설치하고 나면 임시 시스템이 완성된다. 구축된 임시 시스템은 최종 시스템 구축에 사용된다. 임시 시스템 구축은 3장에서 자세히 기술한다. 최종 시스템의 구축은 임시 시스템이 설치된 새 파티션을 루트 디렉토리로 하여 chroot로 진입함으로써 시작된다. chroot는 마치 임시 시스템으로 부팅하여 진입한 효과를 제공해 준다. 먼저 표준 디렉토리들을 생성하고, 최종 C 라이브러리를 구축하며, 최종 링커와 컴파일러를 설치한다. 이렇게 구축된 최종 툴체인을 이용하여 최종 시스템에서 필요한 나머지 패키지들을 모두 빌드하면, 최종 시스템 구축이 완료된다. 자세한 내용은 4장에서 기술한다.

마지막은 LFS 시스템 부팅 시 필요한 시스템 설정 파일 설치와 커널 및 부트로더 설치 단계로, 많은 문헌[5]이 존재하므로 본 논문에서는 언급을 생략한다.

2.2. 구축을 위한 준비 과정

새로운 LFS 시스템 구축을 위하여 가장 먼저 준비할 사항은 동작하는 리눅스 시스템이다. 이 시스템에서 모든 구축이 진행된다. Ubuntu, Fedora, openSuse 등어떤 시스템도 빌드 호스트로 사용될 수 있으나, 일정한 개발 도구 요건을 갖추고 있어야 한다. 이 요건의 만족 여부는 준비된 스크립트 (version-check.sh, library-check.sh)를 실행하면 쉽게 확인할 수 있다. 이어지는 준비 사항은 다음과 같다.

2.2.1. 새 디스크 파티션 및 마운트

여분의 디스크를 확보하여 파티션하고 파일 시스템을 생성한 후 마운트 하는 것이다. 이 과정부터 호스트 시스템의 root 사용자로 작업을 한다. LFS 시스템 구축

에서는 새 파티션의 마운트 위치를 /mnt/lfs로 하기 위해 환경변수 'LFS=/mnt/lfs'를 사용한다.

2.2.2. 소스 파일 다운로드

디렉토리 $LFS/sources를 생성하고, 필요한 프로그램 패키지 소스와 패치 파일을 이 디렉토리에 다운로드 한다. 현재 7.10 버전의 LFS에서 사용하는 패키지는 약 70여종이다. 향후 모든 빌드는 sources 디렉토리 아래에서 수행된다.

2.2.3. 임시 디렉토리 및 사용자 생성

이제 임시 시스템의 설치 장소로 임시 디렉토리 $LFS/tools를 생성하고, 임시 시스템 구축을 위한 일반 사용자 계정을 생성한다.

tools 디렉토리에는 임시 시스템을 구성하는 도구들이 설치된다. 이렇게 함으로써 임시 시스템은 호스트 시스템과 분리되며, 최종 시스템과도 격리될 수 있다. 최종 시스템이 구축되면 tools 디렉토리에 구축된 임시 시스템은 안전하게 제거될 수 있다. 특히, 중요한 점은 이 디렉토리에 대한 심볼릭 링크를 호스트의 루트(/) 디렉토리에 다음과 같이 걸어둔다는 점이다.

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

이 링크를 통해 호스트에서 빌드 시 디렉토리 /tools는 새 파티션의 tools 디렉토리를 가리키게 된다. 즉, 다음 3장의 임시 시스템 구축 시 /tools는 이 디렉토리를 가리키게 된다. 또한, 4장에서 최종 시스템 구축을 할 때에도 디렉토리 /tools는 동일한 디렉토리를 가리키게 된다. 아울러, 임시 시스템 구축 시 치명적인 작업 실수로 인한 피해를 예방하기 위해 일반 사용자 계정 lfs를 생성하고, 이 계정에 적절한 빌드 환경을 만들어 준다. 먼저 tools와 sources 디렉토리의 소유자 권한을 부여한다. 다음은 사용자 lfs가 로그인할 때 마다 다음과 같이 쉘 환경이 초기화 되도록 두 bash 시작 파일을 다음과 같이 설정한다. 먼저 .bash_profile의 내용이다.

```
exec env -i HOME=/root TERM="$TERM" \
PS1='\u:\w\$'  /bin/bash
```

이 파일은 로그인 쉘이 열릴 때 실행되므로, 여기에 모든 기존 환경 변수를 무효화 하는 명령을 기재하고, 세 가지 환경변수만을 정의한 다음, 새 bash 프로그램으로 대체시키는 명령을 기재한다. 대체된 bash 프로그램에 의해 생긴 쉘은 로그인 쉘이 아니므로 .bashrc 파일을 실행한다. 다음은 .bashrc 파일의 내용이다.

```
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=x86_64-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
```

우선 set +h 명령으로 명령 위치 검색시 해쉬 테이블을 사용하지 않도록 한다. LFS 환경변수는 2.2.1절에서 가정한 마운트 위치를 가리킨다. 임시 시스템 구축에 사용되는 크로스 컴파일러 구축을 위해 target 시스템 이름을 환경변수 LFS_TGT로 정의한다. 이 변수의 의미는 3.1절에서 설명한다. PATH 환경 변수는 /tools/bin이 제일 앞에 위치하도록 정의한다. 이는 set +h 명령과 함께, 앞으로 임시 시스템에서 임시 툴들이 tools 디렉토리에 설치되는 대로 이들이 즉시 사용될 수 있도록 한다. 이제 이러한 쉘 환경이 적용되도록 명령 'su - lfs'로 root에서 lfs로 사용자 전환 한다.

Ⅲ. 임시 시스템 구축

이 장은 임시 시스템 구축 과정을 살펴본다. 이 장에서 구축된 임시 시스템은 다음 장에서 최종 LFS 시스템을 구축하는데 사용된다. 임시 시스템은 호스트와 최종 두 시스템과의 완전한 분리를 위해 새로운 파티션의 tools 디렉토리에 설치한다. 이 장의 모든 작업은 2장에서 준비된 사용자 lfs로 수행한다.

3.1. 크로스 툴체인 설치

먼저 크로스 툴체인을 설치한다. 크로스 툴체인의 주요 도구는 크로스 정적 링커와 크로스 gcc 컴파일러로 구성된다. 크로스 툴체인은 그 성격상 호스트 시스템의 영향을 받지 않기 때문에 임시 시스템의 라이브러리를

크로스 컴파일 할 때 사용된다.

정적 링커를 포함하는 binutils 패키지와 컴파일러를 포함하는 gcc 패키지를 컴파일 할 때, 크로스 툴체인을 얻기 위해, configure 옵션[6] 중 target 옵션을 triplet x86_64-lfs-linux-gnu으로 설정한다. 이 target triplet은 기본 triplet의 vendor 부분만을 pc에서 lfs로 살짝 수정한 것으로, 현 빌드 시스템에서 실행 가능한 바이너리를 생성하는 크로스 툴체인을 얻을 수 있게 한다. 또한 결과물인 툴체인이 /tools 디렉토리에 설치되도록 prefix 옵션을 /tools로 설정한다. 이 /tools는 앞에서 $LFS/tools로 링크되었음을 상기하라. 기타, 호스트 시스템의 영향을 피하기 위한 옵션 설정과 당장 필요하지 않은 기능들을 비활성화 하는 configure 옵션들을 추가로 설정한다. 크로스 툴체인 구축에서 binutils 구축을 gcc 구축 보다 우선 실행한다. 이는 크로스 컴파일러 gcc나 임시 시스템 라이브러리 glibc를 컴파일 할 때 binutils 구축 결과물인 정적 링커 (ld)나 어셈블러 (as)에 대한 다양한 테스트가 실행되어 그 결과에 따라 소프트웨어 기능이 취사선택되기 때문이다.

3.2. 임시 C 라이브러리 설치

라이브러리 glibc 패키지를 위에서 구축한 크로스 툴체인을 이용하여 설치한다. 즉, 위에서 얻은 크로스 링커와 크로스 컴파일러를 사용하여 glibc를 크로스 컴파일 한다. 이는 configure 옵션 host를 x86_64-lfs-linux-gnu로 설정함으로써 유도된다. 위에서 언급한 바와 같이, glibc의 크로스 컴파일은 원천적으로 호스트에 대한 의존을 방지한다. prefix 옵션은 역시 /tools로 설정하여 라이브러리가 /tools 디렉토리에 설치되도록 유도한다. 라이브러리 설치를 마친 후, dummy 프로그램을 컴파일 하여, 그 결과 사용되는 동적 링커가 /tools/lib64/ld-linux-x86-64.so.2임을 확인함으로써 새로 설치된 라이브러리의 동적 링커가 사용됨을 반드시 검증한다.

3.3. 툴체인 설치

임시 시스템 툴체인 구축의 마지막 단계로, 3.1절에서 얻은 크로스 툴체인을 사용하여 binutils와 gcc 패키지를 다시 크로스 컴파일 한다. 호스트 시스템의 컴파일 툴을 이용하지 않는다는 점 외에, 3.1절과 다른 점은 이번 크로스 컴파일은 3.2절에서 얻은 라이브러리에 동적으로 링크된 툴체인이 얻어진다는 점이다.

먼저, binutils 패키지를 크로스 컴파일 한다. 이를 위해 MAKEFILE의 환경변수를

```
CC = $x86_64-lfs-linux-gnu-gcc
AR = $x86_64-lfs-linux-gnu-ar
```

과 같이 설정한다. 또한, configure 옵션 with-lib-path를 /tools/lib으로 설정하여 앞에서 새로 구축한 C 라이브러리에 동적 링크되게 한다. 끝으로, 설치된 바이너리 유틸리티들이 /tools 디렉토리에 설치되도록 prefix 디렉토리를 /tools로 설정한 다음 크로스 컴파일 한다. 이렇게 설치된 유틸리티 중 정적 링커 ld는 /tools/lib에 동적 링크되어 구축되었을 것이다. 여기서, 추가로, 추후 최종 시스템에서 사용할 수 있도록 /usr/lib에 동적 링크된 ld-new도 다시 빌드하여 예비해 둔다.

gcc 패키지도 binutils와 유사한 방식으로 크로스 컴파일 되도록 환경변수를 동일하게 설정한다. 또한, gcc가 새 라이브러리의 동적 링커를 사용하도록 하기 위해 세개의 헤더 파일 linux64.h, linux.h, sysv4.h의 소스를 다음과 같이 수정한다. 모든 /lib/ld, /lib64/ld, /lib/ld 앞에 /tools를 추가하고, 모든 /usr/include를 /tools/include로 수정한다. 컴파일 완료 후에는 라이브러리 설치 후 시행한 것과 동일한 검증을 시행하여 올바른 동적 라이브러리가 사용되는지를 확인한다.

이상으로, 임시 시스템의 핵심 툴체인 구축이 완료되었다. 이 툴체인은 호스트 시스템에 의존하지 않는 완전히 독립적인 것으로, 이하 임시 시스템에서 필요한 나머지 유틸리티의 컴파일에 이용된다. bash, make 등을 포함한 나머지 유틸리티들이 /tools/bin에 설치되고 나면 이제 임시 시스템은 완성된 것으로, 실행 커널을 제외하고는 더 이상 어떤 요소도 호스트 시스템에 의존하지 않게 된다.

Ⅳ. 최종 시스템 구축

앞 장의 임시 시스템은 새로운 파티션의 tools 디렉토리에 설치되어 호스트 시스템과 분리되어 있다. 이 장에서는 임시 시스템을 이용하여 최종 시스템을 구축하는 주요 과정을 살펴본다. 최종 시스템은 동일 파티션의 루트 아래 usr, bin 등과 같은 표준 폴더에 구축된다.

이제 임시 시스템이 저장된 파티션을 루트 디렉토리로 하여 부팅하면, 호스트 시스템과는 완전히 격리된 최종 시스템 구축 환경이 될 것이나, 아직 부팅 환경이 준비되어 있지 않으므로, 부팅한 효과를 얻을 수 있는 chroot 환경을 이용한다. 단, 이보다 먼저 가상 커널 파일 시스템을 준비한다. 앞 장은 일반 사용자 계정으로 작업하였으나, 이 장의 구축은 root 계정으로 작업한다.

4.1. 가상 커널 파일 시스템 준비

일부 프로그램은 커널과 통신을 필요로 한다. 커널은 통상 이를 가상 커널 파일 시스템을 생성하여 제공한다. /dev, /proc, /sys 등에 마운트 되는 가상 커널 파일 시스템은 시스템 부팅 시 자동 생성되나, chroot 환경에서는 이를 수동으로 설정해 주어야 한다. 먼저, 새 파티션의 최상위에 이들 마운트 디렉토리를 생성하고, 가상 커널 파일 시스템을 마운트 한다.

4.2. chroot 환경 진입

chroot 환경으로 진입하기 위해 아래와 같은 명령을 사용한다. chroot 명령은 루트 디렉토리를 새로운 디렉토리 ($LFS)로 변경한다. $LFS는 새로운 파티션이 마운트 된 위치를 가리킨다.

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

위 명령은 루트 디렉토리를 임시 시스템이 설치된 새 파티션으로 변경하고, env - i 명령을 실행하여 모든 기존 환경변수를 무효화한다. 다음, 네 가지 환경변수만을 정의한다. 특히, PATH 변수는 통상적인 명령어 검색 표준 디렉토리 외에 말미에 /tools/bin를 포함하고 있다. 이는 최종 시스템의 구축이 진행되면서, 앞 부분에 위치한 표준 디렉토리에 최종 명령어들이 설치되면 즉시 /tools 아래 위치한 임시 시스템 명령들은 사용되지 않게 함이다. 끝으로 ‘bash −login’ 명령의 실행으로 로긴 쉘로 진입된다. +h는 명령어 검색시 해쉬 테이블 기억을 사용하지 않고, 무조건 PATH 변수가 쓰이도록 한다. 이는 앞서 언급한 최종 툴들이 설치 되는대로 /tools 디렉토리의 임시 툴들은 사용되지 않게 하려 함이다.

4.3. 표준 디렉토리 생성 및 필수 파일 생성

chroot 환경 진입 후 처음 할 일은 최종 시스템에서 사용할 표준 디렉토리를 생성하는 것이다. 즉, bin, usr, boot, root, lib, etc와 같은 디렉토리들을 생성한다. 필요한 파일 중 /etc/passwd, /etc/group 파일에는 root 계정 로그인을 위해 필요한 내용을 삽입한다.

4.4. 최종 C 라이브러리 및 최종 툴체인 설치

최종 시스템의 첫 패키지 설치는 라이브러리 설치이다. 먼저, 리눅스 API 헤더를 /usr/include 디렉토리에 설치한다. 다음 최종 C 라이브러리 glibc를 표준 라이브러리 디렉토리 /lib, /usr/lib 등에 설치한다. 동적 링커가 라이브러리 검색 경로에 사용하는 /etc/ld.so.conf 파일도 생성한다. 라이브러리를 설치한 직후, 사용하고 있는 툴체인을 조정한다. 즉, 현재 사용 중인 /tools 아래 정적 링커와 컴파일러가 새로 빌드된 라이브러리를 사용하도록 조정한다. 정적 링커는 3.3절에서 예비해 둔 ld-new로 대체한다. 컴파일러 gcc가 새로운 동적 링커를 가리키고 올바른 start 파일과 헤더 파일을 찾도록 gcc specs 파일을 수정한다. 이제, 조정된 툴체인의 컴파일과 링킹 기능이 올바르게 동작되는지 테스트 해 보는 것이 중요하다. 동적 링커는 /lib에 있는 것이어야 하며, start 파일은 /usr/lib에 있는 것이어야 하고, 헤더 파일 검색 경로는 /usr/include 등이어야 한다.

이제, binutils와 gcc 패키지를 사용하여 최종 툴체인을 설치한다. 최종 툴체인은 /usr/bin에 설치된다. binutils 패키지를 먼저 컴파일 하여 as, ar, ld, objdump 등과 같은 도구를 설치하고, gcc 패키지를 컴파일 하여 gcc, c++, cpp 등과 같은 컴파일 도구를 구축한다. 이제, 최종 툴체인은 임시 시스템의 툴체인을 대신하여 사용될 것이며, 최종 시스템의 나머지 모든 유틸리티 패키지 구축에 사용된다.

4.5. 최종 시스템 패키지 설치

최종 시스템에는 50여개 정도의 유틸리티 패키지가 설치된다. 이에는 shadow 패키지가 있으며 이 패키지가 설치될 때 root 패스워드가 설정된다. bash 패키지가 설치되면 ‘exec /bin/bash −login +h’를 실행하여 새로운 bash 프로그램을 실행한다. 최종 패키지가 설치되고 나면 부팅 환경을 갖추기 전 까지, 새로 chroot 환경으로 진입할 때 마다 4.1절에서 설명한 가상 커널 파일 시스템을

생성하고, 아래와 같은 수정된 chroot 명령을 실행한다. 앞의 4.2절에서 사용된 /tools 디렉토리가 더 이상 사용되지 않고 최종 시스템의 표준 디렉토리가 사용된다.

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

본 연구에서는 도시바 노트북(2.80 GHz CPU, 윈도우10) 상의 VMware 워크스테이션 10에 설치한 64-bit Ubuntu Desktop 14.04[7]를 호스트로 하여 LFS 시스템을 구축하였다. 그림 1은 최종 시스템 구축 시 각 패키지 설치에 소요된 시간을 보인다. 설치 시간은 임시 시스템의 첫 설치 패키지인 binutils의 설치시간을 1 SBU (Standard Build Unit)로 하여 표시한 상대 시간이다.

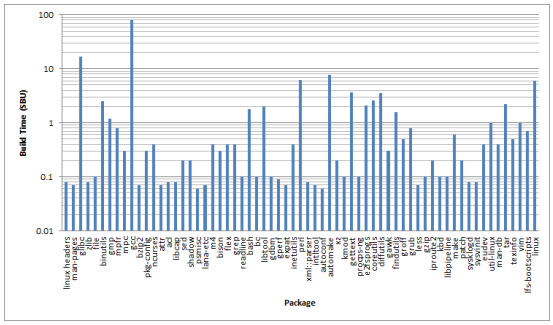

Fig. 1 Package build time of the final system

V. 결 론

본 논문은 LFS 리눅스 시스템 구축 과정을 분석 정리하였다. LFS의 구축 과정은 크로스 컴파일로 중간 임시 시스템을 구축하고, 임시 시스템을 이용하여 최종 시스템을 구축함으로써 호스트 시스템의 헤더 파일이나 라이브러리의 영향을 받지 않는 독립된 리눅스 시스템을 초보자도 큰 실수 없이 구축할 수 있게 해 주어, 그 과정이 교육적이다. 본 논문에서 분석 정리한 빌드 과정을 잘 이해한 후 LFS 사이트의 지시대로 기본 LFS 시스템을 구축해 보면 리눅스 시스템의 구조 이해에 도움이 될 것이며, 향후 더 고급의 맞춤형 임베디드 리눅스 시스템 구축 기술 습득으로 나아가는 데도 도움이 될 것이다.

ACKNOWLEDGMENTS

This work was supported by Dong-eui University Grant.(201601350001)

REFERENCES

[1] LFS Project. Linux From Scratch [Internet]. Available: http://www.linuxfromscratch.org/lfs/.

[2] B. M. Chang, *Unix/Linux: From Usage to Programming*, 1st ed. South Korea, Life and Power Press, 2012.

[3] Distrowatch.com (2017, January). Search Distributions [Internet]. Available: https://distrowatch.com/search.php?category=Source-based.

[4] J. Preshing (2014, November). How to Build a GCC Cross-compiler [Internet]. Available: http://preshing.com/20141119/how-to-build-a-gcc-cross-compiler/.

[5] M. Kalle, et. al.. *Running Linux*. 5h ed. New York. O'Reilly, 2006.

[6] A. Griffith. *The Compete Reference GCC*. 1st ed. New York. MGH Osborne, 2002.

[7] Ubuntu Home Page (2016, April). Ubuntu Desktop [Internet], Available: https://www.ubuntu.com/desktop

이계상(Kyesang Lee)

KAIST 전기및전자공학과 공학박사
현 동의대학교 정보통신공학과 교수
※관심분야 : 차세대 인터넷 프로토콜, 안드로이드 플랫폼 등.