



马哥教育

IT 人的高薪职业学院

软件包管理

讲师：王晓春

本章内容



马哥教育

IT 人的高薪职业学院

- ◆ 软件运行环境
- ◆ 软件包基础
- ◆ rpm包管理
- ◆ yum管理
- ◆ 定制yum仓库
- ◆ dnf管理
- ◆ 编译安装
- ◆ Ubuntu软件管理

◆ ABI : Application Binary Interface

Windows与Linux不兼容

ELF(Executable and Linkable Format)

PE (Portable Executable)

库级别的虚拟化：

Linux: WINE

Windows: Cygwin

◆ API : Application Programming Interface

POSIX : Portable OS

◆ 程序源代码 --> 预处理 --> 编译 --> 汇编 --> 链接

静态编译：.a

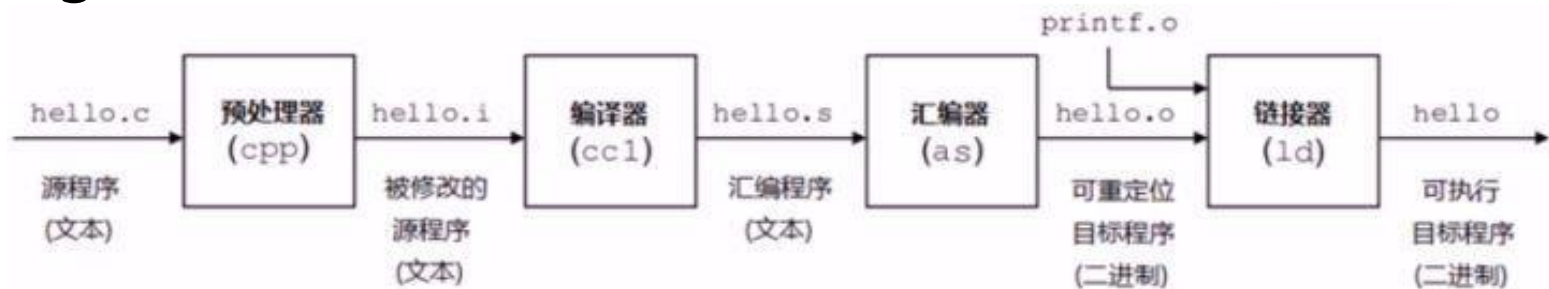
动态编译：.so

gcc 编译程序



◆ gcc 编译程序主要经过四个过程：

- 预处理 (Pre-Processing)
- 编译 (Compiling)
- 汇编 (Assembling)
- 链接 (Linking)

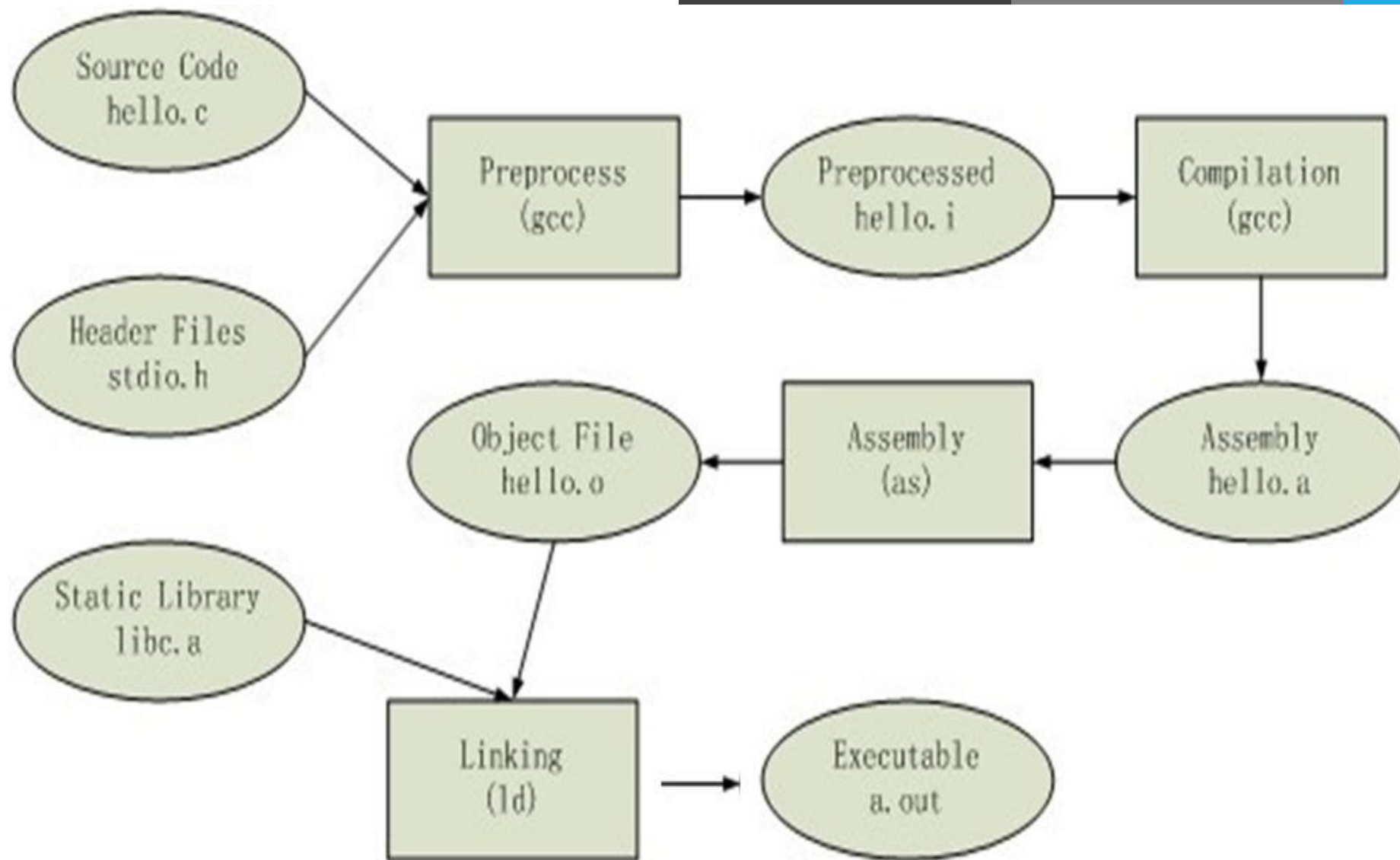


◆ gcc编译过程

```
gcc -E hello.c -o hello.i
gcc -S hello.i -o hello.s
gcc -c hello.s -o hello.o
gcc hello.o -o hello
gcc hello.c -o hello
```

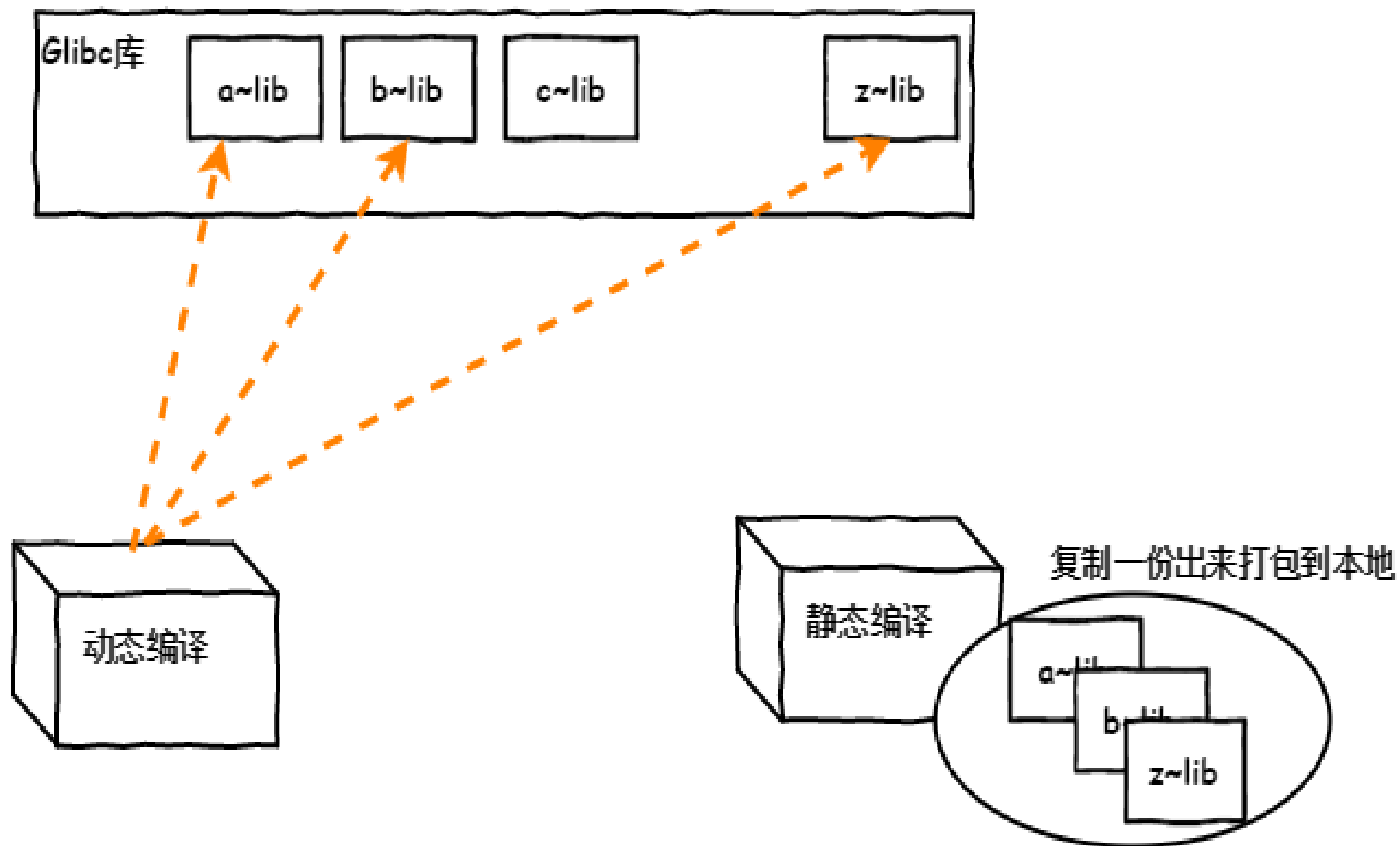
对hello.c文件进行预处理，生成了hello.i 文件
对预处理文件进行编译，生成了汇编文件
对汇编文件进行编译，生成了目标文件
对目标文件进行链接，生成可执行文件
直接编译链接成可执行目标文件

C程序编译过程

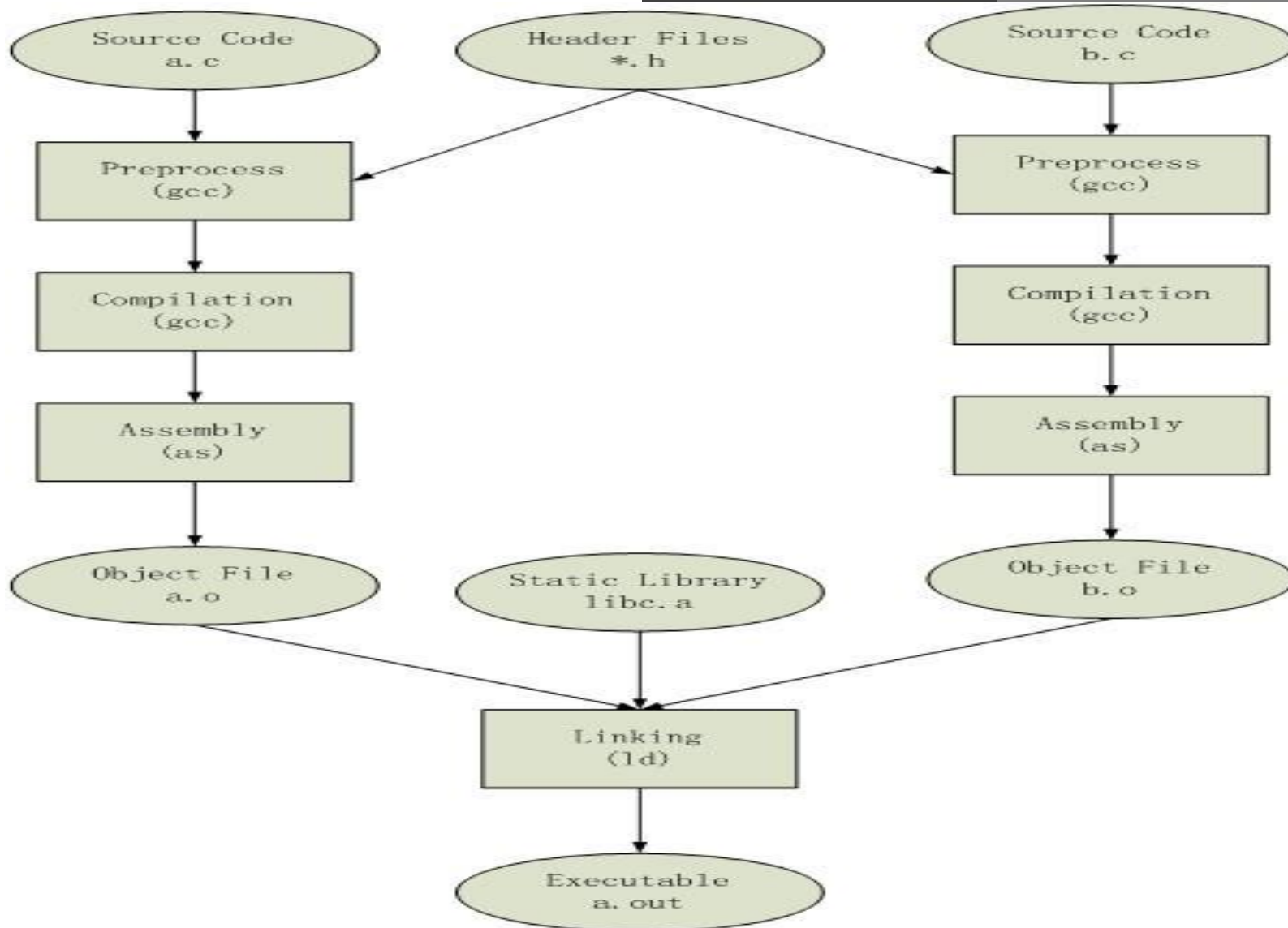


- ◆ 链接主要作用是把各个模块之间相互引用的部分处理好，使得各个模块之间能够正确地衔接，分为静态链接和动态链接
- ◆ 静态链接
 - 把程序对应的依赖库复制一份到包
 - libxxx.a
 - 嵌入程序包
 - 升级难，需重新编译
 - 占用较多空间，迁移容易
- ◆ 动态链接
 - 只把依赖加做一个动态链接
 - libxxx.so
 - 连接指向
 - 占用较少空间，升级方便

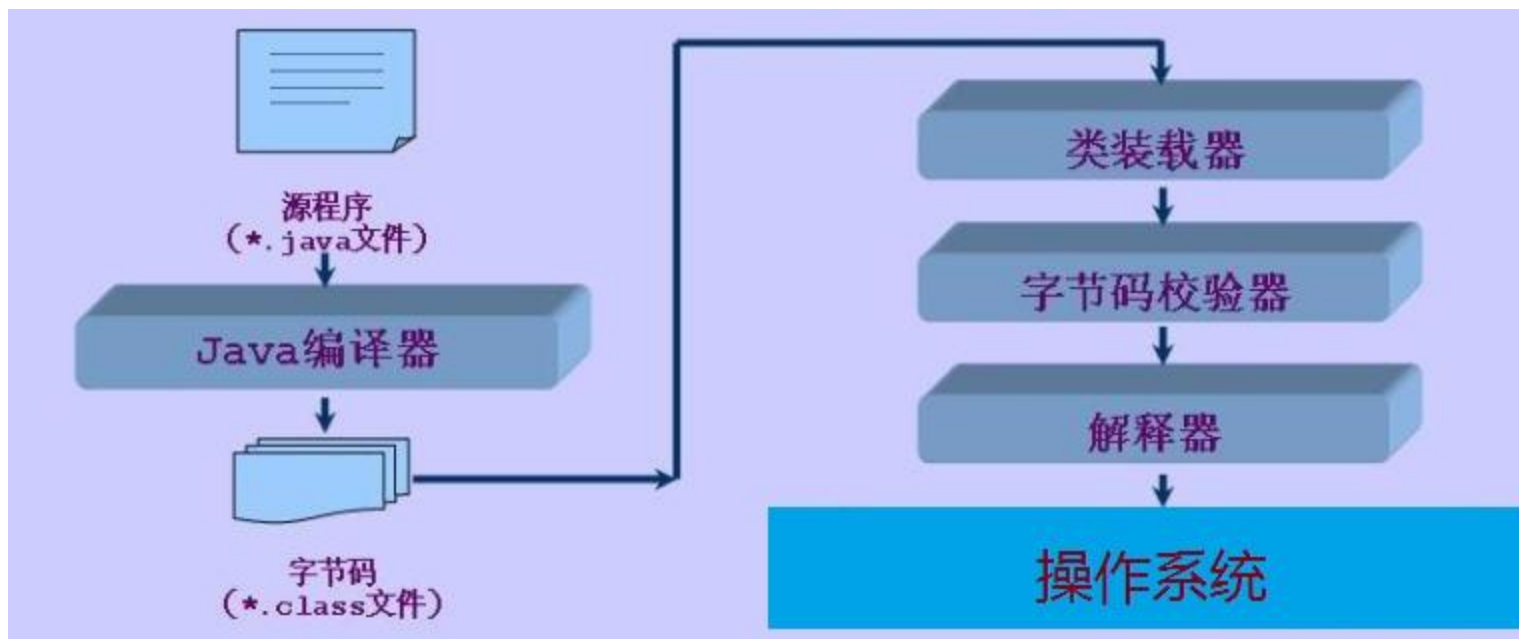
静态和动态链接



C程序静态链接



Java程序运行



开发语言



马哥教育

IT 人的高薪职业学院

◆ 系统级开发

C

C++

◆ 应用级开发

java

Python

go

php

perl

delphi

ruby

- ◆ 最初只提供了.tar.gz的打包的源码文件，用户必须自己编译每个想在GNU/Linux上运行的软件。用户急需系统能提供一种更加便利的方法来管理这些软件，当Debian诞生时，这样一个管理工具也就应运而生，它被命名为dpkg。从而著名的“package”概念第一次出现在GNU/Linux系统中，稍后Red Hat才开发自己的“rpm”包管理系统
- ◆ 包的组成：
 - 二进制文件、库文件、配置文件、帮助文件
- ◆ 程序包管理器：
 - debian : deb文件, dpkg包管理器
 - redhat : rpm文件, rpm包管理器
 - rpm : Redhat Package Manager
 - RPM Package Manager

- ◆ 源代码：name-VERSION.tar.gz|bz2|xz
VERSION: major.minor.release
- ◆ rpm包命名方式：
name-VERSION-release.arch.rpm
例：bash-4.2.46-19.el7.x86_64.rpm
VERSION: major.minor.release
release：release.OS
常见的arch：
x86: i386, i486, i586, i686
x86_64: x64, x86_64, amd64
powerpc: ppc
跟平台无关：noarch

◆ 包：分类和拆包

Application-VERSION-ARCH.rpm: 主包

Application-devel-VERSION-ARCH.rpm 开发子包

Application-utils-VERSION-ARCH.rpm 其它子包

Application-libs-VERSION-ARCH.rpm 其它子包

◆ 包之间：可能存在依赖关系，甚至循环依赖

◆ 解决依赖包管理工具：

yum：rpm包管理器的前端工具

apt：deb包管理器前端工具

zypper：suse上的rpm前端管理工具

dnf：Fedora 18+ rpm包管理器前端管理工具

◆ 查看二进制程序所依赖的库文件

`ldd /PATH/TO/BINARY_FILE`

◆ 管理及查看本机装载的库文件

`ldconfig` 加载配置文件中指定的库文件

`/sbin/ldconfig -p` 显示本机已经缓存的所有可用库文件名及文件路径

映射关系

配置文件：`/etc/ld.so.conf, /etc/ld.so.conf.d/*.conf`

缓存文件：`/etc/ld.so.cache`

◆ 程序包管理器：

功能：将编译好的应用程序的各组成文件打包一个或几个程序包文件，从而方便快捷地实现程序包的安装、卸载、查询、升级和校验等管理操作

◆ 包文件组成 (每个包独有)

RPM包内的文件

RPM的元数据，如名称，版本，依赖性，描述等

安装或卸载时运行的脚本

◆ 数据库(公共)：/var/lib/rpm

程序包名称及版本

依赖关系

功能说明

包安装后生成的各文件路径及校验码信息

◆ 管理程序包的方式：

使用包管理器：rpm

使用前端工具：yum, dnf

◆ 获取程序包的途径：

(1) 系统发版的光盘或官方的服务器

CentOS镜像：

<https://www.centos.org/download/>

<http://mirrors.aliyun.com>

<http://mirrors.sohu.com>

<http://mirrors.163.com>

(2) 项目官方站点

◆ (3) 第三方组织：

Fedora-EPEL：

Extra Packages for Enterprise Linux

Rpmforge:RHEL推荐，包很全

搜索引擎：

<http://pkgs.org>

<http://rpmfind.net>

<http://rpm.pbone.net>

<https://sourceforge.net/>

◆ (4) 自己制作

- ◆ 注意：第三方包建议要检查其合法性
来源合法性,程序包的完整性

◆ CentOS系统上使用rpm命令管理程序包：

安装、卸载、升级、查询、校验、数据库维护

安装：

```
rpm {-i|--install} [install-options] PACKAGE_FILE...
```

-v: verbose

-vv:

-h: 以#显示程序包管理执行进度

```
rpm -ivh PACKAGE_FILE ...
```

◆ [install-options]

--test: 测试安装，但不真正执行安装，即dry run模式

--nodeps : 忽略依赖关系

--replacepkgs | replacefiles 覆盖安装，解决原文件缺失

--nosignature: 不检查来源合法性

--nodigest : 不检查包完整性

--noscripts : 不执行程序包脚本

%pre: 安装前脚本

--nopre

%post: 安装后脚本

--nopost

%preun: 卸载前脚本

--nopreun

%postun: 卸载后脚本

--nopostun

◆ 升级：

◆ rpm {-U|--upgrade} [install-options] PACKAGE_FILE...

◆ rpm {-F|--freshen} [install-options] PACKAGE_FILE...

upgrade：安装有旧版程序包，则“升级”

如果不存在旧版程序包，则“安装”

freshen：安装有旧版程序包，则“升级”

如果不存在旧版程序包，则不执行升级操作

rpm -Uvh PACKAGE_FILE ...

rpm -Fvh PACKAGE_FILE ...

--oldpackage：降级

--force: 强制安装

◆ 注意：

- (1) 不要对内核做升级操作；Linux支持多内核版本并存，因此直接安装新版本内核
- (2) 如果原程序包的配置文件安装后曾被修改，升级时，新版本提供的同一个配置文件不会直接覆盖老版本的配置文件，而把新版本文件重命名(FILENAME.rpmnew)后保留

- ◆ rpm {-q|--query} [select-options] [query-options]
- ◆ [select-options]
 - a : 所有包
 - f : 查看指定的文件由哪个程序包安装生成
 - p rpmfile : 针对尚未安装的程序包文件做查询操作
 - whatprovides CAPABILITY : 查询指定的CAPABILITY由哪个包所提供
 - whatrequires CAPABILITY : 查询指定的CAPABILITY被哪个包所依赖
- ◆ rpm2cpio 包文件|cpio -itv 预览包内文件
- ◆ rpm2cpio 包文件|cpio -id "*.conf" 释放包内文件

◆ [query-options]

- changelog : 查询rpm包的changelog
- c : 查询程序的配置文件
- d : 查询程序的文档
- i : information
- l : 查看指定的程序包安装后生成的所有文件
- scripts : 程序包自带的脚本
- provides : 列出指定程序包所提供的CAPABILITY
- R : 查询指定的程序包所依赖的CAPABILITY

◆ 常用查询用法：

-qi PACKAGE, -qf FILE, -qc PACKAGE, -ql PACKAGE, -qd PACKAGE
-qpi PACKAGE_FILE, -qpl PACKAGE_FILE, ...
-qa

◆ 包卸载：

rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--notriggers]
[--test] PACKAGE_NAME ...

当包卸载时，对应的配置文件不会删除，以FILENAME.rpm.save形式保留

◆ rpm {-V|--verify} [select-options] [verify-options]

S file Size differs

M Mode differs (includes permissions and file type)

5 digest (formerly MD5 sum) differs

D Device major/minor number mismatch

L readLink(2) path mismatch

U User ownership differs

G Group ownership differs

T mTime differs

P capabilities differ

◆ 包来源的合法性验证及完整性验证

完整性验证：SHA256

来源合法性验证：RSA

◆ 公钥加密

对称加密：加密、解密使用同一密钥

非对称加密：密钥是成对儿的

public key: 公钥，公开所有人

secret key: 私钥，不能公开

◆ 导入所需要公钥

`rpm -K|checksig rpmfile` 检查包的完整性和签名

`rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7`

CentOS 7发行版光盘提供：RPM-GPG-KEY-CentOS-7

`rpm -qa "gpg-pubkey*"`

◆ 数据库重建：

/var/lib/rpm

◆ rpm {--initdb|--rebuilddb}

initdb: 初始化

如果事先不存在数据库，则新建之

否则，不执行任何操作

rebuilddb：重建已安装的包头的数据库索引目录

- ◆ CentOS: yum, dnf
- ◆ YUM: Yellowdog Update Modifier , rpm的前端程序 , 可解决软件包相关依赖性 , 可在多个库之间定位软件包 , up2date的替代工具
 - yum repository: yum repo , 存储了众多rpm包 , 以及包的相关的元数据文件 (放置于特定目录repodata下)
 - 文件服务器 :
 - http://
 - https://
 - ftp://
 - file://

◆ yum客户端配置文件：

/etc/yum.conf：为所有仓库提供公共配置

/etc/yum.repos.d/*.repo：为仓库的指向提供配置

仓库指向的定义：

[repositoryID]

name=Some name for this repository

baseurl=url://path/to/repository/

enabled={1|0}

gpgcheck={1|0}

gpgkey=URL

enablegroups={1|0}

failovermethod={roundrobin|priority}

roundrobin：意为随机挑选，默认值

priority:按顺序访问

cost= 默认为1000

◆ yum的repo配置文件中可用的变量：

\$releasever: 当前OS的发行版的主版本号

\$arch: 平台，i386,i486,i586,x86_64等

\$basearch：基础平台；i386, x86_64

\$YUM0-\$YUM9:自定义变量

◆ 示例：

[http://server/centos/\\$releasever/\\$basearch/](http://server/centos/$releasever/$basearch/)

http://server/centos/7/x86_64

<http://server/centos/6/i386>

◆ 阿里云repo文件

<http://mirrors.aliyun.com/repo/>

◆ CentOS系统的yum源

➤ 阿里云：[https://mirrors.aliyun.com/centos/\\$releasever/os/x86_64/](https://mirrors.aliyun.com/centos/$releasever/os/x86_64/)

➤ 清华大学：[https://mirrors.tuna.tsinghua.edu.cn/centos/\\$releasever/os/x86_64/](https://mirrors.tuna.tsinghua.edu.cn/centos/$releasever/os/x86_64/)

◆ EPEL的yum源

➤ 阿里云：[https://mirrors.aliyun.com/epel/\\$releasever/x86_64](https://mirrors.aliyun.com/epel/$releasever/x86_64/)

◆ 阿里巴巴开源软件

<https://opsx.alibaba.com/>

yum-config-manager



- ◆ 生成172.16.0.1_cobbler_ks_mirror_CentOS-X-x86_64_repo
yum-config-manager --add-repo= http://172.16.0.1/cobbler/ks_mirror/7/
- ◆ yum-config-manager --disable “仓库名” 禁用仓库
- ◆ yum-config-manager --enable “仓库名” 启用仓库

◆ yum命令的用法：

```
yum [options] [command] [package ...]
```

◆ 显示仓库列表：

```
yum repolist [all|enabled|disabled]
```

◆ 显示程序包：

```
yum list
```

```
yum list [all | glob_exp1] [glob_exp2] [...]
```

```
yum list {available|installed|updates} [glob_exp1] [...]
```

◆ 安装程序包：

```
yum install package1 [package2] [...]
```

```
yum reinstall package1 [package2] [...] (重新安装)
```

◆ 升级程序包：

```
yum update [package1] [package2] [...]
```

```
yum downgrade package1 [package2] [...] (降级)
```

◆ 检查可用升级：

```
yum check-update
```

◆ 卸载程序包：

```
yum remove | erase package1 [package2] [...]
```

- ◆ 查看程序包information :

yum info [...]

- ◆ 查看指定的特性(可以是某文件)是由哪个程序包所提供 :

yum provides | whatprovides feature1 [feature2] [...]

- ◆ 清理本地缓存 :

清除/var/cache/yum/\$basearch/\$releasever缓存

yum clean [packages | metadata | expire-cache | rpmdb | plugins |
all]

- ◆ 构建缓存 :

yum makecache

- ◆ 搜索：yum search string1 [string2] [...]
以指定的关键字搜索程序包名及summary信息
- ◆ 查看指定包所依赖的capabilities：
yum deplist package1 [package2] [...]
- ◆ 查看yum事务历史：
yum history [info|list|packages-list|packages-info|
summary|addon-info|redo|undo|
rollback|new|sync|stats]
yum history
yum history info 6
yum history undo 6
- ◆ 日志：/var/log/yum.log

◆ 安装及升级本地程序包：

```
yum localinstall rpmfile1 [rpmfile2] [...]
```

(用install替代)

```
yum localupdate rpmfile1 [rpmfile2] [...]
```

(用update替代)

◆ 包组管理的相关命令：

```
yum groupinstall group1 [group2] [...]
```

```
yum groupupdate group1 [group2] [...]
```

```
yum grouplist [hidden] [groupwildcard] [...]
```

```
yum groupremove group1 [group2] [...]
```

```
yum groupinfo group1 [...]
```

◆ yum的命令行选项：

--nogpgcheck：禁止进行gpg check

-y：自动回答为“yes”

-q：静默模式

--disablerepo=repoidglob：临时禁用此处指定的repo

--enablerepo=repoidglob：临时启用此处指定的repo

--noplugins：禁用所有插件

◆ 系统安装光盘作为本地yum仓库：

- (1) 挂载光盘至某目录，例如/mnt/cdrom

```
mount /dev/cdrom /mnt/cdrom
```

- (2) 创建配置文件

```
[CentOS7]
```

```
name=
```

```
baseurl=
```

```
gpgcheck=
```

```
enabled=
```

◆ 创建yum仓库：

```
createrepo [options] <directory>
```

DNF (DaNdiFied)

- ◆ DNF 介绍：新一代的RPM软件包管理器。DNF 发行日期是2015年5月11日，DNF 包管理器采用Python 编写，发行许可为GPL v2，首先出现在Fedora 18 发行版中。在 RHEL 8.0 版本正式取代了 YUM，DNF包管理器克服了YUM包管理器的一些瓶颈，提升了包括用户体验，内存占用，依赖分析，运行速度等
- ◆ 下载安装所需软件包，或者利用extras仓库安装
 - wget http://springdale.math.ias.edu/data/puias/unsupported/7/x86_64/dnf-conf-0.6.4-2.sdl7.noarch.rpm
 - wget http://springdale.math.ias.edu/data/puias/unsupported/7/x86_64/dnf-0.6.4-2.sdl7.noarch.rpm
 - wget http://springdale.math.ias.edu/data/puias/unsupported/7/x86_64/python-dnf-0.6.4-2.sdl7.noarch.rpm
 - wget https://mirrors.aliyun.com/centos/7/extras/x86_64/Packages/python2-libcomps-0.1.8-12.el7.x86_64.rpm
 - wget https://mirrors.aliyun.com/centos/7/extras/x86_64/Packages/libcomps-0.1.8-12.el7.x86_64.rpm
- ◆ 配置文件：/etc/dnf/dnf.conf
- ◆ 仓库文件：/etc/yum.repos.d/ *.repo
- ◆ 日志： /var/log/dnf.rpm.log , /var/log/dnf.log

◆ 帮助：man dnf

◆ dnf 用法：与yum一致

```
dnf [options] <command> [<arguments>...]
```

```
dnf --version
```

```
dnf repolist
```

```
dnf install httpd
```

```
dnf remove httpd
```

```
dnf clean all
```

```
dnf makecache
```

```
dnf list installed
```

```
dnf list available
```

```
dnf search nano
```

```
dnf history undo 1
```

- ◆ 程序包编译安装：
- ◆ Application-VERSION-release.src.rpm --> 安装后，使用rpmbuild命令制作成二进制格式的rpm包，而后再安装
- ◆ 源代码-->预处理-->编译-->汇编-->链接-->执行
- ◆ 源代码组织格式：
 - 多文件：文件中的代码之间，很可能存在跨文件依赖关系
 - C、C++：make 项目管理器
 - configure脚本 --> Makefile.in --> Makefile
 - java: maven

◆ C语言源代码编译安装三步骤：

➤ 1、./configure

(1) 通过选项传递参数，指定启用特性、安装路径等；执行时会参考用户的指定以及Makefile.in文件生成Makefile

(2) 检查依赖到的外部环境，如依赖的软件包

➤ 2、make 根据Makefile文件，构建应用程序

➤ 3、make install 复制文件到相应路径

◆ 开发工具：

autoconf: 生成configure脚本

automake : 生成Makefile.in

◆ 注意：安装前查看README，INSTALL

◆ 开源程序源代码的获取：

官方自建站点：

apache.org (ASF : Apache Software Foundation)

mariadb.org

...

代码托管：

SourceForge.net

Github.com

code.google.com

◆ c/c++编译器: gcc (GNU C Compiler)

◆ 编译C源代码：

准备：提供开发工具及开发环境

开发工具：make, gcc等

开发环境：开发库，头文件

glibc：标准库

实现：通过“包组”提供开发组件

Development Tools

Server Platform Development

生产实践：基于最小化安装的系统建议安装下面相关包

```
yum install gcc gcc-c++ glibc glibc-devel pcre pcre-devel  
openssl openssl-devel systemd-devel zlib-devel vim lrzsz tree screen lsof  
tcpdump wget ntpdate net-tools iotop bc bzip2 zip unzip nfs-utils
```

◆ 第一步：configure脚本

选项：指定安装位置、指定启用的特性

--help: 获取其支持使用的选项

选项分类：

安装路径设定：

--prefix=/PATH：指定默认安装位置,默认为/usr/local/

--sysconfdir=/PATH：配置文件安装位置

System types：支持交叉编译

- Optional Features: 可选特性
 - disable-FEATURE
 - enable-FEATURE[=ARG]
- Optional Packages: 可选包
 - with-PACKAGE[=ARG] 依赖包
 - without-PACKAGE 禁用依赖关系
- 注意：通常被编译操作依赖的程序包，需要安装此程序包的“开发”组件，其包名一般类似于name-devel-VERSION
- ◆ 第二步：make
- ◆ 第三步：make install

◆ 安装后的配置：

- (1) 二进制程序目录导入至PATH环境变量中

编辑文件/etc/profile.d/NAME.sh

```
export PATH=/PATH/TO/BIN:$PATH
```

- (2) 导入帮助手册

编辑/etc/man.config|man_db.conf文件

添加一个MANPATH

- ◆ Debian软件包通常为预编译的二进制格式的扩展名“.deb”，类似rpm文件，因此安装快速，无需编译软件。包文件包括特定功能或软件所必需的文件、元数据和指令
- ◆ dpkg：package manager for Debian，类似于rpm，dpkg是基于Debian的系统的包管理器。可以安装，删除和构建软件包，但无法自动下载和安装软件包或其依赖项
- ◆ APT：Advanced Packaging Tool，功能强大的软件管理工具，甚至可升级整个Ubuntu的系统，基于客户/服务器架构
- ◆ APT工作原理：在服务器上先复制所有DEB包，然后用APT的分析工具genbasedir根据每个DEB包的包头（Header）信息对所有的DEB包进行分析，并将该分析结果记录在文件夹base内的一个DEB索引清单文件中，一旦APT服务器内的DEB有所变动，要使用genbasedir产生新的DEB索引清单。客户端在进行安装或升级时先要查询DEB索引清单，从而获知所有具有依赖关系的软件包，并一同下载到客户端以便安装。当客户端需要安装、升级或删除某个软件包时，客户端计算机取得DEB索引清单压缩文件后，会将其解压置放于/var/cache/apt/，而客户端使用apt-get install或apt-get upgrade命令的时候，就会将这个文件夹内的数据和客户端计算机内的DEB数据库比对，知道哪些DEB已安装、未安装或是可以升级的

- ◆ dpkg 常见用法：`man dpkg`
- `dpkg -i package.deb` 安装包
- `dpkg -r package` 删除包，不建议，不自动卸载依赖于它的包
- `dpkg -P package` 删除包（包括配置文件）
- `dpkg -l` 列出当前已安装的包，类似 `rpm -qa`
- `dpkg -l package` 显示该包的简要说明，类似 `rpm -qi`
- `dpkg -L package` 列出该包中所包含的文件，类似 `rpm -ql`
- `dpkg -S <pattern>` 搜索包含 `pattern` 的包，类似 `rpm -qf`
- `dpkg -s package` 列出该包的状态，包括详细信息，类似 `rpm -qi`
- `dpkg --configure package` 配置包，`-a` 使用，配置所有没有配置的软件包
- `dpkg -c package.deb` 列出 `deb` 包的内容，类似 `rpm -qpl`
- `dpkg --unpack package.deb` 解开 `deb` 包的内容

- ◆ dpkg 示例：
- ◆ 列出系统上安装的所有软件包
`dpkg -l`
- ◆ 列出软件包安装的文件
`dpkg -L bash`
- ◆ 查看/bin/bash来自于哪个软件包
`dpkg -S /bin/bash`
- ◆ 安装本地的 .deb 文件
`dpkg -i /mnt/cdrom/pool/main/z/zip/zip_3.0-11build1_amd64.deb`
- ◆ 卸载软件包
`dpkg -r zip`
- ◆ 注意：一般建议不要使用dpkg卸载软件包。因为删除包时，其它依赖它的包不会卸载，并且可能无法再正常运行

- ◆ Debian 使用APT工具来管理包系统，它与 apt 命令不同。在基于 Debian 的 Linux 发行版中，有各种工具可以与 APT 进行交互，以方便用户安装、删除和管理的软件包。apt-get 是其中一个常用的命令行工具，另外一款较为流行的命令行与 GUI 兼顾的工具是 aptitude，之前最常用的 Linux 包管理命令都被分散在了 apt-get、apt-cache 和 apt-config 这三条命令中
- ◆ 在 2014 年 apt 命令发布第一个稳定版，Ubuntu 16.04 引入新特性之一便是 apt 命令，apt 命令解决了命令过于分散的问题，它包括 apt-get 命令出现以来使用最广泛的功能选项，以及 apt-cache 和 apt-config 命令中很少用到的功能。在使用 apt 命令时，用户不必再由 apt-get 转到 apt-cache 或 apt-config，提供管理软件包所需的必要选项
- ◆ apt 相当于 apt-get、apt-cache 和 apt-config 中最常用命令选项的集合
- ◆ apt 具有更精简但足够的命令选项，而且参数选项的组织方式更为有效。此外，启用的几个特性也非常有帮助。例如：可以在使用 apt 命令安装或删除程序时看到进度条,apt 还会在更新存储库数据库时提示用户可升级的软件包个数
- ◆ apt 与 apt-get 有一些类似的命令选项，但它并不能完全向下兼容 apt-get 命令,也即可用 apt 替换部分 apt-get 系列命令，但不是全部

◆ 查看帮助：apt help

◆ apt与apt-get命令对比

apt 命令	被取代的命令	命令的功能
apt install	apt-get install	安装软件包
apt remove	apt-get remove	移除软件包
apt purge	apt-get purge	移除软件包及配置文件
apt update	apt-get update	刷新存储库索引
apt upgrade	apt-get upgrade	升级所有可升级的软件包
apt autoremove	apt-get autoremove	自动删除不需要的包
apt full-upgrade	apt-get dist-upgrade	在升级软件包时自动处理依赖关系
apt search	apt-cache search	搜索应用程序
apt show	apt-cache show	显示安装细节

◆ apt 特有的命令

apt list 列出包含条件的包（已安装，可升级等）

apt edit-sources 编辑源列表

◆ APT包索引来自/etc/apt/sources.list文件和/etc/apt/sources.list.d目录中定义的存储库的可用包的数据库。要使用存储库中所做的最新更改来更新本地程序包索引

◆ apt命令操作（如安装和删除软件包）记录在/var/log/dpkg.log日志文件中

◆ apt 示例：

◆ 安装包：

```
apt install tree zip
```

◆ 删除包：

```
apt remove tree zip
```

说明：apt remove 中添加 --purge 选项会删除包配置文件，谨慎使用

◆ 更新包索引：

```
apt update
```

◆ 升级包：要升级系统，请首先更新软件包索引，再升级

```
apt upgrade
```

- ◆ 1、查询命令java来自于哪个rpm包
- ◆ 2、yum的配置和使用,包括yum仓库的创建
- ◆ 3、编写系统初始化脚本 reset.sh , 包括别名 , 提示符颜色 , yum仓库配置文件, 安装tree,ftp,lftp,telnet等包
- ◆ 4、在CentOS 7上编译安装 apache 2.4.25 源码包,并启动此服务

- ◆ 博客 : <http://mageedu.blog.51cto.com>
- ◆ 主页 : <http://www.magedu.com>
- ◆ QQ : 1661815153, 113228115
- ◆ QQ群 : 203585050, 279599283

祝大家学业有成

谢 谢

咨询热线 400-080-6560