

第五节 打包解包及其关键技术

詹文翰

zhanwenhan@163.com

课程安排

1

软件工程与软件建模

2

敏捷开发与项目协作

3

开发环境、规范与测试

4

需求、设计与程序框架

5

打包解包及其关键技术

6

压缩解压及其关键技术

7

加密解密与系统集成

8

软件发布与项目收尾

实验难度分级和评分标准

基本要求

各小组“独立”实现一款数据备份软件（对应基础分总分**40分**）：

数据备份：将目录树中的文件数据保存到指定位置

数据还原：将目录树中的文件数据恢复到指定位置

扩展要求

各项目组根据自身情况自行选择扩展要求（对应扩展分总分）。

文件类型支持（10分）：支持特定文件系统的特殊文件（管道/软链接/硬链接等）

元数据支持（10分）：支持特定文件系统的文件元数据（属主/时间/权限等）

自定义备份（各5分）：允许用户筛选需要备份的文件（路径/类型/名字/时间/尺寸）

打包解包（10分）：将所有备份文件拼接为一个文件保存

压缩解压（10分）：通过文件压缩节省备份文件的存储空间

加密解密（10分）：由用户指定密码，将所有备份文件均加密保存（库实现7分）

图形界面（10分）：实现友好易用的GUI界面

定时备份（10分）：允许用户进行设置，进行周期性定时备份和数据淘汰

实时备份（15分）：自动感知用户文件变化，进行自动备份

网络备份（30分）：将数据备份软件从单机模式扩展为网盘模式（10分），还涉及到的功能包括：用户管理（5分）、元数据管理（5分）、传输加密（5分）、增量备份（5分）等。

其它功能：视功能难度讨论加分。

实验难度分级和评分标准

基本要求

各小组“独立”实现一款数据备份软件（对应基础分总分**40分**）：

数据备份：将目录树中的文件数据保存到指定位置

数据还原：将目录树中的文件数据恢复到指定位置

扩展要求

各项目组根据自身情况自行选择扩展要求（对应扩展分总分）。

文件类型支持（10分）：支持特定文件系统的特殊文件（管道/软链接/硬链接等）

元数据支持（10分）：支持特定文件系统的文件元数据（属主/时间/权限等）

自定义备份（各5分）：允许用户筛选需要备份的文件（路径/类型/名字/时间/尺寸）

打包解包（10分）：将所有备份文件拼接为一个大文件保存

压缩解压（10分）：通过文件压缩节省备份文件的存储空间

加密解密（10分）：由用户指定密码，将所有备份文件均加密保存（库实现7分）

图形界面（10分）：实现友好易用的GUI界面

定时备份（10分）：允许用户进行设置，进行周期性定时备份和数据淘汰

实时备份（15分）：自动感知用户文件变化，进行自动备份

网络备份（30分）：将数据备份软件从单机模式扩展为网盘模式（10分），还涉及到的功能包括：用户管理（5分）、元数据管理（5分）、传输加密（5分）、增量备份（5分）等。

其它功能：视功能难度讨论加分。







Unix文件系统

Unix系统中的文件类型：

- 普通文件
- 目录文件
- 字符设备文件
- 块设备文件
- 套接字文件
- 管道文件
- 链接文件

Unix文件系统

Unix系统中的文件类型：

- 普通文件 
- 目录文件 
- 字符设备文件 
- 块设备文件 
- 套接字文件
- 管道文件 
- 链接文件 

Unix文件系统

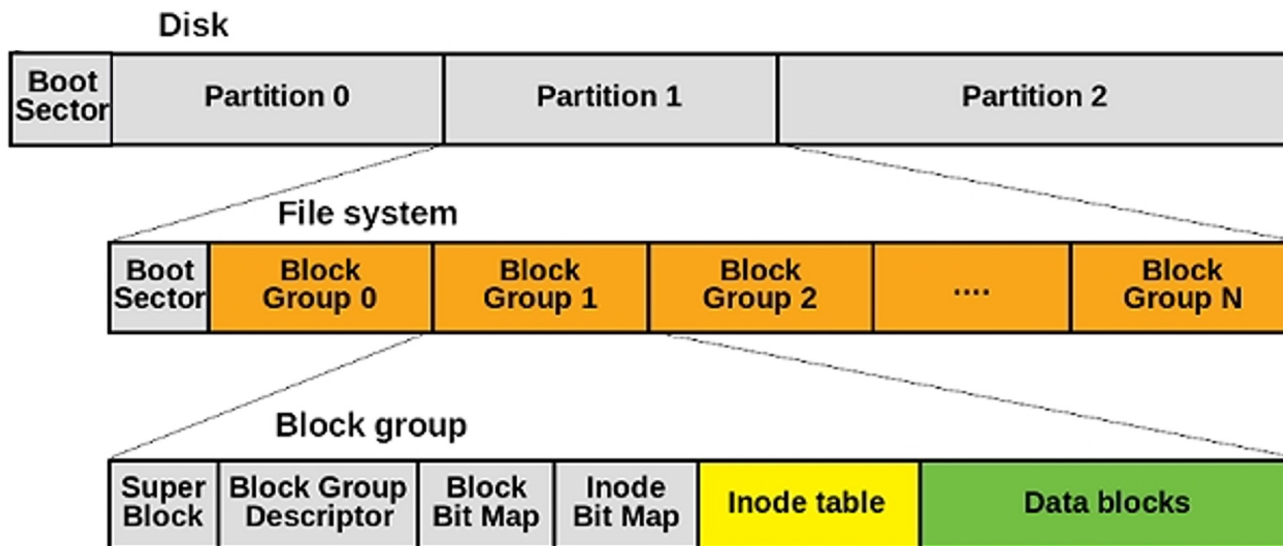
如何保证备份软件的正确性：

- 1、文件类型
- 2、文件路径
- 3、文件数据
- 4、文件元数据（属主/权限/时间）
- 5、文件链接方式（软/硬）

UNIX 文件系统原理

Unix文件系统

典型 Unix 磁盘布局 (ext2 文件系统)

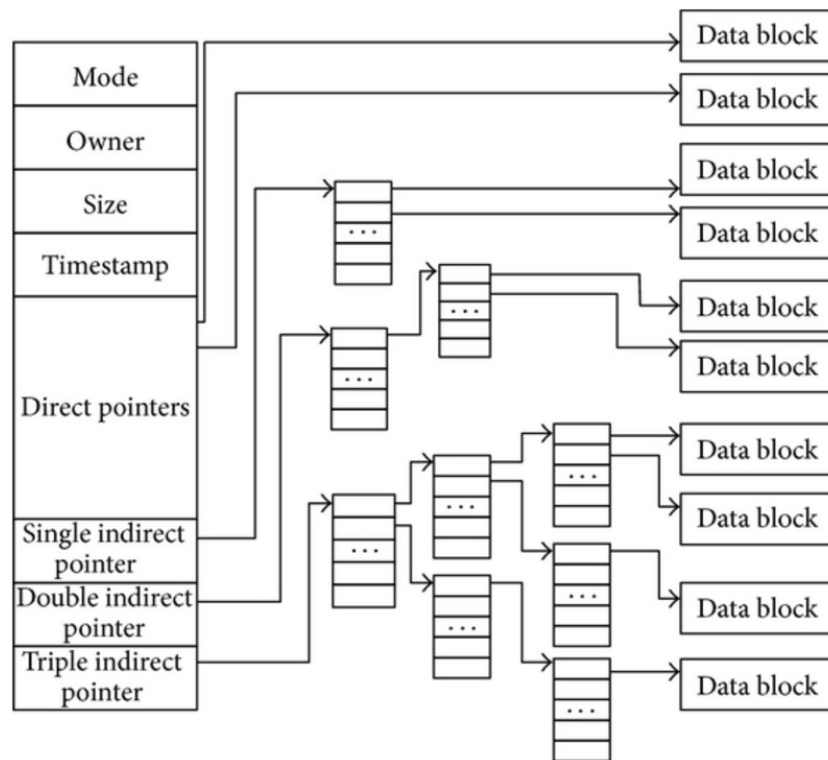


ext2 官方文档: <https://www.nongnu.org/ext2-doc/ext2.html>

Unix文件系统

普通文件的实现

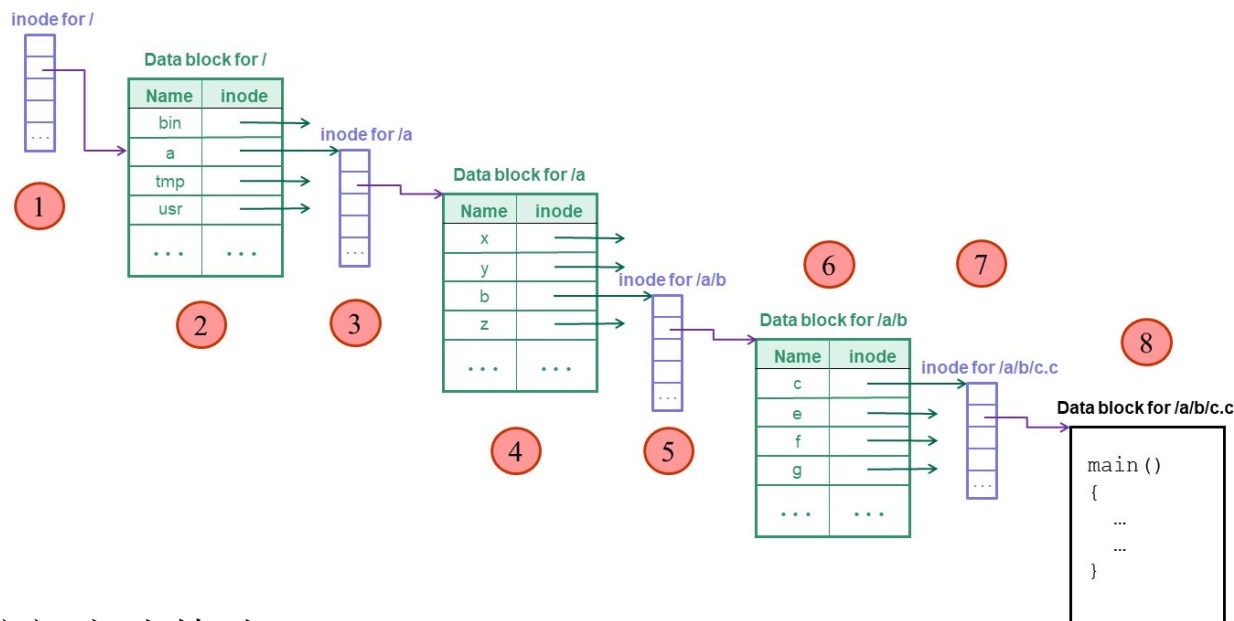
- UNIX 普通文件采用索引结构
- 索引数据结构——inode
- inode存储了文件的所有元数据
- 特殊的索引方式：
 - 直接索引 10个
 - 一级间接索引 1个
 - 二级间接索引 1个
 - 三级间接索引 1个



Unix文件系统

目录文件的实现

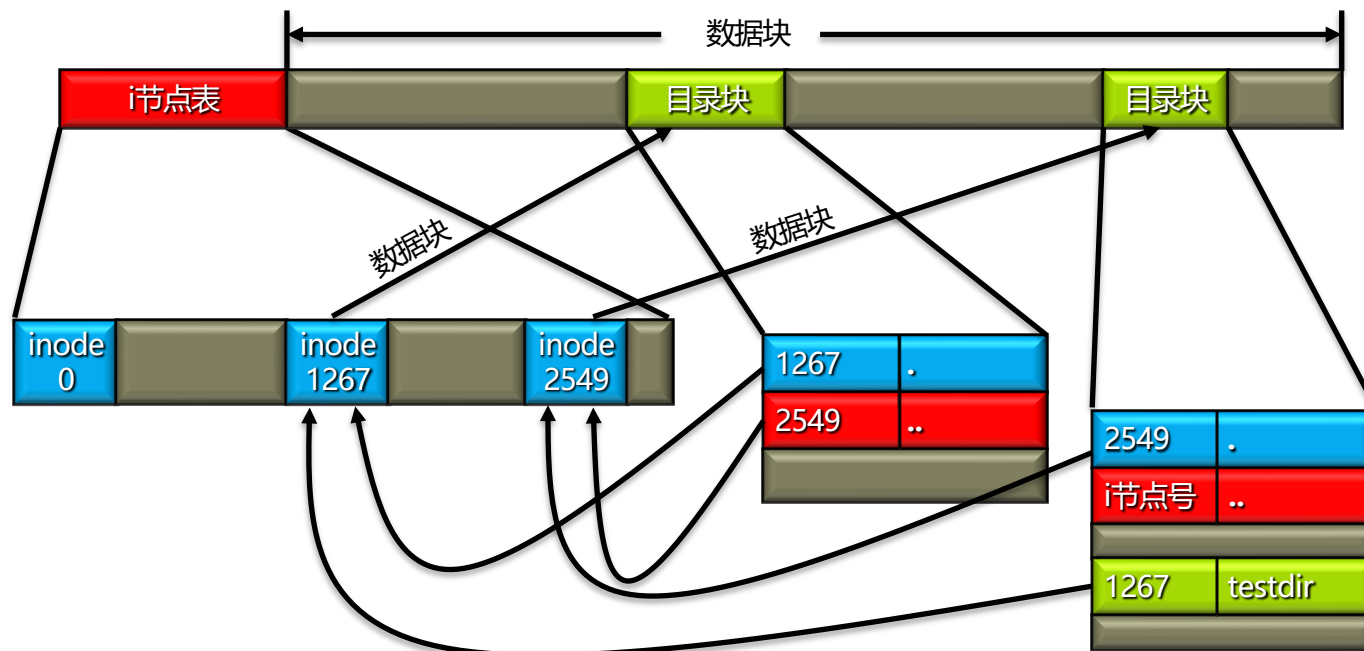
- 目录也是文件！
 - 有特殊结构的文件
 - 以访问 /a/b/c.c 为例



- 访问方式特殊
 - 自动维护目录结构 (.`.` 和 `..`)
 - 保护, 防止误操作

Unix文件系统

例：指向父子目录中目录项之间的关系。




Unix文件系统

链接数（硬链接）

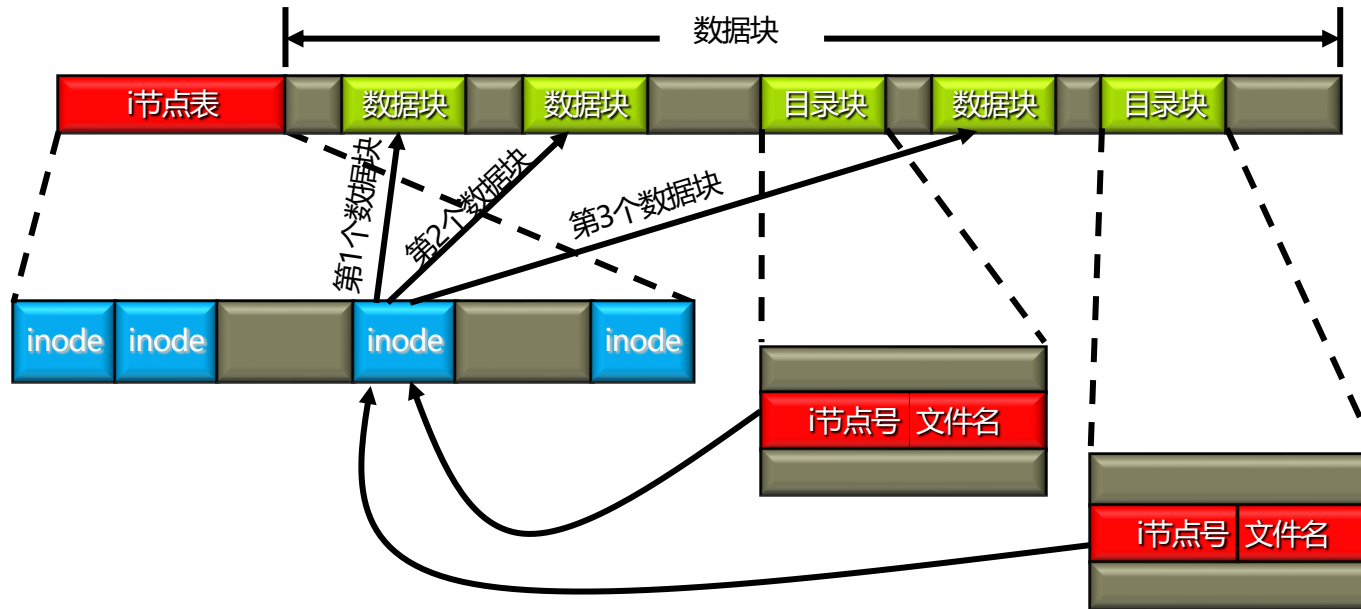
- 指向同一个 inode 的目录项的数量
- 硬链接实现的是 inode 共享
- ./ 和 ../ 会影响到链接数

```
# root @ zhanwenhan-lab in ~ [13:59:57]
$ ls -li
total 52
918153 -rw-r--r-- 1 root root 2097 Mar 22 14:21 f1
917859 -rw-r--r-- 1 root root 16153 Mar 29 11:54 f2
918214 -rw-r--r-- 2 root root 16153 Mar 22 14:21 f3
918214 -rw-r--r-- 2 root root 16153 Mar 22 14:21 hd
```



Unix文件系统

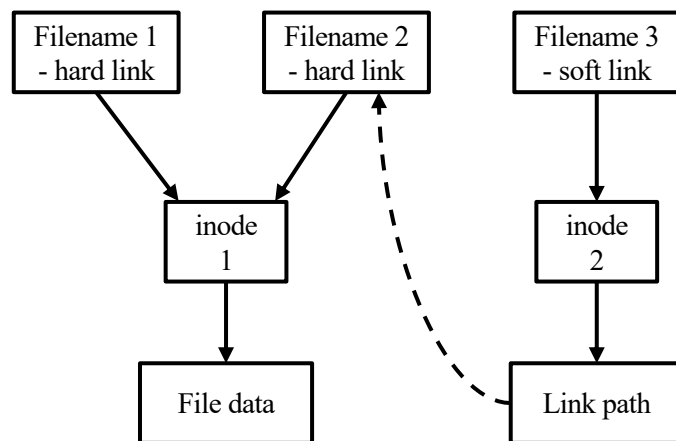
例：指向同一个 inode 的两个目录项。



Unix文件系统

链接文件（软链接）

- 软链接本身就是一个文件，有自己的 inode 和数据块。
- 软链接的数据块中存储了另一个文件或目录的路径，而不是文件或目录的实际数据。



软链接 VS 硬链接

- UNIX 不支持用户为目录创建硬链接，但软链接支持（注意风险）
- 软链接支持跨文件系统进行链接，硬链接不支持
- 硬链接被移动或删除后，其它硬链接仍然有效，但软链接将失效
- 硬链接比软链接访问速度更快，更节省空间（可忽略不计）

Unix文件系统

项目涉及到的系统API（部分）：

- open
- close
- read
- write
- stat
- chown
- chmod
- utime
- opendir
- closedir
- readdir
- rewinddir
- mkdir
- mkfifo
- unlink
- link
- symlink
-

Unix文件系统

项目涉及到的系统API（部分）：

open
close
read
write
stat
chown
chmod
utime
opendir
closedir
readdir
rewinddir
mkdir
mkfifo
unlink
link
symlink
.....

man手册

文件遍历&打包解包

打包解包原理
文件遍历方法

打包解包

打包：将多个文件/目录整合为一个大文件的过程。

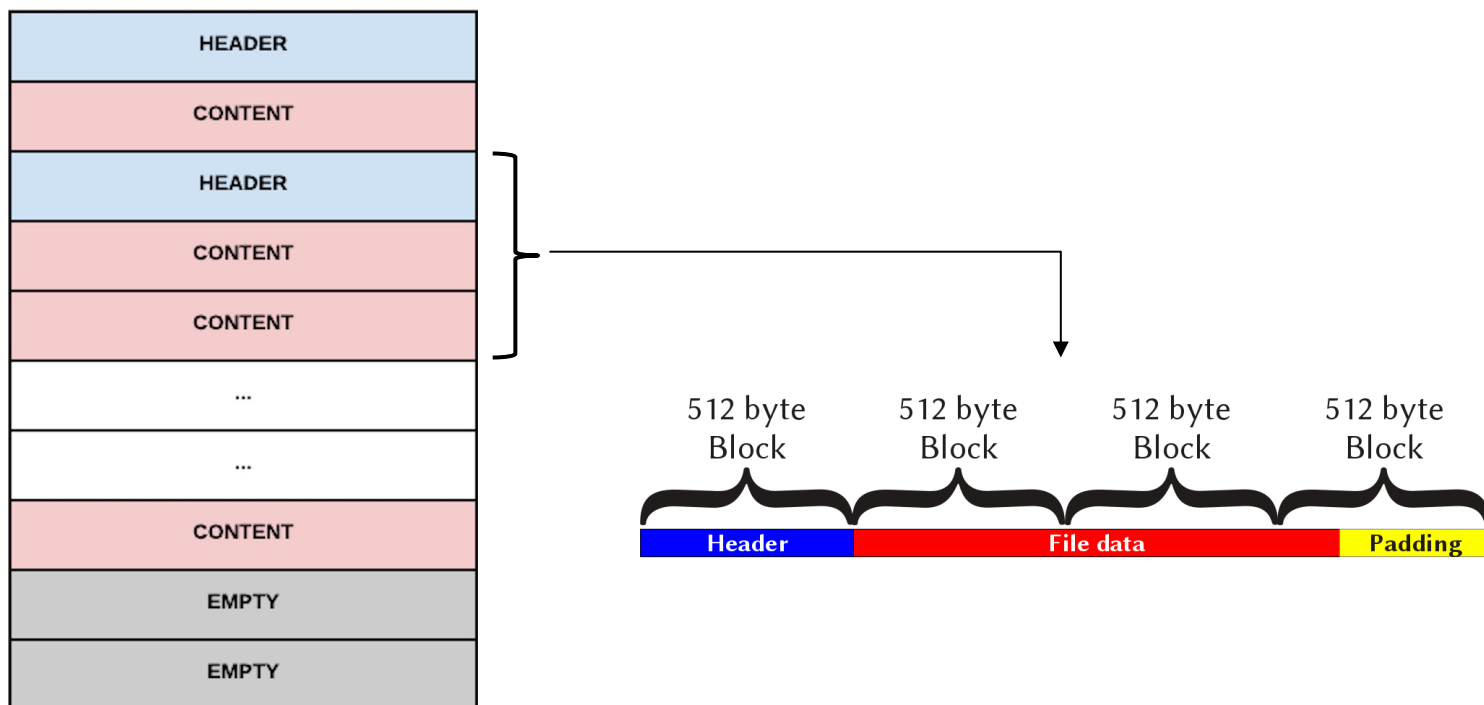
解包：将多个文件/目录从一个大文件中按原样恢复的过程。

大部分压缩软件均支持同时对文件打包。最经典的文件打包程序：**Tar**。

Tar打包原理

Tar是Unix和类Unix系统上的归档打包工具，可以将多个文件合并为一个文件，打包后的文件名亦为“tar”。目前，tar文件格式已经成为POSIX标准。

Tar归档文件由一系列文件对象通过线性排列的方式组合而成。每个文件对象都包含一个或者若干个512字节的记录块，并以一个头记录开头。文件数据按原样写入，但其长度舍入为512字节的倍数。



```

1 union Record
2 {
3     union
4     {
5         // Pre-POSIX.1-1988 format
6         struct
7         {
8             char name[100];        // file name
9             char mode[8];          // permissions
10            char uid[8];            // user id (octal)
11            char gid[8];            // group id (octal)
12            char size[12];          // size (octal)
13            char mtime[12];         // modification time (octal)
14            char check[8];          // sum of unsigned characters in block (octal)
15            char link;              // link indicator
16            char link_name[100];    // name of linked file
17        };
18
19        // UStar format (POSIX IEEE P1003.1)
20        struct
21        {
22            char old[156];          // first 156 octets of Pre-POSIX.1-1988 format
23            char type;              // file type
24            char also_link_name[100]; // name of linked file
25            char ustar[8];          // ustar\000
26            char owner[32];         // user name (string)
27            char group[32];         // group name (string)
28            char major[8];          // device major number
29            char minor[8];          // device minor number
30            char prefix[155];
31        };
32    };
33
34    char block[512]; // raw memory (padded to 1 block)
35 };

```

```

#define REGULAR      0
#define NORMAL      '0'
#define HARDLINK     '1'
#define SYMLINK      '2'
#define CHAR         '3'
#define BLOCK        '4'
#define DIRECTORY    '5'
#define FIFO         '6'

```

实现思路

- 将目录遍历算法和打包算法分开实现和执行
 - 目录遍历算法：
 - 输入：某个目录
 - 输出：目录下的所有文件（全部类型）列表
 - 打包算法：
 - 输入：文件列表
 - 输出：打包文件
 - 解包算法：
 - 输入：打包文件，解包目录
 - 输出：解包数据

实现思路

- 将目录遍历算法和打包算法分开实现的优点和缺点
 - 优点：
 - 便于逻辑解耦
 - 易于实现
 - 便于实现复杂的文件过滤逻辑
 - 缺点：
 - 时间效率和空间效率下降
 - 打包过程中的数据一致性可能会出现问题
 - 目录遍历算法输出文件列表的顺序对解包算法很重要
 - 结论：先序遍历

实现思路

- 实现目录遍历算法应该注意哪些问题
 - 硬连接如何处理？
 - 直接复制 or 保持链接关系？
 - 软连接如何处理？
 - 直接复制 or 仅复制链接？
 - 目录软链接可能导致遍历死循环（lstat）
 - 相对路径 or 绝对路径？
 - 建议将程序的当前路径调整为对应目录再开始遍历
 - 如何进行数据过滤？
 - 在遍历过程中对数据进行筛选
 - 仅保存和继续遍历通过筛选的数据

实现思路

- 实现打包算法应该注意哪些问题
 - 输入安全性检查
 - 源路径存在性，可访问性
 - 目的路径存在性，可访问性
 - 程序当前路径要和遍历时的当前路径一致
 - 文件类型的支持
 - 普通文件
 - 目录文件
 - 软链接文件
 - 管道文件
 - 设备文件
 - 套接字
 - 文件名超长处理
 - 使用多个header块（如何标识？）

实现思路

- 实现解包算法应该注意哪些问题
 - 输入安全性检查
 - 源文件存在性，可访问性
 - 源文件头校验
 - 目的路径存在性，可访问性
 - 目的路径目录为空！
 - 文件恢复
 - 目录文件
 - 软链接文件
 - 管道文件
 - 设备文件
 - 套接字
 - 普通文件（包括数据恢复，注意文件大小）
 - 元数据的恢复
- 文件名超长处理
 - 使用多个header块（如何确认）

参考文件遍历&打包解包代码

实验环节

完善数据备份软件的：
框架代码

完成数据备份软件的：
目录遍历部分
文件过滤部分
打包解包部分
(推荐) 完成单元测试