Задание № 3 по практикуму на ЭВМ.

Реализация хэш-функции ГОСТ Р 34.11-2012

Описание:

https://meganorm.ru/Data/525/52502.pdf
https://tools.ietf.org/html/rfc6986

Обратите внимание на последнюю страницу ГОСТ-а с исправлением пункта 8.3 шага 3.4 процедуры вычисления хэш-функции.

Вторая ссылка – английский вариант в виде RFC.

В данном задании требуется реализовать программы для проверки контрольных сумм, используя функции хэширования ГОСТ Р 34.11-2012. Поведение программы должно полностью управляться аргументами командной строки. Для каждого режима должны быть отдельные бинарные файлы: g256sum и g512sum.

Дальнейшее описание приводится для g256sum, которое полностью аналогично для g512sum за исключением длины значения контрольной суммы.

Описание флагов:

```
g256sum [option]... [file]...
g256sum --help
```

- `[file]` – название файла (путь к файлу), для которого требуется посчитать контрольную сумму. Если параметр отсутствует или равен -, читать со стандартного потока ввода.

- `[option]` – один из следующих флагов:

`-b, --binary`	чтение в бинарном режиме.
`-t, --text`	чтение в текстовом режиме (по умолчанию).
`-c, --check`	режим проверки: посчитать значения контрольных сумм и проверить соответствующие файлы.
`--`	интерпретировать все последующие операнды как названия файлов.

  Следующие опции имеют эффект только в режиме проверки:

`--ignore-missing`	игнорировать отсутствующие файлы и не выводить сообщение об их статусе.
`--quiet`	не печатать `OK` для удачно прошедших проверку файлов.
`--status`	не печатать результаты проверок: предполагается использование кода завершения процесса для определения результата.
`--strict`	выход с ненулевым кодом завершения в случае неправильного формата строки.
`-w, --warn`	выводить сообщения о неправильно форматированных строках в `stderr`.

- `--help` – вывести описание флагов в `stdout`.

- [описание дополнительно реализованных флагов]

Формат выходных данных:

Для каждого файла результатом работы является отдельная строка следующего формата: значение хэш-функции в шестнадцетиричном формате, пробел, символ режима (’*’ – для бинарного, ’ ’ – для текстового), название файла. Примеры:

```
00557be5e584fd52a449b16b0251d05d27f94ab76cbaa6da890b59d8ef1e159d *RFC-10.1.2
be088b3d9a0664855fe44322b1b509d09ff4007494aece1ce4d699b52b20453c  test
```

В режиме проверки, при отсутствии флагов `-w, --warn, --strict`, требуется игнорировать не удовлетворяющие данному формату строки.

Режим проверки

Файлы содержат множество строк вида:

```
be088b3d9a0664855fe44322b1b509d09ff4007494aece1ce4d699b52b20453c  filename1
402c2e4cca2a51e07b2032c92b9bf1f3b30417b927f06f344ec50df47ca1d8d6 *filename2
```

Для каждой строки, каждого файла, программа читает файл по указанному `filename` и считает его контрольную сумму. Если вычисленное значение не совпадает с записанным в той же строке, файл считается не прошедшим проверку. Иначе проверка пройдена.

По умолчанию, для каждой строки в корректном формате, выводится одна стока результата `filename: status`, где `status` – `OK`, `FAILED` или `FAILED open or read` (при отсутствии флага `--ignore-missing`). После того, как были выполнены все проверки, если произошла хотя бы одна ошибка, выводятся предупреждения в `stderr`. Примеры предупреждений:

```
1 line is improperly formatted
1 listed file could not be read
1 computed checksum did NOT match
```

Использование флага `--status` отключает этот вывод. Если хотя бы один указанный файл не был открыт или прочитан (при отсутствии флага `--ignore-missing`), или любая из корректных строк содержит не совпадающую контрольную сумму, или если нет ни одно правильно форматированной строки, программа завершается с ненулевым кодом. Иначе программа завершается успешно.

Примеры работы программы:

1. Результат работы при наличии file1 и отсутствии file3:

```
g256sum file1 file3
stdout > e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  file1
stderr > g256sum: file3: No such file or directory
```

2. Пример работы с флагом `--` и вывода контрольной суммы для бинарного режима:

```
g256sum -b -- -c --strict file1
stderr > g256sum: -c: No such file or directory
stderr > g256sum: --strict: No such file or directory
stdout > e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 *file1
```

3. Три строки имеют неправильный формат, одна строка содержит файл с правильной контрольной суммой:

```
g256sum -c 3form1check.cksum
stdout > file1: OK
stderr > g256sum: WARNING: 3 lines are improperly formatted
```

4. Результат работы для пустого файла:

```
g256sum -c empty.cksum
stderr > g256sum: 'empty.cksum': no properly formatted g256 checksum lines found
```

5. Обе строки имеют неправильный формат:

```
g256sum -c -w 2form.cksum
stderr > g256sum: '2form.cksum': 1: improperly formatted g256 checksum line
stderr > g256sum: '2form.cksum': 2: improperly formatted g256 checksum line
stderr > g256sum: '2form.cksum': no properly formatted g256 checksum lines found
```

6. Файл file1 присутствует и его контрольная сумма совпадает с представленной, file2 - отсутствует:

```
g256sum -c 1check1miss.cksum
stdout > file1: OK
stderr > g256sum: file2: No such file or directory
stdout > file2: FAILED open or read
stderr > g256sum: WARNING: 1 listed file could not be read
```

7. Два примера работы с флагом `--ignore-missing`:

```
g256sum -c --ignore-missing 1miss.cksum
stderr > g256sum: '1miss.cksum': no file was verified
```

```
g256sum -c --ignore-missing 1check1miss.cksum
stdout > file1: OK
```

8. Пример проверки двух файлов, контрольная сумма одного из которых не совпадает:

```
g256sum -c 1check1wrong.cksum
stdout > file1: OK
stdout > file2: FAILED
stderr > g256sum: WARNING: 1 computed checksum did NOT match
```

9. Результат работы при наличии флага `--status`. Код завершения процесса – 1, так как отсутствует file3:

```
g256sum -c --status 1check1miss.cksum
stderr > g256sum: file3: No such file or directory
```

10. Результат работы при наличии флагов `--status` и `--ignore-missing`. Код завершения процесса – 0, так как игнорируется отсутствие file3:

```
g256sum -c --status 1check1miss.cksum
```

Требования к функциональности:

1. Все опции, за исключением `--`, могут передаваться в произвольном порядке.
2. Если переданы неправильные аргументы, вывести сообщение об ошибке (с указанием неправильного аргумента) и напечатать help в `stderr`.
3. Если вы реализуете дополнительную печать, она должна включаться отдельным флагом, с выводом в `stderr`.
4. Все ошибки печатаются в `stderr`.

Требования к коду:

1. Программа должна быть написана на языке C/C++.
2. Код не должен быть скопирован у другого студента.
3. Стремиться к тому, чтобы каждая функция в коде не превышала 25 строк.
4. Стремиться к тому, чтобы каждая строка в коде не превышала 80 символов.
5. Функции и переменные должны иметь осмысленные имена.
6. Компиляция производится gcc версии 4.9+ (лучше 8.1+), с флагами -Wall -O2.
7. Реализация должна быть кроссплатформенной.
8. В архиве с программой должен быть Makefile.

Формат приема заданий:

1. Задания отсылаются на почту is.cmc.2018@yandex.ru.
2. Тема письма в формате “Задание_1_|номер задачи|_Ф_И_О” для задания 1.
3. Тема письма в формате “Задание_|номер|_Ф_И_О” для заданий 2,3.
4. В случае наличия замечаний/ошибок аспирант отправляет комментарий. Процесс повторяется до тех пор, пока аспирант не сообщит, что замечаний больше нет.

5. После устранения всех замечаний, назначается встреча в рамках пары для обсуждения деталей реализации, после чего выставляется оценка, согласно пункту “Формирование оценок”.

Сроки выдачи заданий:

- Задание 1: 3 сентября.
- Задание 2: 1 октября.
- Задание 3: 8 ноября.

Формирование оценок:

- Оценка по заданию выставляется после очной встречи и обсуждения деталей реализации задания.
- Очная встреча назначается после устранения всех замечаний по заданию.
- Каждое задание оценивается в 5 баллов.
- Общая оценка вычисляется как среднее по 3 заданиям.
- Предварительная оценка на момент очной встречи по заданию определяется следующим образом:
  - Устранение всех замечаний в срок 5 недель с момента выдачи задания без учета времени проверки – **5**.
  - Устранение всех замечаний в срок (5, 6] недель с момента выдачи задания без учета времени проверки – **4**.
  - Отправка задания и устранение всех замечаний вне сроков описанных выше – **3**.
  - Невыполнение задания, наличие неустраненных замечаний к зачету – **2**.
- Пояснение фразы *“с момента выдачи задания без учета времени проверки”*:

  Время выделенное на выполнение вами задания не зависит от того, как долго мы его будем проверять.

  То есть, если вы сдадите задание через неделю после его выдачи, а мы будем его проверять 3 недели и после этого вышлем замечания, у вас все еще будет 4 недели на их исправление. Это относится ко всем заданиям, не только ко второму.
- Оценка по заданию по итогам очной встречи может быть изменена, исходя из субъективной оценки аспиранта о выполнении задания студентом, как в меньшую, так и в большую сторону.

Задать вопросы, получить актуальную и оперативную информацию можно в Telegram-чате: https://t.me/joinchat/DpzKQxJXQ_xZAYlYyaYoPQ.