项目三　安装与管理软件包

项目背景

在项目一中曾提到过 GNU 计划与 GPL 授权所产生的自由软件与开放源码，不过，前面的项目都还没有提到真正的开放源码到底是什么。在本项目中，我们将借由 Linux 操作系统的运行文件，理解什么是可运行的程序，以及了解什么是编译器。另外，读者还将学习与程序息息相关的函数库(library)的知识。总之，本项目可以让读者了解如何将开放源码的程序链接到函数库，通过编译成为可以运行的二进制程序(二进制程序)的一系列过程。本项目重点介绍最原始的软件管理方式：使用 Tarball 安装与升级管理软件。

职业能力目标和要求

- 了解开放源码的软件安装与升级。
- 掌握使用传统程序语言进行编译的方法。
- 掌握用 make 进行编译的方法和技能。
- 掌握如何使用 Tarball 管理包。
- 掌握 RPM 安装、查询、移除软件的方法。
- 学会使用 yum 安装与升级软件。

3.1　项目知识准备

3.1.1　开放源码、编译器与可执行文件

Linux 中的软件几乎都经过了 GPL 的授权，所以这些软件均可提供原始程序代码，并且用户可以自行修改该程序源码，以符合个人的需求，这就是开放源码的优点。不过，到底什么是开放源码？这些程序代码到底是什么？Linux 中可以运行的相关软件文件与开放源码之间是如何转换的？不同版本的 Linux 之间能不能使用同一个运行文件？下面将解答相关问题。

在讨论程序代码是什么之前，我们先来谈论什么是可执行文件。在 Linux 系统中，一个文件能不能被运行取决于有没有可运行的权限(具有 x permission)，Linux 系统上的可执行文件其实是二进制文件(二进制程序)，例如/usr/bin/passwd、/bin/touch 等文件。

那么 shell script 是不是可执行文件呢？答案是否定的。shell script 只是利用 shell(例如 bash)这个程序的功能进行一些判断，除了 bash 提供的功能外，最终运行的仍是调用一

些已经编译完成的二进制程序。当然，bash 本身也是一个二进制程序。

使用 file 命令能够测试一个文件是否为 binary 文件。

```
[root@RHEL7-1~]#file /bin/bash
/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.32, stripped
```

如果是二进制而且是可执行文件，就会显示执行文件类别，同时会说明是否使用动态函数库，而如果是一般的脚本，那么就会显示出 text executables 之类的字样。

既然 Linux 操作系统真正识别的是二进制程序，那么该如何制作二进制程序呢？首先使用 vim 来进行程序的撰写，写完的程序就是所谓的原始程序代码。其次，在完成这个源码文件的编写之后，将这个文件编译成操作系统“看得懂”的二进制程序。举个例子来说，在 Linux 中最标准的程序语言为 C，所以使用 C 语言进行原始程序代码的书写，写完之后，以 Linux 上标准的 C 语言编译器 gcc 进行编译，就可以制作出一个可以运行的二进制程序了。

开放源码、编译器、可执行文件总结如下。

- 开放源码：即程序代码，写给用户看的程序语言，但机器并不认识，所以无法运行。
- 编译器：将程序代码编译成机器看得懂的语言，类似翻译者的角色。
- 可执行文件：经过编译器变成二进制程序后，机器看得懂可以直接运行的文件。

3.1.2　make 与 configure

事实上，使用类似 gcc 的编译器来进行编译的过程并不简单，因为一套软件并不会仅有一个程序，而是有大量的程序代码文件，所以除了主程序与副程序需要写出编译过程的命令外，还需要写出最终的链接程序。但是类似 WWW 服务器软件（例如 Apache）或核心的源码这种动辄数百 MB 的数据量，编译命令的量过于庞大。这时就可以使用 make 这个命令的相关功能来进行编译过程的命令简化了。

当运行 make 时，make 会在当前的目录下搜寻 Makefile（或 makefile）这个文件，而 Makefile 里面记录了源码如何编译的详细信息。make 会自动判别源码是否已经改变，从而自动升级执行文件，所以，make 是相当好用的一个辅助工具。

make 是一个程序，会去找 Makefile，那么 Makefile 应该怎么撰写呢？通常软件开发商都会写一个检测程序来检测使用者的操作环境，以及该操作环境是否有软件开发商所需要的其他功能，该检测程序检测完毕后，就会主动创建这个 Makefile 的规则文件。通常这个检测程序的文件名为 configure 或者是 config。

为什么要检测操作环境呢？因为不同版本的核心所使用的系统调用可能不相同，而且每个软件所需要的相关的函数库也不相同。同时，软件开发商不会仅针对 Linux 开发，而是会针对整个 Unix-Like 做开发，所以也必须要检测该操作系统平台有没有提供合适的编译器。一般来说，检测程序所检测的数据有如下几种类型：

- 是否有适合的编译器可以编译本软件的程序代码；
- 是否已经存在本软件所需要的函数库，或其他需要的相关软件；
- 操作系统平台是否适合本软件，包括 Linux 的核心版本；

- 内核的头定义文件(header include)是否存在(驱动程序必须要进行的检测)。

由于不同的 Linux 发行版本的函数库文件的路径、函数库的文件名定义、默认安装的编译器以及内核的版本都不相同,因此理论上,在 CentOS 5.x 上编译出二进制程序后无法在 SuSE 上运行。因为调用的目标函数库位置可能不同,内核版本更不可能相同,所以能够运行的概率微乎其微。当同一套软件在不同的平台上运行时,必须要重复编译。

3.1.3 Tarball 软件

Tarball 文件是将软件的所有源码文件先以 tar 打包,然后再用压缩技术进行压缩从而得到的文件,其中最常见的就是以 gzip 进行压缩。因为利用了 tar 与 gzip 的功能,所以 tarball 文件一般的扩展名会写成 *.tar.gz 或者是简写为 *.tgz。不过,近来由于 bzip2 的压缩率较佳,所以 Tarball 也经常使用 bzip2 的压缩技术进行压缩,这时文件名会写成 *.tar.bz2。

Tarball 是一个软件包,将其解压缩之后,里面的文件通常会有:

- 原始程序代码文件;
- 检测程序文件(可能是 configure 或 config 等文件名);
- 本软件的简易说明与安装说明(Install 或 Readme)。

其中最重要的是 Install 或者 Readme 这两个文件,通常只要能够看明白这两个文件,Tarball 软件的安装就非常容易进行了。

3.1.4 安装与升级软件

软件升级的主要原因有以下几种:

- 需要新的功能,但旧版软件并没有这种功能;
- 旧版软件可能存在安全隐患;
- 旧版软件运行效率不高,或者运行的能力不能满足管理者的需求。

在上面的需求中,尤其需要注意的是第二点,当一个软件有安全隐患时,千万不要怀疑,最好的办法就是立即升级软件,否则有可能造成严重的网络危机。升级的方法可以分为两大类:

- 直接以源码通过编译来安装与升级;
- 直接以编译好的二进制程序来安装与升级。

上面第一点很简单,就是直接以 Tarball 进行检测、编译、安装与配置等操作进行升级。不过,这样的操作虽然让使用者在安装过程中具有很高的选择性,但是比较麻烦。如果 Linux distribution 厂商能够针对自己的操作平台先进行编译等过程,再将编译好的二进制程序发布,那么由于自己的系统与该 Linux distribution 的环境是相同的,所以厂商发布的二进制程序就可以在自己的机器上直接安装,省略了检测与编译等繁杂的过程。

这种预先编译好程序的机制存在于很多 distribution 版本中。如由 Red Hat 系统(含 Fedora/CentOS 系列)发展的 RPM 软件管理机制与 yum 线上升级模式、Debian 使用的 dpkg 软件管理机制与 APT 线上升级模式等。

Tarball 安装的基本流程如下:

(1) 将 Tarball 在厂商的网站上下载下来；

(2) 将 Tarball 解压缩，生成源码文件；

(3) 以 gcc 进行源码的编译(会产生目标文件 object files)；

(4) 以 gcc 进行函数库、主程序和副程序的链接，形成主要的二进制文件；

(5) 将(4)中的二进制文件以及相关的配置文件安装至主机中。

步骤(3)和(4)可以通过 make 命令的功能进行简化，所以整个步骤其实非常简单，只需要在 Linux 系统中至少有 gcc 和 make 两个软件即可。

3.1.5　RPM 与 DPKG

目前在 Linux 界最常见的软件安装方式有以下两种。

(1) DPKG

DPKG 最早是由 Debian Linux 社群开发出来的，通过 DPKG 机制，Debian 提供的软件安装非常简单，同时还能提供安装后的软件信息。衍生于 Debian 的其他 Linux 发行版本大多数也都使用 DPKG 机制来管理软件，包括 B2D、Ubuntu 等。

(2) RPM

RPM 最早是由 Red Hat 公司开发出来的。后来由于软件非常好用，很多发行版就使用这个机制来作为软件安装的管理方式，其中包括 Fedora、CentOS、SuSE 等知名的开发商。

如前所述，DPKG/RPM 机制或多或少都会存在软件依赖性的问题，那么该如何解决呢？由于每个软件文件都提供软件依赖性的检查，如果将依赖属性的数据做成列表，等到实际安装软件时，若存在依赖属性的软件时根据列表安装软件就可以解决依赖性问题。例如，安装 A 需要先安装 B 与 C，而安装 B 则需要安装 D 与 E，那么当要安装 A 时，通过依赖属性列表，管理机制自动去取得 B、C、D、E 同时进行安装，就解决了软件依赖性的问题。

目前新的 Linux 开发商都提供这样的“线上升级”机制，通过这个机制，原版光盘只有第一次安装时用到，其他时候只要有网络，就能够获得开发商所提供的任何软件。在 DPKG 管理机制上开发出了 APT 线上升级机制，RPM 则根据开发商的不同，有 Red Hat 系统的 YUM、SuSE 系统的 Yast Online Update(YOU)、Mandriva 的 urpmi 软件等。线上升级如表 3-1 所示。

表 3-1　各发行版本的线上升级

distribution 代表	软件管理机制	使用命令	线上升级机制(命令)
Red Hat/Fedora	RPM	rpm，rpmbuild	YUM(yum)
Debian/Ubuntu	DPKG	dpkg	APT(apt-get)

RHEL 7 使用的软件管理机制为 RPM 机制，而用来作为线上升级的方式则为 yum。

3.1.6　RPM 与 SRPM

RPM 全名是 Red Hat Package Manager，简称为 RPM。顾名思义，当初这个软件管理的机制是由 Red Hat 公司开发出来的。RPM 是以一种数据库记录的方式将所需要的软件安装到 Linux 系统的一套管理机制。

RPM 最大的特点是将需要安装的软件先编译通过,并且打包成 RPM 机制的包装文件,通过包装文件里默认的数据库记录,记录软件安装时所必须具备的依赖属性软件。当软件安装在 Linux 主机时,RPM 会先依照软件里面的数据库记录查询 Linux 主机的依赖属性软件是否满足,若满足则予以安装,若不满足则不予安装。

但是这也造成一些困扰。由于 RPM 文件是已经打包好的数据,里面的数据已经"编译完成"了,所以该软件文件只能安装在原来默认的硬件与操作系统版本中。也就是说,主机系统环境必须要与当初创建这个软件文件的主机环境相同才行。举例来说,rp-pppoe 这个 ADSL 拨号软件,必须要在 ppp 软件存在的环境下才能进行安装。如果主机没有 ppp 软件,除非先安装 ppp,否则 rp-pppoe 不能成功安装(当然也可以强制安装,但是通常都会出现一些问题)。

所以,通常不同的发行版本发布的 RPM 文件,并不能用在其他的 distributions 上。举例来说,Red Hat 发布的 RPM 文件,通常无法直接在 SuSE 上进行安装。更有甚者,相同发行版本的不同子版本之间也无法互通,例如 RHEL 4. x 的 RPM 文件就无法直接套用在 RHEL 5. x 上。由此可知,使用 RPM 应注意以下问题:

- 软件文件安装的环境必须与打包时的环境需求一致或相当;
- 需要满足软件的依赖属性需求;
- 反安装时需要特别小心,最底层的软件不可先移除,否则可能造成整个系统出现问题。

如果想要安装其他发行版本提供的 RPM 软件文件,需要使用 SRPM。

SRPM 即 Source RPM,也就是 RPM 文件里面含有源码。应特别注意的是,SRPM 所提供的软件内容是源码,并没有经过编译。

通常 SRPM 的扩展名以***. src. rpm 格式命名。虽然 SRPM 提供的是源码,但却不能使用 Tarball 直接安装。这是因为 SRPM 虽然内容是源码,但是仍然含有该软件所需要的依赖性软件说明以及 RPM 文件所提供的数据。同时,SRPM 与 RPM 不同之处是,SRPM 也提供参数配置文件(configure 与 makefile)。所以,如果下载的是 SRPM,那么要安装该软件时,需要完成以下两个步骤:

(1) 将该软件以 RPM 管理的方式编译,此时 SRPM 会被编译成 RPM 文件;

(2) 将编译完成的 RPM 文件安装到 Linux 系统中。

通常一个软件在发布的时候,都会同时发布该软件的 RPM 与 SRPM。RPM 文件必须在相同的 Linux 环境下才能够安装,而 SRPM 是源码的格式,可以通过修改 SRPM 内的参数配置文件,然后重新编译产生能适合 Linux 环境的 RPM 文件,这样就可以将该软件安装到系统,而不必要求与原作者打包的 Linux 环境相同。通过表 3-2 可以看出 RPM 与 SRPM 之间的差异。

表 3-2 RPM 与 SRPM 的比较

文件格式	文件名格式	能否直接安装	内含程序类型	可否修改参数并编译
RPM	xxx. rpm	可	已编译	不可
SRPM	xxx. src. rpm	不可	未编译之源码	可