ДЗ 2: ls

Дата выдачи: 01.10.24
Дедлайн сдачи: 29.10.24

Описание

Требуется реализовать на языке Си программу с функционалом команды **`ls`**, но с обработкой только некоторого подмножества аргументов командной строки, допускаемых командой.

Максимальный балл за задание: **10**.

Функциональные требования

- Программа должна обрабатывать следующие аргументы командной строки
  - **`--all (-a)`**: включить в выдачу файлы и директории, имена которых начинаются с точки
  - **`--almost-all (-A)`**: исключить из выдачи директории **`.`** и **`..`**
  - **`--ignore-backups (-B)`**: исключить из выдачи файлы, имена которых завершаются символом **`~`**
  - **`--directory (-d)`**: не заходить в директории, а печатать информацию о них так же, как о регулярных файлах
  - **`--dereference (-L)`**: проходить по символическим ссылкам и печатать информацию о файлах, на которые они ссылаются, а не о самих ссылках
  - **`--human-readable (-h)`**: выводить размер файлов (в длинном формате) в человеко-читаемом виде (использовать двоичные сокращения)
  - **`--si`**: использовать десятичные сокращения для размера файлов
  - **`--size (-s)`**: выводить размер файлов в блоках (в длинном формате, в начале выдачи)
  - **`--sort <column>`**: сортировать файлы в выдаче
    - **`column=none`**: не сортировать, т.е. использовать исходный порядок перечисления файлов в директории
    - **`column=size`**: сортировать по убыванию размера
    - **`column=time`**: сортировать по убыванию даты последней модификации
  - **`--reverse (-r)`**: изменить порядок сортировки на противоположный
  - **`-S`**: сортировать файлы по размеру
  - **`-t`**: сортировать файлы по дате последней модификации
  - **`-U`**: не сортировать файлы
  - **`-l`**: использовать длинный (подробный) формат вывода

Аргументы могут передаваться в произвольном порядке. Если переданы неизвестные или некорректные аргументы командной строки, программа должна сообщать пользователю причину ошибки (в stderr) и завершаться.

Смысл аргументов полностью совпадает с реализацией команды **`ls`**, при необходимости консультируйтесь с её документацией (**`man ls`**).

- Программа должна состоять из нескольких модулей (допустимо добавление любых необходимых заголовочных файлов, а также файлов с реализацией, помимо указанных ниже)
  - **`main.c`**: основной модуль, осуществляющий разбор аргументов командной строки, вызов функции **`ListPaths`** (см. ниже) и обработку ошибок; не все ошибки приводят к досрочному завершению программы!
  - **`ls.c`**: модуль с реализацией функции **`ListPaths`**; за пределами этого модуля должна быть доступна только функция со следующим прототипом

```
ListErrorCode ListPaths(
    const GenericVector* paths,
    const ListArgs* args,
    FILE* out);
```

  - в отдельные модули следует выносить код для работы со строками, путями, информацией о файлах и пр.

- Сборка программы должна осуществляться с помощью Makefile
  - Опции компилятора обязательно должны включать в себя следующие: **`-Wall, -Werror, -std=c11, -O0 / -O2`**
  - Должны быть поддержаны 4 режима сборки: **`release`** (по умолчанию), **`debug`**, **`test`**, **`valgrind`**
  - В режиме **`debug`** сборки выключены оптимизации (**`-O0`**), в режимах **`release`**, **`test`** и **`valgrind`** – включены (**`-O2`**)
  - Режим **`valgrind`** собирает тесты и запускает их командой

```
valgrind --leak-check=full <test-executable>
```

  - Режим **`test`** собирает тесты (вместе с бенчмарками), запускает их, определяет степень покрытия исходного кода тестами, запускает линтер **`clang-tidy`**
  - Тесты должны покрывать все основные краевые случаи использования тестируемых функций

Стилистические требования

- Комбинация styleguide + clang-tidy + **`-Wall -Werror`** остаётся в силе
- Константы можно оформлять в enum'ы (тогда допускается `CAPS_SNAKE_CASE`) и/или в отдельные статические переменные (тогда используйте только венгерскую нотацию: `kMyLovelyConst`)

Дополнительно

Shell globbing

Сейчас, если вашей программе или команде **`ls`** подать на вход регулярное выражение с символами **`*`** или **`?`** (и некоторыми другими), то командная оболочка автоматически развернёт это выражение в отдельные аргументы командной строки для каждого файла, путь до которого удовлетворяет выражению (проверьте!).

Дополнительное задание (до **5** баллов) состоит в собственной реализации globbing'a: если теперь программе передать аргумент с регулярным выражением внутри (но в одинарных кавычках, чтобы отключить globbing shell'a), то программа сама развернёт его в набор подходящих путей. Достаточно реализовать поддержку символов **`*`** и **`?`**.

Использовать функцию **`glob`** нельзя, но поизучать прототип можно :)

Ссылки: тык, тык.