Основы системного программирования

Лабораторная работа № 1.1
Работа с файлами и каталогами

Разработать на языке C для ОС Linux программу, позволяющую выполнять рекурсивный поиск файлов, начиная с указанного каталога, в соответствии с условием из Табл. 3 и вариантом из Табл. 4.

Программа должна представлять собой консольную утилиту, настройка работы которой осуществляется путем передачи аргументов в строке запуска и/или с помощью переменных окружения (`опции` необязательны, аргументы `каталог` и `цель_поиска` — обязательны):

```
lab11abcNXXXXX [опции] каталог цель_поиска
```

Программа должна выполнять рекурсивный поиск файлов, отвечающих критерию, который задается аргументом `цель_поиска` в соответствии с условием из Табл. 3. При обнаружении файла, отвечающего заданным критериям поиска, программа должна вывести в стандартный поток вывода полный путь к этому файлу.

При указании опций -h или -v (или их "длинных" аналогов --help или --version) выполняется вывод информации, заданной опцией, и работа программы завершается. Опции, которые должны поддерживаться программой, приведены в Табл. 1.

При определении переменной окружения `LAB11DEBUG` в стандартный поток ошибок должна выводиться информация о том, что и в каком месте файла нашлось (чтобы было легче понять, почему файл отвечает критериям поиска), а также может выводиться любая дополнительная отладочная информация. Переменные окружения, которые должны поддерживаться программой, приведены в Табл. 2.

Имя программы должно начинаться на lab11, далее должен следовать уникальный для варианта суффикс. Уникальный суффикс составляется из первых букв имени, отчества (если есть) и фамилии студента, выполняющего лабораторную работу. Далее следует номер группы студента. Используются строчные латинские буквы и арабские (в традиционном понимании, т. е. 0..9) цифры. Например, если студента, выполняющего лабораторную, зовут Петр Сергеевич Иванов, его группа — N32451, то имя программы должно быть `lab11psiN32451`.

Проект (исходные коды, заголовочные файлы, Makefile и прочие файлы, необходимые для сборки) должен содержаться в отдельном каталоге с именем, совпадающим с названием программы (lab11abcNXXXXX) и собираться с помощью стандартной утилиты make. Исходные файлы программы на языке C должны компилироваться с помощью gcc. Makefile должен поддерживать как минимум цели all и clean. Если для сборки проекта требуется что-то большее, чем `make all`, или для запуска и проверки проекта требуются какие-либо нетривиальные или неочевидные действия, то инструкции по сборке и запуску проекта следует добавить в файл README.txt в формате plain text и разместить его в каталоге проекта.

Порядок выполнения и сдачи лабораторной работы:

1. Подготовить тестовый набор файлов (отвечающих и неотвечающих критериям поиска), разместить их в тестовом дереве каталогов.
2. Выполнить задание, подготовить все файлы проекта, скомпилировать программу и библиотеку с флагами `-Wall -Wextra -Werror` и устранить все предупреждения и ошибки.
3. Протестировать программу на различных каталогах, убедиться, что ошибок нет, в противном случае вернуться к пункту 2.
4. Выполнить запуск программы (поиск файлов в тестовом дереве каталогов) под управлением valgrind, убедиться, что утечки памяти отсутствуют. Если утечки есть, то сначала устранить их и вернуться к пункту 2. Если утечек нет, то сохранить отчет в файл valgrind.txt и добавить его в каталог проекта.

5. Скомпилировать программу с флагом `-O3`, повторно протестировать программу поиском в различных каталогах. В случае обнаружения ошибок вернуться к пункту 2.
6. Удалить все исполняемые и промежуточные файлы из папки проекта (`make clean`). В архиве должны остаться только файлы *.c, *.h, Makefile, README.txt, valgrind.txt.
7. Заархивировать папку проекта, используя формат tar.gz. (`tar -czvf lab11abcNXXXXX.tar.gz lab11abcNXXXXX/`).
8. Отправить полученный архив на почту преподавателя, который ведет лабораторные (Гирику А.В. на itmo.osp@gmail.com, Горлиной А.В. на gorlina.a.v@mail.ru, Грозову В.А. на va_groz@mail.ru), письмом с темой «*ОСП ЛР11 Фамилия Имя Отчество NXXXXX*»
9. Дождаться ответа по почте или на лабораторном занятии, устранить возможные замечания (повторить с пункта 1).
10. Получить подтверждение от преподавателя, что лабораторная работа выполнена успешно, после чего подготовить отчет в электронной форме (состав отчета см. ниже).
11. Отправить архив с окончательным вариантом проекта и отчетом в формате pdf (не забыть про подпись на первой странице!) на почту преподавателя письмом с темой «*ОСП Отчет по ЛР11 Фамилия Имя Отчество NXXXXX*». Файл отчета должен иметь название NXXXXX_ФамилияИО_ЛР11.pdf.
12. Получить некоторое количество вопросов по отчету от преподавателя и дать на них ответы (а может и не получить, если лабораторная выполнена на хорошем уровне и сомнений в знаниях студента у преподавателя не возникает). Получить от преподавателя подтверждение, что отчет принят.
13. Немного расслабиться и приступить к следующей лабораторной работе :-)

Отчет должен быть подготовлен в формате pdf и содержать:

- правильно оформленную титульную страницу (с подписью студента);
- задание;
- Make-файл;
- отчет valgrind со строчки «HEAP SUMMARY:»
- примеры работы программ (скриншоты);
- исходные тексты программ с комментариями.

Замечание 1. При выполнении лабораторной работы следует использовать функции стандартной библиотеки C и системные вызовы операционной системы. Использовать C++, а следовательно, ввод-вывод в стиле C++ (классы ifstream/ofstream/…), контейнеры и алгоритмы STL (<string>, <vector>, <map>, ...) запрещено.

Замечание 2. В программах должна присутствовать обработка ошибок: в случаях, если пользователь задал неверную комбинацию опций, указал файлы, которые невозможно открыть, и т.д. программа должна выдавать диагностическое сообщение на консоль (в стандартный поток ошибок и/или лог-файл), прежде чем завершиться.

Замечание 3. При обходе дерева каталогов нужно учитывать, что доступ к некоторым файлам и каталогам может завершиться с ошибкой (например, по причине отсутствия прав доступа и т.д.). В таком случае следует вывести сообщение об ошибке в стандартный поток ошибок и продолжить обход. Для упрощения реализации обхода символические ссылки (и аналогичные средства — жесткие ссылки, bind mount'ы и т.д.) можно игнорировать.

Замечание 4. Категорически запрещается использовать статические массивы (с размерами, заданными на этапе компиляции) для любых данных, размер которых зависит от входных данных или условий запуска. Для хранения таких данных необходимо использовать динамическую память и определять объем необходимой памяти в зависимости от ситуации. Статические массивы можно использовать в тех ситуациях, когда известен максимальный размер обрабатываемых данных (и он не превышает размеров стека или максимального размера статических массивов, допускаемого компилятором).

Замечание 5. Информационные сообщения выводятся программой в стандартный поток вывода, сообщения об ошибках — в стандартный поток ошибок. С помощью определения переменной окружения `LAB11DEBUG` можно включить вывод отладочных сообщений программой в стандартный поток ошибок.

Замечание 6. Для обработки опций командной строки рекомендуется использовать функцию getopt_long(). При необходимости список опций можно расширить.

Замечание 7. Несмотря на то, что для компиляции программ необходимо использовать компилятор gcc, использования расширений GNU C желательно по возможности избегать и ориентироваться на использование стандарта C11 или более позднего.

Замечание 8. Программа должна успешно компилироваться и выполняться в ОС Linux с ядром версии >= 5.0. По возможности следует избегать специфичных для Linux и glibc функций и системных вызовов и стремиться к соответствию стандарту POSIX. При желании и возможности рекомендуется проверить работу программы не только на платформе x86_64 GNU/Linux, но и на других POSIX платформах: x86/ARM/... + *BSD/macOS/...

Таблица 1. Опции командной строки, поддерживаемые программой

Опция	Назначение
-v, --version	Вывод версии программы и информации о программе (ФИО исполнителя, номер группы, номер варианта лабораторной).
-h, --help	Вывод справки по опциям.

Таблица 2. Переменные окружения, поддерживаемые программой

Переменная	Назначение
LAB11DEBUG	Включение вывода отладочной информации.

Таблица 3. Формат аргумента цель_поиска

Вариант (условие)	Назначение
Количество букв в **фамилии** студента, выполняющего работу, **нечетное**.	Выполняется поиск заданной последовательности байтов. Аргумент `цель_поиска` имеет формат 0xhh[hh*], где hh — две шестнадцатеричных цифры (число должно начинаться с префикса 0x). *Пример запуска программы*: `./lab11psiN32451 /home 0xc0ffee` (Начиная с каталога `/home`, рекурсивно выполняется поиск файлов, содержащих последовательность байтов 0xc0 (192), 0xff (255) и 0xee (238).)
Количество букв в **фамилии** студента, выполняющего работу, **четное**.	Выполняется поиск заданной последовательности байтов. Аргумент `цель_поиска` имеет формат строки в кодировке UTF-8. *Пример запуска программы*: `./lab11psiN32451 /home "лавандовый раф"` (Начиная с каталога `/home`, рекурсивно выполняется поиск файлов, содержащих последовательность байтов 0xd0 0xbb ('л'), 0xd0 0xb0 ('а'), ..., 0xd1 0x84('ф').)

Таблица 4. Функции, которые необходимо использовать для реализации поиска файлов и каталогов

Вариант (номер)	Функции
1	opendir()/readdir() без рекурсии
2	opendir()/readdir() с рекурсией
3	ftw()

4	nftw()
5	fts_open()/fts_read()/...