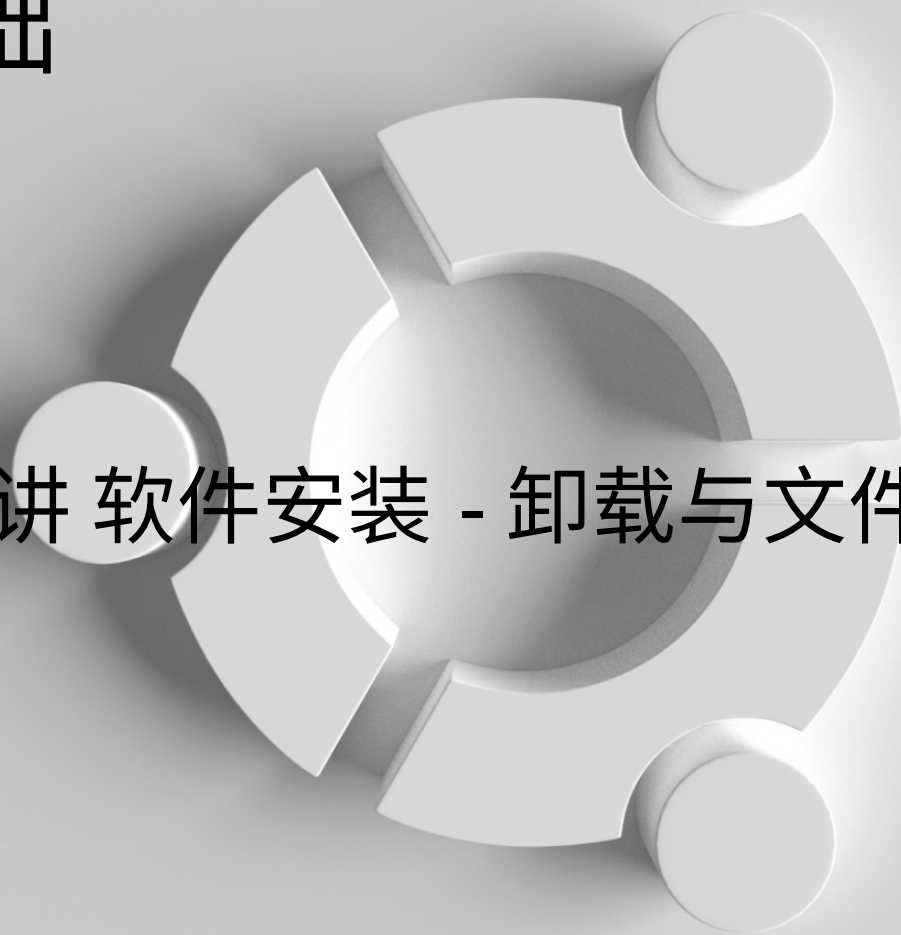


Linux 基础

第 7 讲 软件安装 - 卸载与文件编辑



Debian/Ubuntu 的软件包格式

- .deb 格式是 Debian/Ubuntu 使用的格式。
- .deb 文件是一个压缩包格式，可以解压软件包查看内容。

deb 软件包

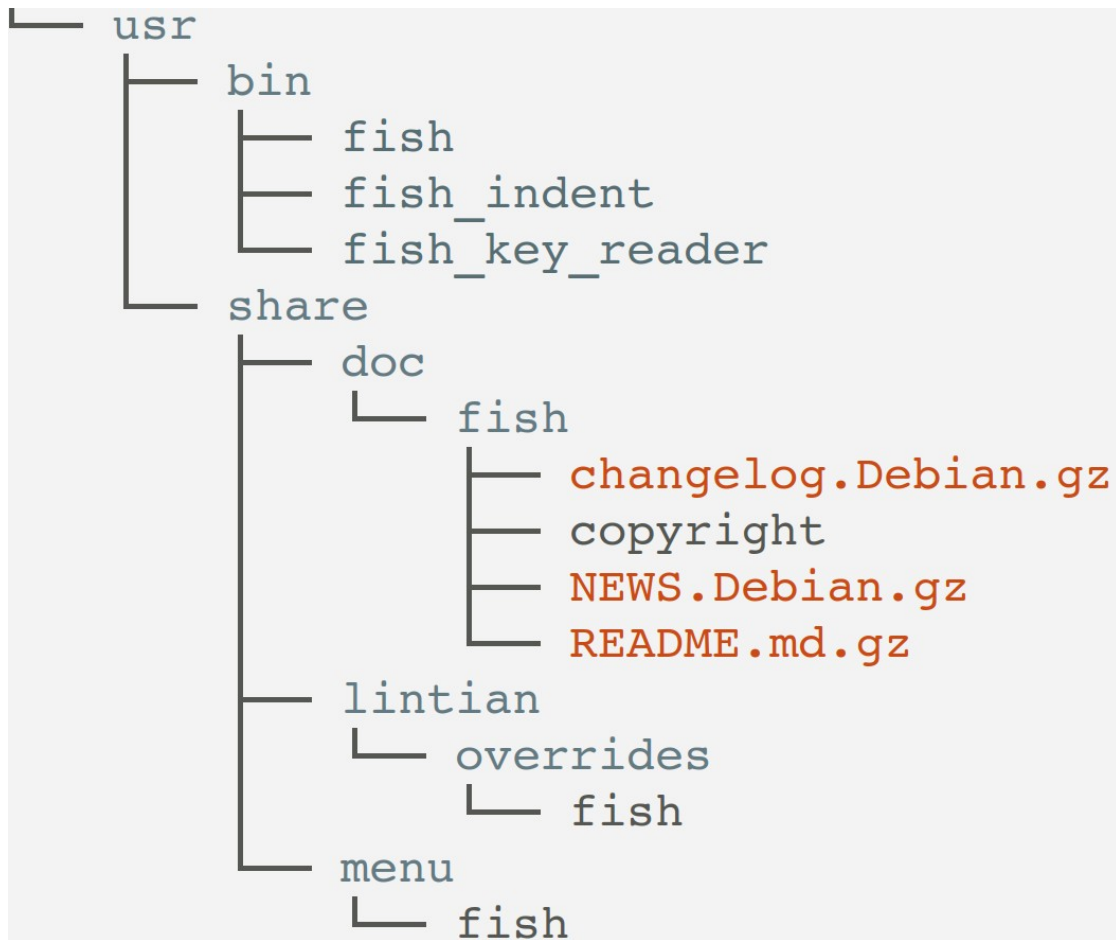
- 解压后的软件包就是已经编译好的程序，配置文件，手册等。
- 软件包中的目录结构对应于系统的目录结构，存放于系统的目录对应的位置，并记录安装信息，这就是安装过程。

解压 deb 包: dpkg

- dpkg 是 Debian 发行版提供的用于本地包管理的工具，解压软件包：

```
dpkg -x fish_3.0.2-1_amd64.deb fish/
```

目录结构示例



dpkg 的问题

- dpkg 不解决包依赖问题，所有依赖的软件包需要自己下载。
- dpkg 不会联网查询软件包信息，只能在本地管理软件包。

dpkg 安装失败示例

```
wy@wxm:$ sudo dpkg --install fish 3.0.2-1 amd64.deb
(正在读取数据库 ... 系统当前共安装有 219875 个文件和
正准备解包 fish 3.0.2-1 amd64.deb ...
正在将 fish (3.0.2-1) 解包到 (3.0.2-1) 上 ...
dpkg: 依赖关系问题使得 fish 的配置工作不能继续:
  fish 依赖于 fish-common (= 3.0.2-1); 然而:
    软件包 fish-common 尚未配置。
  fish 依赖于 libc6 (>= 2.29); 然而:
    系统中 libc6:amd64 的版本为 2.27-3ubuntu1。
  fish 依赖于 libpcre2-32-0 (>= 10.32); 然而:
    未安装软件包 libpcre2-32-0。
  fish 依赖于 libtinfo6 (>= 6); 然而:
    未安装软件包 libtinfo6。
```

dpkg 安装成功示例

```
wy@wxm:$ sudo dpkg --install cowsay_3.03+dfsg2-6_all.deb
正在选中未选择的软件包 cowsay。
(正在读取数据库 ... 系统当前共安装有 219875 个文件和目录。
正准备解包 cowsay_3.03+dfsg2-6_all.deb ...
正在解包 cowsay (3.03+dfsg2-6) ...
正在设置 cowsay (3.03+dfsg2-6) ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
```


移除软件包

- 使用 `dpkg` 移除软件包：

```
sudo dpkg --remove fish
```

```
sudo dpkg --remove fish-common
```

更好的方式

- 在 Debian/Ubuntu 上可以使用 apt 管理软件包。
- apt 根据系统记录的信息来定位软件包。
- 并根据 `/etc/apt/sources.list` 配置的软件源去获取软件包并进行安装。

包管理相关命令说明

`apt` 从软件源安装软件，卸载软件，获取更新，系统更新升级等。

`dpkg` 安装本地deb软件包，卸载软件等。

`apt-cache` 从软件源搜索软件。

`apt-get` 安装/卸载软件，系统更新等。

apt 随 Ubuntu16.04 一起发布。目的在于提供完整的更加结构化的功能，基本上整合了 `apt-get`，`apt-cache`，`apt-config` 三个命令的功能。

使用 apt 安装软件

- apt 在定位到软件包以后，会根据依赖关系自动下载所依赖的软件包。

- 安装 vim：

```
sudo apt install vim
```

获取更新并升级

- 获取软件包更新：

```
sudo apt update
```

- 升级软件包：

```
sudo apt upgrade
```

移除软件包以及清理软件包

- 移除软件包：

```
sudo apt remove [ 软件包名称 ]
```

- 自动清理不需要的软件包：

```
sudo apt autoremove
```

软件包管理总结

- 获取更新: `sudo apt update`
- 升级: `sudo apt upgrade`
- 安装: `sudo apt install [软件包名称]`
- 移除: `sudo apt remove [软件包名称]`
- 自动清理: `sudo apt autoremove`

文本编辑工具： nano

- nano 是终端模式的文本编辑工具， Debian/Ubuntu 等一些主流的发行版都自带 nano。
- nano 支持匹配搜索，代码高亮，能满足基本的编辑配置文件，代码编辑等工作。
- 相对于 vi/vim 来说， nano 比较简单易用。

文本编辑工具： `vim`

- `vim` 是 `vi` 的升级版，作为最古老的代码编辑器之一，`vim` 足够强大，并且扩展性很强，有大量的插件可用。
- 但是 `vim` 上手并不容易，并且需要记住并理解一些基本的命令，只有习惯并熟练以后，才能体会到为什么这个工具到如今依旧如此流行。

目标

- 能够熟练使用 nano 编辑文本就可以了。
- vim 短时间内不容易上手，使用 nano 主要应对需要终端操作的场景。
- 对于桌面环境，可以使用 vscode、gedit、brackets 等 GUI 工具。

nano 启动界面

GNU nano 2.9.3

/etc/nanorc

```
## Sample initialization file for GNU nano.
##
## Please note that you must have configured nano with --enable-nanorc
## for this file to be read! Also note that this file should not be in
## DOS or Mac format, and that characters specially interpreted by the
## shell should not be escaped here.
##
## To make sure an option is disabled, use "unset <option>".
##
## For the options that take parameters, the default value is given.
## Other options are unset by default.
##
## Quotes inside string parameters don't have to be escaped with
## backslashes. The last double quote in the string will be treated as
## its end. For example, for the "brackets" option, "'>]]" will mat$
```

[文件 "/etc/nanorc" 不可写入]

^G 求助
^X 离开

^O 写入
^R 读档

^W 搜索
^ 替换

^K 剪切文字
^U 还原剪切

^J 对齐
^T 拼写检查

nano 基本使用

- 打开文件: `nano [文件名]`
- 保存文件: `Ctrl+S`
- 另存为: `Ctrl+O` , 这时候底部会提示输入文件名, 默认为当前文件名。
- 退出: `Ctrl+X`

nano 配置文件

- 配置文件位置： `/etc/nanorc`
- 配置文件选项一般使用 `set **`，`#` 开头表示注释，比如 `set linenumbers` 表示显示行号。
- 每个选项前使用注释说明了其作用。

nano 修改配置后截图

GNU nano 2.9.3

c/lsp/ch04/mre5.c

```
58         return matchchar(regex[1], regex+1, text, uplow);
59     }
60
61     if (regex[0] == '$' && regex[1] == '\\0')
62         return *text == '\\0';
63
64     if (regex[1] == '*') {
65         if (regex[0] == '.') {
66             while(*text != '\\0') {
67                 if (matchreg(regex+2, text, uplow))
68                     return 1;
69                 text++;
70             }
71         } else {
72             char c = regex[0];
73             char *tbuf = text;
74             while(*tbuf != '\\0') {
```

[行 74/218 (33%), 列 1/35 (2%), 字符 1654/4845 (34%)]

vim 入门

- Ubuntu 系统上可能只提供了 `vi`，这是早期的版本，可以使用 `apt` 安装：

```
sudo apt install vim
```

- 打开文件： `vim [文件名]`。
- 如果文件不存在则会在保存时创建。

vim 基本使用

- vim 的使用和其他编辑器不同， vim 使用了 3 种不同的模式，为了区分，给它们命名为
 - 命令模式
 - 输入模式（底部显示 --INSERT-- 或 -- 插入 --）
 - 底行模式（底部显示 :）

vim 基本使用：开始编辑

- 启动 vim 默认是命令模式。这时候不能直接编辑文件。
- 输入 `i`, `a`, `o` 都可以切换到输入模式，此时底部显示：
--INSERT-- 或 -- 插入 --
- `i` 是在当前位置输入，`a` 会把光标向右移动一个，`o` 是在下一行输入。

vim 基本使用：保存并退出

- 编辑完文件之后，在 vim 中保存退出需要以下操作：
 - 按 ESC 切换回 ‘命令模式’
 - 输入 : 切换到 ‘底行模式’
 - 输入 w 写入文件，输入 q 退出，可以连用，输入 wq 保存并退出

vim 基本使用：不保存退出

- 如果想要放弃本次修改，可以按照以下步骤：
 - 按 ESC 回到 ‘命令模式’
 - 输入 : 切换到 ‘底行模式’
 - 输入 q! 不保存退出

Vim 截图

```
" Press ? for help

.. (up a dir)
/home/wy/c/lsp/
> bin/
> ch01/
> ch02/
> ch03/
> ch04/
> ch05/
> ch06/
▼ ch07/
  alarmchild.c
  daeserv.c
  fork.c
  killall.c
  manyfork.c
  sigchld.c
  wait.c
> ch08/
> ch09/
> ch010/
> ch011/
> ch012/
  morestat.c
  pipestat.c
/home/wy/c/lsp
"ch07/alarmchild.c" 43L, 864C

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6
7 void handle_sig(int sig) {
8     int st;
9     pid_t pid;
10
11     while((pid=waitpid(0, &st, WNOHANG)) > 0) {
12         if (WIFEXITED(st)) {
13             printf("child %d exited\n", pid);
14         } else if (WIFSIGNALED(st)) {
15             printf("child %d terminate by signal\n", pid);
16         }
17         exit(0);
18     }
19 }
20
21 int main(int argc, char *argv[]) {
22
23     pid_t pid = fork();
24
25     if (pid < 0) {
26         perror("fork");
```

Vim 截图

```

> ch02/
> ch03/
> ch04/
> ch05/
> ch06/
▼ ch07/
    alarmchild.c
    daeserv.c
    fork.c
    killall.c
    manyfork.c
    sigchld.c
    wait.c
> ch08/
▼ ch09/
    co.c
    co2.c
    cop.c
    eioserv.c
    eioserv2.c
    eioserv3.c
    epoll.c
    epoll_fork.c
    epoll_sem.c
    ioblock.c
    iotype.c
103 }
104
105 int try_accpet_lock() {
106     struct sembuf mf;
107     mf.sem_num = 0;
108     mf.sem_op = -1;
109     mf.sem_flg = IPC_NOWAIT | SEM_UNDO;
110     return semop(_save_semid, &mf, 1);
111 }
112
113 int release_accpet_lock() {
114     struct sembuf mf;
115     mf.sem_num = 0;
ch09/eioserv3.c 111,1 22%
5 #include <sys/wait.h>
6
7 void handle_sig(int sig) {
8     int st;
9     pid_t pid;
10
11     while((pid=waitpid(0, &st, WNOHANG)) > 0) {
12         if (WIFEXITED(st)) {
13             printf("child %d exited\n", pid);
14         } else if (WIFSIGNALED(st)) {
15             printf("child %d terminate by signal\n", pid);
16         }
/home/wy/c/lsp ch07/alrmchild.c 9,14 12%
```