14장. 리눅스 커널 소스 (The Linux Kernel Sources)

이 장은 특정 커널 함수를 찾기 위해서 리눅스 커널 소스 어디서부터 시작해야 하는지 이 야기한다.

이 책은 C 언어에 대한 지식을 요구하지는 않지만 리눅스 커널의 동작을 보다 잘 이해하려 면 리눅스 커널의 소스를 가지고 있는 것이 좋다. 다시 말하면, 커널의 소스 프로그램은 리 눅스 운영체제를 심도깊게 이해하는데 있어 효과적인 교재이다. 이 장은 커널 소스 전반에 대해 개괄한다. 즉 커널 소스가 어떻게 배열되어 있는지, 특정 코드를 찾으려면 어디서 시작 해야 하는지 설명한다.

어디서 리눅스 커널 소스를 얻을 수 있는가

주요 리눅스 배포판들(Craftworks, Debian, Slackware, Red Hat 등)은 모두 리눅스 커널 소스를 포함하고 있다. 일반적으로 사용자의 리눅스 시스템에 설치된 리눅스 커널은 이 소 스 코드를 컴파일하여 생성한 것이다. 리눅스의 성격상, 소스들이 계속 변경되므로 사용자의 시스템에 설치된 것은 조금 옛날 것이 되고 만다. 최신 버전의 소스 프로그램은 부록 B에 서 언급된 웹 싸이트에서 구할 수 있다. 이들은 ftp://ftp.cs.helsinki.fi과 이를 그 림자처럼 복사하는 다른 웹 싸이트에서 들어 있다. 헬싱키의 웹 싸이트가 가장 최신 버전의 소스를 가지고 있으며, MIT나 Sunsite와 같은 싸이트들로 비교적 최신 버전의 소스를 제공 한다.

웹 싸이트에 접근할 수 없다고 하더라도, 많은 벤더들이 주요 웹 싸이트에 있는 내용들을 CD ROM 형태로 매우 저렴한 가격으로 제공하고 있으므로, 이를 이용하면 될 것이다. 1년에 네번 혹은 매달 정기적으로 업그레이드판을 제공해주는 구독 서비스도 있다. 지역별 리눅스 유저 그룹도 소스를 구하는데 유용한 곳이다[1].

리눅스 커널의 버전 형태는 매우 단순하다. 짝수 버전 커널(예를 들자면 2.0.30)은 안정적 이고 발표된 버전이고, 홀수 버전 커널(예를 들자면 2.1.42)은 모두 개발용 커널이다. 본책 은 안정적인 2.0.30 소스 트리를 기반으로 하고 있다. 개발용 커널은 최신 기능들을 모두 포 함하고 있으며 또한 최신 드라이버들도 모두 지원한다. 개발 커널은 불안정할 수도 있고, 이 는 사용자가 바라지 않는 것이겠지만, 최신 커널을 사용해보는 것은 리눅스 공동체에 있어 중요한 일이다. 그래야 전체 공동체를 위해 테스트를 할 수 있다. 실제 제품으로 나온 커널 이 아닌 것을 써보려고 할 때 시스템 전체를 백업해두는 것이 좋다는 것을 기억하기 바란다.

커널 소스에서 바뀐 것들은 패치(patch) 파일로 배포된다. patch 프로그램은 소스 파일들에 편집된 것들을 적용하는데 사용된다. 따라서, 예를 들어 2.0.29 커널 소스를 가지고 있고, 이 를 2.0.30 소스로 바

꾸고 싶다면, 2.0.30 패치 파일을 구해서 패치를 소스 트리에 적용하면 된 다.

```
$ cd /usr/src/inux
$ patch -p1 < patch-2.0.30
```

이는 전체 소스 트리를 복사할 필요가 없어, 느린 직렬 연결을 통하는 경우 더욱 유용하다. 커널 패치를 구하기 좋은 곳은(공식적이던 비공식적이던) http://www.linuxhq.com 웹 사이트이다[2].

커널 소스는 어떻게 배열되어 있는가

소스 트리의 시작인 /usr/src/linux에서 보면 여러개의 디렉토리가 있다.

arch arch 서브디렉토리는 모든 아키텍쳐에 종속적인 커널 코드를 포함하고 있다. 여기에는 서브디렉토리가 더 있는데, 각각 지원하는 아키텍쳐별로 있다. 예를 들어 i386, alpha같은 이름의 서브디렉토리가 존재한다.

include include 서브디렉토리는 커널 코드를 빌드하는데 필요한 모든 인클루드(include) 파 일들의 대부분을 가지고 있다. 여기에는 지원하는 아키텍쳐별로 하나씩 서브디렉토리가 있 다. /include/asm 서브디렉토리는 현재 아키텍쳐에 필요한 실제 디렉토리로 (예를 들어, include/asm-i386) 소프트 링크되어 있다. 아키텍쳐를 다른 것으로 바꾸려면 커널 makefile을 수정하고 리눅스 커널 환경설정 프로그램으로 돌아와야 한다.

init 이 디렉토리는 커널의 초기화 코드를 가지고 있으며, 커널이 어떻게 동작하는지 보기 시작하기에 좋은 곳이다.

mm 이 디렉토리는 모든 메모리 관리 코드를 가지고 있다. 아키텍쳐 종속적인 메모리 관리 코드는 arch/*/mm/ 아래에 있다. 예를 들어, arch/i386/mm/fault.c 같은 곳에 있다.

drivers 모든 시스템의 디바이스 드라이버는 이 디렉토리에 있다. 이들은 디바이스 드라이버 의 유형별로 좀더 세분화 되면. 예를 들어 블럭 디바이스 드라이버는 block에 있다.

ipc 이 디렉토리는 커널의 프로세스간 통신 코드를 가지고 있다.

modules 이는 단순히 빌드된 모듈을 저장하기 위한 디렉토리이다.

fs 모든 파일 시스템 코드를 가지고 있다. 파일 시스템별로 하나씩 디렉토리가 세분화된다. 예를 들어 vfat, ext2 같은 서브디렉토리가 있다.

kernel 메인 커널 코드가 들어 있다. 아키텍쳐 종속적인 커널 코드는 arch/*/kernel에 있다.

net 커널의 네트워킹 코드가 들어 있다.

lib 이 디렉토리는 커널의 라이브러리 코드를 가지고 있다. 아키텍쳐 종속적인 라이브러리 코드는 arch/*/lib/에 있다.

scripts 이 디렉토리는 커널을 설정하는데 사용되는 스크립트(예를 들어 awk나 tlk 스크립 트)를 가지고 있다.

어디서부터 보기 시작할 것이가

리눅스 커널처럼 방대하고 복합적인 프로그램은 들여다보기에 위압적일 수 있다. 이는 실로 된 커다란 공처럼 끝이 보이지 않는 것이기도 하다. 커널의 한 부분을 보다 보면 관련된 다 른 여러 파일들을 보게되고, 오래지 않아 무엇을 찾으려고 했는지 잊어버리게 된다. 다음 작 은 장들은 어떤 주제를 보려 할때 소스 트리의 어디를 보는게 좋은지 힌트를 제공할 것이다.

시스템 시작과 초기화

인텔 기반 시스템에서, 커널은 loadlin.exe나 LILO가 리눅스 커널을 메모리로 읽어들인 후 커널에 제어권을 넘겨줌으로써 시작한다. 이 부분에 대해서는 arch/i386/kernel/- head.S를 보기 바란다. head.S는 아키텍쳐 종속적인 셋업을 한 후 init/main.c에 있 는 main() 루틴으로 점프한다.

메모리 관리

이 코드는 대부분 mm에 있지만, 아키텍쳐 종속적인 코드는 arch/*/mm에 있다. 페이지 폴 트 처리 코드는 mm/memory.c에 있고, 메모리 매핑과 페이지 캐시 코드는 mm/filemap.c 에 있다. 버퍼 캐시는 mm/buffer.c에, 스왑 캐시는 mm/swap_state.c와 mm/- swapfile.c에 구현되어 있다.

커널

상대적으로 일반적인 코드는 kernel에 있고, 아키텍쳐 종속적인 코드는 arch/*/kernel 에 있다. 스케쥴러는 kernel/sched.c에 있고, fork 코드는 kernel/fork.c에 있다. 하반 부 핸들러 코드는 include/linux/interrupt.h에 있다. task_struct 자료구조는 include/linux/sched.h에서 찾을 수 있을 것이다.

PCI

PCI 유사 드라이버는 drivers/pci/pci.c에 있고, 시스템 범위의 정의들은 include/- linux/pci.h에 되어 있다. 각 아키텍쳐들은 특정 PCI BIOS 코드를 가지고 있는데, 알파의 PCI BIOS 코드는 arch/alpha/kernel/bios32.c에 있다.

프로세스간 통신

이것은 모두 ipc에 들어 있다. 모든 시스템 V IPC 오브젝트들은 ipc_perm 자료구조에 들 어 있고, include/linux/ipc.h에서 찾을 수 있다. 시스템 V 메시지들은 ipc/msg.c에, 공유 메모리는 ipc/shm.c에, 세마포어는 ipc/sem.c에 구현되어 있다. 파이프는 ipc/pipe.c에 구현되어 있다.

인터럽트 처리

커널의 인터럽트 처리 코드는 대부분 모두 마이크로프로세서 (때때로 플랫폼) 종속적이다. 인텔의 인터럽트 처리 코드는 arch/i386/kernel/irq.c에 있고, 정의는 include/asm- i386/irq.h에 되어 있다.

디바이스 드라이버

리눅스 커널 소스 코드의 대부분은 디바이스 드라이버에 있다. 모든 리눅스 디바이스 드라 이버 소스는 drivers에 있지만, 이들은 장치 유형에 따라 세분화 된다. **/block** 블럭 디바이스 드라이버. 예를 들어 IDE 디바이스 드라이버는 ide.c에 있다. 모든 장치가 어떻게 파일 시스템을 가질 수 있으며, 어떻게 초기화되는지 보고 싶다면 drivers/block/genhd.c에 있는 device_setup()을 보기 바란다. 이는 하드 디스크만 초기화하는 것이 아니라, 네트웍을 nfs 파일 시스템에 마운트하려고 한다면 네트웍도 초기 화한다. 블럭 장치에는 IDE와 SCSI 기반 장치가 포함된다.

/char ttys, 시리얼 포트나 마우스같은 문자 기반 장치들을 볼 수 있다.

/cdrom 리눅스의 모든 CDROM 코드가 들어 있다. 특별한 CDROM 장치(Soundblaster CDROM 같은)도 여기서 찾을 수 있다. IDE CDROM 드라이버는 drivers/block에 있는 ide-cd.c에 있고, SCSI CDROM 드라이버는 drivers/scsi에 있는 scsi.c에 있다는 점 에 주의하기 바란다.

/pci 여기에는 PCI 유사 드라이버의 소스가 있다. PCI 서브시스템이 어떻게 매핑되고 초기화 되는지 보기 좋은 곳이다. 알파 AXP PCI 확정 코드는 arch/alpha/kernel/bios32.c에 있고, 이는 볼만한 가치가 있다.

/scsi 모든 SCSI 코드와 함께 리눅스가 지원하는 모든 SCSI 장치들의 드라이버가 있는 곳이 다.

/net 네트웍 장치 디바이스 드라이버를 볼 수 있는 곳이다. DECChip 21040 PCI 이더넷 드라 이버는 tulip.c에 있다.

/sound 모든 사운드 카드 드라이버가 있는 곳이다.

파일 시스템

EXT2 파일 시스템 소스는 fs/ext2/ 디렉토리에 있고 자료구조는 include/linux/- ext2_fs.h, ext2_fs_i.h, ext2_fs_sb.h에 정의되어 있다. 가상 파일 시스템 자료구조 는 include/linux/fs.h에 정의되어 있고, 코드는 fs/*에 있다. 버퍼 캐시와 update 커널 데몬은 fs/buffer.c에 구현되어 있다.

네트웍

네트워킹 코드는 net에 있고, 인클루드(include) 파일들의 대부분은 include/net에 있다. BSD 소켓 코드는 net/socket.c에 있고, IP 버전 4 INET 소켓 코드는 inet/ipv4/- af_inet.c에 있다. 일반적인 프로토콜 지원 코드는 (sk_buff 처리 루틴도 포함하여) net/core/에, TCP/IP 네트워킹 코드는 net/ipv4/에 있다. 네트워크 디바이스 드라이버는 drivers/net에 있다.

모듈

커널 모듈 코드는 일부분은 커널에, 일부분은 modules 패키지에 있다. 커널 코드는 모두 kernel/modules.c에 있고, 자료구조와 커널 데몬 kerneld 메시지는 include/- linux/module.h와 include/linux/kerneld.h에 있다. ELF 오브젝트 파일의 구조는 include/linux/elf.h에서 볼 수 있다.

번역 : 이호, 이대현, 김진석, 심마로
정리 : 이호

역주 1) 국내에서는 컴퓨터 잡지 부록의 형태도 큰 비중을 차지하고 있다. (심마로)

역주 2) 인터넷 주소의 소유권 문제로 http://www.kernelnotes.com이 더 인정받고 있 다. http://www.linuxhq.com은 갱신 빈도가 더 늦다. (심마로)