Основы системного программирования

Лабораторная работа № 1.2
Использование динамических библиотек

Разработать на языке C для ОС Linux:

- программу, позволяющую выполнять рекурсивный поиск файлов, начиная с указанного каталога, с помощью динамических (разделяемых) библиотек-плагинов (использовать в качестве основы программу, разработанную в лабораторной работе 1.1);
- динамическую библиотеку, реализующую заданный вариантом лабораторной работы из Табл. 4 критерий поиска файлов.

Программа должна представлять собой консольную утилиту, настройка работы которой осуществляется путем передачи аргументов в строке запуска и/или с помощью переменных окружения:

```
lab12abcNXXXXX [опции] [каталог]
```

Программа должна выполнять рекурсивный поиск файлов, отвечающих критериям, которые задаются опциями в командной строке. Доступные критерии поиска (и, соответственно, доступные опции) определяются наличием в заданном каталоге динамических библиотек, расширяющих функциональность программы (далее — плагинов).

При запуске без имени каталога для поиска программа выводит справочную информацию по опциям и доступным в момент запуска плагинам. Поддерживаемые программой опции перечислены в Табл. 2. Выводятся все опции, поддерживаемые плагинами, и их описание. По умолчанию плагины ищутся в том же каталоге, где расположен исполняемый файл программы, а если задана опция `-P`, то в каталоге, указанном в этой опции. В случае запуска программы с несколькими опциями, задающими критерии поиска, эти критерии объединяются логической операцией «И» (то же самое, если задана опция `-A`) или логической операцией «ИЛИ» (если задана опция `-O`). Если задана опция `-N`, то после объединения всех условий поиска по «И» или «ИЛИ», оно меняется на противоположное.

Плагины представляют собой динамические библиотеки в формате ELF с произвольным именем и расширением .so и интерфейсными функциями, перечисленными в Табл. 1. Подробное описание API плагинов содержится в файле plugin_api.h.

При обнаружении файла, отвечающего заданным критериям поиска, в стандартный поток вывода выводится полный путь к этому файлу. При определении переменной окружения `LAB1DEBUG` 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 и библиотекой, приведены в Табл. 3.

Имя программы должно начинаться на lab1, далее должен следовать уникальный для варианта суффикс. Имя библиотеки (плагина) должно начинаться на lib, далее должен следовать уникальный для варианта суффикс и расширение .so. 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 — N3245, то имя программы должно быть `lab1psiN3245`, а имя библиотеки — `libpsiN3245.so`.

Проект (исходные коды, заголовочные файлы, Makefile и прочие файлы, необходимые для сборки) должен содержаться в отдельном каталоге с именем, совпадающим с названием программы (lab1abcNXXXX) и собираться с помощью стандартной утилиты make. 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 и библиотеку с помощью проверочных программ lab1test, lab1call, libavg.so и поиском в тестовом дереве каталогов, 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 и библиотеку с помощью проверочных программ lab1test, lab1call и libavg.so и поиском в тестовом дереве каталогов. В случае обнаружения ошибок или предупреждений 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 (`tar -czvf lab1abcNXXXX.tar.gz lab1abcNXXXX/`).
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 ЛР12 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 ЛР12 Фамилия Имя Отчество NXXXXX*». Файл отчета должен иметь название NXXXXX_ФамилияИО_ЛР12.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 ввод-вывод в стиле C++ (классы ifstream/ofstream/…) запрещено. Использовать 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. Плагинами поддерживаются только длинные опции. Плагин может поддерживать несколько опций (например, в целях отладки). Совпадение имен опций в разных плагинах считается ошибкой.

Замечание 6. Информационные сообщения выводятся программой в стандартный поток вывода, сообщения об ошибках — в стандартный поток ошибок. С помощью определения переменной окружения `LAB1DEBUG` можно включить вывод отладочных сообщений программой и плагинами в стандартный поток ошибок. Рекомендуется предварять все сообщения плагина его названием (чтобы было легче разбираться, каким плагином были выведены сообщения).

Замечание 7. Плагины (в качестве потенциальных плагинов можно рассматривать любые файлы с расширением .so) по умолчанию ищутся в том же каталоге, где расположен исполняемый файл. Обратите внимание, что этот каталог может не совпадать с текущим рабочим каталогом. Опция `-P` позволяет задать каталог с плагинами явно.

Замечание 8. 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 9. В вариантах, предусматривающих поиск многобайтовых значений (бинарных, «упакованных» данных) нужно предусмотреть поиск вариантов хранения данных и в little-endian, и в big-endian порядке.

Замечание 10. В вариантах, где необходимо сравнивать некоторые величины с заданными значениями, операторы сравнения задаются следующим образом: eq означает «равно» (***eq***ual), ne означает «не равно» (***n***ot ***e***qual), gt означает «строго больше» (***g***reater ***t***han), lt означает «строго меньше» (***l***ess ***t***han), ge означает «больше или равно» (***g***reater or ***e***qual), le означает «меньше или равно» (***l***ess or ***e***qual).

Замечание 11. Опции `-A`, `-O` и `-N` должны работать следующим образом. Предположим, программе доступно два плагина. Первый поддерживает опцию `--exe`, позволяющую найти любые исполняемые файлы, второй — опцию `--file-len`, позволяющую найти файлы заданной длины. При запуске

```
lab1abcNXXXXX --exe --file-len 1000 /tmp
```

программа должна искать в каталоге /tmp и вложенных в него исполняемые файлы длиной 1000 байтов. При запуске

```
lab1abcNXXXXX -O --exe --file-len 1000 /tmp
```

программа должна искать в каталоге /tmp и вложенных в него или исполняемые файлы, или файлы длиной 1000 (то есть совпадение должно быть хотя бы по одному из критериев). В случае запуска

```
lab1abcNXXXXX -O -N --exe --file-len 1000 /tmp
```

программа должна искать файлы, которые не являются исполняемыми и не имеют длину 1000 байтов.

Таблица 1. Интерфейсные функции библиотеки-плагина

Функция	Назначение
plugin_get_info	Получение информации о поддерживаемых плагином опциях.
plugin_process_file	Проверка соответствия файла заданным критериям.

Таблица 2. Опции командной строки, поддерживаемые программой

Опция	Назначение
-P dir	Каталог с плагинами.
-A	Объединение опций плагинов с помощью операции «И» (действует по умолчанию).
-O	Объединение опций плагинов с помощью операции «ИЛИ».
-N	Инвертирование условия поиска (после объединения опций плагинов с помощью -A или -O).
-v	Вывод версии программы и информации о программе (ФИО исполнителя, номер группы, номер варианта лабораторной).
-h	Вывод справки по опциям.

Таблица 3. Переменные окружения, поддерживаемые программой и динамической библиотекой

Переменная	Назначение
LAB1DEBUG	Включение вывода отладочной информации.

Таблица 4. Опции, поддерживаемые библиотекой-плагином

Вариант (номер)	Описание поддерживаемых плагином опций
1	*Опция*: --crc16 <значение> *Назначение*: поиск файлов, контрольная сумма которых, рассчитанная с помощью алгоритма CRC-16-CCITT, равна заданному значению. Значение суммы задается строкой, содержащей запись числа либо в двоичной (0b…), либо в десятичной, либо шестнадцатеричной (0x…) системах. *Пример*: --crc16 0xbabe
2	*Опция*: --mac-addr <значение> *Назначение*: поиск файлов, содержащих заданное значение MAC-адреса в текстовой форме (в форматах aa-bb-cc-dd-ee-ff, aa:bb:cc:dd:ee:ff, aabbccddeeff, или aa bb cc dd ee ff). *Пример*: --mac-addr b9:c0:ae:12:8d:b3
3	*Опция*: --ipv4-addr <значение> *Назначение*: поиск файлов, содержащих заданное значение IPv4-адреса либо в текстовой форме (x.x.x.x), либо в бинарной (little-endian или big-endian). *Пример*: --ipv4-addr 192.168.8.1
4	*Опция*: --freq-byte <значение> *Назначение*: поиск файлов, в которых заданный байт — самый частый. Значение может задаваться в двоичной (0b...), десятичной или шестнадцатеричной системах (0x...). *Пример*: --freq-byte 0xfe
5	*Опция*: --bit-seq <значение> *Назначение*: поиск файлов, содержащих заданную последовательность битов. Значение последовательности задается строкой, содержащей запись числа либо в

	двоичной (0b…), либо в десятичной, либо шестнадцатеричной (0x…) системах. Длина битовой последовательности может быть произвольной. *Пример*: --bit-seq 0b110110100001
6	*Опция*: --parity <значение> *Назначение*: поиск файлов, содержащих четных и нечетных байтов в отношении, заданном значением опции («odds» — нечетных байтов больше, «evens» — четных байтов больше, «eq» — четных и нечетных поровну). *Пример*: --parity odds
7	*Опция*: --float <значение> *Назначение*: поиск файлов, содержащих текстовую запись вещественного числа в указанном диапазоне значений. Параметр опции имеет формат «b@r», что означает поиск чисел b±r. Вещественное число может быть записано в обычном или научном форматах. *Пример*: --float 3.14@0.5e1
8	*Опция*: --float-bin <значение> *Назначение*: поиск файлов, содержащих заданное вещественное число одинарной точности в бинарной (little-endian или big-endian) форме, т. е. в виде четырех последовательных октетов с соответствующими значениями. Вещественное число может быть записано в обычном или научном форматах. *Пример*: --float-bin 2.71
9	*Опция*: --ipv6-addr <значение> *Назначение*: поиск файлов, содержащих заданное значение Ipv6-адреса в текстовой форме. Допустимы сокращенные формы записи (например, 2001:0db8:0000:0000:0000:0000:0000:7334 — то же самое, что 2001:0db8::7334). *Пример*: --ipv6-addr 2001:db8::1
10	*Опции*: --lines-count <значение>, --lines-count-comp <значение> *Назначение*: поиск файлов, содержащих заданное количество строк. Значение опции --lines-count задает количество строк, значение опции --lines-count-comp — оператор сравнения (eq, ne, gt, lt, ge, le). *Пример*: --lines-count 100 --lines-count-comp ge
11	*Опция*: --anagram <значение> *Назначение*: поиск файлов, содержащих строку или её анаграмму. Например, строка «New York Times» является анаграммой строки «monkeys write». Пробельные символы и регистр букв не учитываются. *Пример*: --anagram "new york times"
12	*Опция*: --crc32 <значение> *Назначение*: поиск файлов, контрольная сумма которых, рассчитанная с помощью алгоритма CRC-32, равна заданному значению. Значение суммы задается строкой, содержащей запись числа либо в двоичной (0b…), либо в десятичной, либо шестнадцатеричной (0x…) системах. *Пример*: --crc32 0xcafebabe
13	*Опция*: --pic <значение> *Назначение*: поиск файлов, содержащих изображения в форматах JPEG, PNG, GIF и BMP. Признаком формата считать соответствующее магическое число в заголовке файла. Значением опции является строка, в которой перечисляются через запятую без пробелов форматы, которые требуется найти. *Пример*: --pic png,gif
14	*Опция*: --exe <значение> *Назначение*: поиск исполняемых файлов в форматах ELF, PE32, a.out и COFF. Признаком формата считать соответствующее магическое число в заголовке файла. Значением опции является строка, в которой перечисляются через запятую без пробелов форматы, которые требуется найти. *Пример*: --exe pe32,elf,a.out
15	*Опция*: --double-bin *Назначение*: поиск файлов, содержащих заданное вещественное число двойной точности в бинарной (little-endian или big-endian) форме, т. е. в виде восьми последовательных октетов с соответствующими значениями. Вещественное число может быть записано в обычном или научном форматах.

	Пример: --double-bin 2.71
16	*Опция*: --bytes <значение> *Назначение*: поиск файлов, содержащих все указанные байты (не обязательно последовательно). Значения байтов задаются разделенными запятыми строками, содержащими запись числа либо в двоичной (0b…), либо в десятичной, либо шестнадцатеричной (0x…) системах. *Пример*: --bytes 10,20,0x18
17	*Опции*: --same-bytes <значение>, --same-bytes-comp <значение> *Назначение*: поиск файлов, содержащих хотя бы один участок из заданного количества повторяющихся байтов. Значение опции --same-bytes задает длину участка, а значение опции --same-bytes-comp — оператор сравнения (eq, ne, gt, lt, ge, le). *Пример*: --same-bytes 10 --same-bytes-comp eq
18	*Опция*: --mac-addr-bin <значение> *Назначение*: поиск файлов, содержащих заданное значение MAC-адреса в бинарной (little-endian или big-endian) форме. *Пример*: --mac-addr-bin a0:12:ce:78:9f:b4
19	*Опция*: --ipv4-addr-bin <значение> *Назначение*: поиск файлов, содержащих заданное значение IPv4-адреса в бинарной (little-endian или big-endian) форме, т. е. в виде четырех последовательных октетов с заданными значениями. *Пример*: --ipv4-addr-bin 192.168.8.1
20	*Опция*: --uuids *Назначение*: поиск файлов, содержащих хотя бы один универсальный уникальный идентификатор (UUID) в наиболее употребимом текстовом формате (32 шестнадцатеричных цифры группами 8-4-4-4-12, например, 123e4567-e89b-12d3-a456-426655440000). *Пример*: --uuids
21	*Опция*: --ccards *Назначение*: поиск файлов, содержащих корректные номера кредитных карт в текстовой форме (либо 16 цифр подряд, либо с разделителями между группами по 4 цифры). Корректность номера проверяется алгоритмом Луна. *Пример*: --ccards
22	*Опция*: --dl-syms <список символов через запятую> *Назначение*: поиск динамических библиотек, содержащих все заданные символы. Значением опции является список символов, разделитель — запятая. *Пример*: --dl-syms plugin_get_info,plugin_process_file
23	*Опции*: --seq-num <значение>, --seq-num-comp <значение> *Назначение*: поиск файлов, содержащих заданное количество последовательностей одинаковых байтов (например, подряд три нулевых байта, затем в другом месте десять подряд байтов со значением 100 и т.д.). Значение опции --seq-num задает количество последовательностей, а значение опции --seq-num-comp — оператор сравнения (eq, ne, gt, lt, ge, le). *Пример*: --seq-num 10 --seq-num-comp ge
24	*Опция*: --ccards-bin *Назначение*: поиск файлов, содержащих корректные номера кредитных карт в бинарной (little-endian или big-endian) форме (8 байтов, по 4 бита на цифру). Корректность номера проверяется алгоритмом Луна. *Пример*: --ccards-bin