SimpleArchiver
Документация

*** Последна версия на проекта: https://github.com/BateBo-2000/Huffman-Compression

Общо описание

SimpleArchiver е конзолна програма за архивиране и разархивиране на файлове и директории, използваща алгоритъм на Хъфман за компресия. Работи чрез команден ред и позволява създаване на архиви, извличане на съдържание, проверка за повреда и извеждане на информация за архивираните обекти, както и бърз начин за промяна. Има собствен бинарен формат. Побитово четене и писане. Разчита на Хъфман компресия.

Архитектура

Архитектурата на проекта е MVC с Command design pattern.

Модела се състои от:
ArchiveMaster - receiver,
ArchiveReader - за четене от архив,
ArchiveWriter - за създаване и промяна на архив,
BitReader, BitWriter - помощни на Хъфман кодера.
 и най-вече HuffmanCompressor - компресия и декомпресия.

View е реализиран чрез ConsoleInterface реализиращ UserInterface, но програмата навсякъде използва UserInterface, за лесно надграждане с графичен такъв.

Controller:
Приема входа от потребителя, намира съответната команда и `и подава съответните аргументи. Всяка команда се грижи сама за входа и коректността си. Всички те използват общ интерфейс и се изпълняват и съхраняват в invoker. Добавянето на нова команда или промяната на съществуваща такава е направено максимално улеснено.

Поддържани команди

zip - Създава архив или добавя файлове/директории към архив.

Формат:
zip archive files...

Поведение:

- създава празен архив ако не съществува такъв на посоченото място
- добавя компресирани файловете
- празни директории се записват
- всеки файл се компресира отделно
- може да добавя безкрайно файлове към архива
- Отказва да добавя съществуващи файлове

unzip - Разархивира съдържанието в директория.

Формат:
unzip archive outDir

Поведение:

- възстановява структурата
- създава нужните директории
- декомпресира всеки файл по отделно
- Ако не се подаде аргумент outDir то всичко се декомпресира в root-а

Info - Извежда информация за архива.

Формат:
info archive

Показва:

- имена на записи
- тип (файл/директория)
- оригинален размер
- компресиран размер
- процент на компресия

Check - Проверява целостта на архива.

Формат:
check archive

За всеки файл:

- прочита компресирания payload

- изчислява hash
- сравнява със записания hash

Несъвпадение или неуспешно четене означава повреда.

update - Командата е дефинирана, но не е реализирана.
При извикване връща съобщение “not implemented”.
Част от функционалността и се извършва от zip.

Няма предоставен начин за махане на файлове от архива за сега.

Архивен формат

Архивът е бинарен файл със следната структура:
Entry записи + индекс в края.
Всеки запис съдържа:
LocalHeader + име + компресиран payload.
В края на архива има:

- MasterHeader — списък с offsets към всички записи
- Footer — offset към MasterHeader

Това позволява:

- бърз достъп до индекс
- без сканиране на целия файл
- append операции

Local Header съдържа:

- magic число - за разпознаване на начало
- флаг за директория
- оригинален размер
- компресиран размер
- hash на payload
- дължина на името

След header-а се записват:
име + payload

Master Header съдържа:

- magic число
- брой записи
- Списък offsets към всеки LocalHeader

Footer съдържа:

- magic число
- offset към MasterHeader
- Чете се от края на файла.

Компресия — Huffman

Използван е алгоритъм на Хъфман:

Стъпки:

- броене на честоти
- построяване на дърво
- генериране на кодове
- битово кодиране
- запис чрез BitWriter

Декомпресия:

- възстановяване на дърво
- битово четене
- декодиране

Сложността на компресията е O(nlogn)
Реализиран е чрез статичен heap - използван като priority queue, понеже се търси по приоритет. И тук и на много места в проекта е използван std::vector заради бързото добавяне в края и възможността за индексиране. Речника на алгоритъма се използва също вектор с булеви за да позволи на програмата да работи с произволни данни, като кодовете могат да нарастват без ограничение, а честотите могат да нарастват до на практика (на теория 2^64) безкрайност.

Проверка за повреда

За всеки payload се пази hash стойност.

При проверка:

- payload се прочита
- пресмята се hash
- сравнява се със записания в LocalHeader

Разлика означава повреден запис.

Памет и производителност

- архивът не се зарежда изцяло в RAM
- индексът (master) е малък и се държи в памет
- payload се чете наведнъж
- append добавя в края без пренаписване
- unzip работи последователно

Ограничения

- update не е реализиран (няма начин да се премахва файл, но zip може да добавя)
- hash е std::hash (не криптографски) - използва се за проверка за повреда на payload
- wildcard matching не е вътрешно реализиран.
- компресорът зарежда всеки отделен файл в паметта (по отделно), те изисква се всеки от файловете да може да бъде побран в паметта по отделно.

Тестов сценарий

Има тестов сценарии в папка test.
За целта тя съдържа:

test
- testFolder
- - testFile1.txt
- - testFile2.txt
- - photo.jpg
- - testTestTestFolder
- - - testFile4.txt

След прилагане на командите от файла Tests.txt:

```
help
zip test\arch.zip test\testFolder
zip test\arch.zip test\testFolder\testFile1.txt
info test\arch.zip
check test\arch.zip
unzip test\arch.zip test\results
exit
```

се получава резултата описан по надолу във файла.
В конзолният интерфейс:

```
SimpleArchiver ready. Type 'help' to list commands.
> help
Available commands:
  - exit
  - help
  - check
  - info
  - unzip
  - update
  - zip

To get help for a command add it as an arg: help <cmd name>
> zip test\arch.zip test\testFolder
Failed to append next file to Archvie.Entry already exists in
archive: test/testFolder/
Entry already exists in archive: test/testFolder/
> zip test\arch.zip test\testFolder\testFile1.txt
Failed to append next file to Archvie.Entry already exists in
archive: test/testFolder/testFile1.txt
Entry already exists in archive: test/testFolder/testFile1.txt
> info test\arch.zip
Archive: test\arch.zip
Entries: 6

[DIR ] test/testFolder/
  original:   0 bytes
  compressed: 0 bytes

[FILE] test/testFolder/photo.jpg
  original:   15262 bytes
  compressed: 17057 bytes
  ratio:      111.761%
```

```
[FILE] test/testFolder/testFile1.txt
  original:   840 bytes
  compressed: 2301 bytes
  ratio:      273.929%

[FILE] test/testFolder/testFile2.txt
  original:   86562 bytes
  compressed: 51940 bytes
  ratio:      60.0032%

[DIR ] test/testFolder/testTestTestFolder/
  original:   0 bytes
  compressed: 0 bytes

[FILE] test/testFolder/testTestTestFolder/testFile4.txt
  original:   26998 bytes
  compressed: 14430 bytes
  ratio:      53.4484%


> check test\arch.zip
Check OK: no corrupted entries.

> unzip test\arch.zip test\results
Unzip completed. Files unzipped in test\results
> exit
```

В папката test:
Ще има две нови неща:

- arch.zip - архивният файл. (размер - 85кб)
- results - папка която съдържа точно копие на папката test

Ресурси

Примерите от moodle.
https://www.programiz.com/dsa/huffman-coding
https://www.geeksforgeeks.org/dsa/huffman-coding-greedy-algo-3/
https://www.reddit.com/r/compsci/comments/5xh380/huffman_coding_explanation_and_c_implementation/
https://www.geeksforgeeks.org/cpp/huffman-coding-in-cpp/
https://youtu.be/XLfgeaYHinM?si=KqDUeV9mvAqSDFt6
https://youtu.be/yue6yqhmfSg?si=ziMdAlM2aLtDqzQM

https://youtu.be/JsTptu56GM8?si=bwdVdSkzcQDfMttH
https://youtu.be/8zKgXFJcWOo?si=lr_hpkenopn4Up3d
https://youtu.be/yPlWylq5zwk?si=vfjGJY1Llwg59YaD
https://stackoverflow.com/questions/11815894/how-to-read-write-arbitrary-bits-in-c-c
https://www.reddit.com/r/learnprogramming/comments/12apng/c_reading_bits_from_a_given_address/
https://en.cppreference.com/w/cpp/types/integer.html
https://en.cppreference.com/index.html
https://en.cppreference.com/w/cpp/filesystem.html
https://en.cppreference.com/w/cpp/header/cstring.html
https://en.cppreference.com/w/cpp/utility/hash.html
https://en.cppreference.com/w/cpp/utility/hash.html
https://en.wikipedia.org/wiki/ZIP_(file_format)
https://en.wikipedia.org/wiki/Archive_file
https://en.wikipedia.org/wiki/Huffman_coding