天津大学本科生实验报告专用纸

学院 软件学院 年级 17 级 专业 软件工程 班级 1 姓名 陈沛圻 学号 3017218052
课程名称 程序应用实践 3 实验日期 2018 年 5 月 31 日 成绩

一、实验目的

编写一个压缩软件，选择两种压缩算法（自选），将用户提交的文件实现压缩并提示用户压缩率并提示用户按压缩率高的算法压缩；该软件还可通过文件格式识别文件是否是本软件压缩并按压缩时的算法解压。

二、实验内容

独立开发压缩软件 “ZIPPEIQ 1.3.0”，版权所有

实现支持三种压缩格式的文件压缩与解压缩：

ZippeiQ 1.3.0 支持 DEFLATE 压缩格式：

使用 Deflate 算法，Deflate 是同时使用了 LZ77 算法与哈夫曼编码（Huffman Coding）的一个无损数据压缩算法，gzip 压缩格式的实现算法也是 DEFLATE，只是在 deflate 格式上增加了文件头和文件尾

ZippeiQ 1.3.0 支持 LZ4 压缩格式：

使用 LZ4 算法：LZ4 是一种无损数据压缩算法，着重于压缩和解压缩速度更多

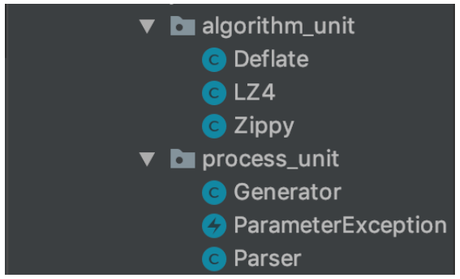
ZippeiQ 1.3.0 支持 SNAPPY 压缩格式：

使用 Snappy 算法：Snappy（以前称 Zippy）是 Google 基于 LZ77 的思路用 C++语言编写的快速数据压缩与解压程序库，并在 2011 年开源。它的目标并非最大压缩率或与其他压缩程序库的兼容性，而是非常高的速度和合理的压缩率。

三、实验步骤

zippeiQ 1.3.0 共分为两块处理单元：

压缩算法计算单元(algorithm_unit)，和压缩解压缩时的文件检验，生成单元(process_unit)



压缩算法计算单元（algorithm_unit）中三种算法的实现：

DEFLATE 算法实现：

jdk 中对 zlib 压缩库提供了支持，压缩类 Deflater 和解压类 Inflater，Deflater 和 Inflater 都提供了 native 方法，直接使用 jdk 提供的压缩类 Deflater 和解压类 Inflater，可以指定算法的压缩级别，这样你可以在压缩时间和输出文件大小上进行平衡。可选的级别有 0（不压缩），以及 1(快速压缩)到 9（慢速压缩），这里使用的是以速度为优先。

代码如下：

```
public class Deflate {
   public static byte[] compress(byte input[]) {
      ByteArrayOutputStream bos = new ByteArrayOutputStream();
      Deflater compressor = new Deflater(1);
      try {
         compressor.setInput(input);
         compressor.finish();
         final byte[] buf = new byte[2048];
         while (!compressor.finished()) {
            int count = compressor.deflate(buf);
            bos.write(buf, 0, count);
         }
      } finally {
         compressor.end();
      }
      return bos.toByteArray();
   }
```

```
public static byte[] uncompress(byte[] input) throws DataFormatException {
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    Inflater decompressor = new Inflater();
    try {
        decompressor.setInput(input);
        final byte[] buf = new byte[2048];
        while (!decompressor.finished()) {
            int count = decompressor.inflate(buf);
            bos.write(buf, 0, count);
        }
    } finally {
        decompressor.end();
    }
    return bos.toByteArray();
}
```

压缩后文件扩展名为.deflate

LZ4 算法实现：

maven 引入第三方库：

```
<dependency>
    <groupId>net.jpountz.lz4</groupId>
    <artifactId>lz4</artifactId>
    <version>1.3.0</version>
</dependency>
```

具体代码实现：

```
public static byte[] compress(byte srcBytes[]) throws IOException {
    LZ4Factory factory = LZ4Factory.fastestInstance();
    ByteArrayOutputStream byteOutput = new ByteArrayOutputStream();
    LZ4Compressor compressor = factory.fastCompressor();
    LZ4BlockOutputStream compressedOutput = new LZ4BlockOutputStream(
          byteOutput, 2048, compressor);
    compressedOutput.write(srcBytes);
    compressedOutput.close();
    return byteOutput.toByteArray();
}
```

```
public static byte[] uncompress(byte[] bytes) throws IOException {
    LZ4Factory factory = LZ4Factory.fastestInstance();
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    LZ4FastDecompressor decompresser = factory.fastDecompressor();
    LZ4BlockInputStream lzis = new LZ4BlockInputStream(
          new ByteArrayInputStream(bytes), decompresser);
    int count;
    byte[] buffer = new byte[2048];
    while ((count = lzis.read(buffer)) != -1) {
        baos.write(buffer, 0, count);
    }
    lzis.close();
    return baos.toByteArray();
}
```

压缩后文件扩展名为.lz4

SNAPPY 算法实现：

maven 引入第三方库：

```
<dependency>
    <groupId>org.xerial.snappy</groupId>
    <artifactId>snappy-java</artifactId>
    <version>1.1.2.5</version>
</dependency>
```

具体代码实现：

```
public static byte[] compress(byte srcBytes[]) throws IOException {
    return  Snappy.compress(srcBytes);
}

public static byte[] uncompress(byte[] bytes) throws IOException {
    return Snappy.uncompress(bytes);
}
```

压缩后文件扩展名为. snappy

实现自己的报错机制，创立类 ParameterException，供软件运行时遇错抛出，代码如下：

```
public class ParameterException extends Exception {
    String msg;
    public ParameterException(String msg){
        super(msg);
    }
}
```

自行编写 exe 文件，代码过长，见./submit/zippeiQ，此处不多加赘述

实现功能：

查看 ZippeiQ 1.3.0 支持的参数

查看 ZippeiQ 1.3.0 支持版本

快捷下载 ZippeiQ 1.3.0 相关依赖以开始运行

查看 ZippeiQ 1.3.0 为您的当前文件预计的压缩效率和时间，您可以通过 ZippeiQ 1.3.0 的推荐选择您想要的压缩格式
压缩当前文件，可选择压缩格式，以及是否显示压缩效率，时间
解压缩已压缩的文件，根据压缩后缀解压缩，可以选择是否显示解压缩时间

具体参数用法可在 submit 目录下使用命令./zippeiQ -H 查看：

```
[➜ submit ./zippeiQ -H                                                    ]
Usage: ./zippeiQ [OPTIONS 1] [OPTIONS 2] [OPTIONS 3] [OPTIONS 4] [filePath]
Options and variables: [defaults in brackets]
 -H, --help          display help info .
 -V, --version          output version information and exit .
 -P, --plan          show all compression plans .
 -I, --install          install zippeiQ .
                        (default in version 1.3.1)
 -C, --compress          compress the file .
                         (default to chose the highest compression rate format)
 -U, --uncompress          compress the file according to the compressed format .

 -T, --time          show the compression/uncompression time .
                     (disable by default)
Options in compression: [used in compress]
 -D, --deflate          use the deflate algorithm to compress files .
 -L, --lz4          use the lz4 algorithm to compress files .
 -S, --snappy          use the snappy algorithm to compress files .
 -R, --rate          show the compression rate .
                     (disable by default, used along with -C --compress)
```

四、实验分析

查看您电脑中当前的 ZippeiQ 版本，执行压缩命令前在/submit 目录下运行命令./zippeiQ -I

或者./zippeiQ - install 下载相关依赖

```
[➜ submit ./zippeiQ --install                                             ]
 /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java -Dmav
en.multiModuleProjectDirectory=/src
  -Dmaven.home=/Applications/IntelliJ IDEA.app/Contents/plugins/maven/lib/maven3
  -Dclassworlds.conf=/Applications/IntelliJ IDEA.app/Contents/plugins/maven/lib/
maven3/bin/m2.conf
  -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=57645:/App
lications/IntelliJ IDEA.app/Contents/bin
  -Dfile.encoding=UTF-8 -classpath /Applications/IntelliJ IDEA.app/Contents/plug
ins/maven/lib/maven3/boot/plexus-classworlds-2.5.2.jar org.codehaus.classworlds.
Launcher -Didea.version=2018.2.4 install
 [INFO] Scanning for projects...
 [WARNING]
 [WARNING] Some problems were encountered while building the effective model for
 chenpeiqi:compress_and_decompress:jar:1.0-SNAPSHOT
 [WARNING] 'build.plugins.plugin.version' for org.apache.tomcat.maven:tomcat7-ma
ven-plugin is missing. @ line 49, column 21
 [WARNING]
 [INFO]
 [INFO] ------------------------------------------------------------------------
 [INFO] Building compress_and_decompress 1.0-SNAPSHOT
```

能够实现 file 类型文件的压缩，如 txt，word，jpg，png 等

压缩前使用参数-P 查看当前选择的文件在不同压缩算法下的压缩结果，以决定用那种方式进行压缩，此时不生成文件

```
➜  submit ./zippeiQ -P ./testZone/testTxt.txt
文件大小： 80532 bytes, 压缩次数： 2000
Deflate:
压缩率： 70.0%
压缩时间： 1204ms
LZ4:
压缩率： 47.0%
压缩时间： 511ms
Snappy:
压缩率： 59.0%
压缩时间： 438ms

建议用.lz4格式压缩此文件
```

压缩时可使用参数-R -T 选择是否显示压缩时间，压缩效率，可以自行选择压缩格式（若用户不指明压缩格式，则默认选择压缩效率最高的压缩格式），若反复压缩，则进行文件覆盖。

```
➜  submit ./zippeiQ -C -R -T  ./testZone/testTxt.txt
正在用建议格式.lz4格式压缩此文件...
成功压缩文件testTxt.txt为testTxt.txt.lz4
压缩率： 47.0%
压缩时间： 436ms
➜  submit ./zippeiQ -C -R -T  ./testZone/testTxt.txt
正在用建议格式.lz4格式压缩此文件...
成功压缩文件testTxt.txt为testTxt.txt.lz4
压缩率： 47.0%
压缩时间： 444ms
➜  submit ./zippeiQ -C --snappy -R -T  ./testZone/testTxt.txt
正在用.snappy格式压缩此文件...
成功压缩文件testTxt.txt为testTxt.txt.snappy
压缩率： 59.0%
压缩时间： 395ms
➜  submit ./zippeiQ -C -R -T --deflate  ./testZone/testImg.jpg
正在用.deflate格式压缩此文件...
成功压缩文件testImg.jpg为testImg.jpg.deflate
压缩率： 55.0%
压缩时间： 443ms
➜  submit
```

解压缩时 zippeiQ 1.3.0 自动按照后缀名选择对应的压缩算法解压缩，可使用参数-T 选择是否显示解压缩时间

```
➜  submit ./zippeiQ -U -T ./testZone/testTxt.txt.snappy
正在解压testTxt.txt.snappy...
解压成功
解压缩后大小： 80532 bytes
解压缩次数： 2000, 时间： 180ms
```

压缩找不到对应压缩格式，解压缩时找不到支持格式的解压缩算法会向用户报错，zippeiQ 1.3.0 拥有自己的报错提示机制

```
➜  submit ./zippeiQ -U -T ./testZone/testTxt.txt
process_unit.ParameterException: 未知格式文件
        at process_unit.Parser.uncomParse(Parser.java:191)
        at App.main(App.java:123)
```

相关限制：

zippeiQ 1.3.0 压缩（解压缩）后文件放在原文件同级目录下，命名格式为 原文件名+对应压缩算法的后缀，暂不支持自定义文件名及压缩路径

zippeiQ 1.3.0 暂不支持过大内存文件的压缩，如时长过长的mov文件以及帧数过多的ppt

zippeiQ 1.3.0 对pdf格式文件，短视频文件（如mov文件）压缩时间较长，且，压缩效率很低压缩此类文件时建议指明压缩算法，或者不建议进行此类文件的压缩

mov文件以最高效率压缩，压缩前后文件大小对比如下

zippeiQ 1.3.0 对以压缩格式文件（如rar文件）的二次压缩效率为零，且耗时较长

zippeiQ 1.3.0经过测试将压缩次数设定为2000，且暂不支持更改压缩次数

zippeiQ 1.3.0暂不支持查看压缩进度

三种算法的优劣的简单介绍：

deflate 更关注压缩率，压缩和解压缩时间会更长； lz4 以及 snappy 压缩算法，均已压缩速度为优先，压缩率会稍逊一筹。

测试：

目录/submit/testZone下有适宜大小的可用于测试的多种文件类型

2019/6/2