《软件开发综合实验》

Comprehensive Experiment on Software Development

肖逸飞

xyf1989@uestc.edu.cn

课程背景

1. 本课程属于计算机专业的**实践类必修**课程，面向高年级学生开设，旨在培养学生对所学知识的**综合**运用能力。
2. 课程将围绕一个**模拟实战**的软件项目，让学生以**团队**为单位，完成从需求分析、系统设计、程序编码到集成测试的**整个软件生命周期**。
3. 通过对软件工程的全流程实践，提升学生的系统建模与分析能力、程序设计与实现能力、团队协作及领导能力，使学生具备足够的面向市场的**工程能力**和**职业素养**。

前置课程

必备：

《程序设计》（C与C++）
《数据结构与算法》
《软件工程》
《软件开发环境》

推荐：

《基于操作系统编程》（Unix）
《UML统一建模语言》
《软件配置管理》（Git）
《操作系统》
《计算机网络》

实验内容及要求

1. 设计并实现一款**数据备份软件**，以项目组形式推进，每组最多**三人**。

2. 基于软件工程方法学进行项目推进，经历从需求分析、系统设计、编码实现、软件测试的整个**软件生命周期**。

3. 实验最终成果包括一款基本可用的**软件**及其对应**文档**。

4. 软件应具有**用户界面**，同时具备**正确性、易用性、健壮性**。

5. 软件文档应包括：**需求分析说明书、系统设计文档、软件测试报告**，同时具备**规范性、一致性、可读性**。

6. 采用现代化**软件开发工具**辅助项目开发，包括但不限于：项目管理工具，UML建模工具，集成开发环境，版本控制工具，软件测试工具。

实验难度分级和评分标准

基本要求

基本要求为所有小组均需实现的项目需求（对应**基础分60分**）。

数据备份：将目录树中的“各种类型”的文件保存到指定位置

数据还原：将目录树中的“各种类型”的文件按“原样”恢复

界面友好：实现易用的GUI界面

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**）。

服务器备份（5-10分）：搭建服务器，允许用户将指定文件或目录备份到服务器。

压缩解压（5-10分）：通过文件压缩节省备份文件的存储空间

打包解包（5-10分）：将所有备份文件拼接为一个大文件保存

自定义备份（5-10分）：允许用户指定仅备份目录树中的部分文件

加密备份（5-10分）：由用户指定密码，将所有备份文件均加密保存

增量备份（5-10分）：每次备份操作仅备份变化文件，以减少负担（主要是网络）

实时备份（5-10分）：自动感知用户文件变化，进行自动备份

网盘备份（10-20分）：将数据备份软件从单机模式扩展为C/S模式（网盘）

其它功能：视功能难度讨论加分。

开发环境

操作系统选择：Linux/Mac/Windows

开发语言选择：C++/Java/Go/Dart：**若选择脚本语言，小组基础分扣10分。**

库的使用：对所有扩展功能，如果使用第三方库/代码实现，只能获得**最低扩展分。**

实验评分标准

小组起评分 = 小组基础分+小组扩展分　　*（但总分不超过100分）*

小组得分 = 小组起评分%*项目完成情况

项目完成情况 = 答辩与演示*25% +
源码质量*15% +
需求分析说明书*15% +
系统设计文档*20% +
软件测试文档*15% +
项目总结*10%

组长得分 = 小组得分
组员得分 = 小组得分-5

提交资料

以小组为单位：

1. 需求分析说明书
2. 系统设计文档
3. 软件测试报告
4. 项目总结
5. 项目答辩PPT
6. 源代码+可执行程序（包含程序构建脚本，推荐dockerfile）
7. 项目演示视频（2分钟以内）

提交方式

1-6：提交到实验平台（pdf格式，打包提交）；
6-7：打包提交到老师邮箱。

截止日期

最后一次实验课（第八次课）。

课程安排

1. 项目概述与技术基础
2. 需求分析
3. 系统设计
4. 项目实现及软件配置
5. 需求变更及敏捷开发
6. 软件测试
7. 知识拓展与项目完善
8. 项目答辩及演示

第一节 技术基础

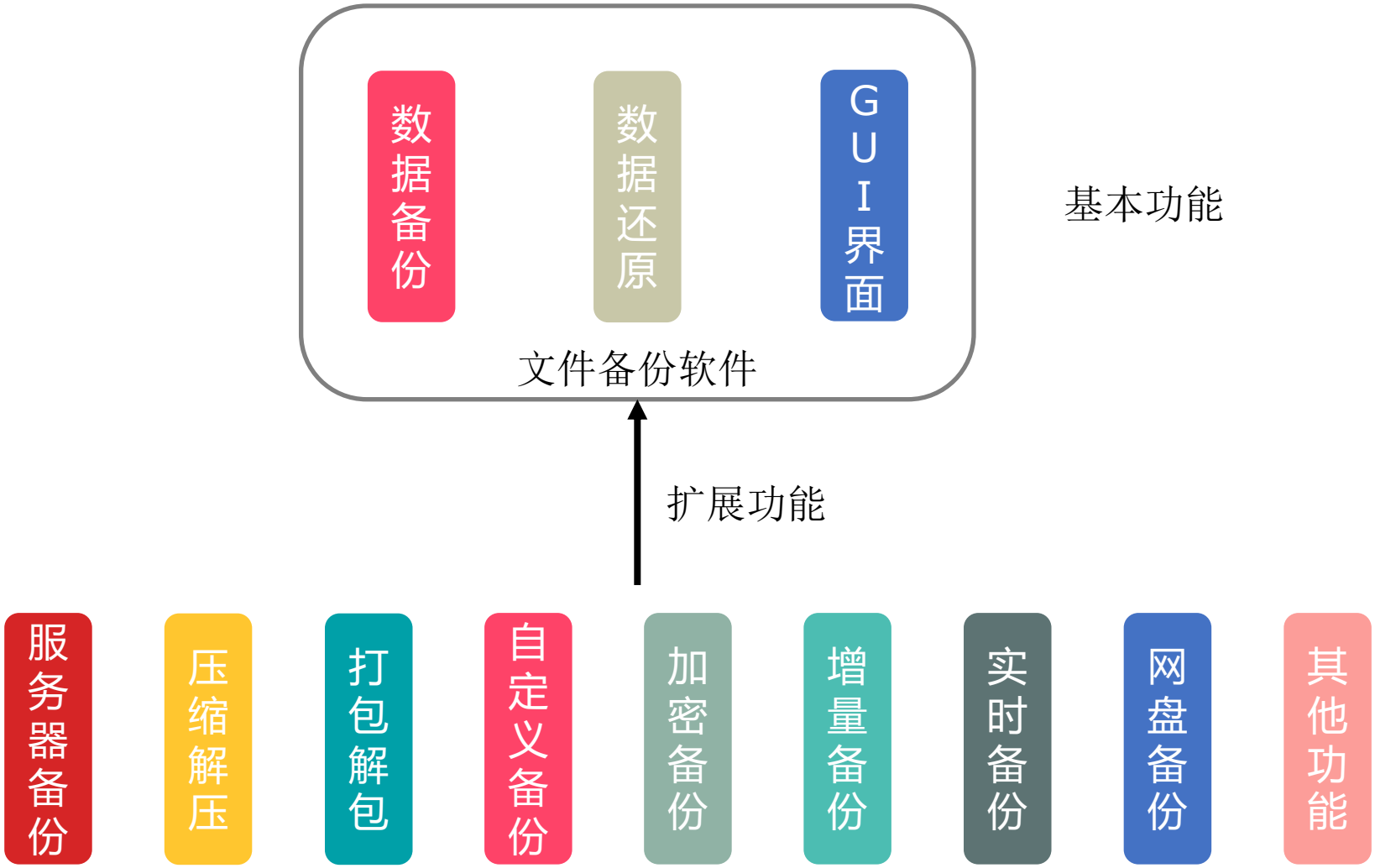
肖逸飞

xyf1989@uestc.edu.cn

基本需求：数据备份

项目概述

基于软件工程方法学，实现一个文件备份软件：

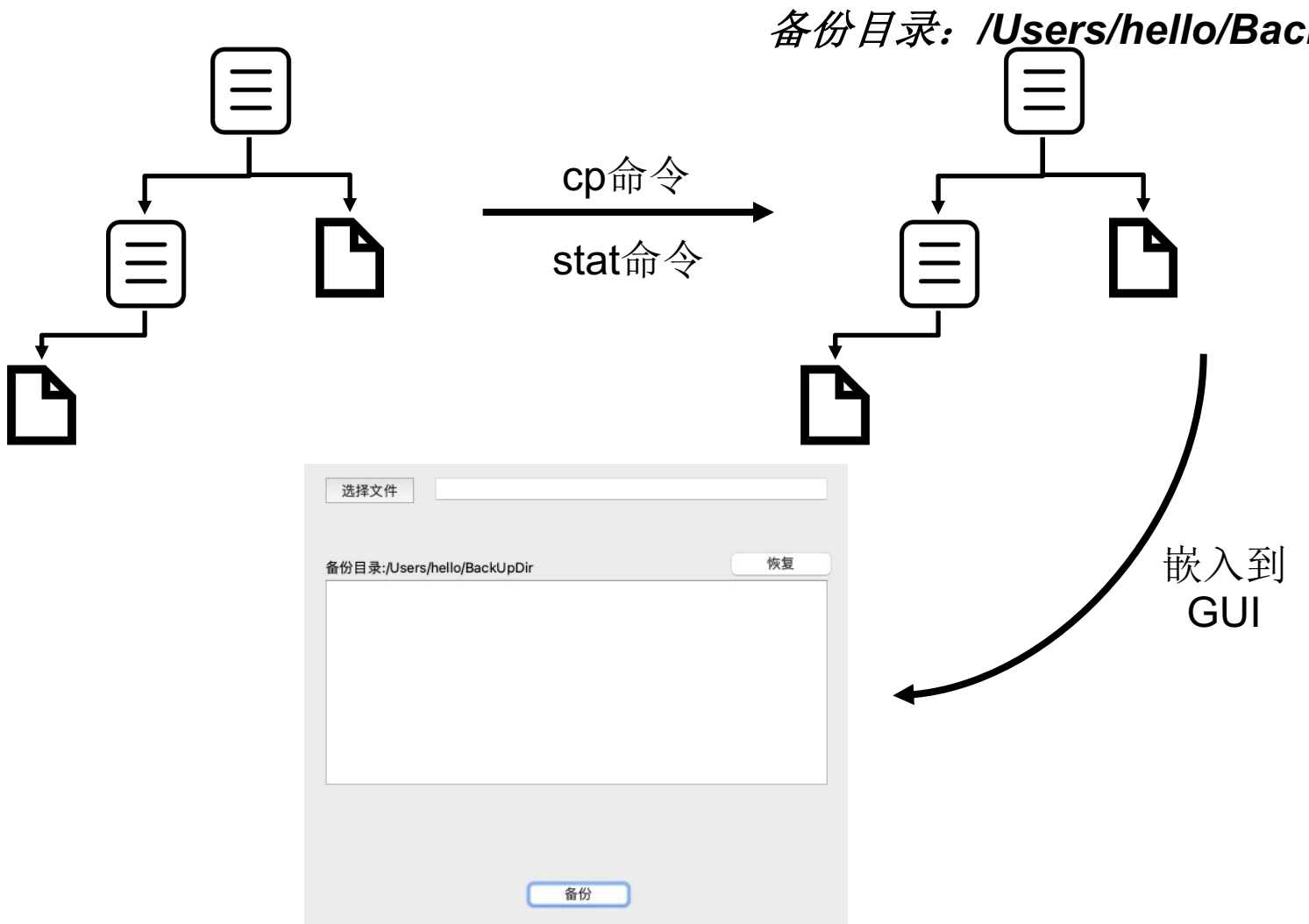


数据备份

1

项目概述与技术基础

实现包含一个数据备份、数据还原、GUI界面的文件备份软件，能够对“各种类型”的文件进行备份。



数据备份

1. GUI编程（Qt、JAVA GUI、 Python GUI、Flutter等等）
2. Unix系统API（cp、stat命令）
3. 文件系统（Unix、Windows）

Qt GUI演示

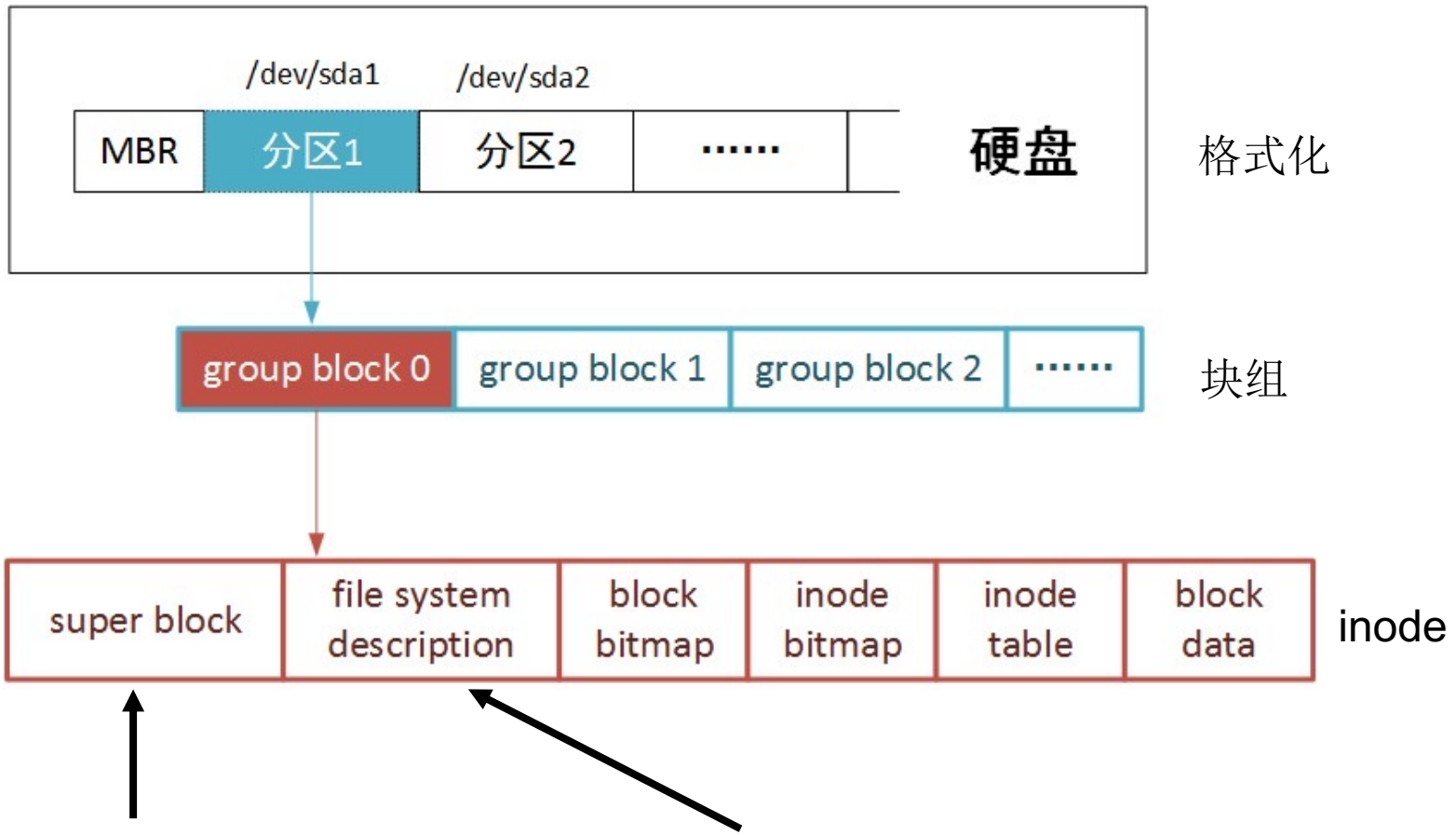
数据备份

1. GUI编程（Qt、JAVA GUI、 Python GUI、Flutter等等）
2. Unix系统API（cp、stat命令）
3. 文件系统（Unix、Windows）

各种类型文件？

Unix文件系统

经典的ext2文件系统结构图

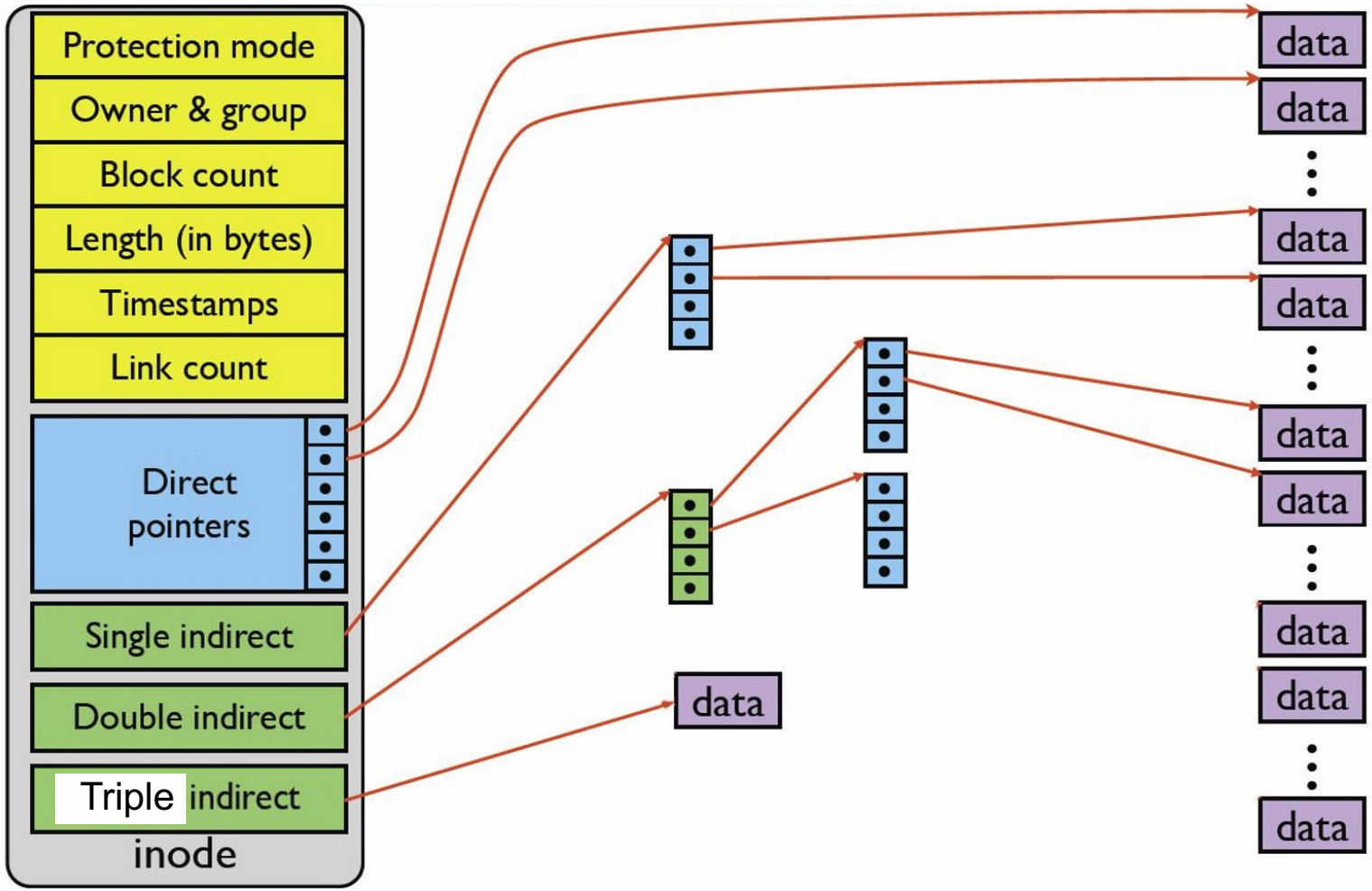


Unix文件系统

ext2文件系统中的普通文件

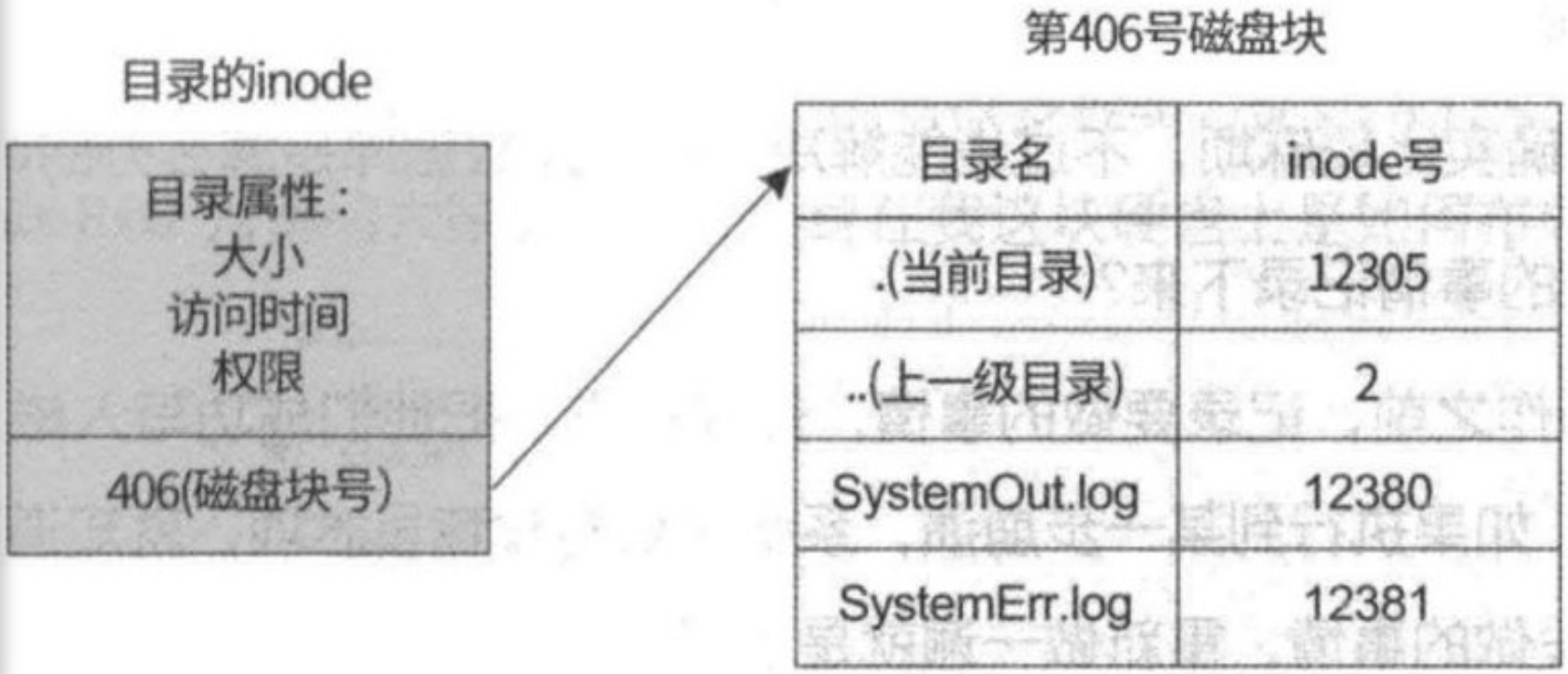
1

项目概述与技术基础



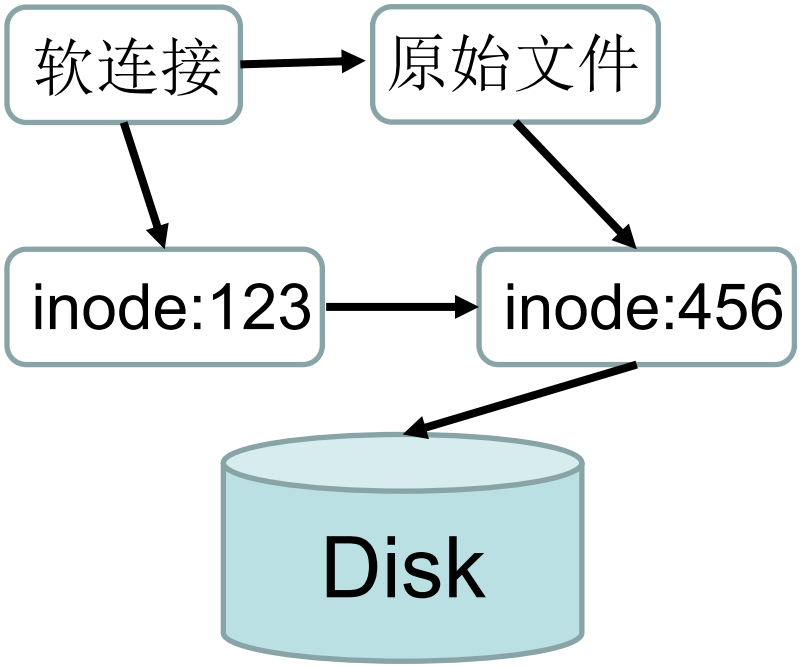
Unix文件系统

ext2文件系统中的目录（也是一种文件），它的内容记录了子文件、子目录的名称和inode ID。

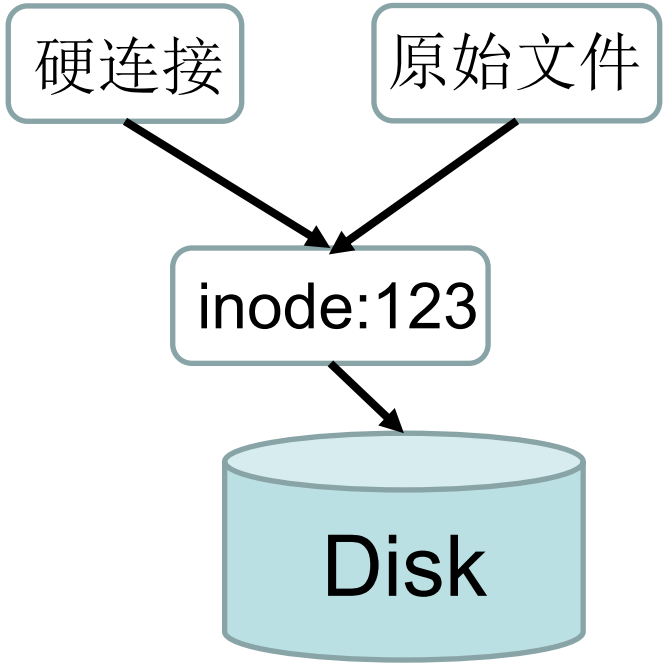


Unix文件系统

文件链接（软连接、硬链接）



有自己的inode，文件内容记录了指向文件的路径。



指向同一个inode，inode的引用计数+1

Unix文件系统

Unix系统中的文件类型：

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

Unix文件系统

1

项目概述与技术基础

如何将目录树中的“各种类型”的文件按“原样”恢复：

1、文件类型
2、对链接文件的处理（软/硬）
3、文件属性（属主/权限/时间）

Unix文件系统

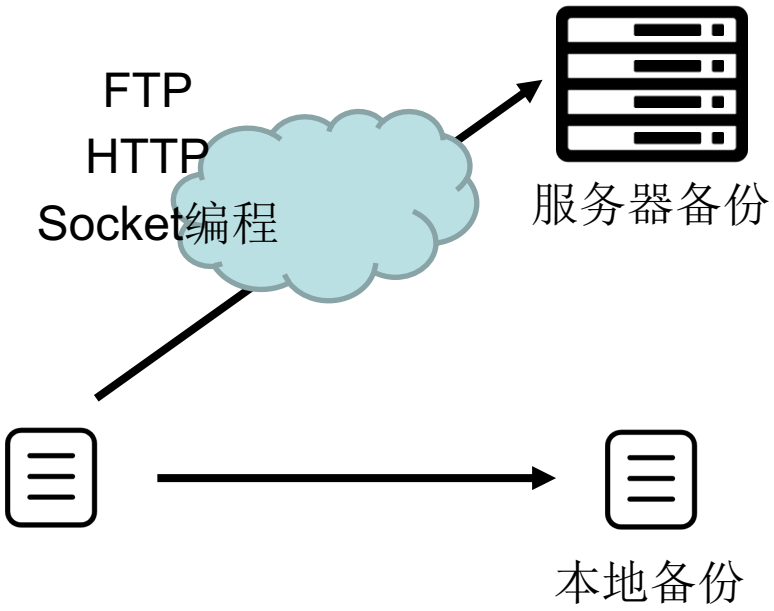
项目涉及到的系统API（部分）：

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

man手册

服务器备份

在基础功能之上，提供服务器备份功能，采用的方式主要有FTP、HTTP、自定义Socket。

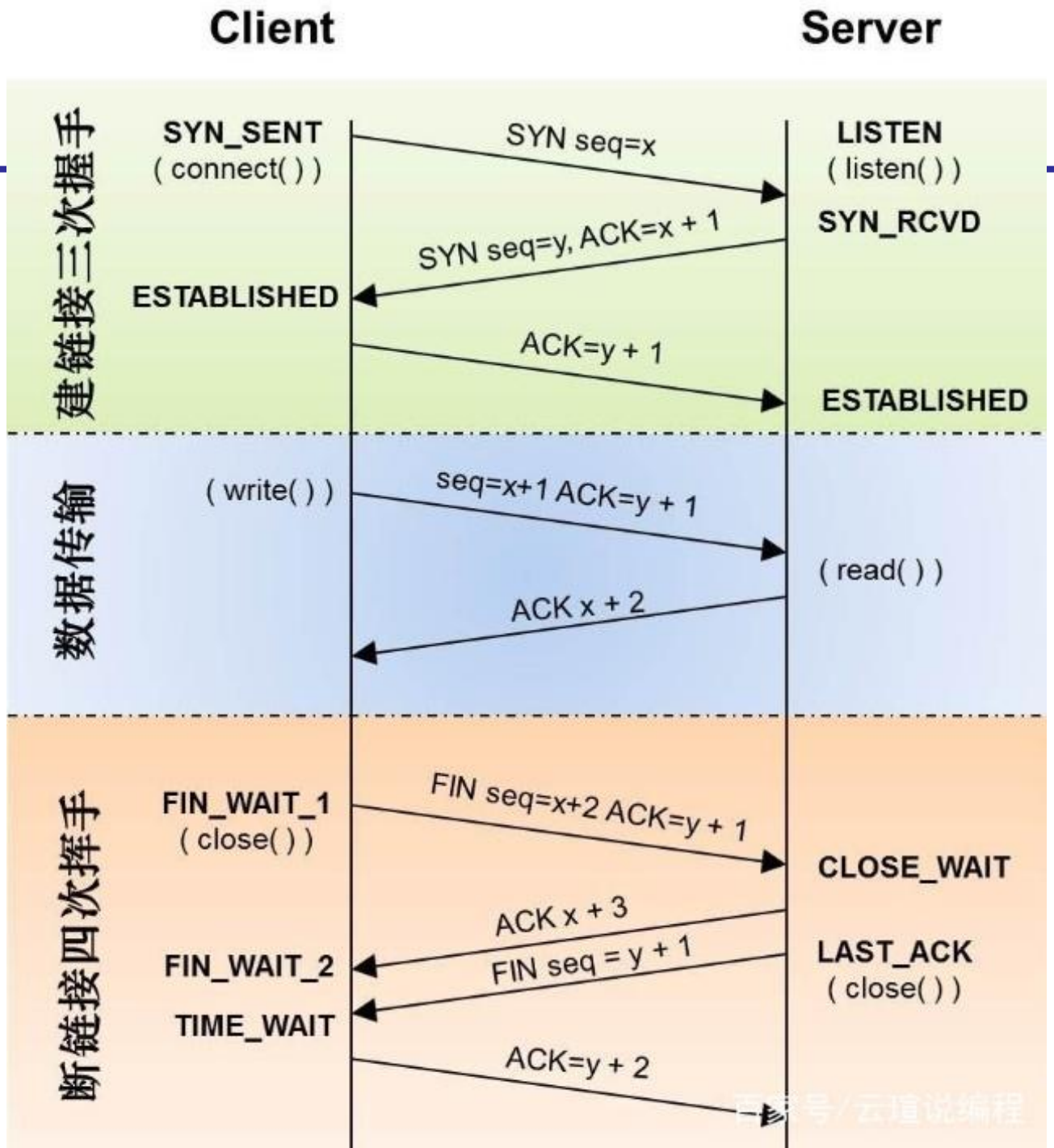


Unix套接字API

涉及到的系统API（部分）：

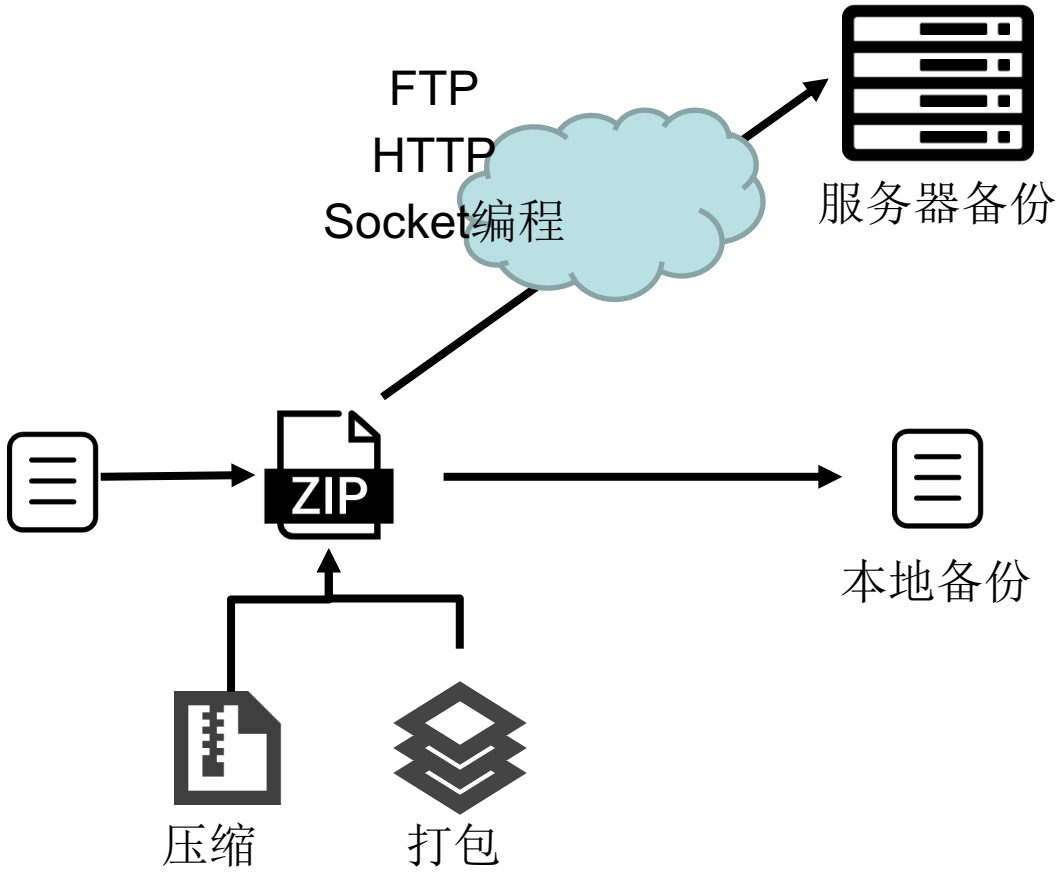
- socket
- bind
- listen
- connect
- accept
- read
- write
- close

1.


Client
Server
建链接三次握手
SYN_SENT
(connect( ))
SYN seq=x
LISTEN
(listen( ))
SYN_RCVD
SYN seq=y, ACK=x + 1
ESTABLISHED
ACK=y + 1
ESTABLISHED
数据传输
(write( ))
seq=x+1 ACK=y + 1
(read( ))
ACK x + 2
断链接四次挥手
FIN_WAIT_1
(close( ))
FIN seq=x+2 ACK=y + 1
CLOSE_WAIT
ACK x + 3
FIN_WAIT_2
FIN seq = y + 1
LAST_ACK
(close( ))
TIME_WAIT
ACK=y + 2

打包、压缩备份

在基础功能之上，提供目录树打包/解包、压缩/解压功能。



打包、压缩备份

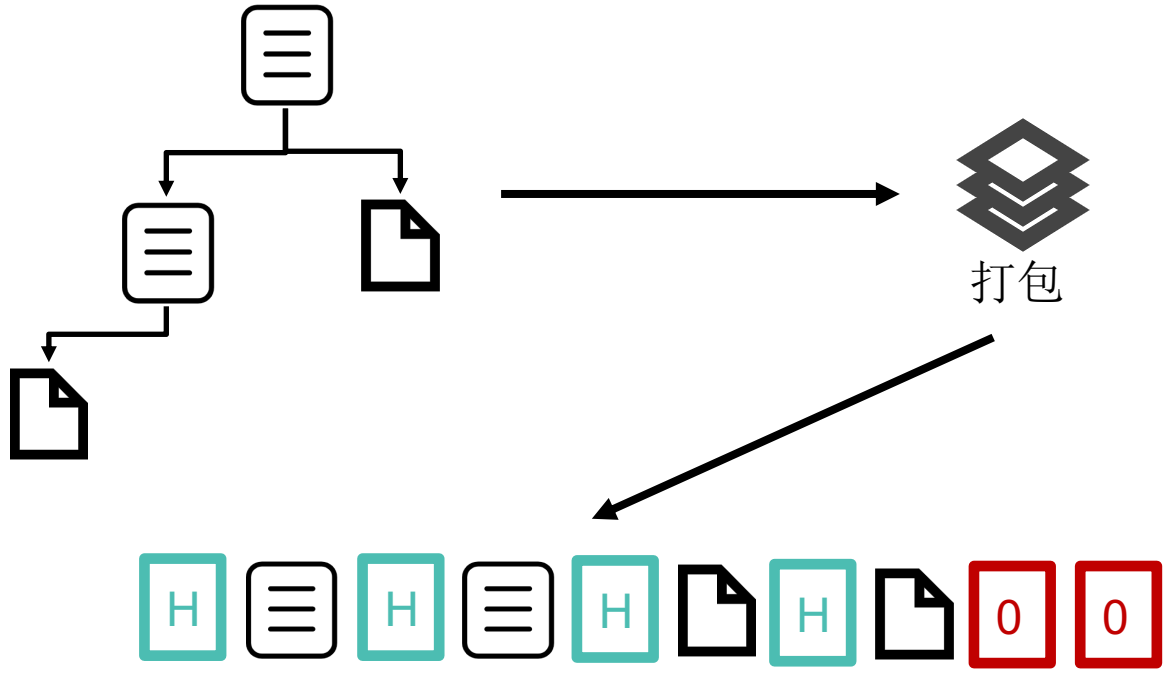
1. 目录树打包与解包
2. 文件压缩与解压

文件打包

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。

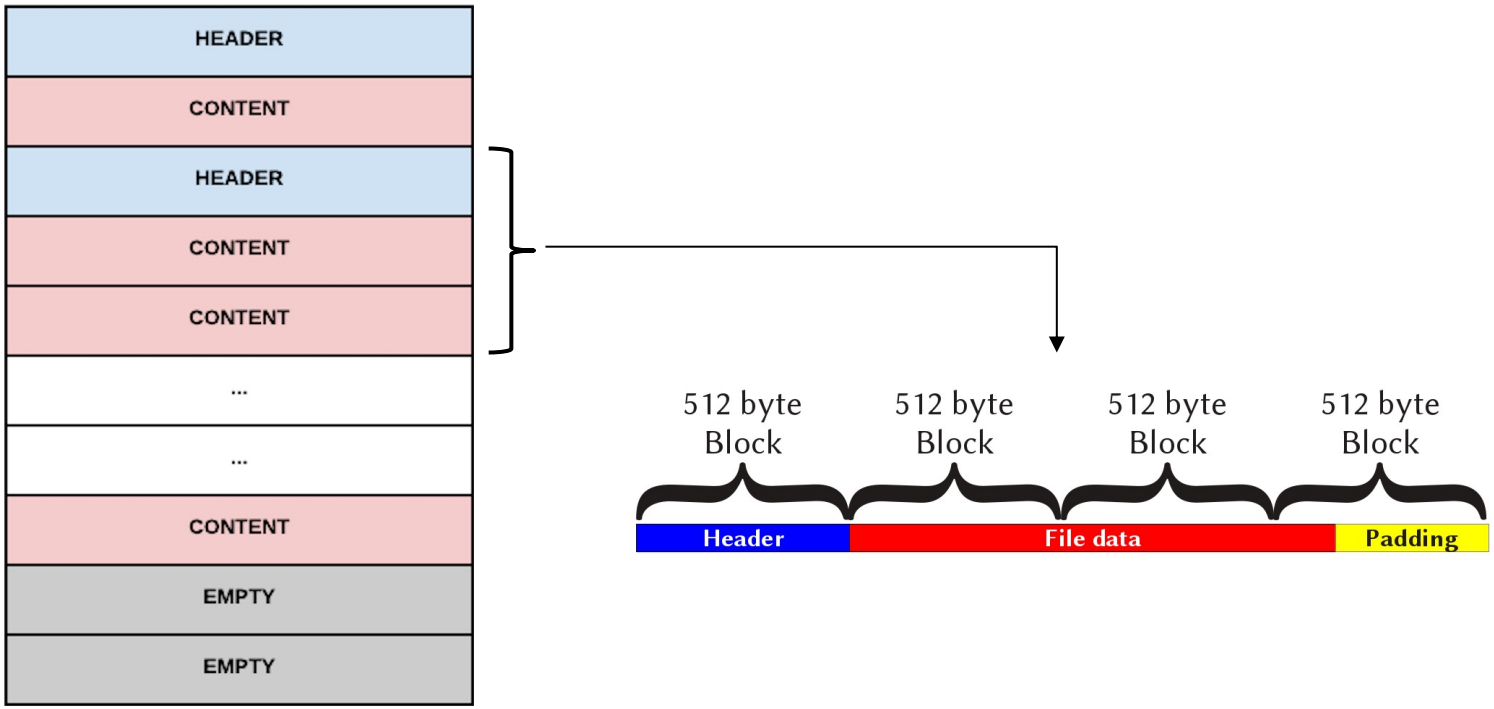
大部分压缩软件支持同时对文件打包。最经典的文件打包程序：Tar。



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。



```
union {
  union { // Pre-POSIX.1-1988 format
    struct {
      char name[100]; // file name                    文件名
      char mode[8]; // permissions
      char uid[8]; // user id (octal)
      char gid[8]; // group id (octal)
      char size[12]; // size (octal)                  文件大小
      char mtime[12]; // modification time (octal)
      char check[8]; // sum of unsigned characters in block,
          with spaces in the check field while calculation is done (octal)
      char link; // link indicator
      char link_name[100]; // name of linked file
    };
    // UStar format (POSIX IEEE P1003.1)
    struct {
      char old[156]; // first 156 octets of Pre-POSIX.1-1988 format
      char type; // file type                         文件类型
      char also_link_name[100]; // name of linked file
      char ustar[8]; // ustar\000
      char owner[32]; // user name (string)
      char group[32]; // group name (string)
      char major[8]; // device major number
      char minor[8]; // device minor number
      char prefix[155];                              文件路径
    };
  };                                                 文件内容
  char block[512]; // raw memory (500 octets of actual data, padded to 1 block)
```

编码解码

在电文传输中，需要将电文中出现的每个字符进行二进制**编码**。在设计编码时需要遵守两个原则：

（1）发送方传输的二进制编码，到接收方解码后必须具有**唯一性**，即解码结果与发送方发送的电文完全一样；

（2）发送的二进制编码尽可能地**短**。

等长编码

每个字符的**编码长度相同**（编码长度就是每个编码所含的二进制位数）。这种编码的特点是解码**简单**且具有**唯一性**，但编码长度并**不一定是最短的**。

不等长编码

每个字符的**编码长度不同**。

压缩（无损）：若对出现频度较高的字符分配相对较短的编码，同时，对出现频度较低的字符分配相对较长的编码，则编码之后的数据的二进制位数可以变小。

压缩编码分类

基于统计的方法：Huffman编码、香农-范诺编码(Shannon-Fano Codes)

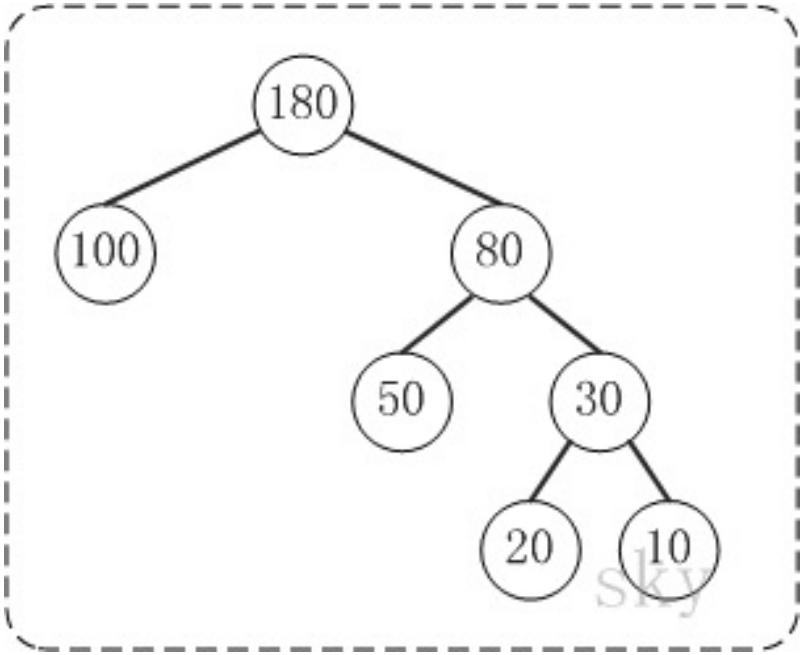
基于字典的方法：LZ77算法、 LZ78算法

混合方法：DEFLATE算法

以Huffman编码和LZ77算法为例。

哈夫曼树

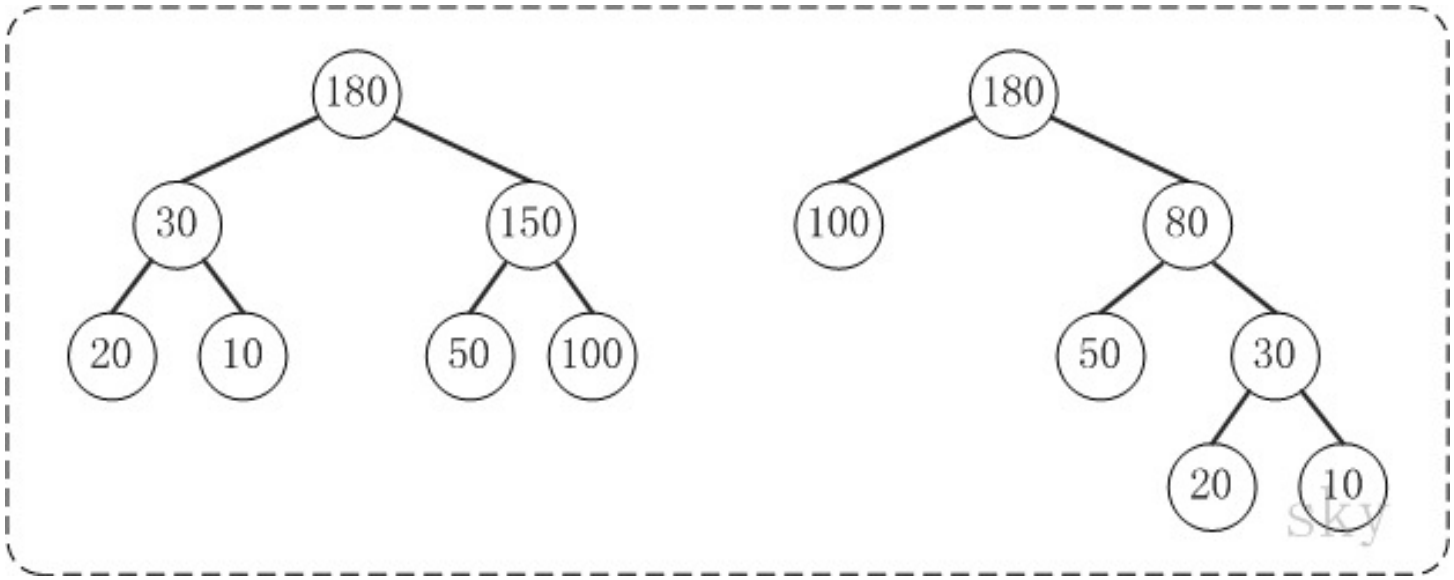
定义：给定n个权值作为n个**叶子结点**，构造一棵二叉树，若树的**带权路径长度**达到最小，则这棵树被称为哈夫曼树。



例子：示例中，
WPL
= 1*100 + 2*50 + 3*20 + 3*10
= 100 + 100 + 60 + 30
= 290

例子

比较下面两棵树：



左边的树WPL= 2*10 + 2*20 + 2*50 + 2*100 = 360
右边的树WPL= 1*100 + 2*50 + 3*20 + 3*10 = 290

问题：基于带有权值的叶子结点w_1、w_2、…，w_n，怎样构建哈夫曼树？

构造哈夫曼树

算法思路：

（1）将w_1、w_2、…，w_n看成是有n 棵树的森林(每棵树仅有一个结点)；

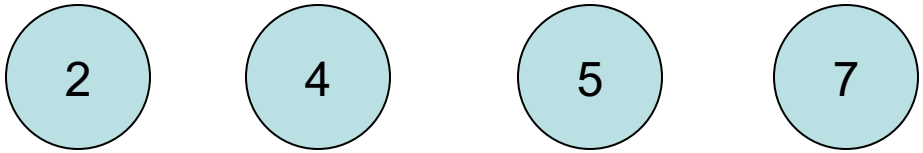
（2）在森林中选出根结点的**权值最小的两棵树进行合并**，作为一棵新树的左、 右子树，且新树的根结点权值为其左、右子树根结点权值之和；

（3）从森林中删除选取的两棵树，并将新树加入森林；

（4）重复(02)、(03)步，**直到森林中只剩一棵树为止**，该树即为所求得的哈夫曼树。

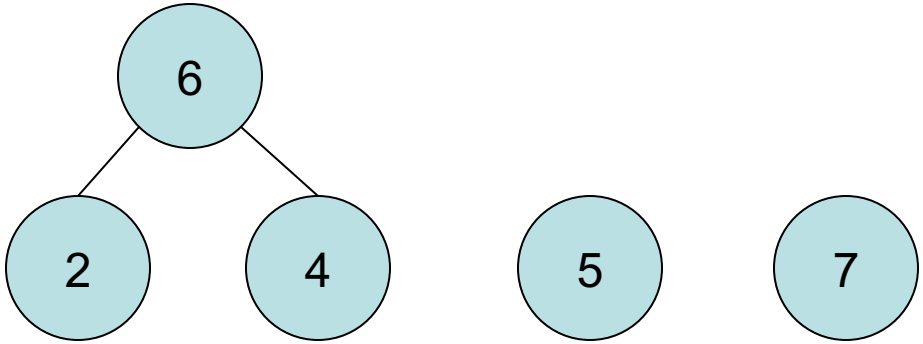
构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



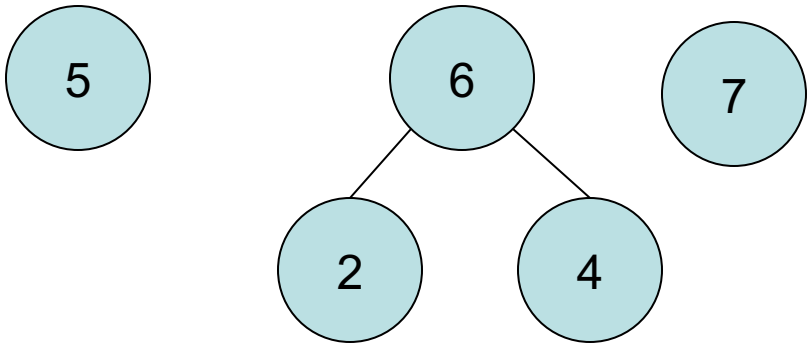
构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



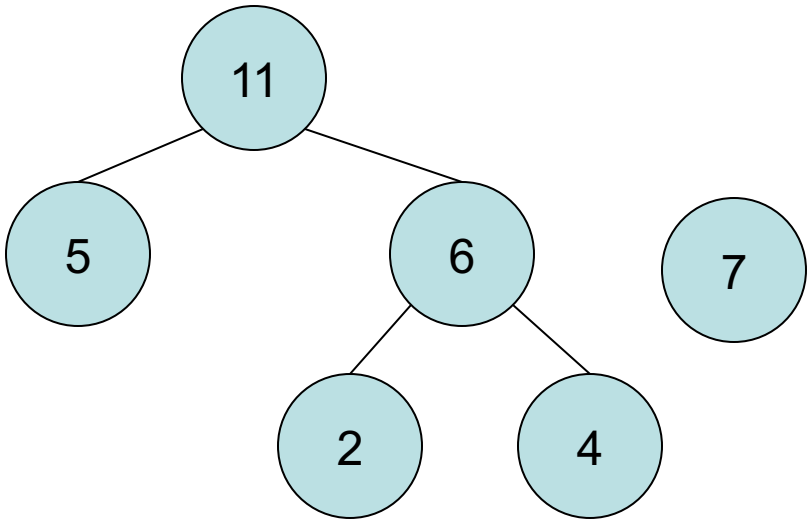
构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



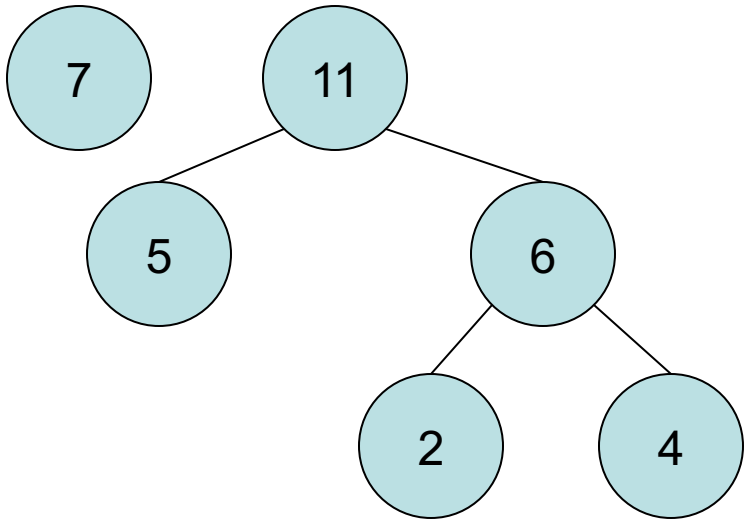
构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



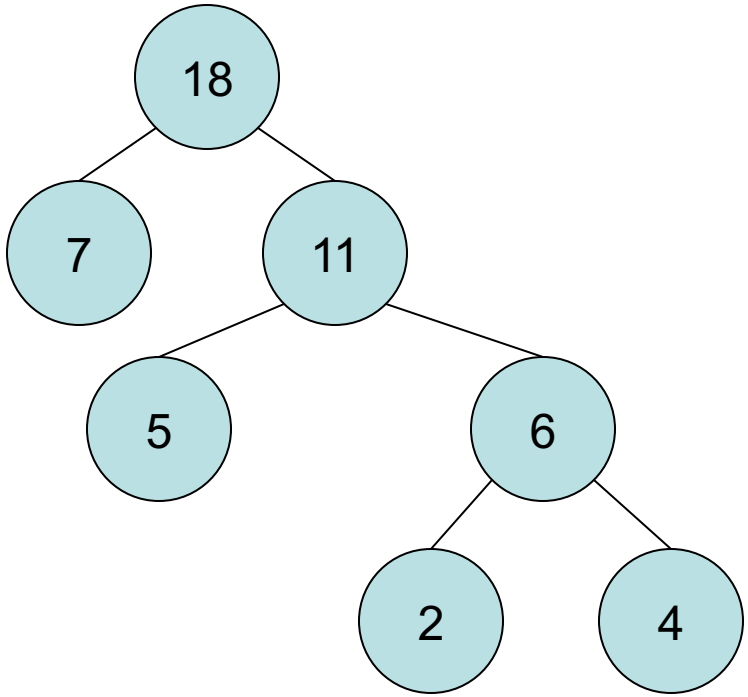
构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



构造哈夫曼树示例

当权值为{7，5，2，4}时，构造哈夫曼树。



哈夫曼编码

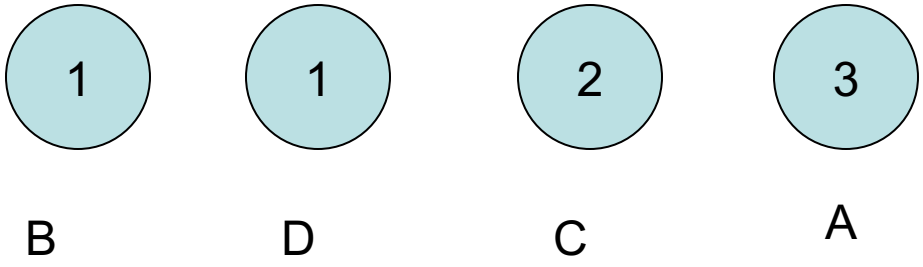
算法思路：

（1）利用字符集中每个字符（叶子结点）的使用频率作为权值构造哈夫曼树。

（2）从根结点开始，为其每个子结点赋予不同的**但顺序相同的**编码（如：左分支赋0，右分支赋1）。将从根结点到该叶子结点的路径编码作为该字符的编码。

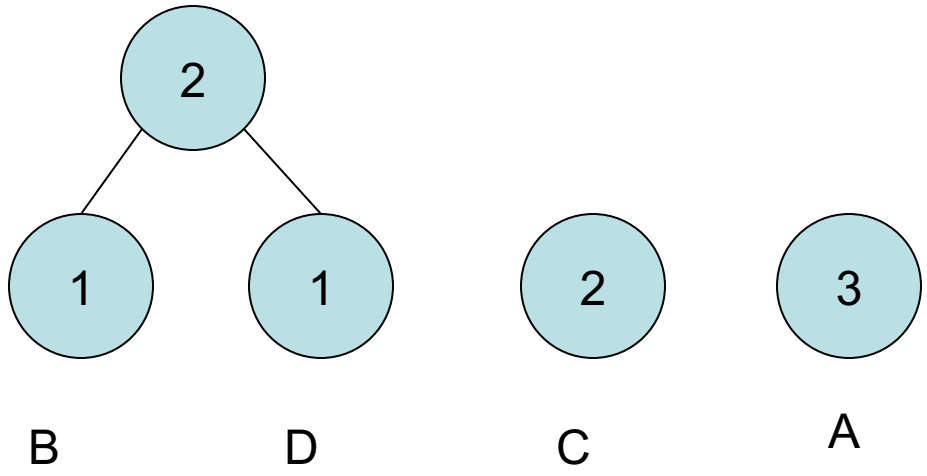
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



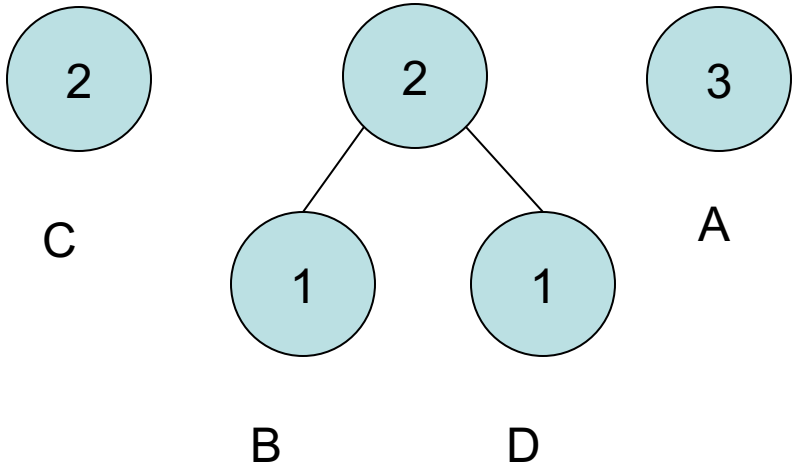
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



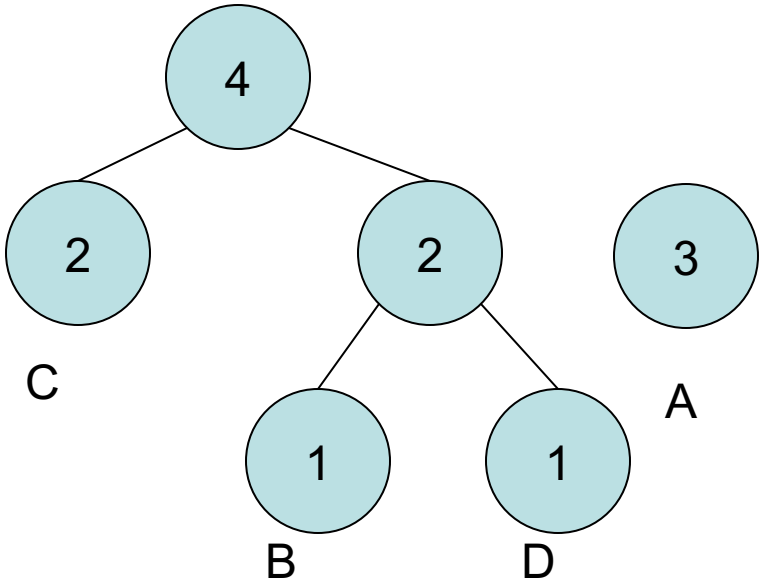
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



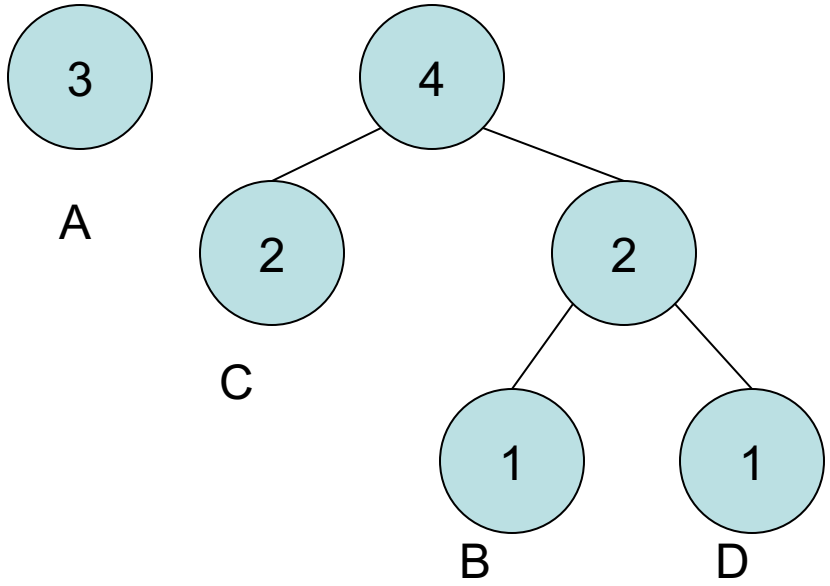
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



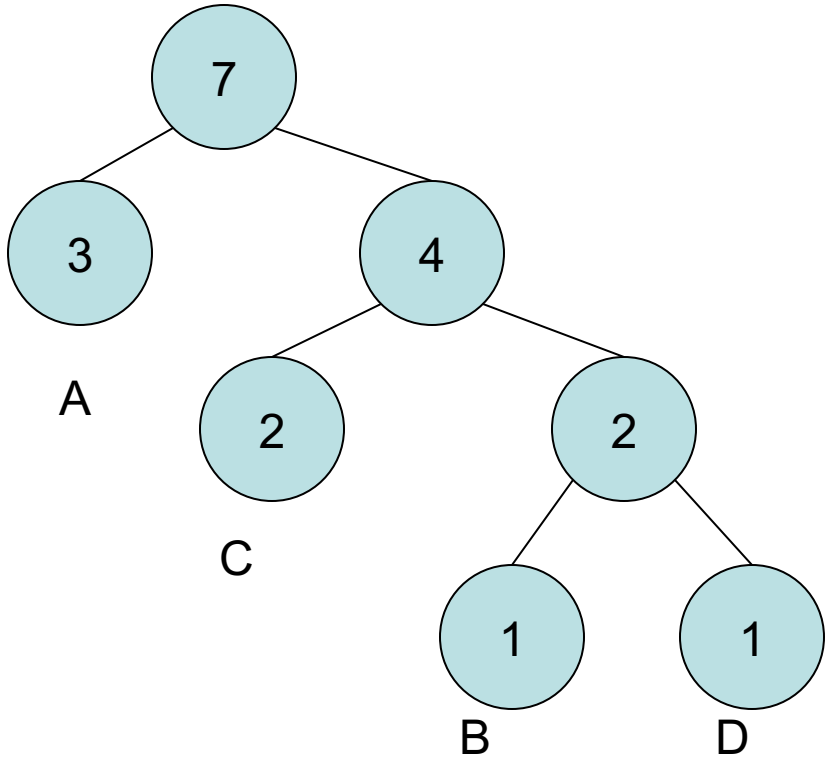
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



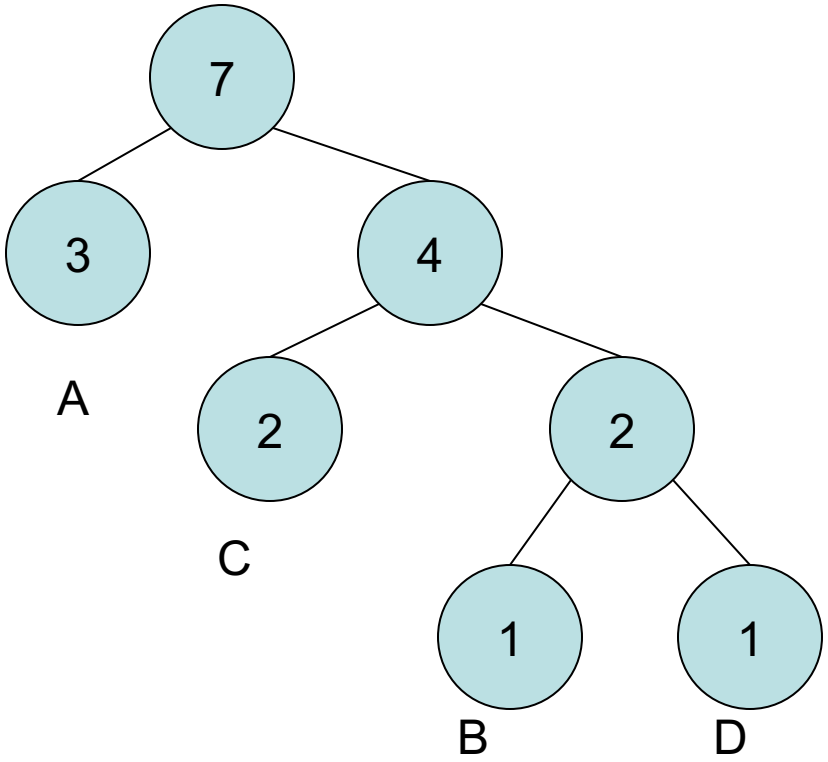
哈夫曼编码举例：ABACCDA

对于字符串“ABACCDA”，共有7个字符，4种字符。其中A、B、C、D出现的次数分别为3、1、2、1。根据权值{3，1，2，1} 构造哈夫曼树



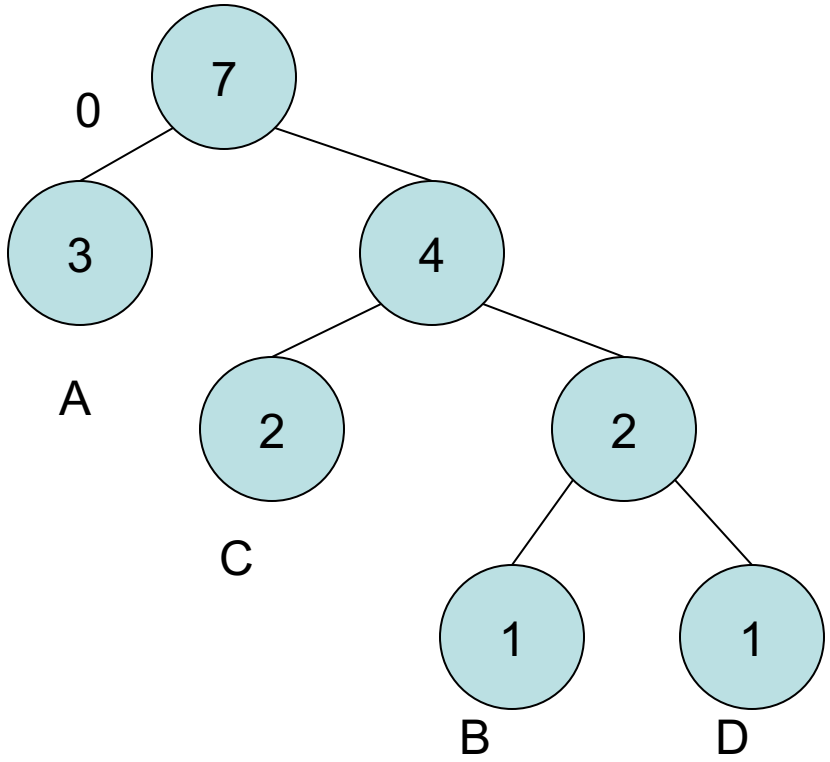
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



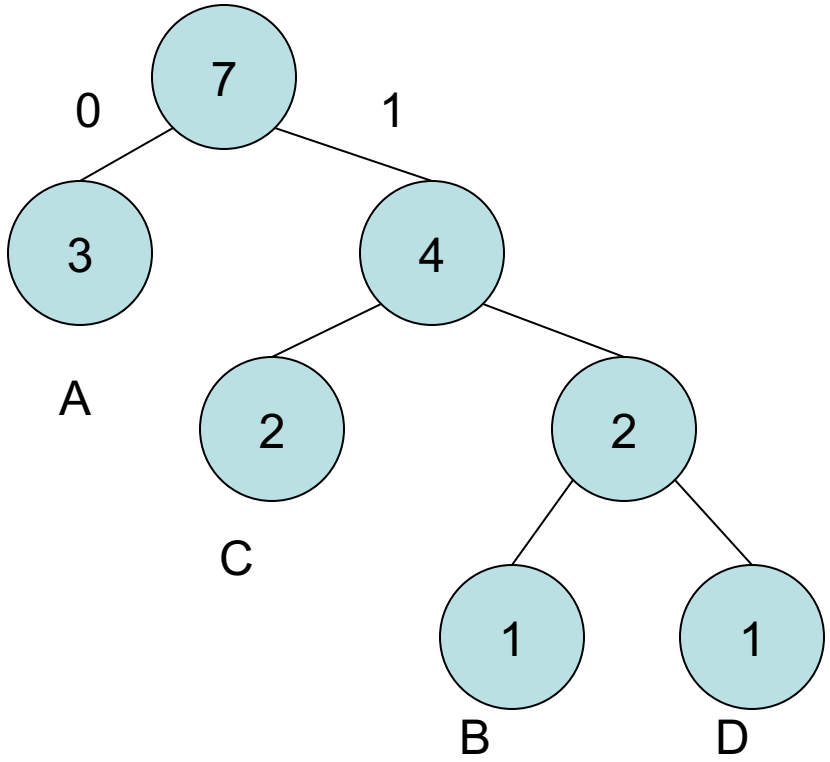
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



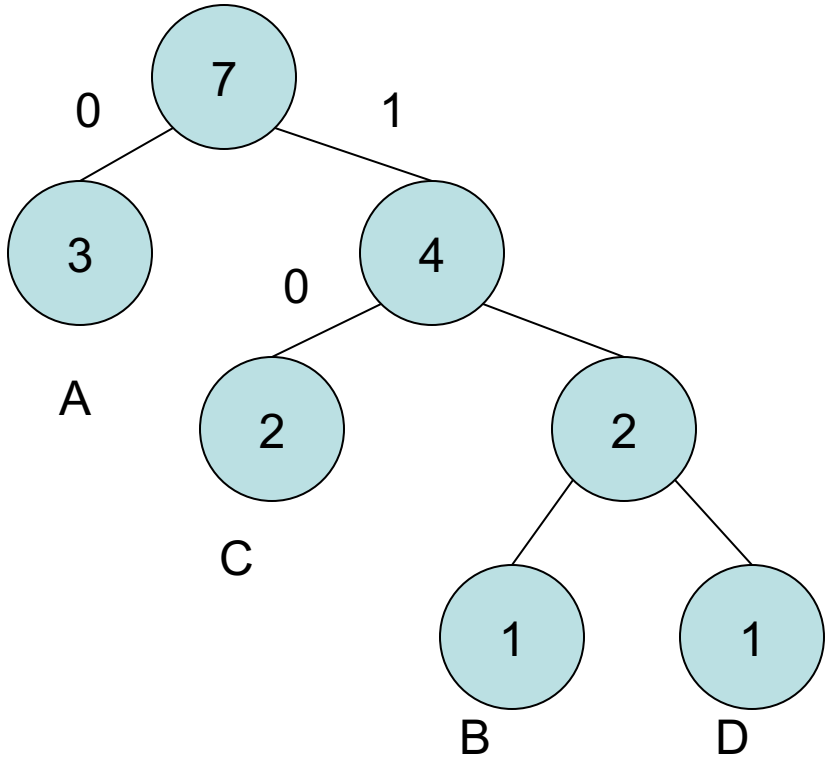
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



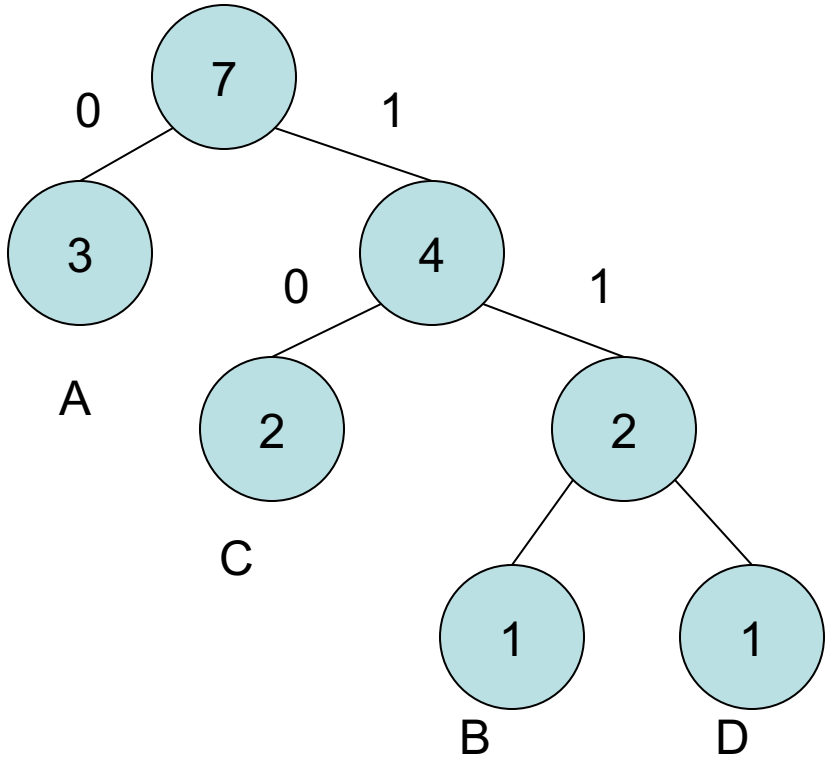
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



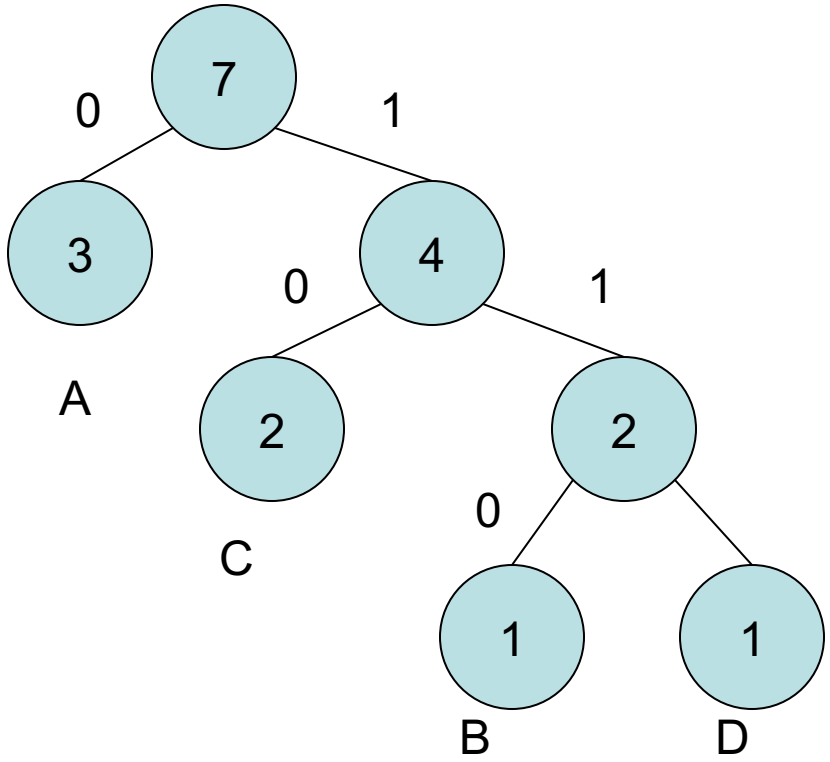
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



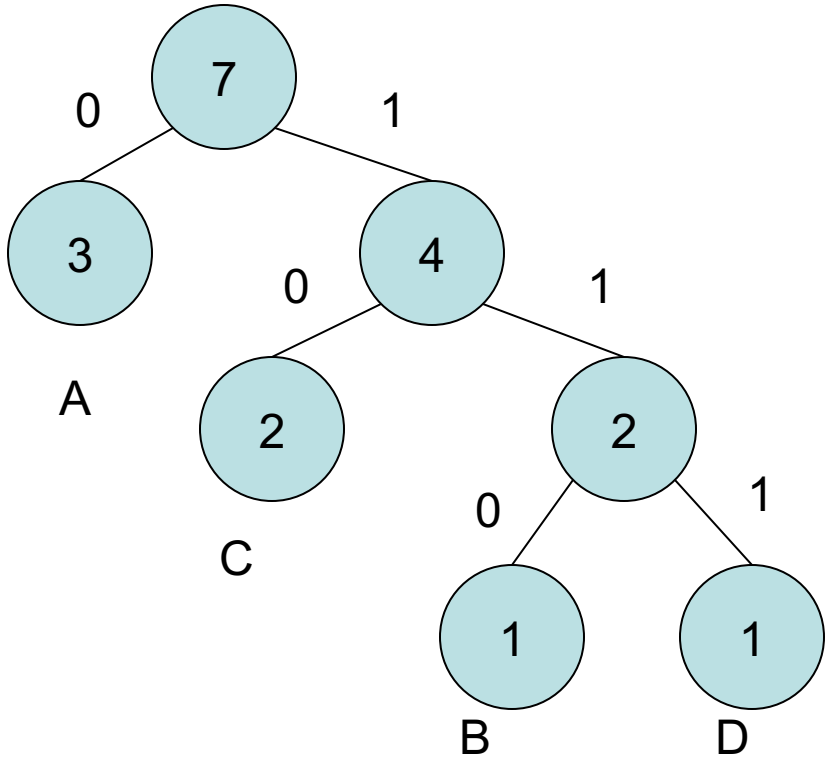
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



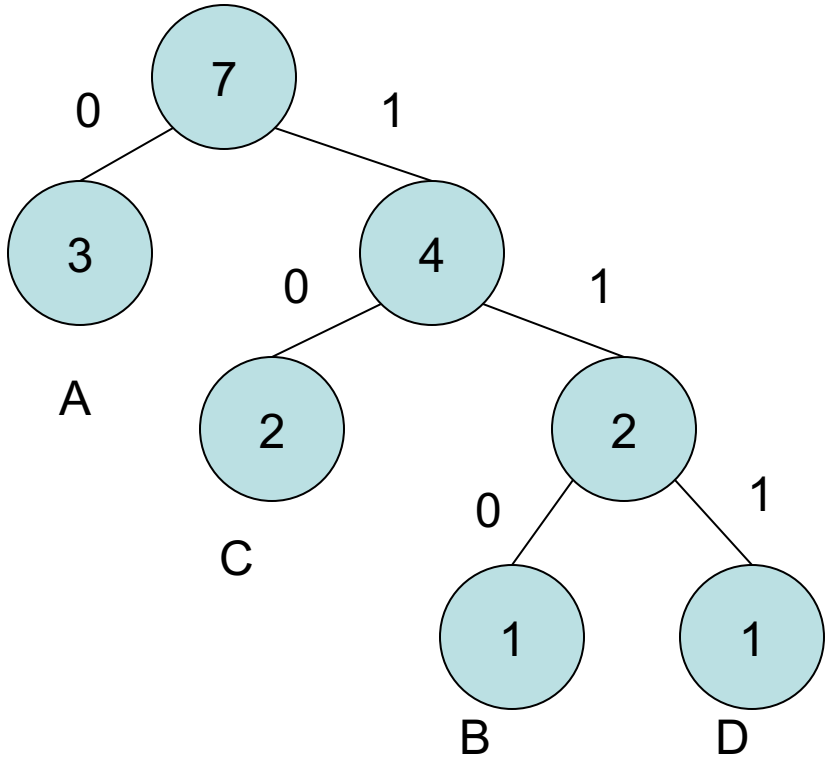
哈夫曼编码举例：ABACCDA

对哈夫曼树进行编码，每一个结点的左分支上面标0，右分支上面标1。



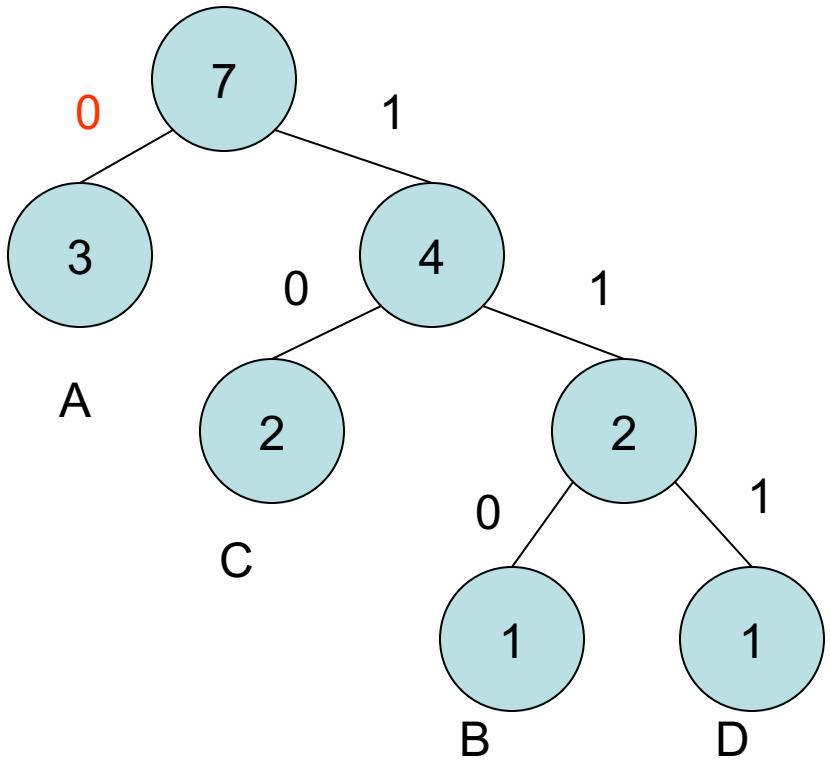
哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



哈夫曼编码举例：ABACCDA

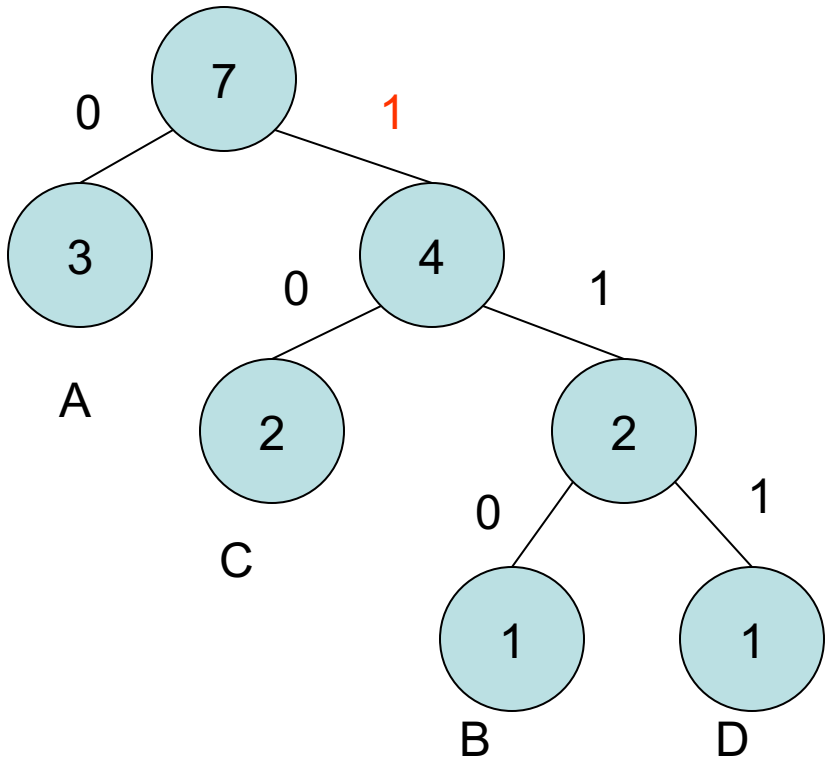
从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



A：0

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。

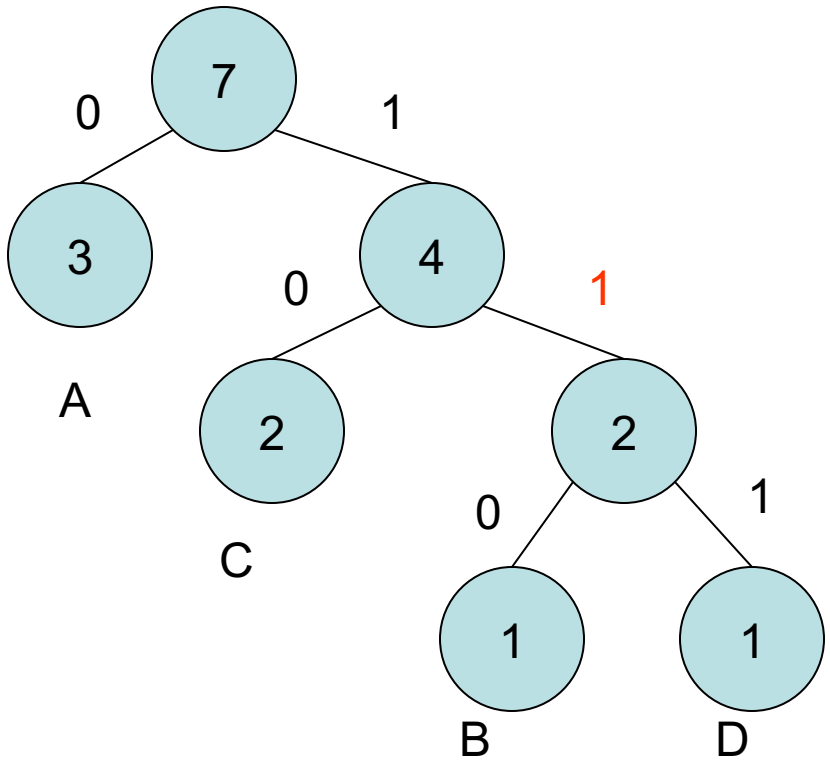


A：0

B：1

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。

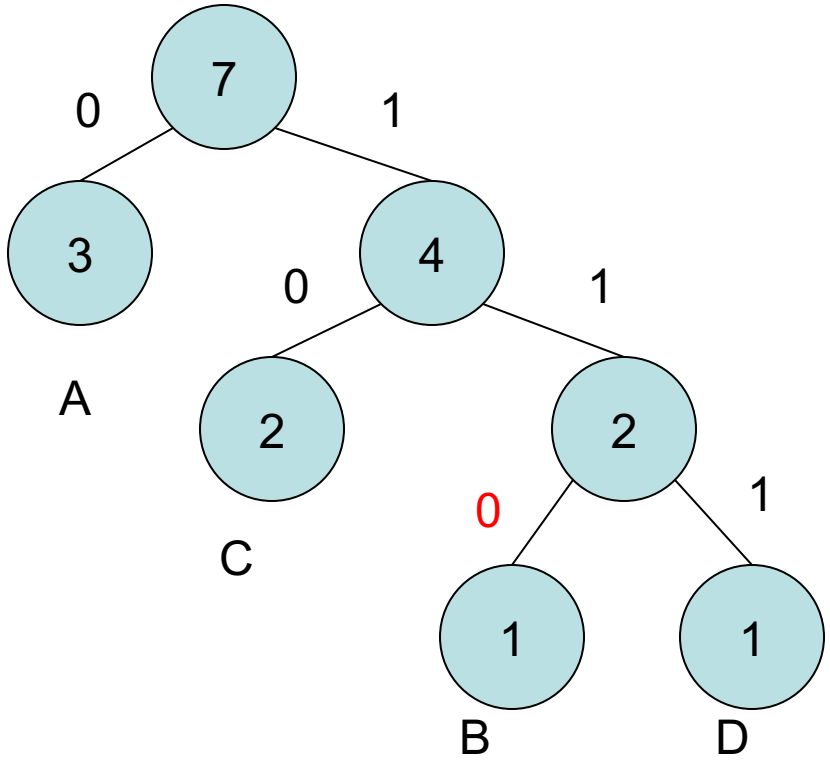


A：0

B：11

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。

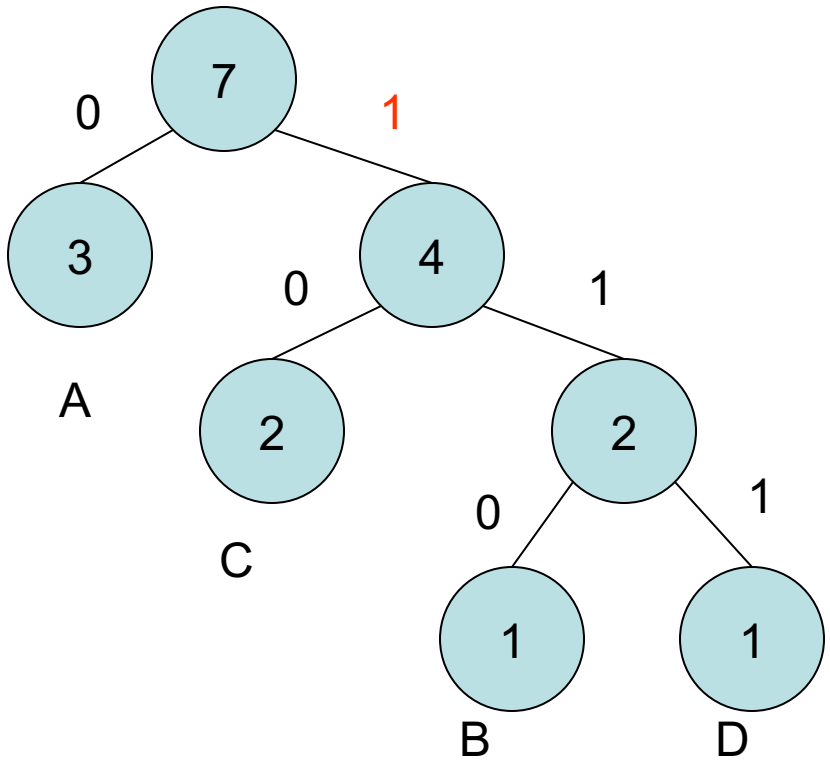


A：0

B：110

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



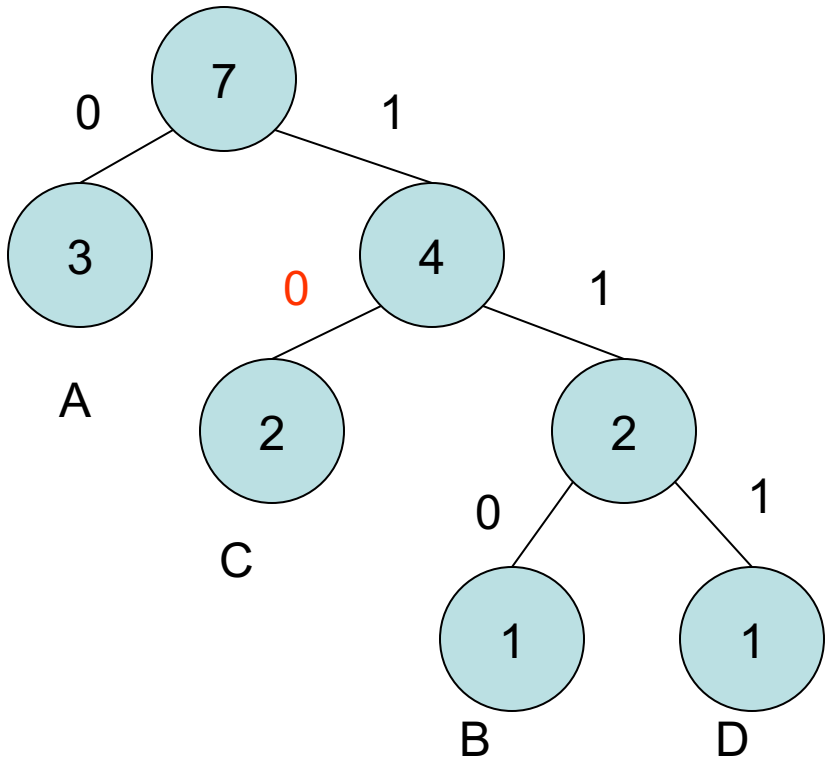
A：0

B：110

C：1

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



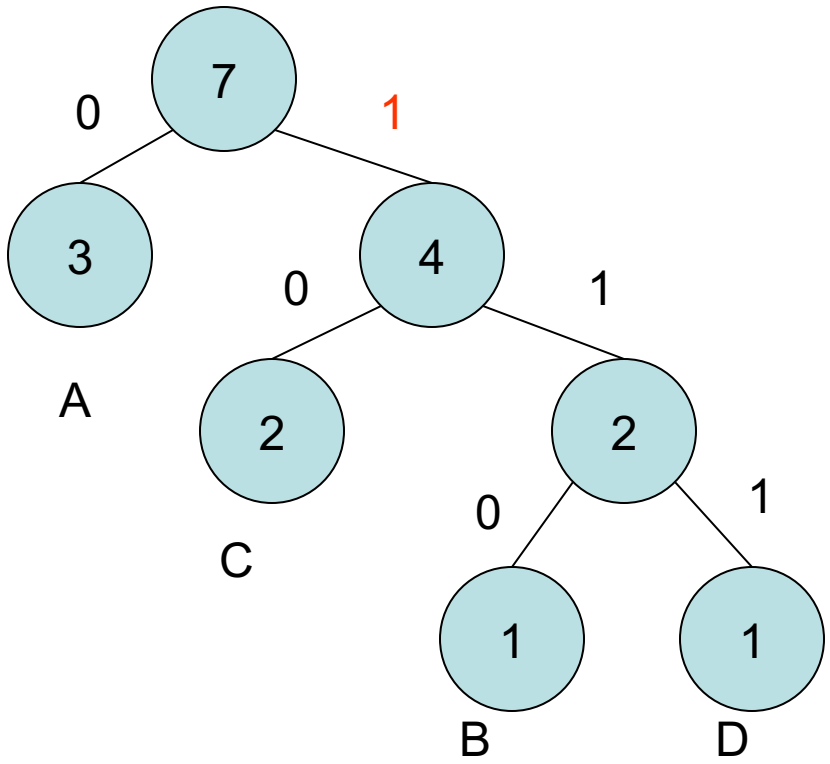
A：0

B：110

C：10

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



A：0

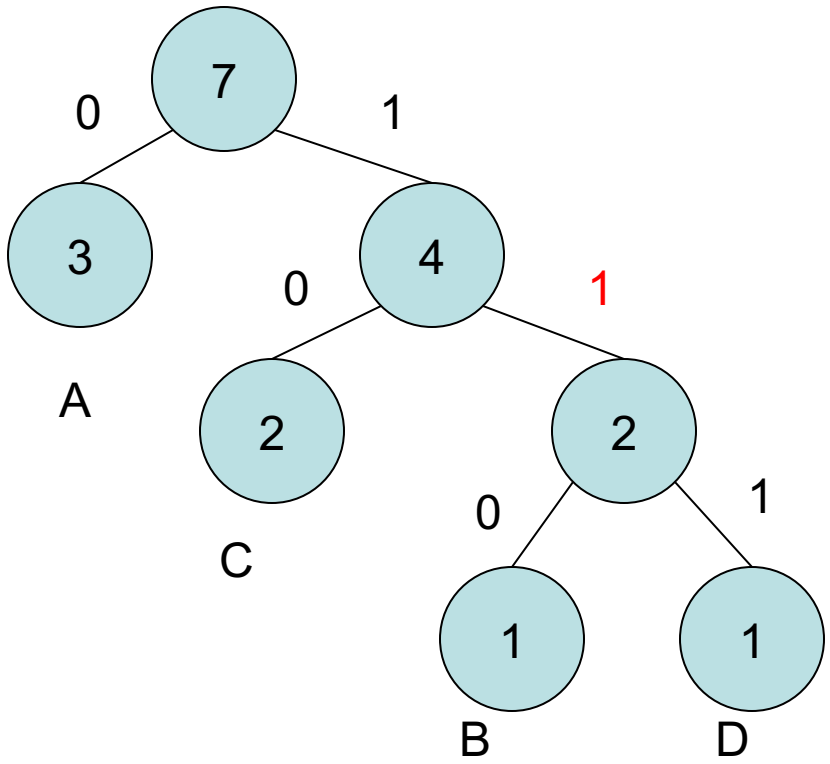
B：110

C：10

D：1

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



A：0

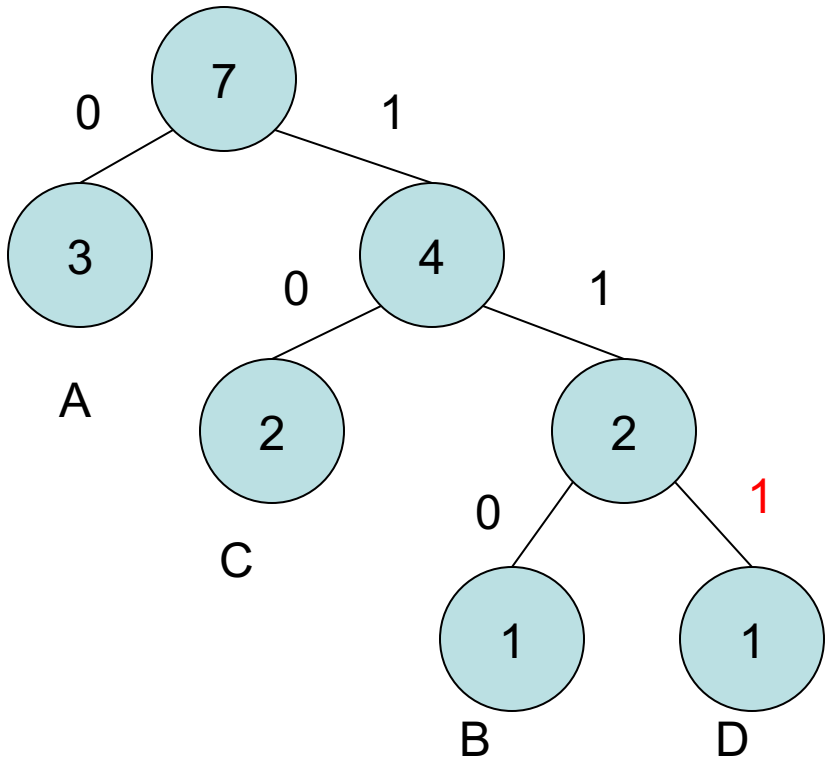
B：110

C：10

D：11

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



A：0

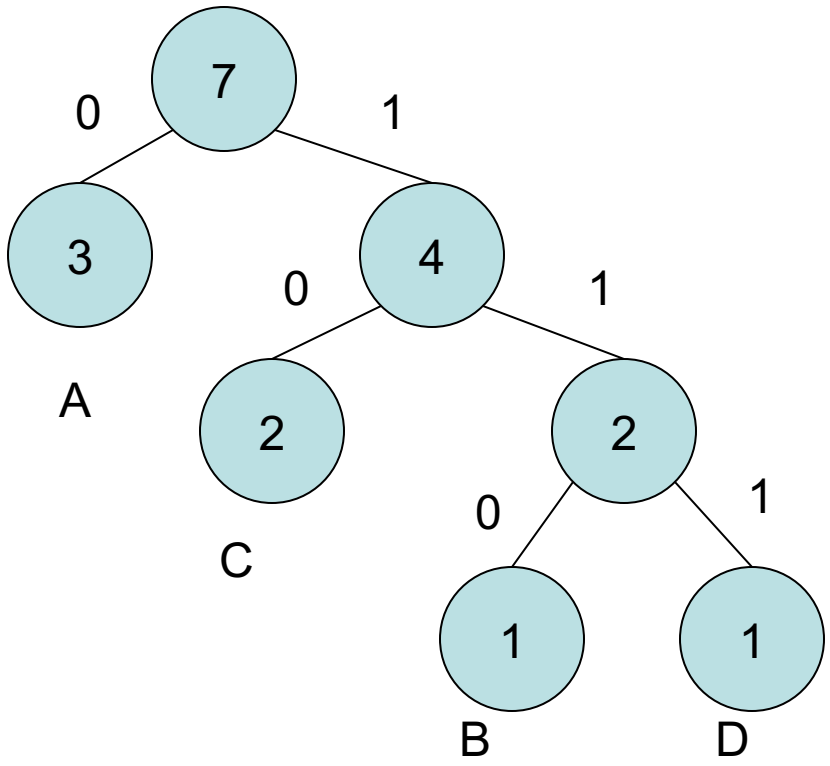
B：110

C：10

D：111

哈夫曼编码举例：ABACCDA

从根结点到各个叶子结点，所经分支上面的0、1序列，就是该叶子结点所代表字符的编码。



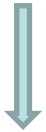
A：0

B：110

C：10

D：111

ABACCDA



0110010101110

使用哈夫曼编码进行文件压缩

目标

思路

基于单词编码进行文件压缩

使用哈夫曼编码进行文件压缩

1

项目概述与技术基础

目标

基于单词编码进行文件压缩

思路

统计词频

使用哈夫曼编码进行文件压缩

1

项目概述与技术基础

目标

基于单词编码进行文件压缩

思路

统计词频

⇩

根据字母出现次数？
根据汉字出现次数？

使用哈夫曼编码进行文件压缩

目标

基于单词编码进行文件压缩

思路

统计词频

⇩

~~根据字母出现次数？~~
~~根据汉字出现次数？~~

⇩

文件在内存都是二进制存储的
根据Byte出现次数

使用哈夫曼编码进行文件压缩

目标

基于单词编码进行文件压缩

思路

统计词频

⇩

~~根据字母出现次数？~~
~~根据汉字出现次数？~~

⇩

文件在内存都是二进制存储的
根据Byte出现次数

对文件以二进制方式读入，然后对每8个bit，即1个Byte（刚好对应256个ACSII字符）进行统计，这样就可以统计出文件对应的256个字符的权值。

使用哈夫曼编码进行文件压缩

目标

基于单词编码进行文件压缩

思路

统计词频

~~根据字母出现次数？~~
~~根据汉字出现次数？~~

文件在内存都是二进制存储的
根据Byte出现次数

根据Byte词频构建哈夫曼树

对文件以二进制方式读入，然后对每8个bit，即1个Byte（刚好对应256个ACSII字符）进行统计，这样就可以统计出文件对应的256个字符的权值。

使用哈夫曼编码进行文件压缩

目标

基于单词编码进行文件压缩

⇧

获得每个单词的编码

⇧

根据Byte词频构建哈夫曼树

思路

统计词频

⇩

~~根据字母出现次数？~~
~~根据汉字出现次数？~~

⇩

文件在内存都是二进制存储的
根据Byte出现次数

⇦

对文件以二进制方式读入，然后对每8个bit，即1个Byte（刚好对应256个ACSII字符）进行统计，这样就可以统计出文件对应的256个字符的权值。

使用哈夫曼编码进行文件压缩

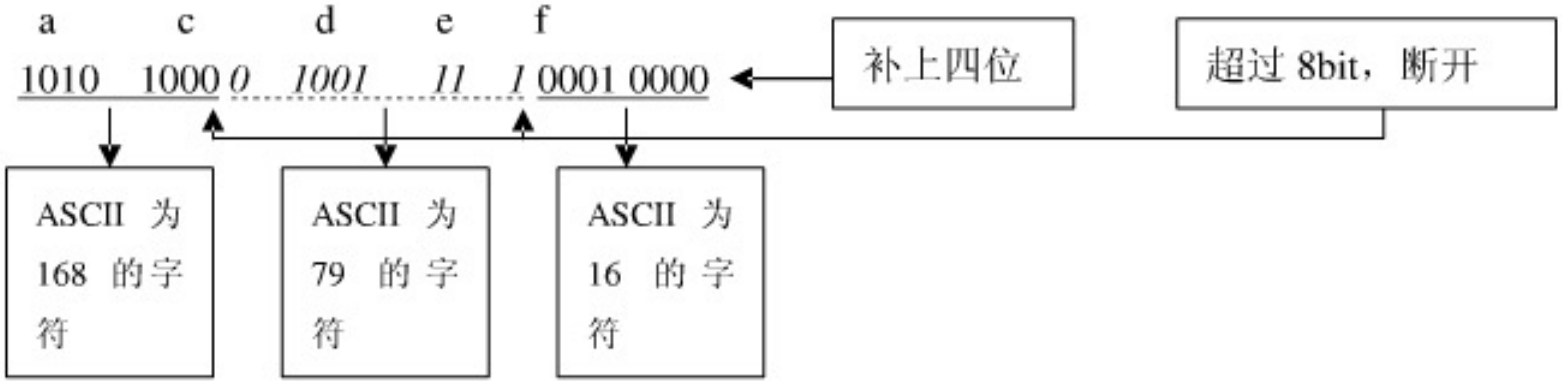
注意：不要用char数据类型来保存每一位编码。这样得到的编码实际上是由字符'0'和字符'1'组成的字符串，不但不能使文件得到压缩，反而会使文件增大。

解决方案：位操作！

举例如下：

已知：a: 1010　b: 00　c: 10000　d: 1001　e: 11　f: 10001　g: 01　h: 1001

如果其中的一段是:: acdef 则处理如下：



注意：需要记录真实编码位数或者补位的位数。否则，最后一个Byte无法正确解码。

使用哈夫曼编码进行文件压缩

需要存储的用于解码的信息：

1、编码之后的数据

2、编码后数据的真实长度

3、解码时需要用的数据（用于创建哈夫曼树）

解码用数据存储方法

方法一：编码对照表

A：0

B：110

C：10

D：111

编码对照表存储方式：

字符+编码位长度+编码（补零）

解码时需要用的数据

方法一：编码对照表

A：0

B：110

C：10

D：111

编码对照表存储方式：

字符+编码位长度+编码（补零）

方法二：词频表

A：3

B：1

C：2

D：1

直接记录原始文件中各字符出现的频率，解码时读取该频率信息，重新生成一棵同样的哈夫曼树。

LZ77算法

基于统计的压缩编码算法（如Huffman编码）需要事先得到数据的出现频率。**但有时（如流数据压缩），这种先验知识很难预先获得。**因此，需要更改为通用的压缩编码算法。

LZ77：基于滑动窗口的动态字典编码。

原理：利用数据的**重复结构信息**来进行数据压缩。

LZ77核心概念

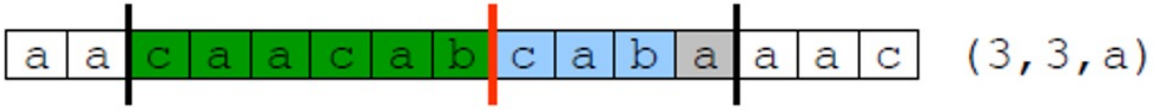
滑动窗口：任一时刻，编解码过程仅关注的数据段。

光标：当前需要编码的字符位置，也是字典区域与前向缓冲之间的边界

字典区域：随滑动窗口滑动，用作编码时的参考依据。

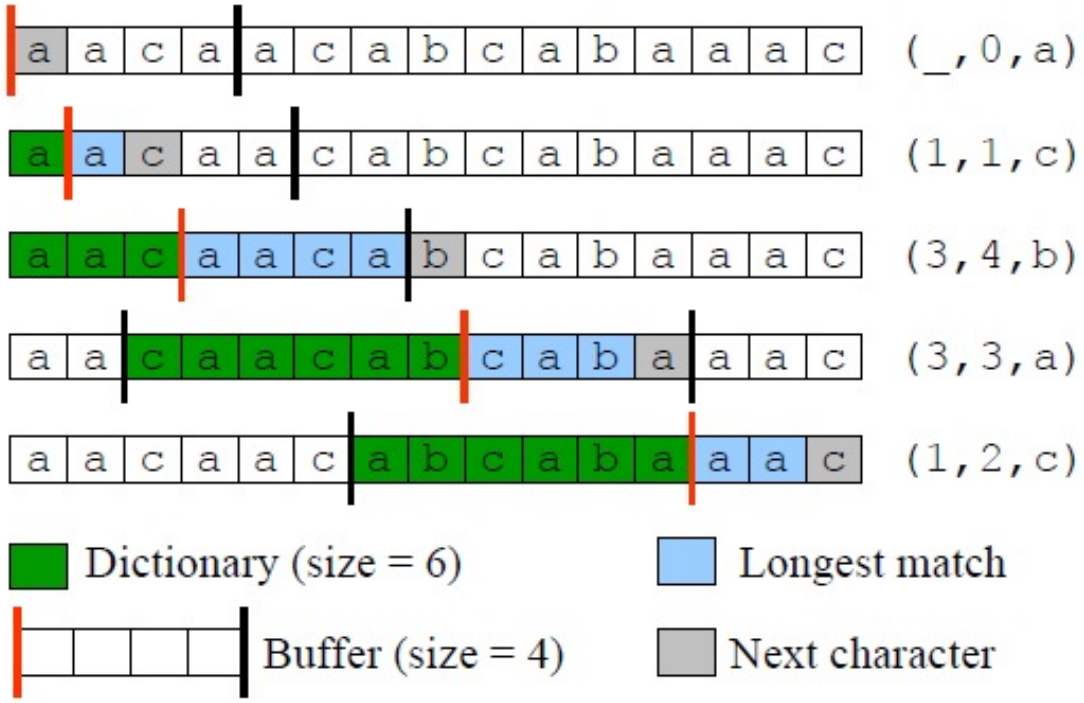
前向缓冲：随滑动窗口滑动，采用最大最长匹配算法来匹配字典（包括前向缓冲本身）中出现过的重复数据。

编码格式：在搜索到匹配数据之后（也可以没匹配到），将匹配结果进行编码。



LZ77算法举例

编码以下数据：aacaacabcabaaac



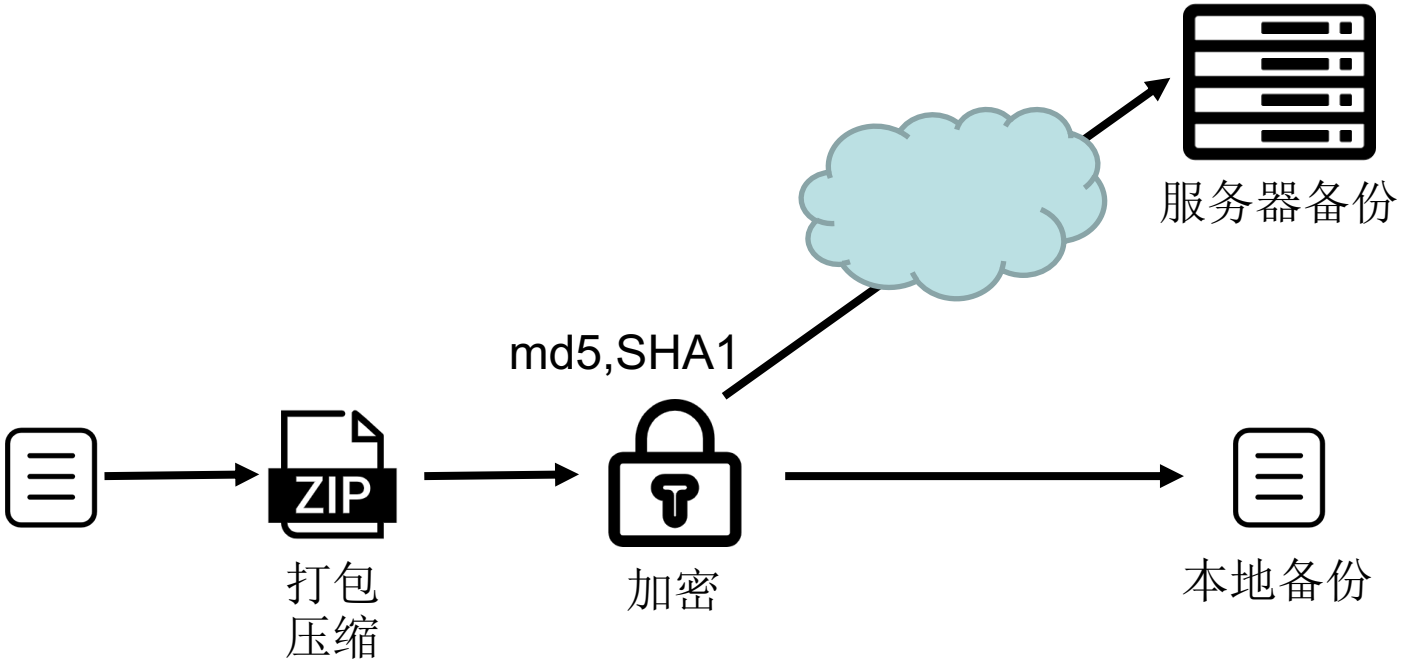
思考：

- 1、哈夫曼编码和LZ77算法各有什么优缺点？
- 2、在本项目中，应如何选择压缩算法？

加密备份

3.加密备份

在基础功能之上，提供加密功能。



对称加密和散列算法

对于数据备份而言，主要考虑**对称加密**和**散列算法**。

散列算法也称为摘要算法，如MD5、SHA1等。

简单对称加密方式可考虑流密码。

更优的对称加密方式为分组加密，如DES、AES等。

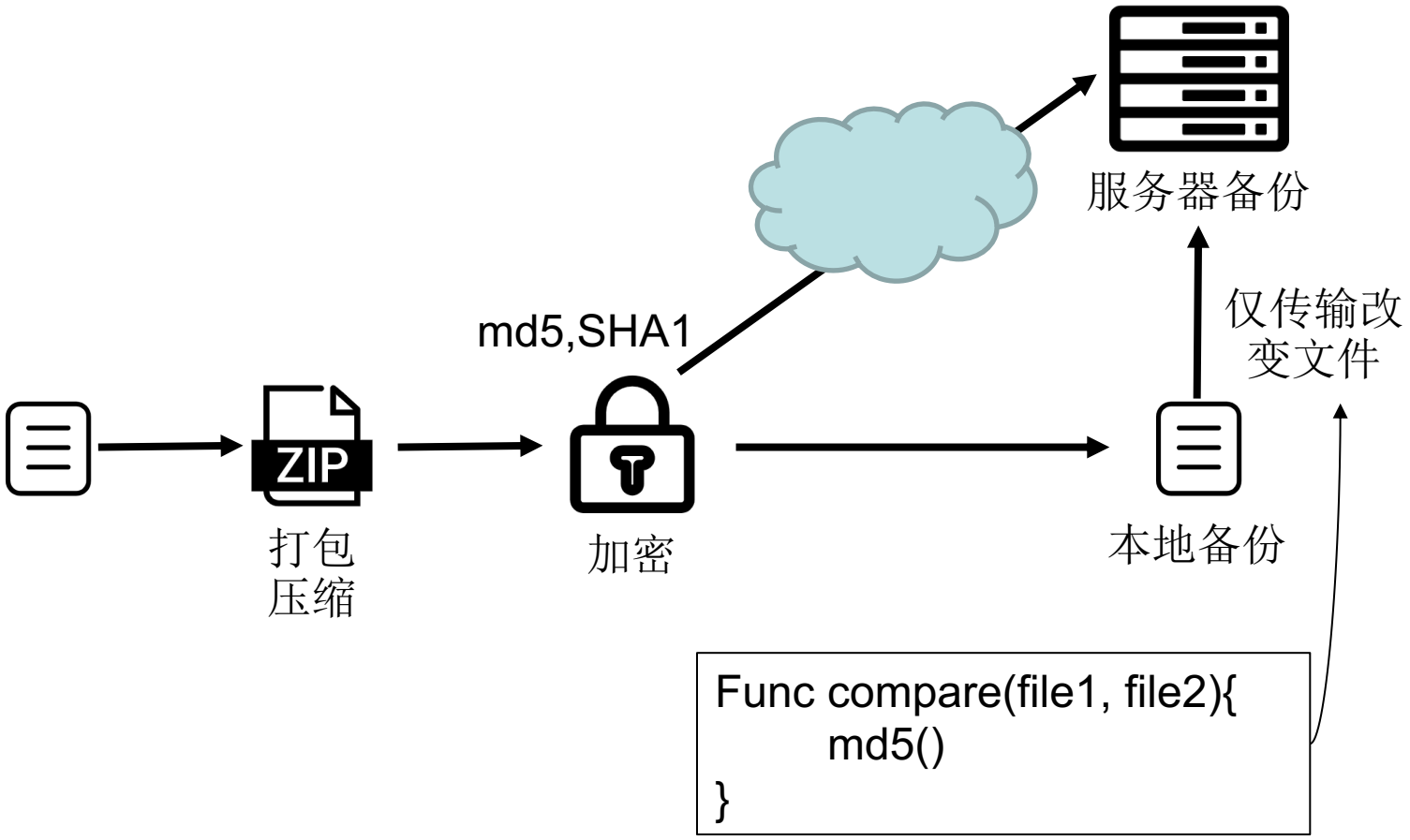
加解密常用库——OpenSSL

OpenSSL中的MD5、AES相关函数：

MD5_Init
MD5_Update
MD5_Final
AES_set_encrypt_key
AES_set_decrypt_key
AES_cbc_encrypt
AES_ecb_encrypt
AES_ofb128_encrypt

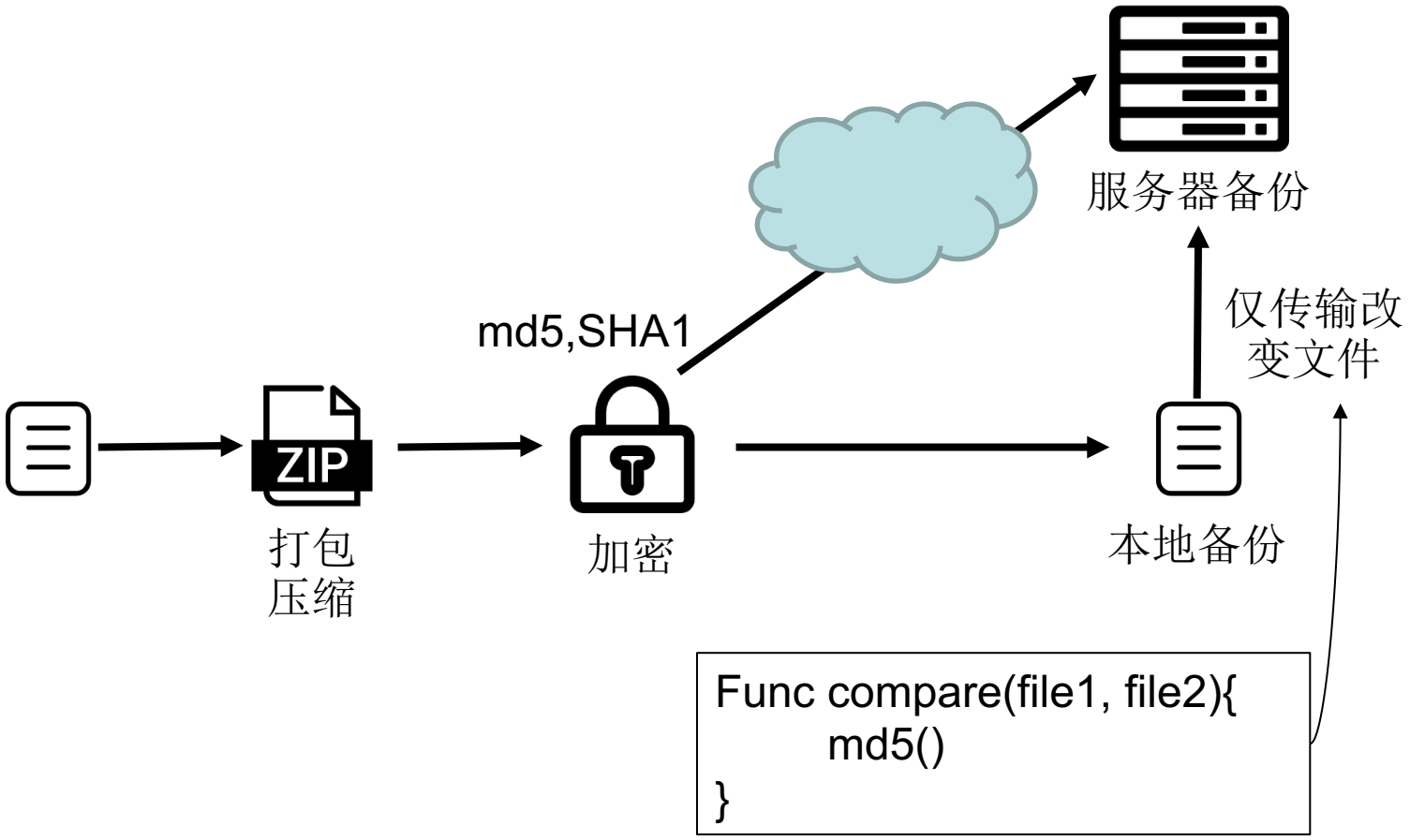
增量备份

在服务器备份之上，提供文件比对功能，仅备份改变的文件。



实时备份

本地设置监听，一旦本地备份目录的数据发生改变，则将改变的文件同步到服务器。



实时备份

实现策略

1. 本地开启一个后台服务，定时备份（比如间隔时间1min）
2. 采用Unix inotify机制进行文件监听

Unix文件系统事件感知

现代操作系统均提供了对自身文件系统状态变化的通知机制，如

1. Windows中的FindFirstChangeNotification系列API
2. Linux中的inotify系列API

通过对文件系统中的某些文件和文件夹的监控，从而感知文件系统的各种变化。

以下对Linux中的inotify系列API进行介绍。

Inotify系列API

inotify是Linux内核中的一个子系统，提供了一种监控文件系统（基于inode的）事件的机制，可以监控文件系统的变化如文件修改、新增、删除等，并可以将相应的事件通知给应用程序，可以同时监控多个文件和目录。

inotify使用文件描述符作为接口，符合Linux中“一切皆文件”的设计思想，可以与Linux中的IO复用机制（如select、poll、epoll）一起使用。

其能够监控的事件包括：

```
IN_ACCESS：文件被访问
IN_MODIFY：文件被修改
IN_ATTRIB，文件属性被修改
IN_CLOSE_WRITE，以可写方式打开的文件被关闭
IN_CLOSE_NOWRITE，以不可写方式打开的文件被关闭
IN_OPEN，文件被打开
IN_MOVED_FROM，文件被移出监控的目录
IN_MOVED_TO，文件被移入监控着的目录
IN_CREATE，在监控的目录中新建文件或子目录
IN_DELETE，文件或目录被删除
IN_DELETE_SELF，自删除，即一个可执行文件在执行时删除自己
IN_MOVE_SELF，自移动，即一个可执行文件在执行时移动自己
```

Inotify系列API

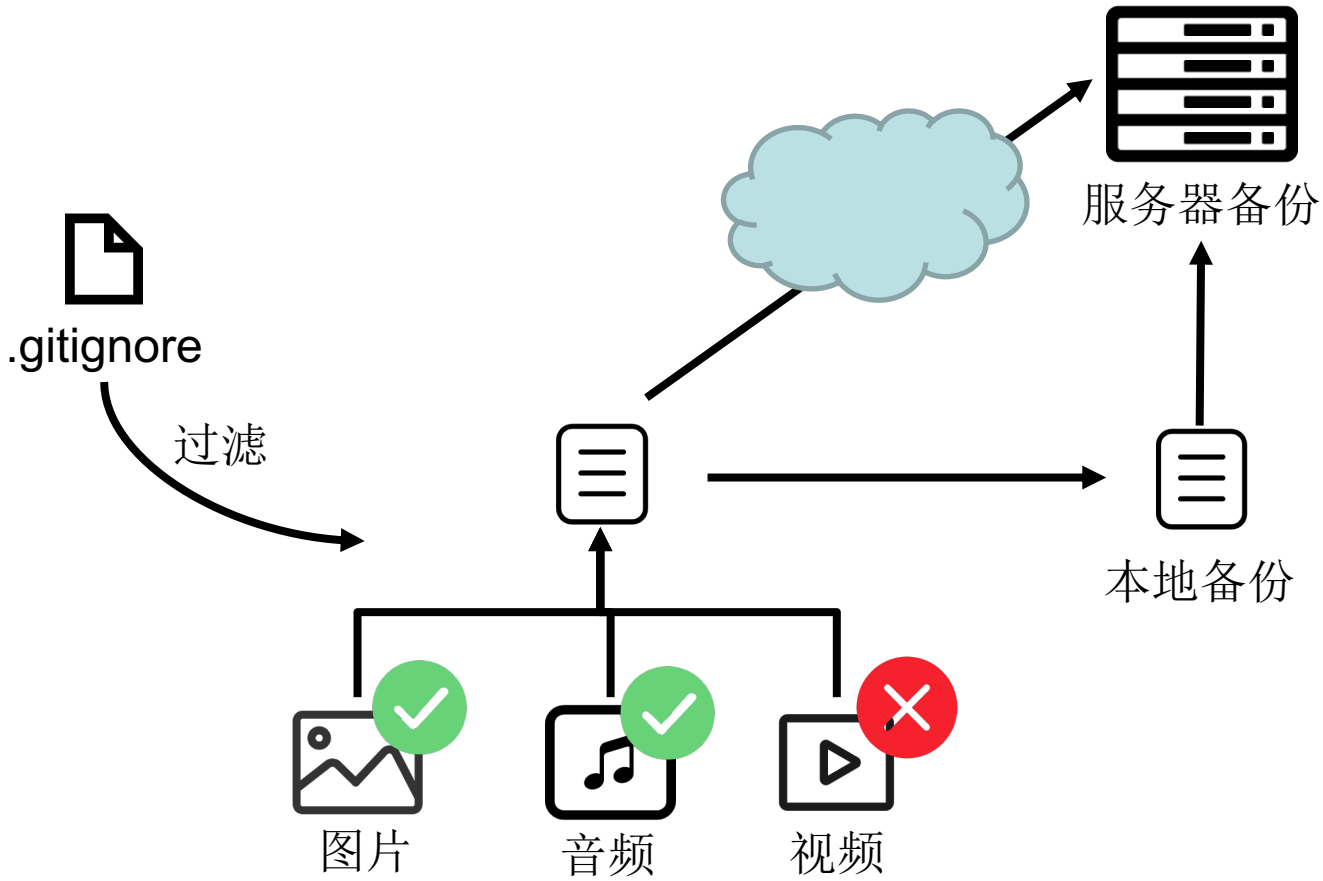
涉及到的系统API（部分）：

inotify_init
inotify_init1
inotify_add_watch
inotify_rm_watch
read
close

注意：对于目录监控，仅部分Linux内核支持递归监控其中的文件和子目录。如当前内核不支持递归监控，则需要手动逐层添加监控。

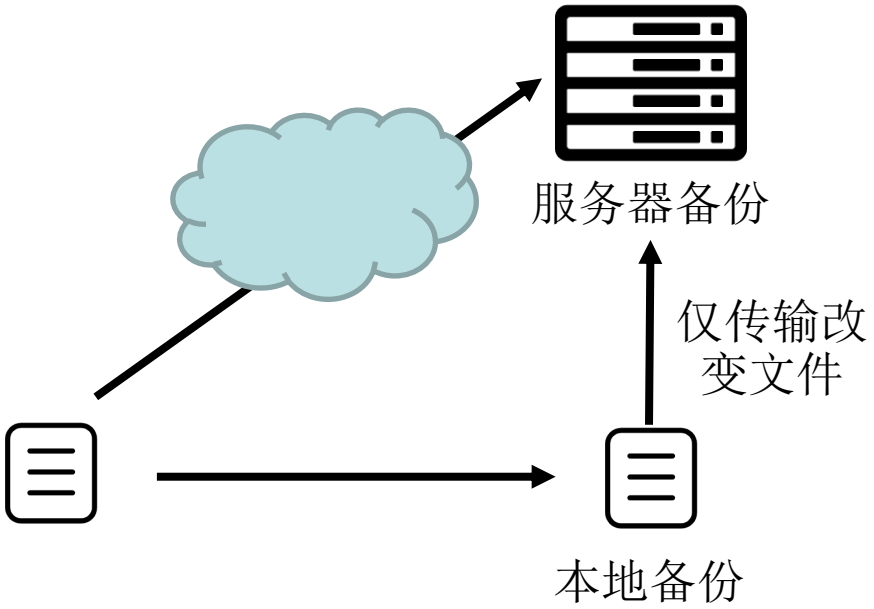
自定义备份

可以建立一个类似.gitignore的备份过滤文件，仅备份指定的文件类型或路径。



网盘备份

在**服务器备份、增量备份**基础上，具有**登录功能，服务器文件上传、浏览、下载、删除功能**（类似于百度网盘）

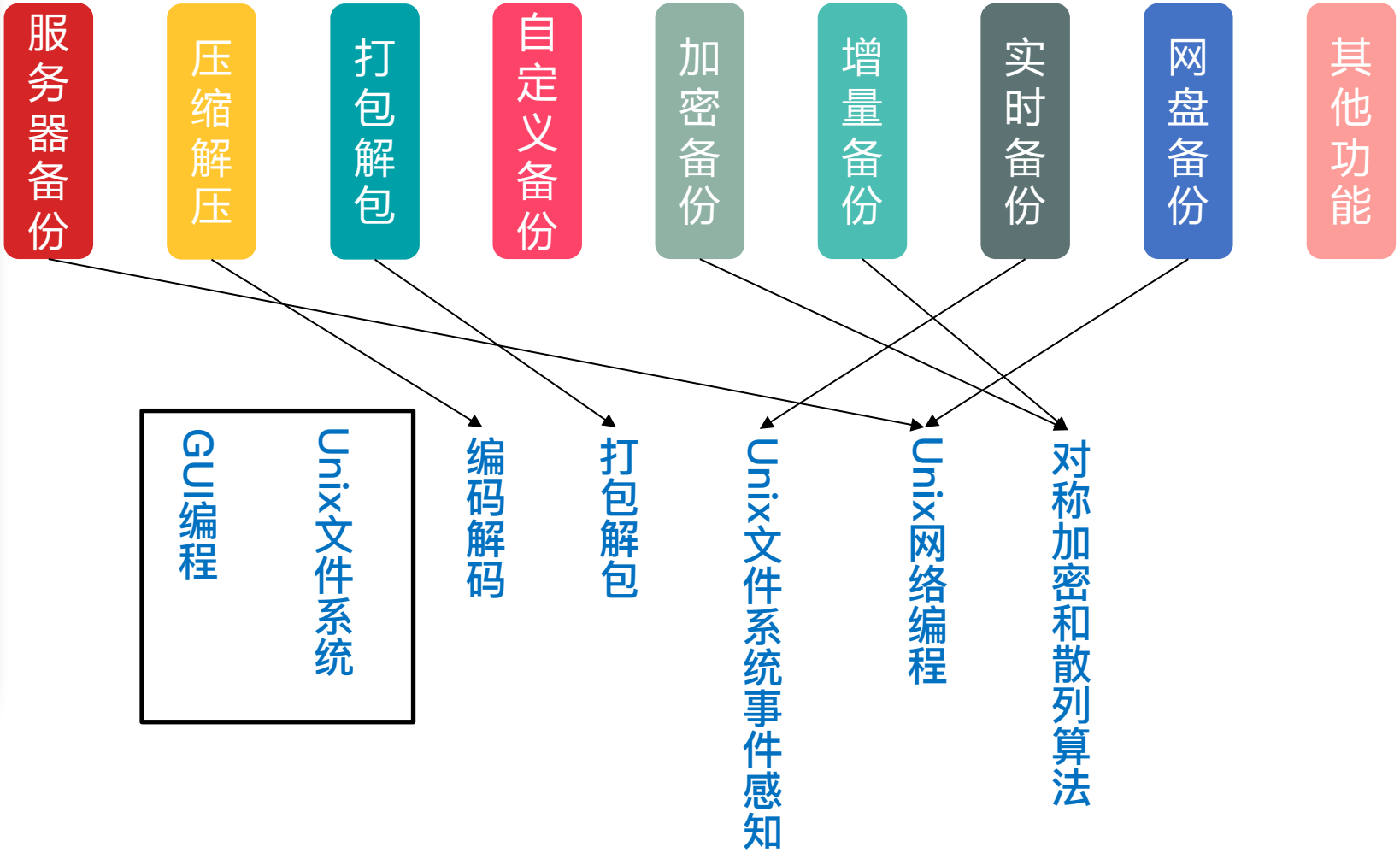


网盘备份

1

项目概述与需求分析

在**服务器备份、增量备份**基础上，具有**登录功能，服务器文件上传、浏览、下载、删除功能**（类似于百度网盘）



实验环节

1. 回顾ppt，将讲解的技术环节进行简单实践（可上外网查阅资料）
2. 组队
3. 组内讨论，确定需求

分组表

【腾讯文档】软件开发综合实验2021-2022-1（肖逸飞班）学生分组
https://docs.qq.com/sheet/DUlp0Q1NYV1JkcVNs

软件开发综合实验1班QQ群