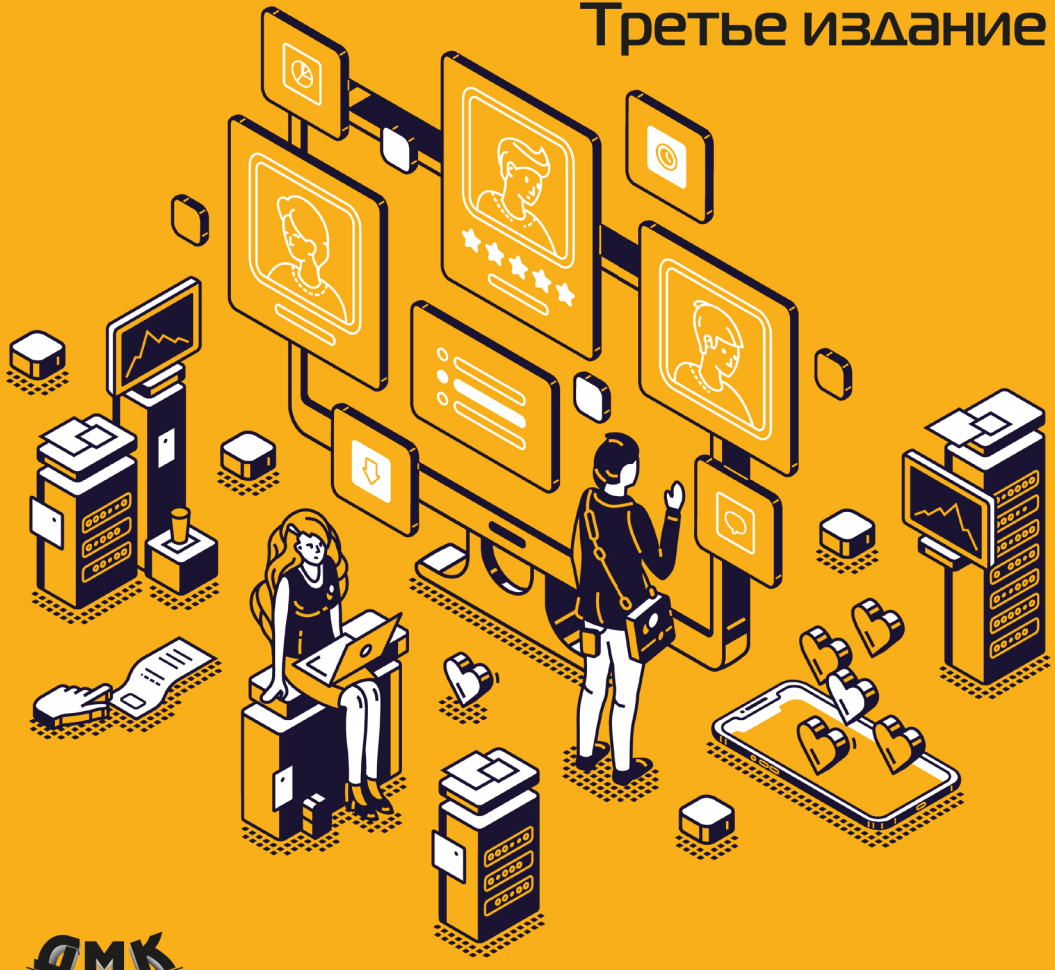
Рэндал Э. Брайант
Дэвид Р. О'Халларон

Компьютерные системы

Архитектура и программирование

Третье издание



ДМК
издательство

Рэндал Э. Брайант
Дэвид Р. О'Халларон

Компьютерные системы:
архитектура и программирование

3-е издание

Computer Systems
A Programmer's Perspective

Third edition

Randal E. Bryant
Carnegie Mellon University

David R. O'Hallaron
Carnegie Mellon University

PEARSON

Компьютерные системы:
архитектура и программирование

3-е издание

Рэндал Э. Брайант
университет Карнеги–Меллона

Дэвид Р. О'Халларон
университет Карнеги–Меллона

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО
Москва, 2022

УДК 004.2
ББК 32.972
Б87

Б87 Рэндал Э. Брайант, Дэвид Р. О'Халларон
Компьютерные системы: архитектура и программирование. 3-е изд. / пер. с англ. А. Н. Киселева. – М.: ДМК Пресс, 2022. – 994 с.: ил.

ISBN 978-5-97060-492-2

В книге описываются стандартные элементы архитектуры, такие как центральный процессор, память, порты ввода-вывода, а также операционная система, компилятор, компоновщик и сетевое окружение. Демонстрируются способы представления данных и программ на машинном уровне, приемы оптимизации программ, особенности управления потоками выполнения и виртуальной памятью, а также методы сетевого и параллельного программирования. Приведенные в книге примеры для процессоров, совместимых с Intel (x86_64), написаны на языке C и выполняются в операционной системе Linux.

Издание адресовано студентам и преподавателям по IT-специальностям, а также будет полезно разработчикам, желающим повысить свой профессиональный уровень и писать программы, эффективно использующие возможности компьютерной архитектуры.

Authorized translation from the English language edition, entitled Computer Systems: A Programmer's Perspective, 3rd Edition, by Randal E. Bryant and David R. O'Hallaron, published by Pearson Education, Inc, publishing as Pearson, Copyright © 2016.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-0-13-409266-9 (англ.)

ISBN 978-5-97060-492-2 (рус.)

Copyright © 2016, 2011, and 2003 by Randal E. Bryant and David R. O'Hallaron, 2021

© Оформление, перевод на русский язык, издание, ДМК Пресс, 2022

Студентам и преподавателям курса 15-213 университета Карнеги–Меллона, вдохновившим нас на переработку и уточнение этого издания

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте **www.dmkpress.com**, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Pearson очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты **dmkpress@gmail.com**.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Вступление

Данная книга предназначена для программистов, желающих повысить свой профессиональный уровень изучением того, что происходит «под кожухом системного блока» компьютерной системы.

Целью авторов является попытка разъяснить устойчивые концепции, лежащие в основе всех компьютерных систем, а также демонстрация конкретных видов влияния этих идей на корректность, производительность и полезные свойства прикладных программ. Многие книги по компьютерным системам написаны *для создателей* таких систем и описывают, как скомпоновать оборудование или реализовать системное программное обеспечение, включая операционную систему, компилятор и сетевой интерфейс. Эта книга, напротив, написана *для программиста* и рассказывает, как прикладные программисты могут использовать свои знания о системах для создания более качественных программ. Конечно, знакомство с требованиями к системам является хорошим первым шагом в обучении их созданию, поэтому эта книга послужит также ценным введением для тех, кто продолжает заниматься созданием аппаратного и программного обеспечения систем. Большинство книг по компьютерным системам также имеют тенденцию сосредоточиваться только на одном аспекте системы, например на аппаратной архитектуре, операционной системе, компиляторе или сети. Эта книга охватывает все эти аспекты и рассматривает их с позиции программиста.

Досконального изучение и освоение изложенных в книге концепций позволит читателю со временем превратиться в редкий тип профессионального программиста, понимающего саму суть происходящего и способного решить любую задачу. Вы сможете писать программы, которые эффективнее используют возможности операционной системы и системного программного обеспечения, действуют правильно в широком диапазоне рабочих условий и параметров, работают быстрее и не содержат уязвимостей для кибератак. При этом будет заложена основа для изучения таких специфических тем, как компиляторы, архитектура компьютерных систем, операционные системы, сети и кибербезопасность.

Что нужно знать перед прочтением

Эта книга посвящена системам с аппаратной архитектурой x86-64, являющиеся последним этапом на пути развития, который прошли Intel и ее конкуренты, начинавшие с микропроцессора 8086 в 1978 году. В соответствии с соглашениями об именовании, принятыми в Intel в отношении их линейки микропроцессоров, этот класс микропроцессоров в просторечии называется «x86». По мере развития полупроводниковых технологий, позволяющих размещать на одном кристалле все больше и больше транзисторов, производительность и объем внутренней памяти процессоров значительно увеличились. В ходе этого прогресса они перешли от 16-разрядных слов к 32-разрядным и выпустили процессор IA32, а совсем недавно произошел переход к 64-разрядным словам и появилась архитектура x86-64.

Мы рассмотрим, как машины с этой архитектурой выполняют программы на языке C в Linux. Linux – одна из операционных систем, ведущих свою родословную от операционной системы Unix, первоначально разработанной в Bell Laboratories. К другим членам этого класса операционных систем относятся Solaris, FreeBSD и MacOS X. В последние годы эти операционные системы сохраняли высокий уровень совместимости благодаря усилиям по стандартизации POSIX и Standard Unix Specification. То есть сведения, что приводятся в этой книге, почти напрямую применимы ко всем этим «Unix-подобным» операционным системам.

Еще незнакомы с C? Совет по языку программирования C

В помощь читателям, плохо знакомым или незнакомым с языком C, авторами предлагаются примечания, подобные данному, для подчеркивания функций, особенно важных для C. Предполагается, что читатели знакомы с C++ или Java.

В тексте содержится множество примеров программного кода, которые мы компилировали и опробовали в системах Linux. Мы предполагаем, что у вас есть доступ к такой системе, что вы можете входить в нее и умеете выполнять простые действия, такие как получение списка файлов или переход в другой каталог. Если ваш компьютер работает под управлением Microsoft Windows, то мы рекомендуем установить одну из множества виртуальных машин (например, VirtualBox или VMWare), которые позволяют программам, написанным для одной операционной системы (гостевой ОС), запускаться в другой (несущей ОС, или хост-ОС).

Также предполагается, что читатель знаком с C или C++. Если весь опыт программиста ограничивается работой с Java, переход потребует от него больше усилий, но авторы окажут всю необходимую помощь. Java и C имеют общий синтаксис и управляющие операторы. Однако в C есть свои особенности (в частности, указатели, явное распределение динамической памяти и форматируемый ввод/вывод), которых нет в Java. К счастью, C – не очень сложный язык, он прекрасно описан в классической книге Брайана Кернигана и Денниса Ритчи [61]. Вне зависимости от «подкованности» читателя в области программирования, эта книга послужит ценным дополнением к его библиотеке. Если прежде вы использовали только интерпретируемые языки, такие как Python, Ruby или Perl, то вам определенно стоит посвятить некоторое время изучению C, прежде чем продолжить читать эту книгу.

В начальных главах книги рассматривается взаимодействие между программами на C и их аналогами на машинном языке. Все примеры на машинном языке были созданы с помощью компилятора GNU GCC на процессоре x86-64. Наличия какого бы то ни было опыта работы с аппаратными средствами, машинными языками или программирования в ассемблере не предполагается.

Как читать книгу

Изучение принципов работы компьютерных систем с точки зрения программиста – занятие очень увлекательное, потому что проходит в интерактивном режиме. Изучив что-то новое, вы тут же можете это проверить и получить результат, что называется, из первых рук. На самом деле авторы полагают, что единственным способом познания систем является их *практическое исследование*: либо путем решения конкретных упражнений, либо написанием и выполнением программ в реально существующих системах.

Система является предметом изучения всей книги. При представлении какой-либо новой концепции в тексте она будет сопровождаться иллюстрацией в форме одной или нескольких *практических задач*, которые нужно сразу же постараться решить, чтобы проверить правильность понимания изложенного. Решения упражнений приводятся в конце каждой главы. Пытайтесь сначала самостоятельно решать эти практические задачи и только потом проверяйте правильность выбранного пути. В конце каждой главы также представлены *домашние задания* различной степени сложности. Каждому домашнему заданию присвоен определенный уровень сложности:

- ♦ для решения достаточно нескольких минут; требуется минимальный объем программирования (или не требуется вообще);
- ♦♦ для решения потребуется до 20 минут. Часто нужно написать и опробовать программный код. Многие такие задания созданы на основе задач, приведенных в примерах;

♦♦♦ для решения потребуется приложить значительные усилия; по времени на решение может уйти до 2 часов. Как правило, для выполнения этих заданий требуется написать и опробовать значительный объем кода;

♦♦♦♦ лабораторная работа, на выполнение которой может уйти до 10 часов.

Все примеры кода в тексте книги отформатированы автоматически (без всякого ручного вмешательства) и получены из программ на C, скомпилированных с помощью GCC и протестированных в Linux. Конечно, в вашей системе может быть установлена другая версия gcc или вообще другой компилятор, генерирующий другой машинный код, но общее поведение примеров должно быть таким же. Весь исходный код доступен на веб-странице книги **csapp.cs.cmu.edu**. Имена файлов с исходным кодом документируются с использованием горизонтальных полос, окружающих отформатированный код. Например, программу из листинга 1 можно найти в файле `hello.c` в каталоге `code/intro/`. Мы советуем обязательно опробовать примеры программ у себя по мере их появления.

Листинг 1. Типичный пример кода

code/intro/hello.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("hello, world\n");
6     return 0;
7 }
```

code/intro/hello.c

Чтобы не раздувать объем и без того большой книги, мы создали несколько приложений, размещенных в интернете, которые содержат сведения, дополняющие основную книгу. Они отмечены в книге заголовками в форме «Приложение в интернете ТЕМА:ПОДТЕМА», где ТЕМА кратко описывает основную тему, а ПОДТЕМА – раздел темы. Например, «Приложение в интернете DATA:BOOL» содержит сведения по булевой алгебре, дополняющие описание представлений данных в главе 2, а «Приложение в интернете ARCH:VLOG» содержит описание приемов проектирования процессоров с использованием языка описания оборудования Verilog, дополняющее обсуждение конструкции процессора в главе 4. Все эти приложения доступны на веб-странице книги (**https://csapp.cs.cmu.edu/3e/waside.html**).

Обзор книги

Книга состоит из 12 глав, охватывающих основные принципы компьютерных систем:

Глава 1. Экскурс в компьютерные системы.

В этой главе описываются основные идеи и темы, относящиеся к компьютерным системам, на примере исследования жизненного цикла простой программы «hello, world».

Глава 2. Представление информации и работа с ней.

Здесь описывается компьютерная арифметика с упором на свойства представлений числа без знака и числа в дополнительном двоичном коде, которые имеют значение для программистов. В данной главе рассматривается представление чисел и, следовательно, диапазон значений, которые можно запрограммировать для отдельно взятого размера слова. Авторы обсуждают влияние

преобразований типов чисел со знаком и без знака и математические свойства арифметических операций. Для начинающих программистов часто оказывается откровением, что сложение (в дополнительном коде) или умножение двух положительных чисел может дать отрицательный результат. С другой стороны, арифметика дополнительного кода удовлетворяет многим алгебраическим свойствам целочисленной арифметики, благодаря чему компилятор может преобразовать операцию умножения на константу в последовательность сдвигов и сложений. Для иллюстрации принципов и применения булевой алгебры авторы используют поразрядные операции C. Формат IEEE с плавающей точкой описывается в терминах представления значений и математических свойств операций с плавающей точкой.

Абсолютное понимание компьютерной арифметики принципиально для создания надежных программ. Например, программисты и компиляторы не могут заменить выражение `(x < y)` на `(x - y < 0)` из-за возможности переполнения. Они не могут даже заменить его выражением `(-y < -x)` из-за асимметричности диапазона отрицательных и положительных чисел в дополнительном коде. Арифметическое переполнение является обычным источником ошибок программирования и уязвимостей в системе безопасности, однако мало в какой книге можно найти описание свойств компьютерной арифметики, сделанное с точки зрения самого программиста.

О чем рассказывается во врезках?

На протяжении всей книги вам будут встречаться подобные примечания. В них приводятся некоторые дополнительные сведения к обсуждаемой теме. Примечания преследуют несколько целей. Одни представляют собой небольшие исторические экскурсы. Например, как появились C, Linux и Internet? Другие разъясняют какие-либо понятия. Например, чем отличаются кеш, множество и блок? Третьи примечания описывают примеры «из жизни». Например, как ошибка в вычислениях с плавающей точкой уничтожила французскую ракету или какие геометрические и функциональные параметры имеют коммерческие жесткие диски. И наконец, некоторые примечания – это всего лишь забавные комментарии.

Глава 3. Представление программ на машинном уровне.

Авторы научат читать машинный код x86-64, созданный компилятором C. Здесь представлены основные шаблоны инструкций для различных управляющих структур, таких как условные операторы, циклы и операторы выбора. Также рассматривается реализация процедур, включая выделение места на стеке, условные обозначения использования реестров и передачу параметров. В главе рассматриваются различные структуры данных, например структуры, объединения и массивы, их размещение в памяти и доступ к ним. Здесь еще будет показано, как выглядят программы с точки зрения машины, что поможет понять распространенные уязвимости, такие как переполнение буфера, и шаги, которые программист, компилятор и операционная система могут предпринять для уменьшения этих угроз. Изучение данной главы поможет повысить профессиональный уровень, потому что при этом появится понимание, как компьютер воспринимает программы.

Глава 4. Архитектура процессора.

В этой главе описываются комбинаторные и последовательные логические элементы, после чего демонстрируется, как эти элементы можно объединить в

информационный канал, выполняющий упрощенный набор инструкций x86-64 с названием «Y86-64». Сначала будет рассматриваться однотактный тракт данных. Его архитектура проста, но не отличается высоким быстродействием. Затем будет представлено понятие *конвейерной обработки*, в которой различные шаги, необходимые для обработки инструкции, реализуются как отдельные этапы. В каждый конкретный момент этапы конвейера могут обрабатывать разные инструкции. Получившийся в результате пятиступенчатый процессорный конвейер намного более реалистичен. Управляющая логика процессора описывается с использованием простого языка описания аппаратных средств – HCL. Проекты аппаратного обеспечения, написанные на HCL, можно компилировать и объединять в симуляторы, а затем использовать для создания описания Verilog, пригодного для производства реального оборудования.

Глава 5. Оптимизация производительности программ.

В этой главе представлен ряд методов повышения производительности кода, при этом идея состоит в том, что программисты учатся писать код на C так, чтобы компилятор мог затем сгенерировать эффективный машинный код. Сначала рассматриваются преобразования, сокращающие объем работы, которую предстоит выполнить программе, и, следовательно, которые должны стать стандартной практикой при написании любых программ для любых машин. Затем мы перейдем к преобразованиям, повышающим степень параллелизма на уровне команд в сгенерированном машинном коде, чтобы повысить их производительность на современных «суперскалярных» процессорах. Для обоснования этих преобразований будет представлена простая модель работы современных процессоров и показано, как измерить потенциальную производительность программы с точки зрения критических путей с использованием графического представления программы. Вы будете удивлены, насколько можно ускорить программу, применив простые преобразования к коду на C.

Глава 6. Иерархия памяти.

Память является для программистов одной из самых «заметных» частей компьютерной системы. До этой главы читатели полагались на концептуальную модель памяти в форме одномерного массива с постоянным временем доступа. На практике память представляет собой иерархию запоминающих устройств разной емкости, стоимости и быстродействия. В главе рассматриваются разные типы памяти, такие как ОЗУ и ПЗУ, а также геометрические параметры и устройство существующих современных дисковых накопителей, организация этих запоминающих устройств в иерархию. Авторы показывают возможность иерархической организации посредством локальности ссылок. Данные идеи конкретизируются представлением уникального взгляда на систему памяти как на «гору памяти» со «скалами» временной локальности и «склонами» пространственной локальности. В заключение рассказывается, как повысить производительность программных приложений путем усовершенствования их временной и пространственной локальности.

Глава 7. Связывание.

В данной главе описывается статическое и динамическое связывание, включая такие понятия, как: перемещаемые и выполняемые объектные файлы, разрешение символов, перемещение, статические библиотеки, разделяемые библиотеки, перемещаемый код и подмена библиотечных функций (library interpositioning). Тема связывания редко рассматривается в книгах по компьютерным системам, но авторы решили включить ее в эту книгу по двум причи-

нам. Во-первых, некоторые типичные ошибки, с которыми сталкиваются программисты, как раз возникают на этапе связывания и особенно характерны для крупных программных пакетов. Во-вторых, объектные файлы, создаваемые компоновщиками, связаны с такими понятиями, как загрузка, виртуальная память и отображение памяти.

Глава 8. Управление исключениями.

В этой главе авторы отступают от однопрограммной модели, вводя общую концепцию потока управления исключениями (не совпадающего с обычным потоком управления путем ветвления в условных операторах и в точках вызова процедур). Мы рассмотрим примеры потоков управления исключениями, существующих на всех уровнях системы, от аппаратных исключений и прерываний низкого уровня до переключения контекста между параллельными процессами, внезапных изменений в потоке управления, вызванных передачей сигналов Linux, и нелокальных переходов в C, разрывающих стройную структуру стека.

В этой части книги будет представлено фундаментальное понятие *процесса* как абстракции выполняющейся программы. Здесь авторы расскажут, как работают процессы, как их создавать и как ими можно управлять из прикладных программ, и покажут, как прикладные программисты могут запустить несколько процессов с помощью системных вызовов Linux. По окончании этой главы вы сможете написать простую командную оболочку для Linux с поддержкой управления заданиями. Эта глава также станет первым знакомством с недетерминированным поведением, возникающим при параллельных вычислениях.

Глава 9. Виртуальная память.

Описывает представление системы виртуальной памяти с целью дать некоторое понимание ее особенностей и принципов работы. Здесь вы узнаете, как разные процессы, действующие одновременно, могут использовать один и тот же диапазон адресов, совместно использовать одни страницы памяти и иметь индивидуальные копии других. В этой главе также описываются вопросы, связанные с управлением виртуальной памятью и манипуляциями с ней. В частности, мы уделим большое внимание инструментам распределения памяти из стандартной библиотеки, таким как `malloc` и `free`. Обсуждение данной темы преследует несколько целей. Прежде всего оно подкрепляет концепцию о том, что пространство виртуальной памяти является всего лишь массивом байтов, который программа может разделить на блоки разного размера для хранения данных. Помогает понять последствия ошибок обращения к памяти в программах, такие как утечки и недействительные ссылки в указателях. Наконец, многие программисты реализуют свои инструменты распределения памяти, оптимизированные под требования и характеристики конкретного приложения. Эта глава в большей степени, чем любая другая, демонстрирует преимущества неразрывного освещения аппаратных и программных аспектов компьютерных систем. Книги о традиционных компьютерных архитектурах и операционных системах обычно представляют виртуальную память только с одной стороны.

Глава 10. Системный уровень ввода/вывода.

В этой главе рассматриваются основные концепции ввода/вывода в системе Unix, такие как файлы и дескрипторы. Авторы описывают совместное использование файлов, принципы работы переадресации ввода/вывода и доступ к метаданным файлов. Здесь также представлен пример разработки надежного пакета буферизованного ввода/вывода, прекрасно справляющегося с любопыт-

ным поведением подсистемы ввода/вывода, известным как *недостача*, когда библиотечные функции возвращают только часть ввода. В главе описывается стандартная библиотека ввода/вывода языка C и ее связь с подсистемой ввода/вывода в Linux с упором на ограничения стандартного ввода/вывода, делающие его непригодным для сетевого программирования. Вообще говоря, темы, охваченные в этой главе, служат основой для двух следующих глав, посвященных сетевому и параллельному программированию.

Глава 11. Сетевое программирование.

Сети являются своеобразными устройствами ввода/вывода для программ, объединяющими многие из понятий, описанных ранее: процессы, сигналы, порядок следования байтов, отображение памяти и динамическое распределение пространства запоминающих устройств. Сетевые программы также являются одними из первых кандидатов на применение приемов параллельного программирования, о котором рассказывается в следующей главе. Данная глава – лишь тонкий срез глобального предмета сетевого программирования, необходимый для создания простенького веб-сервера. Здесь будет представлена модель клиент–сервер, лежащая в основе всех сетевых приложений. Авторы представят взгляд программиста на сеть Интернет и покажут, как писать сетевые клиенты и серверы, используя интерфейс сокетов. И наконец, в главе будет представлен протокол HTTP и разработан простой веб-сервер.

Глава 12. Конкурентное программирование.

Эта глава описывает принципы конкурентного (параллельного) программирования на примере сетевого сервера. Авторы сравнивают и противопоставляют три основных механизма, используемых для создания конкурентных программ: процессы, мультиплексирование ввода/вывода и потоки выполнения – и показывают возможность их использования при создании серверов, способных обслуживать множество одновременных соединений. Здесь же описаны основные принципы синхронизации с использованием семафорных операций P и V, безопасность потоков выполнения и реентерабельность, а также состояние взаимоблокировки. Конкурентное программирование играет важную роль в большинстве сетевых приложений. Также в этой главе описывается конкурентное программирование на уровне потоков выполнения, что позволяет ускорить решение задач на многоядерных процессорах. Чтобы все ядра правильно и эффективно работали над одной вычислительной задачей, требуется тщательная координация потоков, выполняющихся конкурентно.

Что нового в этом издании

Первое издание этой книги было опубликовано в 2003 году, а второе – в 2011. Учитывая быстрое развитие компьютерных технологий, содержимое книги на удивление хорошо сохранило свою актуальность. Принципиальное устройство компьютеров Intel x86, на которых выполняются программы на C под управлением Linux (и других похожих операционных систем), мало изменилось за эти годы. Однако изменения в аппаратных технологиях, компиляторах, интерфейсах программных библиотек и накопленный опыт преподавания потребовали существенного пересмотра материала книги.

Самым большим изменением по сравнению со вторым изданием является переход с представления, основанного на сочетании IA32 и x86-64, к представлению, основанному исключительно на x86-64. Это смещение акцента повлияло на содержимое многих глав. Вот краткое перечисление основных значительных изменений.

Глава 1. Экскурс в компьютерные системы.

Мы переместили обсуждение закона Амдала из главы 5 в эту главу.

Глава 2. Представление информации и работа с ней.

В многочисленных отзывах читатели и рецензенты сообщают, что некоторые сведения в этой главе сложны для понимания. Поэтому мы постарались упростить форму подачи материала, поясняя моменты, которые обсуждаются в строгом математической стиле. Это позволит читателям сначала просмотреть математические детали, чтобы получить общее представление, а затем вернуться к подробному описанию.

Глава 3. Представление программ на машинном уровне.

Мы перешли от представления, основанного на комбинации IA32 и x86-64, к представлению только на основе x86-64. Мы также обновили примеры кода, генерируемые более свежими версиями gcc. Как результат глава претерпела существенные изменения, включая изменение порядка, в котором представлены некоторые концепции. Мы также добавили описание аппаратной поддержки вычислений с плавающей точкой. А для сохранения совместимости добавили приложение в интернете, описывающее машинный код для IA32.

Глава 4. Архитектура процессора.

Мы обновили описание архитектуры процессора, выполнив переход с 32-разрядной архитектуры на архитектуру с поддержкой 64-разрядных слов и операций.

Глава 5. Оптимизация производительности программ.

Мы обновили эту главу, отразив возможности последних поколений процессоров x86-64 в плане производительности. С введением большего количества функциональных модулей и более сложной логики управления разработанная нами модель производительности программ, основанная на представлении программ в форме потока данных, стала предсказывать производительность еще надежнее, чем раньше.

Глава 6. Иерархия памяти.

Мы обновили эту главу с учетом последних технологий.

Глава 7. Связывание.

Мы переписали эту главу, перейдя на архитектуру x86-64, расширили обсуждение использования глобальной таблицы смещений GOT и таблицы компоновки процедур PLT для создания перемещаемого кода и добавили новый раздел о мощном методе связывания, известном как *library interpositioning* (подмена библиотечных функций).

Глава 8. Управление исключениями.

Мы добавили более строгое описание обработчиков сигналов, включив функции, которые можно безопасно вызывать при обработке асинхронных сигналов, специальные рекомендации по написанию обработчиков сигналов и использование `sigsuspend` для приостановки обработчиков.

Глава 9. Виртуальная память.

Эта глава изменилась незначительно.

Глава 10. Системный уровень ввода/вывода.

Мы добавили новый раздел о файлах и иерархии файлов, но в остальном эта глава изменилась незначительно.

Глава 11. Сетевое программирование.

Мы представили новые методы протоколонезависимого и потокобезопасного сетевого программирования с использованием современных функций `getaddrinfo` и `getnameinfo`, пришедших на смену устаревшим и нереентерабельным функциям `gethostbyname` и `gethostbyaddr`.

Глава 12. Параллельное программирование.

Мы расширили обсуждение параллельного программирования, добавив потоки выполнения, использование которых позволяет программам быстрее решать свои задачи на многоядерных машинах.

Также мы добавили и пересмотрели множество практических и домашних заданий по всей книге.

Происхождение книги

Книга родилась из вводного курса, разработанного в университете Карнеги–Меллона (УКМ) осенью 1998 года и получившего название «15-213: Введение в компьютерные системы». С тех пор курс читается в каждом семестре, и в каждом семестре его слушают более 400 студентов, от второкурсников до аспирантов, самых разных специальностей. Данный курс стал основой для большинства других курсов по компьютерным системам в университете Карнеги–Меллона.

Идея этого курса заключалась в том, чтобы познакомить студентов с компьютерами, взглянув на них с другой стороны. Мало кто из студентов смог бы самостоятельно построить компьютерную систему. С другой стороны, от большинства обучающихся и даже инженеров по вычислительной технике требуется повседневное использование компьютеров и умение программировать. Поэтому авторы данной книги решили начать знакомство с системами с точки зрения программиста и при выборе тем использовали следующий своеобразный фильтр: тема будет освещаться только в том случае, если она связана с производительностью, корректностью или с полезными свойствами пользовательских программ на C.

К примеру, исключены темы, связанные с аппаратными сумматорами и конструкцией шин. В курсе также имелись темы, посвященные машинному языку, но вместо подробного рассмотрения языка ассемблера мы предпочли сконцентрироваться на том, как компилятор транслирует конструкции языка C в машинный код, включая операции с указателями, циклы, вызовы процедур и возврат из них. Кроме того, мы решили более широко взглянуть на систему как на комплекс аппаратных и программных средств и включили в книгу такие темы, как связывание, загрузка, процессы, сигналы, оптимизация производительности, виртуальная память, ввод/вывод, а также сетевое и параллельное программирование.

Данный подход позволил сделать курс практичным, конкретным, наглядным и на редкость интересным для студентов. Ответная реакция с их стороны и со стороны коллег по факультету была незамедлительной и положительной, и авторы книги поняли, что преподаватели из других учебных заведений тоже смогут воспользоваться их наработками. Это и стало предпосылкой появления данной книги, написанной лекционным конспектом курса и которую мы теперь обновили, чтобы отразить изменения в технологиях и подходах к реализации систем.

Благодаря выходу новых изданий и переводам этой книги на разные языки она и многие ее варианты стали частью учебных программ по информатике и компьютерной инженерии в сотнях колледжей и университетов по всему миру.

Преподавателям: курсы на основе этой книги

Преподаватели могут использовать эту книгу для проведения нескольких видов курсов по компьютерным системам. В табл. 1 перечислены пять категорий таких курсов. Конкретный курс зависит от требований учебной программы, личного вкуса, а также опыта и способностей студентов. Курсы в табл. 1 перечислены слева направо в порядке близости ко взгляду программиста на систему. Вот их краткое описание.

УК. Курс «Устройство компьютеров» с традиционными темами, раскрытыми в нетрадиционном стиле. Охватываются такие традиционные темы, как логическая модель, архитектура процессора, язык ассемблера и системы памяти. При этом больше внимания должно уделяться программной стороне. Например, обсуждение данных должно быть связано с типами данных и операциями в программах на C, а обсуждение ассемблерного кода основываться не на рукописном машинном коде, а на сгенерированном компилятором C.

УК+. Курс УК с дополнительным упором на аппаратную сторону и производительность прикладных программ. По сравнению с УК, студенты, слушающие курс УК+, узнают больше об оптимизации кода и об улучшении производительности памяти своих программ на языке C.

ВКС. Базовый курс «Введение в компьютерные системы», разработанный для подготовленных программистов, которые понимают влияние оборудования, операционной системы и системы компиляции на производительность и правильность прикладных программ. Существенное отличие от УК+ – отсутствие охвата низкоуровневой архитектуры процессора. Вместо этого программисты учатся работать с высокоуровневой моделью современного процессора. Курс ВКС хорошо вписывается в 10-недельную четверть, но при необходимости может быть продлен до 15-недельного семестра, если проходить его в неторопливом темпе.

ВКС+. Базовый курс «Введение в компьютерные системы» с дополнительным охватом таких тем системного программирования, как ввод/вывод системного уровня, сетевое и параллельное программирование. В университете Карнеги–Меллона это семестровый курс, охватывающий все главы данной книги, кроме низкоуровневой архитектуры процессора.

СП. Курс «Системное программирование». Этот курс похож на ВКС+, но в нем не преподается оптимизация операций с плавающей запятой и производительности, а также уделяется больше внимания системному программированию, включая управление процессами, динамическое связывание, ввод/вывод на уровне системы, сетевое программирование и параллельное программирование. Преподаватели могут добавить информацию из других источников для обсуждения дополнительных тем, таких как демоны, управление терминалом и межпроцессные взаимодействия в Unix.

Как показывает табл. 1, эта книга дает студентам и преподавателям массу возможностей. Если вы хотите, чтобы ваши ученики познакомились с низкоуровневой архитектурой процессоров, то этот вариант доступен в курсах УК и УК+. С другой стороны, если вы хотите переключиться с текущего курса «Устройство компьютеров» на курс ВКС или ВКС+, но опасаетесь вносить радикальные изменения сразу, то можете организовать постепенный переход к ВКС. Вы можете начать с курса УК, который преподносит традиционные темы нетрадиционным способом, а освоившись с этим материалом – переходить к УК+ и в конечном итоге к ВКС. Если студенты не имеют опыта программирования на C (например, они программировали только на Java), то

вы можете потратить несколько недель на чтение лекций о C, а затем переходить к чтению курса УК или ВКС.

Таблица 1. Пять категорий курсов о компьютерных системах, основанных на книге «Компьютерные системы: архитектура и программирование». Курс ВКС+ – это курс 15-213 в университете Карнеги–Меллона.

Примечание: символ ⊙ означает частичный охват главы, как то: (1) только аппаратная часть; (2) без динамического распределения памяти; (3) без динамического связывания; (4) без представления данных с плавающей точкой

Глава	Тема	Курс				
		УК	УК+	ВКС	ВКС+	СП
1	Экскурс в компьютерные системы	•	•	•	•	•
2	Представление данных	•	•	•	•	⊙(4)
3	Машинный язык	•	•	•	•	•
4	Архитектура процессора	•	•			
5	Оптимизация кода		•	•	•	
6	Иерархия памяти	⊙(1)	•	•	•	⊙(1)
7	Связывание			⊙(3)	⊙(3)	•
8	Управление исключениями			•	•	•
9	Виртуальная память	⊙(2)	•	•	•	•
10	Ввод/вывод на уровне системы				•	•
11	Сетевое программирование				•	•
12	Параллельное программирование				•	•

Наконец, отметим, что курсы УК+ и СП могут образовать хорошую последовательность из двух семестров (или четверти и семестра). Или же можно подумать о чтении курса ВКС+ как состоящего из ВКС и СП.

Преподавателям: примеры лабораторных работ в классе

Курс ВКС+ в университете Карнеги–Меллона получил очень высокие оценки от студентов. Медианный балл 5,0/5,0 и средний балл 4,6/5,0 являются типичными оценками студентов курса. Основными достоинствами студенты называют забавные, увлекательные и актуальные лабораторные работы, которые доступны на веб-странице книги. Вот примеры лабораторных работ, которые поставляются с книгой.

Представление данных.

Эта лабораторная работа требует от студентов реализовать простые логические и арифметические функции с использованием строго ограниченного подмножества языка C. Например, они должны вычислить абсолютное значение числа, используя только битовые операции. Эта лабораторная работа помогает студентам понять представление типов данных в языке C на двоичном уровне и особенности побитовых операций.

Двоичные бомбы.

Двоичная бомба – это программа, которая передается студентам в виде скомпилированного файла. При запуске она предлагает пользователю ввести шесть разных строк. Если при вводе будет допущена ошибка, то бомба «взрывается» – выводит сообщение об ошибке и регистрирует событие на сервере оценки. Студенты должны «обезвредить» свои уникальные бомбы, дизассемблировав

программы и определив, как должны выглядеть эти шесть строк. Лабораторная работа учит студентов понимать язык ассемблера, а также заставляет их научиться пользоваться отладчиком.

Переполнение буфера.

Студенты должны изменить поведение двоичного выполняемого файла, используя уязвимость переполнения буфера. Эта лабораторная работа учит студентов осторожности обращения со стеком и показывает опасности кода, уязвимого для атак переполнения буфера.

Архитектура.

Некоторые домашние задания из главы 4 можно объединить в лабораторную работу, где студенты изменяют HCL-описание процессора, добавляя новые инструкции, изменяя политику прогнозирования ветвлений или добавляя и удаляя обходные пути и регистрируя порты. Полученные процессоры можно моделировать и запускать с помощью автоматических тестов, которые обнаруживают большинство возможных ошибок. Эта лабораторная работа позволяет студентам познакомиться с захватывающими аспектами проектирования процессоров, не требуя полного знания языков логического проектирования и описания оборудования.

Производительность.

Студенты должны оптимизировать производительность основных функций приложения, таких как свертка или транспонирование матриц. Эта лабораторная работа наглядно демонстрирует свойства кеш-памяти и дает студентам опыт низкоуровневой оптимизации программ.

Кеш.

Эта лабораторная работа является альтернативой лабораторной работе «Производительность». В ней студенты должны написать симулятор кеша общего назначения, а затем оптимизировать базовые функции программы транспонирования матрицы так, чтобы минимизировать количество промахов кеша. При этом мы используем инструмент Valgrind для трассировки реальных адресов.

Командная оболочка.

Студенты создают свою программу командной оболочки Unix с поддержкой управления заданиями, включая комбинации клавиш **Ctrl+C** и **Ctrl+Z**, а также команды `fg`, `bg` и `jobs`. В этой лабораторной работе учащиеся впервые знакомятся с параллельным программированием и получают четкое представление об управлении процессами в Unix, сигналах и их обработке.

Распределение памяти.

Студенты реализуют свои версии `malloc`, `free` и (необязательно) `realloc`. Эта лабораторная работа помогает студентам получить четкое представление о структуре и организации данных и требует от них оценки различных компромиссов между эффективностью потребления памяти и времени выполнения.

Прокси.

Студенты реализуют параллельный веб-прокси, находящийся между браузером и остальной частью Всемирной паутины. Эта лабораторная работа знакомит студентов с такими темами, как веб-клиенты и серверы, и связывает воедино многие концепции курса, такие как порядок следования байтов, файловый ввод/вывод, управление процессами, сигналы, обработка сигналов, отображение памяти, сокеты и параллельное выполнение. Студентам нравится видеть, как работают их программы, обслуживающие реальные веб-браузеры и веб-серверы.