彻底理解链接器：四，库与可执行文件的生成

Original 码农的荒岛求生 码农的荒岛求生 2018-09-22 21:00

收录于话题
#链接器
6个 >

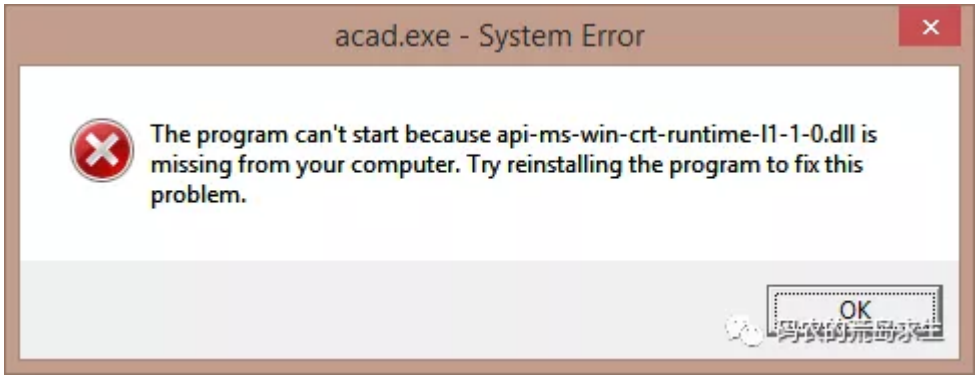
我们继续来看动态链接。

我们知道静态库在编译链接期间就被打包copy到了可执行文件，也就是说静态库其实是在编译期间(Compile time)链接使用的，那么动态库又是在什么时候才链接使用的呢，动态链接可以在两种情况下被链接使用，分别是load-time dynamic linking(加载时动态链接) 以及 run-time dynamic linking(运行时动态链接)，接下来我们分别讲解一下。

- load-time dynamic linking(加载时动态链接)

首先可能有的同学会问，什么是load-time呢，load_time翻译过来也就是加载时，那么什么又是加载呢?
我们大家都玩过游戏，当我们打开游戏的时候经常会跳出来一句话：“加载中，请稍后。。。”和这里的加载意思差不多。这里的加载指的是程序的加载，而所谓程序的加载就是把可执行文件从磁盘搬到内存的过程，因为程序最终都是在内存中被执行的。至于这个过程的详解内容我会在接下来的文章《加载器与可执行文件》一文中给大家详细讲解。在这里我们只需要简单的把加载理解为程序从磁盘复制到内存的过程，加载时动态链接就出现在这个过程。

当把可执行文件复制到内存后，且在程序开始运行之前，操作系统会查找可执行文件依赖的动态库信息(主要是动态库的名字以及存放路径)，找到该动态库后就将该动态库从磁盘搬到内存，并进行符号决议(关于符号决议，参考符号决议一节)，如果这个过程没有问题，那么一切准备工作就绪，程序就可以开始执行了，如果找不到相应的动态库或者符号决议失败，那么会有相应的错误信息报告为用户，程序运行失败。比如Windows下比较常见的启动错误问题，就是因为没有找到依赖的动态库。Linux下同样会有类似信息提示用户程序启动失败。



到这里，同学们应该对加载时动态链接应该有一个比较清晰的了解了。从总体上看，加载时动态链接可以分为两个阶段：阶段一，将动态库信息写入可执行文件；阶段二，加载可执行文件时依据动态库信息进行动态链接。

阶段一，将动态库信息写入可执行文件

在编译链接生成可执行文件时，需要将使用的动态库加入到链接选项当中，比如在Linux下引用libMath.so，就需要将libMath.so加入到链接选项当中（比如libMath.so放到了/usr/lib下，那么使用命令 gcc ... -lMath -L/user/lib ... 进行编译链接），所以使用这种方式生成的可执行文件中保存了依赖的动态库信息，在Linux可使用一个简单的命令ldd来查看。

阶段二：加载可执行文件时依据动态库信息进行动态链接
由于在阶段一生成的可执行文件中保存了动态库信息，当可执行文件加载完成后，就可以依据此信息进行中动态库的查找以及符号决议了。

通过这个过程也可以清楚的看到静态库和动态库的区别，使用动态库的可执行文件当中仅仅保留相应信息，动态库的链接过程被推迟到了程序启动加载时。

为加深你对加载时动态链接这个过程的理解，我们用一个类比来结束本小节，沿用前几节读书的例子，我们正在读的书中引用了《码农的荒岛求生》以及其它著作，那么加载时动态链接就好比，读者开始准备读这本书的时候（还没有真正的读）就

把所有该书当中引用的资料著作都找齐放到一旁准备查看，当我们真正看到引用其它文献的地方时就可以直接在一旁找到该著作啦。在这个类比当中，开始读书前的准备工作就好比加载时动态链接。

接下来我们讲解第二种动态链接，run-time dynamic linking(运行时动态链接) 。

- run-time dynamic linking(运行时动态链接)

上一小节中我们看到如果我们想使用加载时动态链接，那么在编译链接生成可执行文件阶段时需要告诉编译器所依赖的动态库信息，而run-time dynamic linking 运行时动态链接则不需要在编译链接时提供动态库信息，也就是说，在可执行文件被启动运行之前，可执行文件对所依赖的动态库信息一无所知，只有当程序运行到需要调用动态库所提供的代码时才会启动动态链接过程。

我们在上一节中介绍了load-time，也就是程序加载时，那么程序加载完成后就开始程序执行了，那么所谓run-time(运行时)指的就是从程序开始被CPU执行到程序执行完成退出的这段时间。

所以运行时动态链接这种方式对于“动态链接”阐释的更加淋漓尽致，因为可执行文件在启动运行之前都不知道需要依赖哪些动态库，只在运行时根据代码的需要再进行动态链接。同加载时动态链接相比，运行时动态链接将链接这个过程再次推迟往后推迟，推迟到了程序运行时。

由于在编译链接生成可执行文件的过程中没有提供所依赖的动态库信息，因此这项任务就留给了程序员，在代码当中如果需要使用某个动态库所提供的函数，我们可以使用特定的API来运行时加载动态库，在Windows下通过LoadLibrary或者LoadLibraryEx，在Linux下通过使用dlopen、dlsym、dlclose这样一组函数在运行时链接动态库。当这些API被调用后，同样是首先去找这些动态库，将其从磁盘copy到内存，然后查找程序依赖的函数是否在动态库中定义。这些过程完成后动态库中的代码就可以被正常使用了。

相对于加载时动态链接，运行时动态链接更加灵活，同时将动态链接过程推迟到运行时可以加快程序的启动速度。

为了和加载时动态链接作比对，我们继续使用上一小节当中读书的例子，加载时动态链接就好比在开始准备读一本书之前，将该书中所有引用到的资料文献找齐全，而运行时动态链接则不需要这个过程，运行时动态链接就好比直接拿起一本书开始看，看到有引用的参考文献时再去找该资料，找到后查看该文献然后继续读我们的书。从这个例子当中运行时动态链接更像是我们平时读书时的样子。

至此，两种动态链接的形式我们就都已经清楚了，接下来我们看一下动态链接下生成的可执行文件。

动态链接下可执行文件的生成

在静态链接下，链接器通过将各个目标文件的代码段和数据段合并拷贝到可执行文件，因此静态链接下可执行文件当中包含了所依赖的所有代码和数据，而与之对比的动态链接下可执行文件又是什么样的呢？

其实我们在动态库这一节中已经了解了动态链接下可执行文件的生成，即，在动态链接下，链接器并不是将动态库中的代码和数据拷贝到可执行文件中，而是将动态库的必要信息写入了可执行文件，这样当可执行文件在加载时就可以根据此信息进行动态链接了。为方便理解，我们将该信息仅仅认为是动态库都名字，真实情况当然要更复杂一点，这里我们以Linux下可执行文件即ELF文件为例（这一系列的文章重点关注最本质的原理思想，所以这里讨论的同样适合Windows下的可执行文件即exe文件）。

在前几节中我们将可执行文件简单的划分为了两段，数据段和代码段，在这里我们继续丰富可执行文件中的内容，如图所示，在动态链接下，可执行文件当中会新增两段，即dynamic段以及GOT（Global offset table）段，这两段内容就是是我们之前所说的必要信息。

可执行文件

dynamic段中保存了可执行文件依赖哪些动态库，动态链接符号表的位置以及重定位表的位置等信息。关于dynamic以及GOT段的作用限于篇幅就不重点阐述了。如果你对GOT段的具体作用很好奇的话，欢迎关注微信公共账号，码农的荒岛求生。
当加载可执行文件时，操作系统根据dynamic段中的信息即可找到使用的动态库，从而完成动态链接。

这里需要强调一点，在编译链接过程中，可以同时使用动态库以及静态库。这两种库的使用并不冲突，那么在这种情况下生成的可执行文件中，可执行文件中包含了静态库的数据和代码，以及动态库的必要信息。

至此，关于静态库，静态链接，动态库，动态链接就讲述到这，那么接下来的问题就是静态库和动态库都有什么样的优缺点。

动态库vs静态库

在计算机的历史当中，最开始程序只能静态链接，但是人们很快发现，静态链接生成的可执行文件存在磁盘空间浪费问题，因为对于每个程序都需要依赖的libc库，在静态链接下每个可执行文件当中都有一份libc代码和数据的拷贝，为解决该问题才提出动态库。

在前几节我们知道，动态链接下可执行文件当中仅仅保留动态库的必要信息，因此解决了静态链接下磁盘浪费问题。动态库的强大之处不仅仅于此，我们知道对于现代计算机系统，比如PC，通常会运行成百上千个程序（进程），且程序只有被加载到内存中才可以使用，如果使用静态链接那么在内存中就会有成百上千份同样的libc代码，这对于宝贵的内存资源同样是极大的浪费，而使用动态链接，内存中只需要有一份libc代码，所有的程序（进程）共享这一份代码，因此极大的节省了内存资源，这也是为什么动态库又叫共享库。

动态库还有另外一个强大之处，那就是如果我们修改了动态库的代码，我们只需要重新编译动态库就可以了而无需重新新编译我们自己的程序，因为可执行文件当中仅仅保留了动态库的必要信息，重新编译动态库后这些必要都信息是不会改变的（只要不修改动态库的名字和动态库导出的供可执行文件使用的函数），编译好新的动态库后只需要简单的替换原有动态库，下一次运行程序时就可以使用新的动态库了，因此动态库的这种特性极大的方便了程序升级和bug修复。我们平时使用都客户端程序，比如我们常用QQ，输入法，播放器，都利用了动态库的这一优点，原因就在于方便升级以bug修复，只需要更新相应的动态库就可以了。

动态库的优点不止于此，我们知道动态链接可以出现在运行时（run-time dynamic link），动态链接的这种特性可以用于扩展程序能力，那么如何扩展呢？你肯定听说过一样神器，没错，就是插件。你有没有想过插件是怎么实现的？实现插件时，我们只需要实现几个规定好的几个函数，我们的插件就可以运行了，可这是怎么做到的呢，答案就在于运行时动态链接，可以将插件以动态的都方式实现。我们知道使用运行时动态链接无需在编译链接期间告诉链接器所使用的动态库信息，可执行文件对此一无所知，只有当运行时才知道使用什么动态库，以及使用了动态库中哪些函数，但是在编译链接可执行文件时又怎么知道插件中定义了哪些函数呢，因此所有的插件实现函数必须都有一个统一的格式，程序在运行时需要加载所有插件（动态库），然后调用所有插件的入口函数（统一的格式），这样我们写的插件就可以被执行起来了。

动态库都强大优势还体现在多语言编程上。我们知道使用Python可以快速进行开发，但Python的性能无法同C/C++相比(因为Python是解释型语言，至于什么是解释型语言我会在后面码农的荒岛求生系列文章当中给大家详细讲解)，有没有办法可以兼具Python的快速开发能力以及C/C++的高性能呢，答案是可以的，我们可以将C/C++代码编译链接成动态库，这样python就可以直接调用动态库中的函数了。不但Python，Perl以及Java等都可以通过动态库的形式调用C/C++代码。动态库的使用使得同一个项目不同语言混合编程成为可能，而且动态库的使用更大限度的实现了代码复用。

了解了动态库的这么多优点，那么动态库就没有缺点吗，当然是有的。

首先由于动态库是程序加载时或运行是才进行链接的，因此同静态链接相比，使用动态链接的程序在性能上要稍弱于静态链接，这时因为对于加载时动态链接，这无疑会减慢程序都启动速度，而对于运行时链接，当首次调用到动态库的函数时，程序会被暂停，当链接过程结束后才可以继续进行。且动态库中的代码是地址无关代码（Position-Idependent Code，PIC），之所以动态库中的代码是地址无关代码是因为动态库又被成为共享库，所有的程序都可以调用动态库中的代码，因此在使用动态库中的代码时程序要多做一些工作，这里我们不再具体展开讲解到底程序多做了哪些工作，对此感兴趣当同学可以参考CSAPP（深入理解计算机系统）。这里我们说动态链接的程序性能相比静态链接稍弱，但是这里的性能损失是微乎其微的，同动态库可以带来的好处相比，我们可以完全忽略这里的性能损失，同学们可以放心的使用动态库。

动态库的一个优点其实也是它的缺点，即动态链接下的可执行文件不可以被独立运行（这里讨论的是加载时动态链接，load-time dynamic link），换句话说就是，如果没有提供所依赖的动态库或者所提供的动态库版本和可执行文件所依赖的不兼容，程序是无法启动的。动态库的依赖问题会给程序的安装部署带来麻烦，在Linux环境下尤其严重，以笔者曾参与开发维护的一个虚拟桌面系统为例，我们在开发过程中依赖的一些比较有名的第三方库默认不会随着安装包发布，这就会导致用户在较低版本Linux中安装时经常会出现程序无法启动的问题，原因就在于我们编译链接使用都动态库和用户Linux系统中都动态库不兼容。解决这个问题的方法通常有两种，一个是用户升级系统中都动态库，另一个是我们讲需要都第三方库随安装包一起发布，当然这是在取得许可的情况下。

在了解了动态库的优缺点后，接下来我们来看一下静态库。

静态链接是最古老也是最简单的链接技术。静态链接都最大优点就是使用简单，编译好的可执行文件是完备的，即静态链接下的可执行文件不需要依赖任何其它的库，因为静态链接下，链接器将所有依赖的代码和数据都写入到了最终的可执行文件当中，这就消除了动态链接下的库依赖问题，没有了库都依赖问题就意味着程序都安装部署都得到了极大都简化。请大家不要小看这一点，这对当今那些拥有海量用户的后端系统来说至关重要，比如类似微信这种量级的系统，其后端会部署在成千上万台机器上，这么多的机器其系统的安装部署以及升级会给运维带来极大挑战，而静态链接下的可执行文件由于不依赖任何库，因为部署非常方便，仅仅用一个新的可执行文件进行覆盖就可以了，因此极大的简化了系统部署以及升级。笔者之前所在的某电商广告后端系统就完全使用静态链接来简化部署升级。

而静态库的缺点相信大家都已经清楚了，那就是静态链接会导致可执行文件过大，且多个程序静态链接同一个静态库的话会导致磁盘浪费的问题。

到这里关于静态库和动态库的讨论就告一段落了，相信大家对于这两种链接类型都有了清晰都认知。接下来让我们稍作休息，开始链接器的下一个重要功能，重定位。

《彻底理解链接器：五，重定位》，欢迎关注微信公众号，码农的荒岛求生，获取更多内容。

收录于话题 #链接器 6

‹ 上一篇
彻底理解链接器：三，库与可执行文件的生成

下一篇 ›
彻底理解链接器：五，重定位

People who liked this content also liked

一年赚了100多万，美金！

码农的荒岛求生