### МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

# Изучаем программирование на



Дэвид Гриффитс Дон Гриффитс





### Статические и динамические библиотеки

#### Легко заменяемый код

Вы уже знаете, какой мощью обладают стандартные библиотеки.

Пришло время применить эту мощь в *собственном* коде. Здесь вы узнаете, как создавать **собственные библиотеки** и **использовать один и тот же код в нескольких приложениях**. Вы научитесь разделять код во время выполнения программы с помощью **динамических библиотек**. И дочитав эту главу, вы сможете писать масштабируемый, простой и эффективный в управлении код.

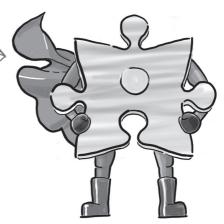
Код, который вы можете принести с собой в банк	388
Угловые скобки предназначены для стандартных заголовков	390
Но что если вы хотите разделять код?	391
Разделяем заголовочные файлы	392
Разделяем объектные файлы .о с помощью полного пути	393
Архив содержит объектные файлы	394
Создайте архив с помощью команды ar	395
И наконец, скомпилируйте другие свои программы	396
Гренажерный зал Head First выходит на международный уровень	401
Подсчет калорий	402
Но все немного сложнее	405
Программы состоят из множества частей	406
Динамическая компоновка происходит во время выполнения программы	408
Можно ли скомпоновать архив во время выполнения программы?	409
Сначала создадим объектный файл	410
На разных платформах динамические библиотеки называются по-разному	411
Ваш инструментарий языка Си	423





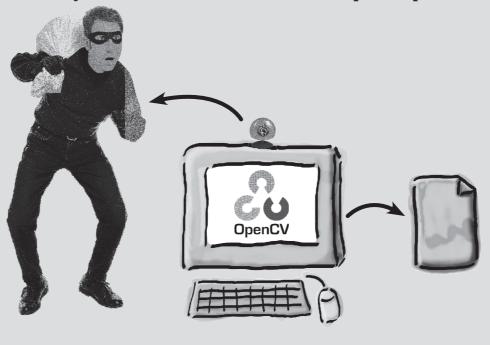


Это птица? Это самолет? Нет, — это перемещаемый объектный файл с метаданными.



## Лабораторная работа 2 OpenCV

Представьте, что компьютер в ваше отсутствие может присматривать за домом и сообщать о подозрительных личностях, которые бродят вокруг. В лабораторной работе 2 вы создадите детектор для обнаружения незваных гостей, используя возможности языка Си и мастерство OpenCV.



### Процессы и системные вызовы

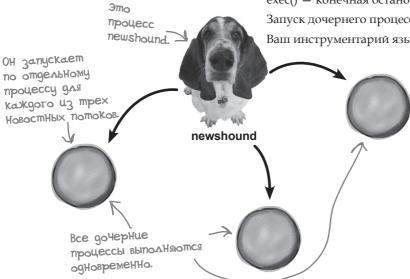
#### Разрушая границы

## 9

#### Пришло время мыслить нестандартно.

Вы уже знаете, что с помощью командной строки можно собирать сложные приложения из небольших утилит. Но что если вы хотите вызывать другие программы из собственного кода? В этой главе вы научитесь использовать системные сервисы для создания процессов и управления ими. Благодаря этому вы получите доступ к электронной почте, браузеру и любому установленному внешнему приложению. К концу этой главы вы сможете выйти за рамки языка Си.

Системные вызовы - это ваша прямая связь 434 с операционной системой Кто-то проник в систему 438 Безопасность не единственная проблема 439 Функция exec() дает вам больший контроль 440 Существует много функций ехес() 441 Функции с поддержкой массивов execv(), execvp(), execve() 442 Передача переменных среды 443 Большинству системных вызовов присущи одни 444 и те же проблемы Читаем новости с помощью RSS 452 exec() — конечная остановка для вашей программы 456 Запуск дочернего процесса с помощью fork() + exec() 457 Ваш инструментарий языка Си 463



### Межпроцессорное взаимодействие

### Общение — это хорошо

## 10

#### Создание процессов — это только первый шаг.

Что если вам захочется управлять запущенным процессом? Что если вам захочется отправить ему данные? Или прочитать его вывод? Благодаря межпроцессному взаимодействию процессы могут выполнять работу сообща. Вы увидите, что мощь вашего кода может возрасти многократно, если позволить ему общаться с другими программами системы.

```
Перенаправление ввода и вывода
                                                       466
Типичный процесс изнутри
                                                       467
При перенаправлении всего лишь
меняются потоки данных
                                                       468
С помощью fileno() можно получить дескриптор
                                                       469
                                                       474
Иногда нужно подождать
Поддерживайте связь со своим детищем
                                                       478
                                                       479
Соединяйте свои процессы с помощью каналов
                                                       480
Практический пример: открытие новостей в браузере
                                                       481
Дочерний процесс
Родительский процесс
                                                       481
Открытие веб-страницы в браузере
                                                       482
                                                       487
Смерть процесса
Перехват сигналов для запуска собственного кода
                                                       488
sigaction регистрируется с помощью функции sigaction()
                                                      489
Перепишем код с использованием обработчика сигнала
                                                       490
Чтобы посылать сигналы, используйте команду kill
                                                       493
Ваш инструментарий языка Си
                                                       502
```

```
#include <stdio.h>

int main()
{
  char name[30];
  printf("Введите свое имя: ");
  fgets(name, 30, stdin);
  printf("Привет, %s\n", name);
  return 0;
}
```

File Edit Window Help
> ./greetings
Введите свое имя: ^C
>

Если вы нажмете Ctrl + C, программа перестанет разотать. но почему?

### Cokembi n pasoma b Cemn

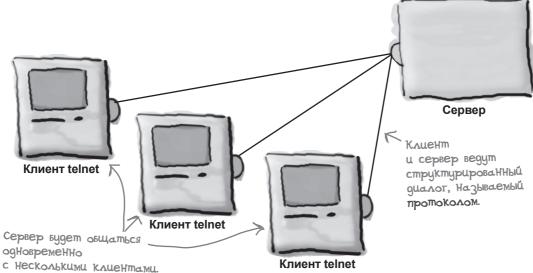
### Нет места лучше, чем 127.0.0.1

## 11

#### Программам на разных компьютерах тоже нужно общаться.

Вы уже научились использовать ввод/вывод для организации взаимодействия с файлами и узнали, каким образом могут общаться два процесса на одном и том же компьютере. Теперь вы сможете *разговаривать со всем остальным миром*, создавая код на языке Си, способный взаимодействовать с другими программами по Сети в любой точке *планеты*. К концу этой главы вы научитесь создавать как серверные, так и клиентские приложения.

Интернет-сервер «Тук-тук»	504
Обзор сервера «Тук-тук»	505
Как сервер разговаривает с Интернетом	506
Сокеты — это не совсем обычные потоки данных	508
Иногда сервер стартует не так, как положено	512
Говорила же мама всегда делать проверку на ошибки	513
Прием данных от клиента	514
Сервер может общаться только с одним человеком	
в отдельный момент времени	521
Вы можете клонировать процесс для каждого клиента	522
Написание веб-клиента	526
Клиент всегда прав	527
Создаем сокет для IP-адреса	528
getaddrinfo() получает адреса доменов	529
Ваш инструментарий языка Си	536



### Tomoku

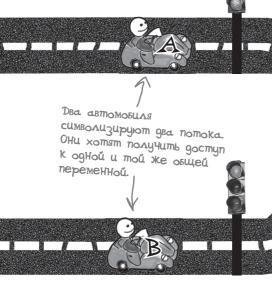
#### Это параллельный мир

## 12

### Программам часто приходится выполнять несколько заданий одновременно.

С помощью стандартных POSIX-потоков вы можете сделать код более отзывчивым, разделяя его на несколько параллельно выполняемых частей. Но будьте осторожны! Потоки являются мощным инструментом, однако вряд ли вы захотите, чтобы они столкнулись друг с другом. В этой главе вы научитесь расставлять светофоры и наносить разметку так, чтобы в вашем коде не образовывались пробки. В результате вы узнаете, как создавать POSIX-потоки и как с помощью механизмов синхронизации сохранять целостность хрупких данных.

Задачи выполняются последовательно или нет	538
и разбиение на процессы не всегда подходит	539
Простые процессы выполняют только одну задачу за один раз	540
Наймите дополнительный персонал — используйте потоки	541
Как создавать потоки	542
Создаем поток с помощью pthread_create	543
Код небезопасен с точки зрения многопоточности	548
Вам нужно добавить светофоры	549
Используйте мьютексы как светофоры	550
Ваш инструментарий языка Си	557

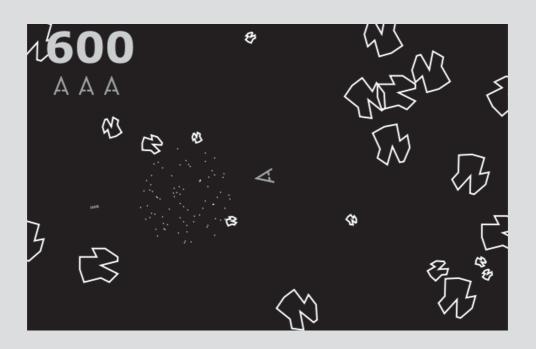


Светофоры не позволят двум потокам получить одновременно доступ к одной и той же общей переменной.

Общая переменная

## Лабораторная работа 3 Бластероиды

В лабораторной работе 3 вы отдадите дань уважения одной из самых популярных и долговечных видеоигр. Пришло время написать «Бластероиды»!



### Приложение I На закуску Топ-10 фактов (которым мы не уделили внимание)

#### Даже после всего сказанного еще кое-что осталось.

Есть несколько фактов, о которых, как нам кажется, вы должны знать. С одной стороны, нам не хотелось делать эту книгу объемной настолько, чтобы ее нельзя было поднять без специальной физической подготовки. Но с другой стороны, мы считаем, что кое-какие моменты заслуживают хотя бы краткого упоминания и проигнорировать их было бы неправильно. Поэтому прежде чем окончательно отложить книгу в сторону, ознакомьтесь со следующими темами.



№ 1. Операторы	576
№ 2. Директивы препроцессора	578
№ 3. Ключевое слово static	579
№ 4. Определение размера	580
№ 5. Автоматизированное тестирование	581
№ 6. Еще немного о дсс	582
№ 7. Еще немного o make	584
№ 8. Средства разработки	586
№ 9. Создание пользовательских интерфейсов	587
№ 10. Справочные материалы	588

### Приложение [] Все темы

#### Вспомнить все

#### Всегда хотелось собрать все замечательные факты о Си в одном месте?

Здесь мы собрали все темы и правила, которые были затронуты в этой книге. Взгляните на них и постарайтесь вспомнить все, о чем мы вам рассказали. Для каждого факта указана глава, где он был рассмотрен, в случае чего вы можете запросто вернуться назад. Возможно, вы даже захотите вырвать эти страницы и приклеить их на стену.



### 1 Начинаем работать с языком Си



#### Хотите узнать, как думает компьютер?

Вам нужно написать высокопроизводительный код для новой игры? Запрограммировать контроллер Arduino? Или использовать продвинутую стороннюю библиотеку в своем приложении для iPhone? Тогда язык Си вам поможет. Си работает на гораздо более низком уровне, чем большинство других языков программирования, поэтому понимание Си позволит вам лучше разобраться в том, что на самом деле происходит внутри компьютера. Си также может помочь в освоении других языков. Поэтому хватайте свой компилятор и немедленно приступайте к делу.

### Си — это язык для компактных быстрых программ

Язык Си предназначен для написания компактных быстрых программ. Он более низкоуровневый, чем большинство других языков программирования. Это означает, что созданный с его помощью код наиболее понятен для компьютера.

#### Как работает Си



Все начинается с создания исходного файла. Исходный файл содержит код на языке Си, пригодный для восприятия человеком.

Исходный код пропускается через компилятор. Компилятор ищет ошибки и, если все хорошо, компилирует исходный код.

Компилятор создает новый файл, который называется исполняемым. Этот файл содержит машинный код — поток единиц и нулей, понятный компьютеру. Это и есть программа, которую вы запускаете.

Си используется там, где важны скорость и объем. Большинство операционных систем написано на Си. Многие компьютерные языки также написаны на Си, равно как и большая часть игровых программ.

Вы можете столкнуться с тремя стандартами языка Си. ANSI С был создан в конце 1980-х гг. и используется для наиболее старого кода. В 1999 г. многие аспекты языка были исправлены в стандарте С99. А в 2011 г. появился современный стандарт С11 с некоторыми новыми интересными возможностями. Отличия между разными версиями не очень большие, и мы будем указывать на них время от времени.

### Наточите свой карандаш

Попытайтесь угадать, что делает каждый из этих фрагментов. Опишите действия, которые, по вашему мнению, выполняет этот код. int card\_count = 11; if (card count > 10) puts("Карты в колоде удачные. Повышаем ставку."); int c = 10;while (c > 0) { puts("Я не должен писать код в виде классов"); c = c - 1;/\* Подразумевается, что имя короче 20 латинских (и 10 кириллических) символов. \*/ \_\_\_\_\_\_ char ex[20]; puts("Введите имя вашей девушки:"); **.**..... scanf("%19s", ex); printf("Дорогая %s. \n\n\tC тобой покончено.\n", ex); char suit = 'H'; **.**..... switch(suit) { case 'C': \_\_\_\_\_\_ puts("Clubs(Трефы)"); break; case 'D': puts("Diamonds(Бубны)"); ••••• break; case 'H': puts("Hearts(Черви)"); break; default: •••••• puts("Spades(Пики)");

### Наточите свой карандаш

Решение

**Не волнуйтесь, если вам что-то непонятно.** Далее в этой книге будут даны более подробные пояснения.

```
int card_count = 11; int - yeaoe yucao.
                                                     Создаем целочисленную переменную
                                                     и устанавливаем для нее значение 11.
if (card count > 10)
                                                    Счетчик больше 10?
    puts ("Карты в колоде удачные. Повышаем
                                                     Если да, то выводим сообщение
ставку."
                                                    в командной строке.
Заесь текст выводится в командной строке или на
терминал.
                  Фитурные сковки
                                                      Создаем целочисленную переменную
               определяют влочнию
int c = 10;
                                                    и устанавливаем для нее значение 10.
                  инструкцию.
while (c > 0) {
                                                    До тех пор пока значение больше 0,...
    puts("Я не должен писать код в виде
                                                    ...выводим сообщение...
классов");
                                                    ...и уменьшаем счетчик.
    c = c - 1;
                                                    Конец кода, который должен повторяться.
/* Подразумевается, что имя короче 20 латинских
                                                    Это комментарий.
(и 10 кириллических) символов. */
                                                    Создаем массив из 20 символов.
char ex[20];
                                                    Выводим сообщение на экран.
puts("Введите имя вашей девушки: ");

это означает «поместить все, что

scanf("%19s", ex); ввел пользователь, внутрь массива ех».
                                                    Сохраняем в массив то, что ввел пользователь.
                                                     Выводим сообщение,
                                                     включающее введенный текст.
printf("Дорогая %s. \n\n\tC тобой покончено.\n",
ex);
                      3qecb cmpoka cumbonob
                      вставляется на место %s.
                                                     Создаем символьную переменную,
char suit = 'H';
                    Инструкция switch проверяет одну
                                                    помещаем в нее букву «H».
                    и ту же переменную на разные
switch(suit) {
                                                    Проверяем значение переменной.
                    значения.
case 'C':
                                                    Оно равно «С»?
    puts("Clubs(Трефы)");
                                                    Если да, то выводим «Clubs (Трефы)».
    break:
                                                    Затем пропускаем остальные проверки.
case 'D':
                                                    Оно равно «D»?
    puts("Diamonds(Бубны)");
                                                    Если да, то выводим «Diamonds (Бубны)».
                                                    Затем пропускаем остальные проверки.
    break:
                                                    Оно равно «Н»?
case 'H':
                                                    Если да, то выводим «Hearts (Черви)».
    puts("Hearts(Черви)");
                                                    Затем пропускаем остальные проверки.
    break:
                                                    В противном случае...
default:
                                                    ...выводим «Spades (Пики)».
    puts("Spades(Пики)");
                                                   На этом проверки заканчиваются.
```

### Как выглядит программа на Си целиком

Чтобы создать полноценную программу, вам необходимо написать свой код на языке Си и сохранить его в *исходный файл*. Исходные файлы можно создавать с помощью любого текстового редактора, имя файла при этом, как правило, заканчивается на .с.

Давайте взглянем на типичный исходный файл на языке Си.

Программы на Си обычно начинаются с комментария.

Комментарий описывает назначение кода, содержащегося в файле, и, возможно, содержит информацию о лицензии и авторском праве. Острой необходимости добавлять его сюда (или в любую другую часть файла) нет, но так принято, и большинство программистов на Си рассчитывают подобный комментарий там увидеть.

Комментарий начинается У

эти звездочки необязательны и нужны только для красоты.

Комментарий заканчивается-

\* Программа для подсчета количества карт в колоде. \* Этот код выпущен под публичной Лас-Вегасской лицензией. \* (c)2014, Команда колледжа по блек-джеку.

#include <stdio.h>

2

Дальше следует **—** часть

с подключением внешних файлов.

Си - очень лаконичный язык, и без использования внешних библиотек он практически ни на что не способен. Компилятору необходимо сообщить, какой внешний код использовать, подключая заголовочные файлы соответствующих библиотек. Чаще других будет встречаться заголовок stdio.h. Это библиотека stdio, содержащая код для вывода данных на терминал и считывания их оттуда.

```
int main()
{
   int decks;
   puts("Введите количество колод.");
   scanf("%i", &decks);
   if (decks < 1) {
      puts("Вы ввели недопустимое количество колод.");
      return 1;
   }
   printf("Всего карт %i\n", (decks * 52));
   return 0;
}</pre>
```

 Последнее, что вы найдете в исходном файле, это функции.

Весь код на Си выполняется внутри функций. Самая важная функция в любой программе, написанной на Си, называется главной — main(). Она является отправной точкой для всего кода в вашей программе.

Давайте рассмотрим функцию main() более подробно.