Библиотеки и оптимизации

Николай Пономарев

22 сентября 2025 г.

Санкт-Петербургский
государственный университет

Про сдачу домашек

В курсе по Python Вас научили пользоваться Git и GitHub

- Каждая домашняя работа оформляется как Pull Request в свой репозиторий
- Добавьте преподавателя в коллабораторы
- Одно задание — один PR
- Следите за оформлением кода, коммитов, комментариев и т.д.
- Дедлайн будем считать по дате запроса review (вдруг HwProj опять устанет)

Важно

Начиная с этой домашки в HwProj должна быть ссылка на Pull Request!

Библиотеки

- Многие задачи решали до Вас неоднократно
- Такие готовые решения принято называть «библиотеками»
- Иногда требуется создавать свои библиотеки
- Чаще — достаточно использовать уже готовые

Библиотеки

- Многие задачи решали до Вас неоднократно
- Такие готовые решения принято называть «библиотеками»
- Иногда требуется создавать свои библиотеки
- Чаще — достаточно использовать уже готовые

Определение

Библиотекой в Си называют набор заголовочных файлов, доступных другим приложениям, и файлов с реализацией объявлений из соответствующего заголовочного файла. Это могут быть как .c-файлы, так и уже прекомпилированные файлы.

Стандартные библиотеки в Си

На самом деле Вы использовали уже довольно много разных библиотек:

stdio.h	printf, scanf, ...
stdlib.h	malloc, calloc, free, ...
math.h	abs, sin, cos, ...
string.h	strcmp, strcpy, ...
stdbool.h	true, false, ...
	...

Связывание со стандартными библиотеками

Большая часть функций живёт в libc, однако математика живёт в libm

```c
#include <math.h>
#include <stdio.h>
int main(void)
{
    double x = 0;
    printf("Enter number: ");
    scanf("%lf", &x);

    printf("Square root of %lf is equal to %lf\n", x, sqrt(x));
}
```

Нужно собирать как `gcc -Wall -Wextra -pedantic rootOf3.c -lm -o rootOf3`

Модули

- Способ группировки кода в логически обособленные группы
- В C это реализуется с помощью заголовочных файлов и файлов с реализацией
  - `.h` и `.c`
- В отдельный модуль выносятся объявления типов данных и функции, которые делают одно дело
  - Например, разные функции сортировки
  - Или всё для работы с матрицами
- В интерфейсную часть модуля выносится только то, что может использовать другой код
  - Меньше знаешь — крепче спишь
- Функции, используемые только для реализации, пишутся только в `.c`-файле
  - Например, функция разделения массива для быстрой сортировки или swap

Модули

Заголовочный файл:

```
#pragma once

// Комментарий к функции 1
int functionOne(int x, int y);

// Комментарий к функции 2
void functionTwo();
```

.c-файл:

```
#include <имя заголовочного файла.h>

#include <все остальные библиотеки>

int functionOne(int x, int y)
{
    ...
}

void functionTwo()
{
    ...
}
```

Тонкости

- Реализации функций в .h-файле писать нельзя
  - Иначе будет беда, если один .h-ник подключат в два .c-шника
- Комментарии обязательны
- #pragma once обязательна
- Подключать «свой» заголовочный файл в .c обязательно
- Файлы .h/.c всегда ходят парами, кроме файла с main

Пример простой собственной библиотеки

См. workdir/prime

```
$ gcc -Wall -Wextra -pedantic main.c -o main
/usr/bin/ld: /tmp/cc4YmGLo.o: in function `main':
main.c:(.text+0x3f): undefined reference to `isPrime'
collect2: error: ld returned 1 exit status
```

Не выйдет! Нужно так:

```
$ gcc -Wall -Wextra -pedantic prime.c -c
$ gcc -Wall -Wextra -pedantic prime.o main.c -o main
```

Статическая линковка

Способ линковки, при котором библиотеки встраиваются в исполняемый файл

Плюсы:

- Высокая переносимость
- Повышенная безопасность
- Скорость исполнения

Минусы:

- Увеличение размера исполняемого файла
- Сложность управления версиями библиотек

Динамическая линковка

Способ связывания, при котором система загружает библиотеки по ходу исполнения программы

Плюсы:

- Меньший размер исполняемого файла
- «Простота» управления версиями библиотек

Минусы:

- Проблемы с переносимостью
- Временные затраты на поиск и загрузку библиотек по ходу исполнения

Динамическая линковка vs. Статическая линковка

```
$ gcc -Wall -Wextra -pedantic rootOf3.c -lm -o rootOf3
$ ldd rootOf3
 linux-vdso.so.1 (0x00007f4e00c64000)
 libm.so.6 ⇒ /usr/lib/libm.so.6 (0x00007f4e00b16000)
 libc.so.6 ⇒ /usr/lib/libc.so.6 (0x00007f4e00800000)
 /lib64/ld-linux-x86-64.so.2 ⇒ /usr/lib64/ld-linux-x86-64.so.2
  ↪  (0x00007f4e00c66000)
$ du -h rootOf3
16K   rootOf3

$ gcc -Wall -Wextra -pedantic rootOf3.c -lm -o rootOf3 -static
$ ldd rootOf3
 not a dynamic executable
$ du -h rootOf3
948K  rootOf3
```

Компиляторные оптимизации

`gcc -O1 -O2 -O3 ...`

- раскрытие циклов (избавление от условных переходов)
- замена инструкций (умножение на 2 эквивалентно битовому сдвигу)
- раскрытие внутренних методов (избавление от вызовов подпрограмм)
- использование SSE (одновременная обработка блоков данных)
- ...

`gcc -Os` — оптимизация по размеру

Compiler Explorer: https://godbolt.org/

Домашнее задание

Реализовать приложение, сортирующее поступающий в стандартный поток ввода набор целых чисел (не более 100 штук, это гарантируется). Числа разделены пробелами, последним символом является перенос строки.
Процедура сортировки должна быть реализована на языке ассемблера *достаточно оптимальным образом*. Кодом возврата приложения является количество элементов, участвовавших в сортировке и изменивших свою позицию.
Дополнительные соглашения:

- Чтение поступающих на вход сортируемых элементов необходимо производить с помощью команды `scanf(...)`.
- Файлы с кодом приложения должны иметь расширение `.c`, файл с реализацией сортировки на ассемблере должен иметь расширение `.s`.

Не забудьте приложить инструкцию по сборке!

Полезные ссылки

- C FAQ: «I'm wondering what to put in .c files and what to put in .h files. (What does ".h" mean, anyway?)»
  - https://c-faq.com/cpp/hfiles.html
- How to handle dynamic and static libraries in Linux
  - https://opensource.com/article/20/6/linux-libraries
  - Статья для тех, кто хочет понять, как создавать библиотеки пригодные для связывания
- Про компоновку, dependency hell и обратную совместимость
  - https://habr.com/ru/articles/220961/
  - Про то, к каким побочным эффектам может приводить повсеместное бездумное использование динамической линковки и как их можно лечить