SENSATION

Data and Knowledge Engineering / Hodgepodge

정적 라이브러리(Static Library) & 공유 라이브러리(Shared Library)

2012.09.14 22:47 신고

라이브러리(Library)란?

라이브러리(Library)는 다른 프로그램들과 링크되기 위하여 존재하는, 하나 이상의 서브루틴(Subroutine)이나 함수(Function)들의 집합 파일 말하는데 함께 링크(link)될 수 있도록 보통 **미리 컴파일된 형태인 오브젝트코드(Object code)** 형태로 존재한다.(Object들은 자주 사용하는 함수의 소스를 컴파일하여 만들 수 있음.) 라이브러리는 코드 재사용을 위해 조직화된 오래된 기법 중의 하나이며, 많은 다른 프로그램들에서 사용할 수 있도록, 운영체계나 소프트웨어 개발 환경 제공자들에 의해 제공되는 경우가 많다.

라이브러리라는 기술이 생긴 이유는 코드의 재사용(**자주 사용하는 함수들의 쉽게 사용 가능**) 및 부품화 실현, 소스를 제공하지 않음으로서 중요 기술의 유출을 방지할 수 있고, 라이브러리는 사용하는 개발자들로서는 대형 어플리케이션 개발 시간을 단축시킬 수 있다는 장점들이 주어지기 때문이다. 라이브러리 내에 있는 루틴들은 두루 쓸 수 있는 범용일 수도 있지만, 3차원 애니메이션 그래픽 등과 같이 특별한 용도의 함수로 설계될 수도 있다.

라이브러리들은 사용자의 프로그램과 링크되어, 실행이 가능한 완전한 프로그램을 이룬다.(**라이브러리는 미리 컴파일되어 있어서 링크만 하면 사용 가능하기 때문에 컴파일 시간이 단축된다.**) 이러한 링크는 대개 정적으로 연결(static link)되지만, 시스템에 따라 동적으로 연결(dynamic link)될 수도 있다.

- 표준 시스템 라이브러리: /lib와 /usr/lib에 위치
- 라이브러리의 이름은 대개 lib로 시작

정적 라이브러리(.a)

정적 라이브러리를 사용하여 컴파일을 하면 링커가 프로그램이 필요로 하는 부분을 라이브러리에서 찾아 실행파일 에다가 바로 복사한다. 실행파일에 다 들어가기 때문에 실행할 때 라이브러리가 필요없다. 하지만 크기가 그만큼 커지고, 같은 코드를 가진 여러 프로그램을 실행할 경우 코드가 중복이 되니 그만큼 메모리를 낭비하게 된다.

정적 라이브러리(.a)는 ar 프로그램에 의해 만들어진다.

```
[root@localhost ~]$ gcc -c func1.c func2.c
[root@localhost ~]$ ls
func1.c func1.o func2.c func2.o program.c
[root@localhost ~]$ ar crv libfoo.a func1.o func2.o
a - func1.o
a - func2.o
[root@localhost ~]$ gcc program.c -o program libfoo.a
[root@localhost ~]$ ls
func1.c  func1.o  func2.c  func2.o  lib.h  program  program.c  libfoo.a
[root@localhost ~]$ ./program
```

위에 명령은 func1.o와 func2.o를 정적 라이브러리 libfoo.a에 추가한다. 만약 libfoo.a 파일이 없다면 새로 만든다.(ar -t libfoo.a 명령어를 통해 아카이브에 있는 파일 리스트를 출력할 수 있다.)
참고로 program.c 는 func1과 func2를 호출하는 프로그램이고, lib.h는 func1과 func2의 정보가 담겨있다.

> 라이브러리 이름이 lib로 시작한다면 -l 옵션을 사용하면 앞에 lib와 확장자를 안 써도 된다. 그리고 이번에 만든게 기본 라이브러리 폴더에 없으므로(-l 은 기본 라이브러리 폴더를 뒤진다.) -L. 을 이용해서 현재 폴더에서 찾도록 한다.
>
> ```
> [root@localhost ~]$ gcc program.c -o program -lfoo -L.
> ```

공유 라이브러리(.so/.sa)

공유 라이브러리(Shared Library)는 많이 쓰는 라이브러리 종류이다. 공유 라이브러리를 사용하여 컴파일을 하면 링커가 실행파일에다가 단지 "실행될 때 우선 이 라이브러리를 로딩시킬 것"이라는 표시를 해 둔다. 그러면 실행할 때 라이브러리에 있는 컴파일된 코드를 가져와 사용한다.

이렇게 하면 정적 라이브러리를 사용하는 것보다 파일 크기가 작아지고, 커널이 메모리에 한 카피의 공유 메모리를 보유하면서 다중의 프로그램들과 공유하기 때문에 메모리가 적게 차지하고, 사용 후 메모리에서 삭제되기 때문에 메모리 사용에 효율적이다. 리눅스는 기본적으로 공유 라이브러리가 있으면 그것과 링크를 시키고, 그렇지 않으면 정적 라이브러리를 가지고 링크 작업을 한다.

배포할 때 공유 라이브러리를 함께 배포해야 한다. 그렇지 않으면 실행 시 라이브러리를 찾을 수 없다는 에러메시지가 나타난다.

공유 라이브러리 작동 마법은 Procedure Linkage Table(PLT)라고 하는 데이터 청크이다. 이것은 프로그램이 호출하는 모든 함수를 나열하고 있는 프로그램의 테이블이다. 프로그램이 시작되면 PLT는 각 함수용 코드를 포함하여 함수를 로딩했던 주소에 대한 런타임 링커를 쿼리한다. 그런 다음

테이블의 모든 엔트리를 채우고 그곳으로 옮겨간다. 각 함수가 호출될 때 PLT의 엔트리는 로딩된 함수로 단순히 직접 점프한다. 하지만 여분의 인다이렉션 레이어를 남겨둔다는 것을 기억해야 한다. 각 함수 호출은 점프를 통해 테이블로 바뀐다.

공유 라이브러리 구성

공유 라이브러리 파일은 확장자가 파일포맷이 ELF이면 .so, a.out이면 .sa이다. 그리고 그 뒤에 버전 숫자가 붙는데, 메이저 넘버(major number)와 마이너 넘버(minor number)이다. 메이저 넘버는 라이브러리 버전들 간 잠재적 비호환성을 나타내고, 마이너넘버는 버그 픽스들만을 나타낸다. 보통 메이저 넘버와 더 높은 마이너 넘버를 가진 라이브러리가 안전하다. 일반적으로 libexample.so 파일은 libexample.so.N에 링크되고 다시 이것은 libexample.so.N.M에 링크되나. N은 가장 높은 메이저 넘버이고, M은 가장 높은 마이너 넘버이다. 링커에 -lexample을 지정하면 최근 버전에 대한 심볼릭 링크인 libexample.so을 찾는다.

다음은 공유 라이브러리를 만드는 기본적인 절차이다.

```
[root@localhost ~]$ gcc -fPIC -c *.c
[root@localhost ~]$ gcc --shared -Wl,-soname,libfoo.so -o libfoo.so func1.
[root@localhost ~]$ nm -D libfoo.so
[root@localhost ~]$ sudo ldconfig -n .
[root@localhost ~]$ gcc -o program program.o -L. -lfoo
[root@localhost ~]$ ldd program
```

```
[root@localhost ~]$ gcc -fPIC -c *.c
[root@localhost ~]$ gcc -shared -Wl,-soname,libfoo.so.1 -o libfoo.so.1.0 *
[root@localhost ~]$ ln -s libfoo.so.1.0 libfoo.so.1
[root@localhost ~]$ ln -s libfoo.so.1 libfoo.so
[root@localhost ~]$ LD_LIBRARY_PATH=`pwd`:$LD_LIBRARY_PATH ; export LD_LIB
[root@localhost ~]$ su
[root@localhost ~]# cp libfoo.so.1.0 /usr/local/lib
[root@localhost ~]# /sbin/ldconfig
[root@localhost ~]# ( cd /usr/local/lib ; ln -s libfoo.so.1 libfoo.so )
```

- -fPIC 옵션은 프로세스의 가상 주소 공간 안의 다른 지역에 로드될 수 있도록 코드를 컴파일 한다. 실행 될 때 그 위치가 정해지기 때문에 가상 공간 안에 고정된 위치를 가지면 안된다.
- -shared 옵션은 공유 라이브러리를 만든다는 옵션이다.
- -Wl 옵션은 gcc에게 콤마로 부분된 다음 옵션을 링커에게 알려주기 위한 것이다.
- -soname,libfoo.so.1 라이브러리에 대한 soname을 고정시키는 부분이다.
- LD_LIBRARY_PATH=`pwd`:$LD_LIBRARY_PATH ; export LD_LIBRARY_PATH: 현재 폴더를 공유 라이브러리 탐색 경로에 넣어준다.
- ldconfig은 /etc/ld.so.conf에 지정된 디렉토리를 찾아다니면서 발견된 각각의 동적 라이브러리들에 대해서 그 라이브러리의 soname에 의해서 불리어지는 심볼릭 링크들을 만들어 낸다. 위에 soname에 주어진 이름으로 새로운 심볼릭 링크를 만든다.공유 라이브러리를 이용해서 컴파일을 할 경우 .so를 먼저 경로에서 찾고 없으면 .a를 찾아 주소값을 결정한다.

실행 파일이 메모리에 적재되면 /lib/ld-linux.so에 의해 동적 링킹이 일어난다. (/lib/ld.so는

a.out 형식, /lib/ld-linux.so는 ELF 실행 포맷에서 사용한다.) 그리고 /lib/ld.so 모듈은 /etc/ld.so.cache 파일을 이용하여 실행 파일에서 필요로 하는 soname과 일치하는 라이브러리를 찾는다.(/etc/ld.so.cache 파일은 라이브러리가 어떠한 파일에 포함되어 있는지에 대한 정보를 가지고 있다. /etc/ld.so.cache 는 ldconfig를 통해서 만들어진다.

ldconfig는 /etc/ld.so.conf에 지정된 디렉토리를 찾아다니면서 발견된 동적 라이브러리들에 대한 soname에 의해서 불리어지는 심볼릭 링크를 만들어낸다. ld.so가 파일의 이름을 얻으려고 할 때 이것은 soname을 발견한 파일의 디렉토리를 선택하는 그러한 일을 한다. 그리고 이렇게 함으로서 우리가 라이브러리를 추가할 때마다 ldconfig를 수행할 필요는 없게 되는 것이다. ldconfig는 리스트에 새로운 디렉토리를 만들어 널때만 실행시키면 된다. soname은 라이브러리 내에 포함되어 있는데, 위에 -soname libmyfile.so.1에서 libmyfile.so.1이 soname이다. soname은 버전을 관리하는 것과 관계가 있다.

동적 링커는 /etc/ld.so.conf를 통해 제어된다. 이곳에 기본적으로 검색할 디렉토리 리스트가 포함되어 있는데, 환경변수 LD_LIBRARY_PATH가 ld.so.conf에 나열된 경로에 앞선다. 따라서 사용자들은 이들 설정을 오버라이드 할 수 있다.

ldd를 통해서 실행 파일에 어떤 공유 라이브러리가 필요한지 알 수 있다.

```
$ ldd ls
linux-gate.so.1 => (0xb7fa2000)
librt.so.1 => /lib/librt.so.1 (0x00700000)
libacl.so.1 => /lib/libacl.so.1 (0x00868000)
libselinux.so.1 => /lib/libselinux.so.1 (0x00d33000)
libc.so.6 => /lib/libc.so.6 (0x0089a000)
libpthread.so.0 => /lib/libpthread.so.0 (0x00c2b000)
/lib/ld-linux.so.2 (0x0087d000)
libattr.so.1 => /lib/libattr.so.1 (0x007bd000)
libdl.so.2 => /lib/libdl.so.2 (0x009f6000)
libsepol.so.1 => /lib/libsepol.so.1 (0x006bc000)
```

참고 사이트

- http://unix.co.kr/data/sopro/?p=26
- http://sunnykwak.egloos.com/949540
- http://kldp.org/HOWTO/html/Program-Library-HOWTO/index.html
- http://www.ibm.com/developerworks/kr/library/l-shlibs.html
- http://www.linuxlab.co.kr/docs/98-11-2.htm

Comments(0)

Recent Posts

- NOX iso 다운로드
- Putty 세션이 종료 되더라도 프로세..
- 스타크래프트 다운로드
- Ubuntu 보조디스크 마운트(mount) 하기
- 여러 파일에 대해 같은 명령구 실행..
- MariaDB 설치 및 데이터베이스 생성/..

Categories

Recent Comments

Links

- 한양대학교 컴퓨터공학부
- 구글 학술검색
- MathWorld
- Wolfram|Alpha: Computational Knowledge..
- Free IT eBooks
- Online LaTeX Equation Editor
- 지식로그
- Hadoop Bigdata 네이버 까페
- 한국정보과학회
- DBGuide.Net
- Deoker.com

- ESL POD
- W3Schools Online Web Tutorials
- Majortests