NOTE:
This is a version of Documentation/stable_api_nonsense.txt translated
into korean
This document is maintained by barrios <minchan.kim@gmail.com>
If you find any difference between this document and the original file or
a problem with the translation, please contact the maintainer of this file.

Please also note that the purpose of this file is to be easier to
read for non English (read: korean) speakers and is not intended as
a fork. So if you have any comments or updates for this file please
try to update the original English file first.

==========
이 문서는
Documentation/stable_api_nonsense.txt
의 한글 번역입니다.

역자: 김민찬 <minchan.kim@gmail.com>
감수: 이제이미 <jamee.lee@samsung.com>
==========

리눅스 커널 드라이버 인터페이스
(여러분들의 모든 질문에 대한 답 그리고 다른 몇가지)

Greg Kroah-Hartman <greg@kroah.com>

이 문서는 리눅스가 왜 바이너리 커널 인터페이스를 갖지 않는지, 왜 변하지
않는(stable) 커널 인터페이스를 갖지 않는지를 설명하기 위해 쓰여졌다.
이 문서는 커널과 유저공간 사이의 인터페이스가 아니라 커널 내부의
인터페이스들을 설명하고 있다는 것을 유념하라. 커널과 유저공간 사이의
인터페이스는 응용프로그램이 사용하는 syscall 인터페이스이다. 그 인터페이스는
오랫동안 거의 변하지 않았고 앞으로도 변하지 않을 것이다. 나는 pre 0.9에서
만들어졌지만 최신의 2.6 커널 배포에서도 잘 동작하는 프로그램을 가지고
있다. 이 인터페이스는 사용자와 응용프로그램 개발자들이 변하지 않을 것이라고
여길수 있는 것이다.

초록

여러분은 변하지 않는 커널 인터페이스를 원한다고 생각하지만 실제로는
그렇지 않으며 심지어는 그것을 알아채지 못한다. 여러분이 원하는 것은
안정되게 실행되는 드라이버이며 드라이버가 메인 커널 트리에 있을 때
그런 안정적인 드라이버를 얻을 수 있게 된다. 또한 여러분의 드라이버가
메인 커널 트리에 있다면 다른 많은 좋은 이점들을 얻게 된다. 그러한 것들이
리눅스를 강건하고, 안정적이며, 성숙한 운영체제로 만들어 놓음으로써
여러분들로 하여금 바로 리눅스를 사용하게 만드는 이유이다.

소개

커널 내부의 인터페이스가 바뀌는 것을 걱정하며 커널 드라이버를 작성하고
싶어하는 사람은 정말 이상한 사람이다. 세상의 대다수의 사람들은 이 인터페이스를
보지못할 것이며 전혀 걱정하지도 않는다.

먼저, 나는 closed 소스, hidden 소스, binary blobs, 소스 wrappers, 또는 GPL로

배포되었지만 소스 코드를 갖고 있지 않은 커널 드라이버들을 설명하는 어떤 다른 용어들에 관한 어떤 법적인 문제에 관해서는 언급하지 않을 것이다. 어떤 법적인 질문들을 가지고 있다면 변호사와 연락하라. 나는 프로그래머이므로 여기서 기술적인 문제들만을 설명하려고 한다. (법적인 문제를 경시하는 것은 아니다. 그런 문제들은 엄연히 현실에 있고 여러분들은 항상 그 문제들을 인식하고 있을 필요는 있다.)

자, 두가지의 주요 주제가 있다. 바이너리 커널 인터페이스들과 변하지 않는 커널 소스 인터페이들. 그것들은 서로 의존성을 가지고 있지만 바이너리 문제를 먼저 풀고 넘어갈 것이다.

바이너리 커널 인터페이스

우리가 변하지 않는 커널 소스 인터페이스를 가지고 있다고 가정하자. 그러면 바이너리 인터페이스 또한 자연적으로 변하지 않을까? 틀렸다. 리눅스 커널에 관한 다음 사실들을 생각해보라.

- 여러분들이 사용하는 C 컴파일러의 버젼에 따라 다른 커널 자료 구조들은 다른 alignmnet들을 갖게 될것이고 다른 방법으로(함수들을 inline으로 했느냐, 아니냐) 다른 함수들을 포함하는 것도 가능한다. 중요한 것은 개별적인 함수 구성이 아니라 자료 구조 패딩이 달라진다는 점이다.
- 여러분이 선택한 커널 빌드 옵션에 따라서 커널은 다양한 것들을 가정할 수 있다.
  - 다른 구조체들은 다른 필드들을 포함할 수 있다.
  - 몇몇 함수들은 전혀 구현되지 않을 수도 있다(즉, 몇몇 lock들은 non-SMP 빌드에서는 사라져 버릴수도 있다).
  - 커널내에 메모리는 build optoin들에 따라 다른 방법으로 align될수 있다.
  - 리눅스는 많은 다양한 프로세서 아키텍쳐에서 실행된다. 한 아키텍쳐의 바이너리 드라이버를 다른 아키텍쳐에서 정상적으로 실행시킬 방법은 없다.

커널을 빌드했던 C 컴파일러와 정확하게 같은 것을 사용하고 정확하게 같은 커널 구성(configuration)을 사용하여 여러분들의 모듈을 빌드하면 간단히 많은 문제들을 해결할 수 있다. 이렇게 하는 것은 여러분들이 하나의 리눅스 배포판의 하나의 배포 버젼을 위한 모듈만을 제공한다면 별일 아닐 것이다. 그러나 각기 다른 리눅스 배포판마다 한번씩 빌드하는 수를 각 리눅스 배포판마다 제공하는 다른 릴리즈의 수와 곱하게 되면 이번에는 각 릴리즈들의 다른 빌드 옵션의 악몽과 마주하게 것이다. 또한 각 리눅스 배포판들은 다른 하드웨어 종류에(다른 프로세서 타입과 다른 옵션들) 맞춰져 있는 많은 다른 커널들을 배포한다. 그러므로 한번의 배포에서조차 여러분들의 모듈은 여러 버젼을 만들 필요가 있다.

나를 믿어라. 여러분들은 이러한 종류의 배포를 지원하려고 시도한다면 시간이 지나면 미칠지경이 될 것이다. 난 이러한 것을 오래전에 아주 어렵게 배웠다...

변하지않는 커널 소스 인터페이스들

리눅스 커널 드라이버를 계속해서 메인 커널 트리에 반영하지 않고 유지보수하려고 하는 사름들과 이 문제를 논의하게 되면 훨씬 더 "논란의 여지가 많은" 주제가 될 것이다.

리눅스 커널 개발은 끊임없이 빠른 속도로 이루어지고 있으며 결코

느슨해진 적이 없다. 커널 개발자들이 현재 인터페이스들에서 버그를
발견하거나 무엇인가 할수 있는 더 좋은 방법을 찾게 되었다고 하자.
그들이 발견한 것을 실행한다면 아마도 더 잘 동작하도록 현재 인터페이스들을
수정하게 될 것이다. 그들이 그런 일을 하게되면 함수 이름들은 변하게 되고,
구조체들은 늘어나거나 줄어들게 되고, 함수 파라미터들은 재작업될 것이다.
이러한 일이 발생되면 커널 내에 이 인터페이스를 사용했던 인스턴스들이 동시에
수정될 것이며 이러한 과정은 모든 것이 계속해서 올바르게 동작할 것이라는
것을 보장한다.

이러한 것의 한 예로써, 커널 내부의 USB 인터페이스들은 이 서브시스템이
생긴 이후로 적어도 3번의 다른 재작업을 겪었다. 이 재작업들은 많은 다른
문제들을 풀었다.
  - 데이터 스트림들의 동기적인 모델에서 비동기적인 모델로의 변화. 이것은
    많은 드라이버들의 복잡성을 줄이고 처리량을 향상시켜 현재는 거의 모든
    USB 장치들의 거의 최대 속도로 실행되고 있다.
  - USB 드라이버가 USB 코어로부터 데이터 패킷들을 할당받로록 한 변경으로
    인해서 지금의 모든 드라이버들은 많은 문서화된 데드락을 수정하기 위하여
    USB 코어에게 더 많은 정보를 제공해야만 한다.

이것은 오랫동안 자신의 오래된 USB 인터페이스들을 유지해야 하는 closed 운영체제들과는
완전히 반대되는 것이다. closed된 운영체제들은 새로운 개발자들에게 우연히 낡은
인터페이스를 사용하게 할 기회를 주게되며, 적절하지 못한 방법으로 처리하게 되어
운영체제의 안정성을 해치는 문제를 야기하게 된다.

이 두가지의 예들 모두, 모든 개발자들은 꼭 이루어져야 하는 중요한 변화들이라고
동의를 하였고 비교적 적은 고통으로 변경되어졌다. 리눅스가 변하지 않는 소스
인터페이스를 고집한다면, 새로운 인터페이스가 만들어지게 되며 반면 기존의 오래된
것들, 그리고 깨진 것들은 계속해서 유지되어야 하며 이러한 일들은 USB 개발자들에게
또 다른 일거리를 주게 된다. 모든 리눅스 USB 개발자들에게 자신의 그들의 업무를
마친 후 시간을 투자하여 아무 득도 없는 무료 봉사를 해달라고 하는 것은 가능성이
희박한 일이다.

보안 문제 역시 리눅스에게는 매우 중요하다. 보안 문제가 발견되면 그것은
매우 짧은 시간 안에 수정된다. 보안 문제는 그 문제를 해결하기 위하여
여러번 내부 커널 인터페이스들을 재작업하게 만들었다. 이러한 문제가
발생하였을 때 그 인터페이스들을 사용하는 모든 드라이버들도 동시에
수정되어 보안 문제가 앞으로 갑작스럽게 생기지는 않을 것이라는 것을
보장한다. 내부 인터페이스들의 변경이 허락되지 않으면 이러한 종류의 보안
문제를 수정하고 그것이 다시 발생하지 않을 것이라고 보장하는 것은 가능하지
않을 것이다.

커널 인터페이스들은 계속해서 정리되고 있다. 현재 인터페이스를 사용하는
사람이 한명도 없다면 그것은 삭제된다. 이것은 커널이 가능한한 가장 작게
유지되며 존재하는 모든 가능성이 있는 인터페이스들이 테스트된다는 것을
보장한다(사용되지 않는 인터페이스들은 유효성 검증을 하기가 거의 불가능하다).

무엇을 해야 하나

자, 여러분이 메인 커널 트리에 있지 않은 리눅스 커널 드라이버를 가지고
있다면 여러분은 즉, 개발자는 무엇을 해야 하나? 모든 배포판마다 다른
커널 버젼을 위한 바이너리 드라이버를 배포하는 것은 악몽이며 계속해서
변하고 있는 커널 인터페이스들의 맞처 유지보수하려고 시도하는 것은 힘든
일이다.

간단하다. 여러분의 커널 드라이버를 메인 커널 트리에 반영하라(우리는 여기서
GPL을 따르는 배포 드라이버에 관해 얘기하고 있다는 것을 상기하라. 여러분의
코드가 이러한 분류에 해당되지 않는다면 행운을 빈다. 여러분 스스로 어떻게든
해야만 한다). 여러분의 드라이버가 트리에 있게되면 커널 인터페이스가
변경되더라도 가장 먼저 커널에 변경을 가했던 사람에 의해서 수정될 것이다.
이것은 여러분의 드라이버가 여러분의 별다른 노력없이 항상 빌드가 가능하며
동작하는 것을 보장한다.

메인 커널 트리에 여러분의 드라이버를 반영하면 얻게 되는 장점들은 다음과 같다.
- 관리의 드는 비용(원래 개발자의)은 줄어줄면서 드라이버의 질은 향상될 것이다.
- 다른 개발자들이 여러분의 드라이버에 기능들을 추가 할 것이다.
- 다른 사람들은 여러분의 드라이버에 버그를 발견하고 수정할 것이다.
- 다른 사람들은 여러분의 드라이버의 개선점을 찾을 줄 것이다.
- 외부 인터페이스 변경으로 인해 여러분의 드라이버의 수정이 필요하다면 다른
  사람들이 드라이버를 업데이트할 것이다.
- 여러분의 드라이버는 별다른 노력 없이 모든 리눅스 배포판에 자동적으로
  추가될 것이다.

리눅스는 다른 운영 체제보다 "쉽게 쓸수 있는(out of the box)" 많은 다른 장치들을
지원하고 어떤 다른 운영 체제보다 다양한 아키텍쳐위에서 이러한 장치들을 지원하기
때문에
이러한 증명된 개발 모델은 틀림없이 바로 가고 있는 것이다.

이 문서의 초안을 검토해주고 코멘트 해준 Randy Dunlap, Andrew Morton, David
Brownell,
Hanna Linder, Robert Love, 그리고 Nishanth Aravamudan에게 감사한다.