EVENODD

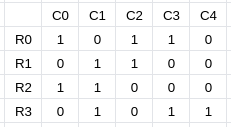
实现一个由EVENODD编码的编码器。本文档分为以下六个部分：

1. EVENODD介绍
2. 功能实现
3. 代码包说明
4. 作品提交
5. 评价标准
6. 参考文献

**EVENODD介绍：**

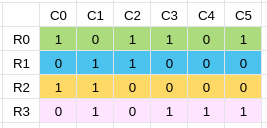
EVENODD[1]由一个质数p定义。它可以将一个规模为(p-1)\*p的阵列经过编码成为一个规模为(p-1)\*(p+2)的阵列。它可以容任意的两错，即任意一列或者两列的数据丢失，都可以由剩余数据进行恢复。我们称前p列为“数据列” (data column)，后两列为“校验列” (parity column)。其中，第0个校验列我们称为“行校验列”(row parity)，第1个校验列我们称为“对角线校验列” (diagonal parity)。

下面，我们以p=5为例说明EVENODD的编码过程。如图一所示为一个4\*5的数据阵列，阵列中的每个元素为一个比特。



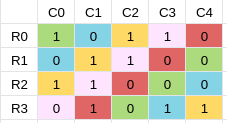
图一：4\*5的数据列。Ri表示第i行，Ci表示第i列。

行校验列的计算由同一行所在的数据列的数据经过异或计算所得。例如，如图二所示，行校验列C5的第0行结果为1+0+1+1+0 = 1。



图二：行校验列的计算

对角线校验列的计算较为复杂。首先我们需要理清位于同一对角线的数据。对于坐标为(Ri, Cj)的数据，它所在的对角线的编号为 (i+j)%p 。如图三所示，标记为相同颜色的数据位于同一对角线上

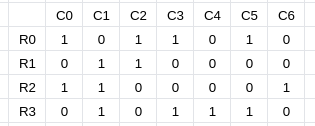


图三：对角线

计算对角线校验列之前，我们首先计算对角线编号为(p-1)上的数据的异或值，我们记为syndrome。例如，在我们的例子中，syndrome 为图三中标记为红色的数据的异或值，1+0+0+0 = 1。

计算对角线校验列第i行的数据时，我们找到编号为i的对角线上的数据，计算他们的异或值，并将结果与syndrome进行异或，最终得到的值即为所求。例如，要求对角线校验列第0行的数据，我们找到编号为0的对角线，即图三中标记为绿色的对角线，计算对角线数据的异或值1+0+0+0=1，然后将其与syndrome进行异或，所以最终结果为1+1 = 0。

那么，经过编码后，我们得到如图四所示的4\*7的阵列



图四：EVENODD(p=5)的例子

EVENODD可以至多容两个错。请参赛者根据EVENODD论文[1]学习它的修复算法。

**功能实现：**

1. 写数据
2. 读数据
3. 数据修复

下面，我们依次介绍这三种功能：

**功能一: 写数据**

命令：$> ./evenodd write <file\_name> <p>

参数：

<file\_name>: 输入文件的路径。

<p>: EVENODD的参数p，为一个质数。

例子：$> ./evenodd write /path/to/file 5

要求：

1. 请参赛者根据参数<p> 检查本地数据文件夹，如果数据文件夹不存在，那么请新建数据文件夹，前缀为"disk\_"。例如，在我们的例子中，p=5，那么我们检查数据文件夹"disk\_0", "disk\_1", ..., "disk\_6"是否存在，若不存在，则需要新建数据文件夹；否则，沿用已有的数据文件夹。
2. 请参赛者对数据文件进行EVENODD编码。编码过后，产生p+2个数据块，分别存储在p个不同的文件夹下。例如，在我们的例子中，p=5，那么经过编码后会产生7个块，分别存在"disk\_0", "disk\_1", ... "disk\_6", 这7个文件夹下。

其他：

1. 输入文件是一个任意大小的文件。
2. 请自行对文件进行划分。
3. 请自行对划分后的数据块进行命名。

**功能二：读数据**

命令：$> ./evenodd read <file\_name> <save\_as>

参数:

<file\_name>: 所读文件的名字。

<save\_as>: 读出来的文件存为“save\_as”。

例子：$> ./evenodd read /path/to/file tmp\_file

要求：

1. 将文件内容正确读出并存在本地文件系统中。例如，在我们的例子中，读出“/path/to/file”, 并将它在本地文件系统中存储为"tmp\_file"。
2. 支持在至多两个数据文件夹损坏的情况下，将数据正常读出。
3. 若要读的文件并不存在，那么请打印出一行出错信息：“File does not exist！”。
4. 若要读的文件中丢失块的个数多于两个，那么请打印出一行出错信息：“File corrupted!”。

**功能三：数据修复**

命令：$> ./evenodd repair <number\_erasures> <idx> ...

参数:

<number\_erasures>: 损坏文件夹的数量。

<idx>: 损坏数据文件夹的编号。

例子: $> ./evenodd repair 2 0 1

要求：

1. 将存储于原来数据文件夹的数据进行修复，且修复出的数据与原来存储与数据相同。
2. 如果损坏数据文件夹的数量大于2，那么打印一行出错信息“Too many corruptions!”。

**代码包说明：**

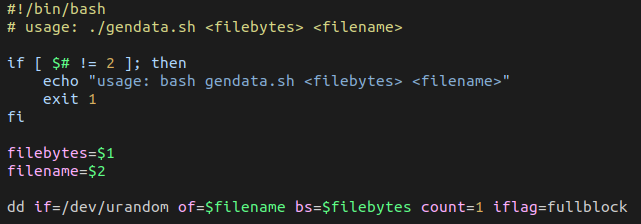
本次比赛提供的代码包包含一下内容：

1. 一个生成测试数据的脚本“gendata.sh”
2. 一个c程序框架“evenodd.c”
3. 一个编译脚本“compile.sh”

下面对这三个文件进行说明。

**gendata.sh:**

脚本内容如图五所示：



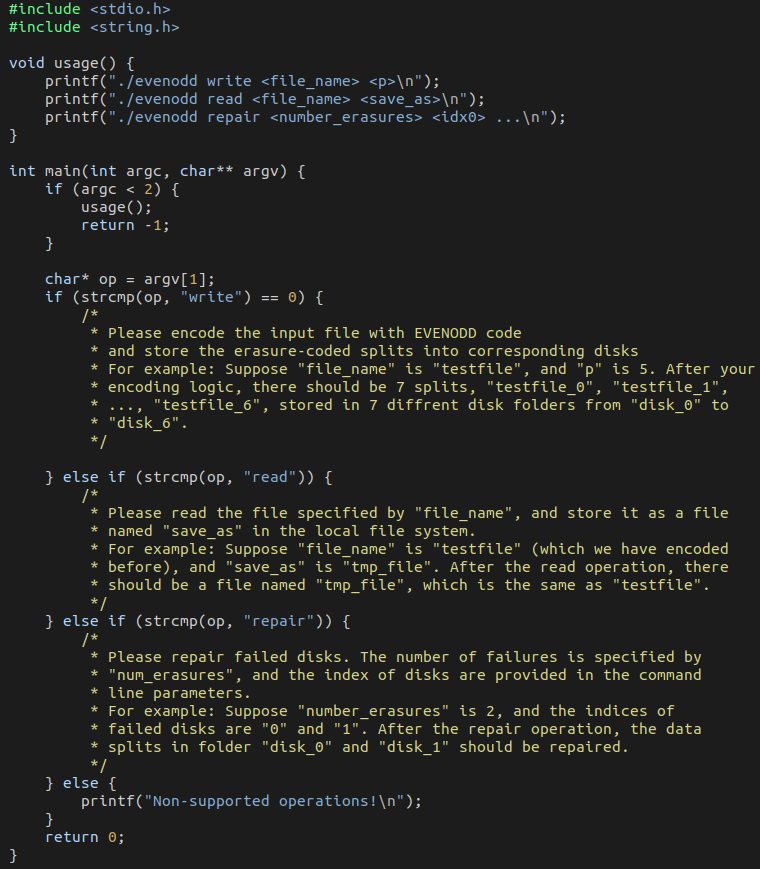
图五：gendata.sh

参赛者可以运行该脚本生成任意大小的文件用于测试。例如，下面的命令可以生成一个大小为5个字节，名字为“testfile”的一个文件。

$> ./gendata 5 testfile

**evenodd.c:**

本次比赛提供了一个c程序框架”evenodd.c"。其内容如下：



图六：evenodd.c

请参赛者以evenodd.c为基础进行拓展，分别实现三个功能，并做性能优化。

**compile.sh:**

代码包中提供的编译脚本compile.sh如图七所示：

descript

图七：compile.sh

运行编译脚本`./compile.sh`即可生成一个可执行文件“evenodd”，用于测试各项功能。

**作品提交：**

请参赛者将"evenodd.c"以及编译脚本"compile.sh"进行打包提交。打包后文件格式可以为“zip”, “tar”, 或者“tar.gz”。

**评价标准：**

1. 比赛将从“正确性“和"性能"两个方面对参赛作品进行评价。
2. 我们将运行“compile.sh”编译参赛者提交的作品，并挑选不同大小的文件和不同的EVENODD参数p进行测试。
3. 在2的基础上，我们将测试每个操作的时间，以评价参赛者程序的性能。
4. 所有参赛者的作品将在同一个平台进行最终测试。该平台为一个干净的ubuntu18.04的机器。参赛者仅可以使用ubuntu 18.04自带的系统库，不可使用其他的第三方库。否则，测试平台将无法正常运行参赛者提交的作品。
   1. 在初赛作品提交截止日期之前，华为云将为所有参赛队伍提供为期一周的测试机器。参赛队伍可以在该机器上测试正确性，以及在有限资源下的编解码性能。该机器为一个2核vcpu，4G内存的ubuntu18.04服务器。
   2. 组委会最终的测试平台提供与华为云相同的测试环境，并为所有参赛队伍提供三次测试机会。
5. 组委会将在所有选择本赛题的队伍中选出至多15支队伍进入决赛环节。

参考文献

【1】M. Blaum, J. Brady, J. Bruck and Jai Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," in *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 192-202, Feb. 1995, doi: 10.1109/12.364531.