Азбука халтурщика-ARMатурщика
разработка встраиваемых систем

основы бытовой автоматики,
систем управления и сбора данных

© ruOpenWrt
© HackSpace «Чебураторный завод»
© Консорциум хоббитов России
© Bill Collis (Часть I)
© Joe Martin, Craig Libuse (Часть V)

Оглавление

Введение **2**

I Введение в практическую электронику **3**

An Introduction to Practical Electronics, Microcontrollers and Software Design © Bill Collis 4

1 1 Введение в практическую электронику 13 **5**

1.1 Ваше обучение по специальности «Технология» 6

1.1.1 Цели обучения технологиям Ново-Зеландской программы 6

1.2 Ключевые компетенции Ново-Зеландской программы 7

2 Вводная электронная схема **10**

2.1 Где купить комплектующие? 10

2.2 Использование мультиметра 14

2.3 2.2 Определение сопротивления резистора по цветовому коду 16 16

2.4 2.3 Светодиоды 17 . . . 16
2.5 2,4 Некоторые технические характеристики светодиода 17 . . . 17
2.6 2.5 Задание на исследование светодиода 17 . . . 17
2.7 2.6 Добавление выключателя в схему 18 . . . 17
2.8 2.7 Задание на установку выключателя 18 . . . 17
2.9 2,8 Важные понятия схемы 19 . . . 17
2.10 2.9 Изменение величины сопротивления 19 . . . 18
2.11 2.10 Добавление транзистора в схему 20 . . . 18
2.12 2.11 Чтение схем 21 . . . 18
2.13 2.12 Входная цепь — LDR 22 . . . 18
2.14 2.13 Рабочая схема датчика темноты 23 . . . 18
2.15 2,14 Защитные цепи - использование диода 24 . . . 19
2.16 2.15 Задача исследования диода 24 . . . 19
2.17 2.13 Финальная схема датчика темноты 23 . . . 19

3 3 Вводное конструирование печатной платы 26 20

4 4 Пайка, припой и паяльники 41 21

5 5 Введение в теорию электроники 49 22

6 6 Введение в электронику микроконтроллера 63 23

7 7 Входные цепи микроконтроллера 91 24

8 8 Обзор программирования 104 25

9 9 Введение в поток выполнения программы 112 26

10 10 Вводное программирование: использование подпрограмм 126 27

11 11 Вводное программирование: Использование переменных 134 28

12 12 Основные дисплеи 161 29

13 13 Проект портативного аудиоусилителя на TDA2822M 174 30

14 14 Основы логического программирования 187 31

15 15 Разработка алгоритма: Система сигнализации 202 32

16 16 Основы теории цепей постоянного тока 215 33

17 17 Основы планирования проекта 236 34

18 18 Пример дизайна системы: Таймер клеевого пистолета 268 35

19 19 Основные интерфейсы и их программирование 273 36

20 20 Основы интерфейса аналого-цифрового преобразования 295 37

21 21 Основы проектирования системы 314 38

22 22 Основы проектирования системы: Тайм-трекер 317 39

23 23 Основы вычислений времени 330 40

24 24 Основы строковых переменных 340 41

25 25 Силовые интерфейсы 353 42

26 26 Теория источников питания 370 43

27 27 Типичные вопросы тестирования 2011/12/13 годов 395 44

28 28 Расширенное программирование: Массивы 397 45

29 29 Подтягивающие резисторы AVR 402 46

30 30 Дополнительноподключение клавиатуры 403 47

31 31 Тонкости циклов Do-Loop & While-Wend 417 48

32 32 Подключение двигателя постоянного тока 423 49

33 33 Пример расширенной системы: Будильник 452 50

34 34 Резистивный сенсорный экран 468 51

35 35 Пример проектирования системы: Регулятор температуры 475 52

36 36 Расширенное программирование: Машины состояний 478 53

37 37 Переработанный проект будильника 501 54

38 38 Студенческий проект: Расширенный оконный контроллер 514 55

39 39 Альтернативные техники кодирования машин состояния 524 56

40 40 Сложно: последовательная связь 526 57

41 41 Цифровой радиоканал 597 58

42 42 Введение в I2C 617 59

43 43 Студенческий проект: Таймер полива теплицы 631 60

44 44 Проект Велосипедного аудиоусилителя 642 61

45 45 Графические LCD 648 62

46 46 Проект Отслеживания температуры GLCD 660 63

47 47 Прерывания 672 64

48 48 Таймеры/Счётчики 692 65

49 49 Проект скроллинга графического LED дисплея: массивы и таймеры 698 66

50 50 Проект медицинского прибора: реализация таймера 709 67

51 Проект часов на 7-сегментном индикаторе реализация на сдвоенном таймере 68

52 52 ИС драйвера дисплея MAX 7219/7221 739 69

53 53 Подключение через мобильную связь: ADH8066 744 70

54 54 Передача данных через Internet 778 71

55 55 Задание: математика в реальном мире 816 72

56 56 Цветной графический LCD на основе SSD1928 825 73

57 57 Светофор: помощь и решение 865 74

58 58 Компьютерное программирование: низкоуровневые детали 869 75

59 59 USB-программатор: USBASP 876 76

60 60 Программатор USBTinyISP 877 77

61 61 Программирование на Си и AVR 881 78

62 62 Объектно-Ориентированное Программирование (ООП) на C_+^+ и AVR 929 79

63 63 Современные (2014) отладочные платы на AVR 953 80

64 64 Eagle: создание собственной библиотеки 970 81

65 65 Практические методы 979 82

66 66 ЧПУ 990 83

67 67 Индекс 1008 84

II Основы электроники 85

68 Линейные схемы на пассивных элементах, основы электротехники 87

69 Симуляция и расчет схем в ngSPICE 88

70 KiCAD 89
70.1 Установка под ⊞Windows . . . 91
70.2 Установка под Linux . . . 92
70.3 Создание проекта в менеджере проектов kicad . . . 93
70.4 Создание принципиальной схемы в eeschema (часть 1) . . . 95
70.5 Инструмент Добавить компоненты . . . 97
70.6 eeschema: редактор электрических схем . . . 99
70.7 Библиотеки компонентов . . . 99
70.8 gerbview: просмотр фотошаблонов . . . 100
70.9 Программа Wings3D для создания 3D моделей . . . 100
70.9.1 Установка Wings3D под ⊞Windows . . . 100

71 Простейшие полупроводниковые элементы **101**
71.1 Оптоэлектроника . . . 101
71.2 Схемы на биполярных транзисорах . . . 101
71.3 Схемы на на полевых транзисорах . . . 101

72 Операционные усилители **102**

73 Источники питания **103**
73.1 Батарейное питание . . . 103
73.2 Линейные стабилизаторы . . . 103
73.3 Импульсные преобразователи на ШИМ-контроллерах . . . 103
73.4 Цепи защиты и гашения кондуктивных помех . . . 103

74 Цифровая электроника **104**

75 Компьютерные интерфейсы **105**
75.1 Поколение 90х: COM, LPT, ISA . . . 106
75.1.1 Резервный программатор AVR “пять проводков” . . . 106
75.2 Сеть CAN . . . 106
75.3 Интерфейсные модули USB . . . 106
75.3.1 Универсальный высокоскоростной конвертер FTDI FT2232H . . . 106
75.3.2 JTAG-адаптер . . . 106
75.3.3 Отладочный модуль CAN . . . 106
75.4 Интерфейсные модули Ethernet . . . 106

76 ПЛИС **107**

77 Датчики **108**

78 Электропривод и исполнительные устройства **109**

III Основы конструирования РЭС 110

79 Пакеты моделирования на основе OpenFOAM **111**

80 Обеспечение теплового режима **112**

81 Электромагнитная совместимость **113**
81.1 Кондуктивные помехи . . . 113
81.2 Компоновочные модели и оптимизация кабельной сети . . . 113

IV Технология РЭС 114

82 Инструменты и оборудование **115**
82.1 Паяльное оборудование . . . 115
82.1.1 Паяльник . . . 115
82.1.2 Паяльная станция . . . 117
82.2 JTAG-адаптер . . . 120
82.3 Отладочные платы . . . 120
82.3.1 Arduino /Atmel Mega AVR8/ . . . 120
82.3.2 Cortex-Mx . . . 120

82.3.3 CubieBoard /Cortex-A8 AllWinner A10/ . . . 120
82.3.4 Raspberry Pi /ARM11 BCM3032/ . . . 120
82.3.5 BlackSwift /MIPS/ . . . 120
82.3.6 VoCore /MIPS/ . . . 120
82.4 Радиомонтажный инструмент . . . 120
82.4.1 Pro'sKit . . . 121
82.5 Измерительное оборудование . . . 133
82.5.1 Мультиметр . . . 133
82.5.2 Осциллограф . . . 137
82.5.3 Логический анализатор . . . 137
82.5.4 Генератор сигналов . . . 137
82.5.5 Рыльцеметр RLC . . . 137
82.6 Электроинструмент . . . 137
82.6.1 Дрелъ . . . 138
82.6.2 Лобзик . . . 141
82.6.3 Жвигатель . . . 142

83 Трассировка плат и подготовка производства в KiCAD 145
83.1 Технология ЛУТ (Лазерный УТюг) . . . 145
83.2 Технология фоторезиста . . . 145
83.3 Формат Gerber и подготовка промышленного производства . . . 145

84 FreeCAD 146
84.1 Установка под ⊞Windows . . . 149
84.2 Чертеж . . . 151

84.3 Эскиз 151
84.4 Деталь 151
84.5 Сборка 151
84.6 Автогенерация конструкторской докуметации 151
84.7 Скрипты и пользовательские расширения 151

V Станки 152

85 A special note to engineers reading this book 154

86 Machining for engineen and engineering for machinists 155

87 Navigating the seas of machining 156

88 looking at engineering from the craftsman's perspedive 157

89 Seeing produdion from the point of view of both the engineer and machinist 159

90 About the Joe Martin 160

91 Dedication 162

92 Правила техники безопасности 164

93 Предисловие 165

94 Введение **166**

95 Галерея фотографий проектов **167**

96 Профиль Мастера — Скотти Хьюитт... 24 **168**

97 РАЗДЕЛ I — О станочной обработке **169**

97.1 1. Получение информации об обработке..... 27 170
97.2 2. Вам нужен токарный станок. фрезерный станок или оба ? 29 170
97.3 3. Материалы для металлообработки... 39 170
97.4 4. Процессы для металлообработки 170
97.5 5. Использование ручных инструментов и абразивов 57 170
97.6 6. Режущий инструмент для металлообработки 170
97.7 7. Измерительные инструменты и измерения... 85 170
97.8 8. Смазочно-охлаждающие жидкости и масла... 99 170
97.9 9. Основные термины мехобработки 101 170
97.1010. Смазка и обслуживание станка 107 170

VI SECTION 2-LATHE OPERATIONS **171**

98 Craftsman Profile-Jerry Kieffer 112 **172**

99 Jerry Kieffer's Flying Pendulum Clock 114 **173**

100. Lathe work holding 115 **174**

10 12. Lathe operating instnlclions 121 175

10 23. Tail slock lools and operations 141 176

10 34. Ri ser blocks 145 177

10 45. Supporting long or thin work 149 178

10 56. Gelling started in thread cutting. 157 179

10 67. Knurled fini shes. 167 180

10 78. Watchmaking and clockmaking tools 171 181

10 89. Milling operations on a lathe 177 182

VII SECTION 3-MIll1NG OPERATIONS 183

109 Craftsman Profile- Augie Hiscano 180 184

110 1. Holding parts for milling. 183 185

111 2. Mill operating in structions.. . . . . 191 186

112 3. Squaring up a block 205 187

113 4. The rotary table and index ing attachment 209 188

114 5. Gears and Geartrains 2 19 189

115 6. Accessories for milling 190

116 • Horizontal milling conversion 235 191

117 • Rotary column attachmcnt 237 192

VIII SECTION 4-OTHER MACHINING TOPICS 193

118 Craft sman Profiles-Dan Lutz and Paul White . 242 194

119 1. Setting up a small workshop 247 195

120 2. Lathe and mill alignment and adj ustments .. 25 1 196

121 3. Enginering drawings 259 197

122 4. Frequently asked questions 265 198

123 5. Making a bus iness o ut of a hobby .. .Joe Martin's 199

124 and Sherline's story 273 200

125 6. Using CNC in a home shop 309 201

IX SECTION S-PROJECTS AND RESOURCES 202

126 Cra ftsman Profi le-Bob Bres lauer 3 16 203

127 Machini st's tips 3 18 204

128 1. Plans and projects you can build 3 19 205

129 1. Miniature Tap Handle ... a beginning 206

130 project you can use in your shop 32 1 207

131 2. Mill vise "sofe jaws. 325 208

132 3. Lay ing out a circul ar hole pattern 209

133 fordrilling, a handy skill to learn. 327 210

134 4. "Millie"... a small oscillating steam 211

135 engine by Ed Warren, a simple project 212

136 from the pages of Modeltec magazine) 329 213

137 5. Ordering plans for the Little Ange l 214

138 hit 'n miss engine ... an advanced 215

139 machining project by Bob Shores 333 216

140 2. Contests and awards for tabletop machini sts . . 335 217

141 3. Exploded views and part number li sting 218

142 Model 4000 and 4400 Lathes 340 219

143 Model 5000 and 5400 Milling machines . . . 341 220

144 Model 2000 8-Direction Mill Column 342 221

145 Part number listing 343 222

146 4. A simple RPM gage for your latheor mill 345 223

147 Harold Cli sby and the first Sherline lathe 346 224

148 S. Index 347 225

149 Conversion faclors 350 226

X Станочное оборудование 227

150 Маркировка моделей станков производства СССР 230

151 1A616: станок токарно-винторезный 235
151.1 Назначение и области применения . . . 238
151.2 Распаковка и транспортировка . . . 238
151.3 Фундамент станка, монтаж и установка . . . 238
151.4 Подготовка станка к первоначальному пуску . . . 238
151.5 Паспортные данные . . . 238
151.6 Описание основных узлов . . . 238
151.7 Смазка . . . 238
151.8 Первоначальный пуск . . . 238
151.9 Указания по технике безопасности . . . 238
151.10 Настройка . . . 238
151.11 Регулирование . . . 238
151.12 Ведомость комплектации . . . 238

152 Основы ЧПУ и цифрового производства 239
152.1 CAM-пакеты для FreeCAD . . . 239

XI Основы теории систем автоматического управления 240

153 Математический аппарат 241
153.1 Передаточная функция . . . 241

153.2 Устойчивость САУ . . . 241
153.3 Сети Петри . . . 241
153.4 Автоматы Маркова . . . 241

154 Релейное управление **242**

155 Пропорциональные САУ **243**

156 ПИДn-регуляторы **244**

XII Разработка ПО для встраиваемых систем 245

157 Архитектура программных систем **246**
157.1 Литература . . . 246
157.2 Ортогональность . . . 246
157.2.1 Что такое ортогональность? . . . 247
157.2.2 Неортогональная система . . . 247
157.2.3 Преимущества ортогональности . . . 248
157.2.4 Увеличение производительности . . . 249
157.2.5 Снижение риска . . . 249
157.2.6 Проектные группы . . . 250
157.2.7 Проектирование . . . 251
157.2.8 Инструментарии и библиотеки . . . 252
157.2.9 Написание текста программы . . . 254
157.2.10 Тестирование . . . 255

157.2.11 Документация 256
157.2.12 Жизнь в условиях ортогональности 256

158 Вспомогательные скрипты на языке Python 258
158.1 Установка под ⊞Windows 259
158.2 Запуск 261
158.3 Дополнительные материалы 266

159 Make: управление сборкой проектов 267

160 VCS: системы контроля версий 268
160.1 CVS 268
160.2 Subversion 268
160.3 Git 268
160.3.1 GitHub 268

161 Основы Си и C_+^+ 269
161.0.2 Установка MinGW (win32) 269
161.1 Особенности C_+^+ в embedded 269

162 LLVM и разработка собственных компиляторов 270

163 Лексический и синтаксический анализ 271
163.1 Лексер и лексический анализ, утилита flex 272
163.2 Компилятор Паскаля 275

163.3 Дополнительная литература 275

164 Сборка кросс-компилятора GNU toolchain 276

XIII Микроконтроллеры Cortex-Mx 277

165 Отладочные платы 278
165.1 STM32DISCOVERY /Cortex-M3 STM32F103/ 278
165.2 STM32F4DISCOVERY /Cortex-M4 STM32F407/ 278

XIV Периферия 279

XV Встраиваемый emLinux 280
165.3 Загрузчик syslinux 282
165.3.1 Закачка 282
165.3.2 Установка под ⊞Windows на флешку 283
165.3.3 **syslinux.cfg** 284

166 azLinux 287
166.1 Требования к системе сборки (**BUILD**-хост) 289
166.2 Понятие пакет 290
166.3 Клонирование проекта azLinux 292
166.4 Общий порядок сборки 293

166.5 Фиксация переменных . . . 294
166.6 dirs: Создание дерева каталогов . . . 294
166.7 gz: Загрузка архивов исходников . . . 298
166.8 APP: Приложение . . . 299
166.9 HW: Поддерживаемое железо . . . 300
166.9.1 i386 . . . 301
166.9.2 ARM . . . 305
166.9.3 MIPS . . . 305
166.10 CPU: Конфигурации процессоров . . . 306
166.10.1 i386 . . . 306
166.10.2 ARM . . . 307
166.10.3 MIPS . . . 307
166.11 Пакеты . . . 308
166.11.1 **versions.mk**: Версии пакетов . . . 308
166.11.2 tc: сборка кросс-компилятора . . . 309
166.11.3 core: сборка основной системы . . . 314
166.11.4 libs: сборка библиотек **${LIBS}** . . . 326
166.11.5 apps: сборка прикладных пакетов **${APPS}** . . . 326
166.11.6 user: сборка пользовательского кода . . . 326
166.11.7 root: формирование корневой файловой системы . . . 326
166.11.8 boot: сборка загрузчика syslinux/grub/uboot . . . 329
166.12 qmu: запуск собранной системы в эмуляторе . . . 329
166.13 netboot: Сетевая загрузка . . . 331
166.14 Прошивка на устройство . . . 331
166.15 RT-патч . . . 331

167 Библиотека libSDL2 332

168 OpenGL 333

169 RealTime: режим жесткого реального времени в Linux 334

170 netboot: Сетевая загрузка 335

171 cross 336

172 BuildRoot 337

173 Особенности OpenWrt 338
173.1 Реализация microGUI 338

174 Приложения для X Window 339

175 Программирование сетевых приложений 340

176 Сборка кросс-компиляторя GNU мальтийским крестом 341

XVI IDE 342

177 eclipse 345
177.1 Установка ECLIPSE под Windows 347
177.2 Установка ECLIPSE под Linux 347

177.3 Установка CDT . . . 348
177.4 Установка PyDev . . . 348
177.5 Установка TeXlipse . . . 349
177.6 Редактирование файлов в формате XML и производных . . . 349
177.7 Проверка орфографии . . . 350

178 Code::Blocks **353**

179 (g)Vim **354**
179.1 Установка под ⊞Windows . . . 356
179.2 Выход из (g)Vim . . . 359
179.2.1 Выход с автосохранением . . . 359
179.3 Переход в режим редактирования . . . 359
179.4 Переход в режим команд . . . 359
179.5 Запись редактируемого файла . . . 359
179.6 Перезагрузка файла . . . 360
179.7 Отмена последних изменений (undo) . . . 360

XVII Замечания для авторов **361**
179.8 Набор репозиториев на GitHub . . . 362
179.9 Верстка в LaTeX . . . 362

XVIII Подготовка публикаций в LaTeX **364**
179.10 Установка MikTeX под ⊞Windows . . . 366

179.10 Структура документа . . . 367
179.11.1 Заголовочный файл или блок . . . 367
179.11.2 Стили документа . . . 367
179.11.3 Пакеты . . . 367
179.11.4 Автор и название . . . 367
179.11.5 Верстка титульных страниц . . . 367
179.11.6 Оглавление . . . 367
179.12 Верстка слайдов . . . 367
179.13 Список литературы и цитирование . . . 367
179.14 Команды секционирования: часть, глава, раздел,.. . . . 369
179.15 Таблицы . . . 369
179.16 Формулы . . . 369
179.17 Перекрестные ссылки и гипессылки . . . 369
179.18 Листинги скриптов и текстовых данных . . . 369
179.19 Подготовка иллюстраций . . . 369
179.19.1 Графики GNUPLOT . . . 370
179.19.2 Схемы и графы в GraphViz . . . 371
179.19.3 PGF/TikZ . . . 371
179.19.4 GLE . . . 371
179.19.5 Xy-pic . . . 372
179.20 Верстка электронных изданий . . . 372

XIX Символьная и численная математика 373

180 Общие сведения о компьютерной математике 375

181 Пакет Maxima 378
181.1 Установка Maxima под ⊞Windows 378
181.2 Калькулятор 379

XX Куча 380

Список литературы 381

Введение

Первоначально этот материал задумывался как комплект документации к платам BlackSwift и VoCore, но постепенно превратился в толстенный учебник для студентов ВУЗов и научных работнков по специализациям, связанным с применением цифровой электроники и компьютерной техники.

Большой упор был сделан на использование открытого некоммерческого программного обеспечения, с целью удешевления учебного процесса, уменьшения себестоимости ваших проектов[1], и стимулирования вашего участия в развитии этих программных пакетов.

Лицензия на эту книгу пока не выбрана, так что она пока просто пишется в духе OpenSource: любой может использовать ее часть, изменять или дополнять, до тех пор, пока не накладываются какие-либо административные, финансовые или юридические ограничения на распространение и развитие оригинальной версии или ее открытых форков.

Приглашаем всех желающих участвовать в развитии этого учебного пособия на форум ruOpenWrt, нам нужна обратная связь по качеству материала, результаты тестирования на вас или ваших студентах, дополнения и замечания.

Мы признательны Bill Collis за разрешение использовать материалы его книги «An Introduction to Practical Electronics, Microcontrollers and Software Design»[**bcollis**] в русскоязычном варианте «Азбуки» (Часть I), и конечно он вполне заслуженно включен в основные соавторы этой книги.

[1] вряд ли ли у вас окажется лишняя пачка килобаксов на покупку пары коммерческих САПР, по крайней мере пока ваш стартап не взлетит в Top$100K

Часть I

Введение в практическую электронику

Эта часть основана на книге:

An Introduction to Practical Electronics, Microcontrollers and Software Design
Second Edition, 01 May-2014
© Bill Collis
www.techideas.co.nz

Мы признательны автору за разрешение использовать материалы его книги в русскоязычном варианте «Азбуки», и конечно он вполне заслуженно включен в основные соавторы этой книги.

We are grateful to the author for permission to use materials of his book in the russian version of «Azbuka», and of course he was deservedly included in the main co-authors of this book.

```
From: Bill Collis <Bill.Collis@..........nz>
Date: 2014-11-24 0:53 GMT+04:00
Subject: Electronis Book
To: "dponyatov@gmail.com" <dponyatov@gmail.com>

Hi Dmitry
thanks for your email.
I am looking at the future of the book myself and thinking I will open source
it. If you will only be in using it in Russian language then that is ok and you
need to reference the original book.

Thanks
Bill
```

Глава 1

1 Введение в практическую электронику 13

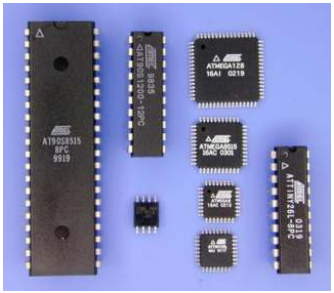
Эта книга ©[1] имеет слеующий ряд основных направлений:

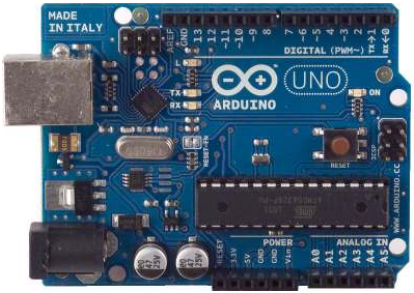
- Распознавание электронных компонентов и их правильное использование
- Наработка цельного набора компетенций по основам электроники
- Использование макетных плат
- Навыки ручной пайки
- Использование закона Ома для выбора токоограничивающих резисторов
- Делитель напряжения
- Использование EDA CAD[2] для разработки и подготовки производства печатных плат

[1] оригинал: [**bcollis**] B.Collis The Introduction to Practical Electronics...

[2] [E]lectronic [D]esign [A]utomation, САПР автоматизации проектирования электроники

- Программирование микроконтроллеров и их сопряжение с внешними устройствами
- Использование транзистора в режиме ключа
- Теория источников питания
- Принципы и схемы электропривода
- Навыки отладки схем, их тестирования и испытаний
- Следование принципам обучения через практику
- Безопасные приемы работы





1.1 Ваше обучение по специальности «Технология»

1.1.1 Цели обучения технологиям Ново-Зеландской программы

- **Технологическая практика**
  - **Чоткость**: разработка ясных описаний для ваших технологических проектов.
  - **Планирование**: думать прежде чем делать, и использовать во время работы документацию: блок-схемы, принципиальные схемы, чертежи разводки плат, диаграммы и эскизы.

  - **Наработка навыков**: сборка, отладка и тестирование электронных схем, проектирование и изготовление печатных плат, написание программ для микроконтроллеров.
- **Технологические знания**
  - **Моделирование**: прежде чем строить готовое электронное устройство, сначала важно понять как оно работает путем моделирования и/или макетирования аппаратного и программного обеспечения.
  - **Технологические продукты**: знания о компонентах и их характеристиках.
  - **Технологические системы**: электронное устройство является более, чем набором компонентов, это функционирующая система с входами, выходами и контролирующим процессом.
- **Природа технологии**
  - **Значение технологических достижений**: знания об электронных компонентах, особенно микроконтроллерах, как основе современных технологий.
  - **Роль технологии в обществе**: электронные устройства в настоящее время играют центральную роль в инфраструктуре нашего современного общества; подчинили ли они нас себе, как они изменили нашу жизнь?

1.2 Ключевые компетенции Ново-Зеландской программы

- **Знания**: для меня предметом технологии является все что относится к знанию. Моя цель: заставить студентов понимать технологии, заложенные в электронные устройства. Для достижения этого понимания студенты должны активно учиться[3] в работе на самом раннем этапе, чтобы они могли построить собственное понимание предмета и пойти дальше, чтобы стать хорошими решалами про-

[3] в оригинале **enage**, англо-калька с себуанского, NZ

блем. В начале обучения электронике это требует от студентов восприимчивости к инструкциям, которые им дают, и поиск ясности, когда они не понимают их.

Для этого на занятиях рассматриваются много новых и различных элементов знаний, и студентам выдаются задания на решение проблем, чтобы помочь им мыслить логически. Копирование чужого ответа наказывается, но приветствуется совместная работа. В основе обучения лежит построение правильных концептуальных моделей и анализ в контексте "большой картины".

- **Взаимодействие**: работа в парах и группах, это важно как в классе, так и в любой другой ситуации в жизни; мы все должны договариваться и разделять ресурсы и оборудование с другими людьми; поэтому крайне важно активное общение и помощь друг другу.
- **Использование языка символов и текстов**: сердцем нашего предмета является язык, который мы используем для обмена информацией в электронных схемах, планах, алгоритмах и синтаксисе компьютерных языков программирования; так что способность распознавать и правильно использовать символы и диаграммы для работы, которую мы делаем, имеет критическое значение.
- **Самоконтроль**: студенты принимают на себя личную ответственность за собственное обучение; они принимают вызов, надеясь найти ответы в книгах или найти учителя, способного объяснить им, что делать. Это значит, что студенты должны взаимодействовать с рабочим материалом.

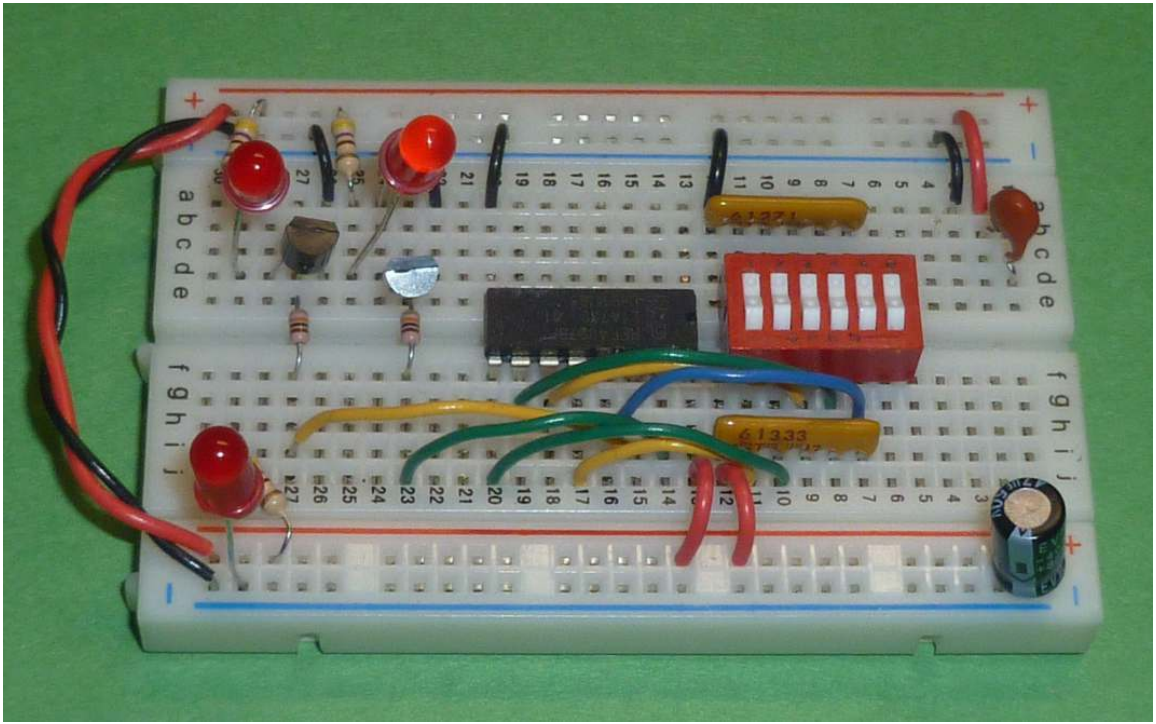
  Иногда ответы приходят легко, иногда нет; часто наша тема требует много проб и ошибок (в основном ошибок). Студенты должны знать, что у них будут трудные времена, пока не будет изучена бóльшая часть. И не сдаться в поиске понимания.
- **Участие и содействие**: мы живем в мире, который невероятно зависит от технологии, особенно электроники; студенты должны развивать осознание важности этой области человеческого творчества в нашей повседневной жизни, и понимать, что наши проекты имеют и социальную функцию, а не только техническую.

Глава 2

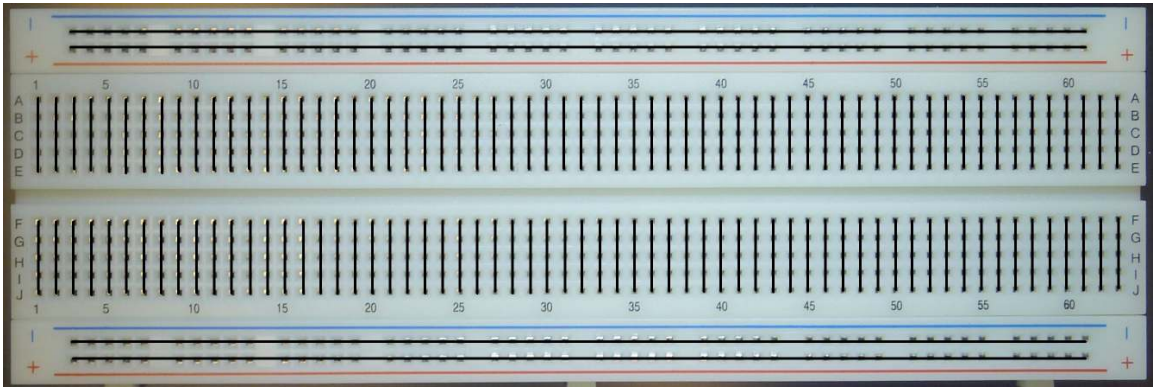
Вводная электронная схема

2.1 Где купить комплектующие?

В Новой Зеландии есть некоторое количество отличных поставщиков компонентов с разумными ценами, включающих www.surplustronics.co.nz, и www.activecomponents.com. Зарубежные поставщики, которых я использую, включают www.digikey.co.nz, www.sparkfun.com, ebay.com и aliexpress.com

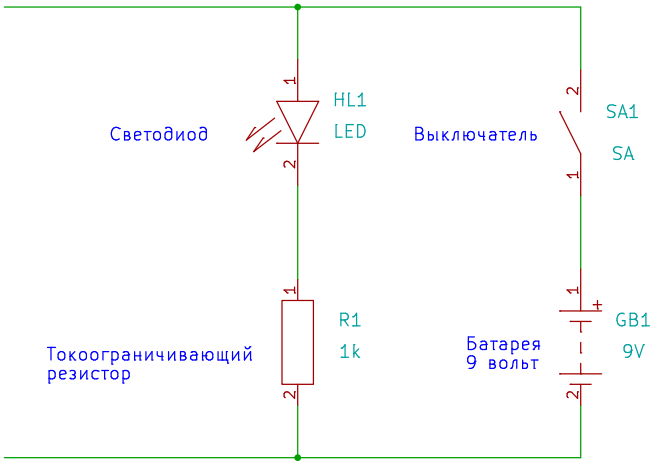


Breadboard ([б]еспаечная [м]акетная [п]лата , БМП, “вафля”) — пластмассовый блок с отверстиями и металлическими полоскóвыми зажимами, создающими соединения между элементами схемы. Отверстия расположены так, что компоненты и отрезки провода могут быть соединены вместе формируя схему, без использования паяльника. Верхние и нижние ряды, как правило, используются для шин питания, красный сверху для плюса, и внизу черный/синий для минуса (общий провод). На длинных вафлях шины питания поделены на отдельные сегменты, и требуют соединения короткими перемычками.



Эта схема 2.1 может быть собрана вот так 2.1, обратите внимание, что светодиод должен находиться в правильном положении. Если у вас есть светодиод и резистор, соединенные в замкнутый контур, светодиод должен загореться.

Принципиальная схема



Компоновка

Светодиоду требуется для включения около 2 В[1], батарея на 9 В, так что с напряжением все нормально. Но если вы подключите светодиод напрямую к батарее, он сгорит! Для светодиодов главный рабочий параметр — допустимый рабочий ток, обычно он не превышает 10..20 mA[2]. Так что 1 k[3] резистор используется для ограничения тока через светодиод.

Если вы отключите любой провод в схеме, она перестает работать, схема должна быть завершенной, чтобы электроны могли течь.

[1] [В]ольт
[2] 1 [м]иллиАмпер = 0.001 [А]мпер
[3] 1 [К]илоОм = 1000 Ом

2.2 Использование мультиметра

Возьмите мультиметр82.5.1 и измерьте напряжение на резисторе, близко ли оно к 7 В ? Также измерьте ток через диод, не превышает ли он допустимый ?

Напряжение измеряется в [В]óльтах подключением измерительного прибора (мультиметра) параллельно элементу, при этом нужно

- режим измерения включить на режим измерения постоянного напряжение $V-$/DCV (или переменного, обозначается как $V\sim$/ACV) а
- диапазон измерения выставить на максимальное значение напряжения
- Последовательно уменьшая диапазон измерения, найдите диапазон, в котором мультиметр показывает наибольшее количество знаков после запятой.

Если диапазон слишком большой, прибор покажет значение в районе 0, а если слишком низкий — выведет [1]. Обычно работают в диапазоне, соответствующем максимальому напряжению питания (в нашем случае 20 V[4]), иногда для слабых напряжений переключаясь на одну..две ступени ниже. Но — возможны случаи[5] когда напряжение в части схемы на порядки (в десятки..тысячи раз) превосходит напряжение питания.

Ток измеряется в [А]мпéрах подключением измерительного прибора (мультиметра) последовательно с элементом в разрыв цепи, при этом нужно

[4] [V]olt = [В]ольт

[5] в любых схемах содержащих индуктивности: катушки прóвода, трансформаторы, электродвигатели, динамики и т.п.

- режим измерения включить на режим измерения постоянного тока $A-$/DCA (или переменного, обозначается как $A\sim$/ACA) а
- диапазон измерения выставить на максимальное значение тока
- Последовательно уменьшая диапазон измерения, найдите диапазон, в котором мультиметр показывает наибольшее количество знаков после запятой.

Обратите внимание, что напряжение измеряется вольтметром на полностью собранной схеме, а для измерения тока нужно изменять схему, включая в нее амперметр. Если измерения тока нужно проводить на готовом устройстве, иногда ставят 2хконтактный джампер (как на материнских платах компьютеров): при измерении амперметр подключают к его контактам, а потом джампер замыкают специальной съемной проводящей перемычкой. Этом прием вы можете использовать в своих устройствах для регулярного измерения тока потребления, и расчета потребляемой мощности:

$$W_{\text{потребляемая}}[\text{Ватт}] = U_{\text{питания}}[\text{Вольт}] \times I_{\text{устройства/элемента}}[\text{Ампер}] \tag{2.1}$$

Переключив мультиметр в режим прозвонки (тéстера), можно проверить

- наличие электрического соединения между двумя точками обязательно отключенной от источника питания схемы,
- исправность диода или светодиода,
- исправность конденсатора большой емкости и
- любые другие случаи когда нужно определить что две точки электрически соединены между собой, постоянно или временно.

Если между щщупами мультиметра в режиме прозвонки[6] сопротивление протекающему току не превышает 1 КилоОм[7], раздается звуковой сигнал.

Если щщупы подключить к исправному разряженному конденсатору большой емкости ("электролиту"), раздастся короткий целчок или даже "пик".

Тестером также можно определить тип и цоколе́вку[8] транзистора; как это сделать описано в ??.

Диоды проверяются двумя подключениями в разной полярности — при красном щщупе на аноде (+) диода (в прямой полярности включения) диод проводит ток, а в обратной полярности нет (красный щщуп на катоде (-)).

Проверьте светодиод: отключите его из схемы, и проверьте как диод. Если светодиод исправен, в прямой полярности светодиод будет очень слабо светиться.

2.3 2.2 Определение сопротивления резистора по цветовому коду 16

.

2.4 2.3 Светодиоды 17

.

[6] = тестера

[7] см. паспорт на прибор

[8] "раскопытку"

2.5 2,4 Некоторые технические характеристики светодиода 17

.

2.6 2.5 Задание на исследование светодиода 17

.

2.7 2.6 Добавление выключателя в схему 18

.

2.8 2.7 Задание на установку выключателя 18

.

2.9 2,8 Важные понятия схемы 19

.

2.10 2.9 Изменение величины сопротивления 19

.

2.11 2.10 Добавление транзистора в схему 20

.

2.12 2.11 Чтение схем 21

.

2.13 2.12 Входная цепь — LDR 22

.

2.14 2.13 Рабочая схема датчика темноты 23

.

2.15 2,14 Защитные цепи - использование диода 24

.

2.16 2.15 Задача исследования диода 24

.

2.17 2.13 Финальная схема датчика темноты 23

.

Глава 3

3 Вводное конструирование печатной платы 26

Глава 4

4 Пайка, припой и паяльники 41

Глава 5

5 Введение в теорию электроники 49

Глава 6

6 Введение в электронику микроконтроллера 63

Глава 7

7 Входные цепи микроконтроллера 91

Глава 8

8 Обзор программирования 104

Глава 9

9 Введение в поток выполнения программы 112

Глава 10

10 Вводное программирование: использование подпрограмм 126

Глава 11

11 Вводное программирование: Использование переменных 134

Глава 12

12 Основные дисплеи 161

Глава 13

13 Проект портативного аудиоусилителя на TDA2822M 174

Глава 14

14 Основы логического программирования 187

Глава 15

15 Разработка алгоритма: Система сигнализации 202

Глава 16

16 Основы теории цепей постоянного тока 215

Глава 17

17 Основы планирования проекта 236

Глава 18

18 Пример дизайна системы: Таймер клеевого пистолета 268

Глава 19

19 Основные интерфейсы и их программирование 273

Глава 20

20 Основы интерфейса аналого-цифрового преобразования 295

Глава 21

21 Основы проектирования системы 314

Глава 22

22 Основы проектирования системы: Тайм-трекер 317

Глава 23

23 Основы вычислений времени 330

Глава 24

24 Основы строковых переменных 340

Глава 25

25 Силовые интерфейсы 353

Глава 26

26 Теория источников питания 370

Глава 27

27 Типичные вопросы тестирования 2011/12/13 годов 395

Глава 28

28 Расширенное программирование: Массивы 397

Глава 29

29 Подтягивающие резисторы AVR 402

Глава 30

30 Дополнительноподключение клавиатуры 403

Глава 31

31 Тонкости циклов Do-Loop & While-Wend 417

Глава 32

32 Подключение двигателя постоянного тока 423

Глава 33

33 Пример расширенной системы: Будильник 452

Глава 34

34 Резистивный сенсорный экран 468

Глава 35

35 Пример проектирования системы: Регулятор температуры 475

Глава 36

36 Расширенное программирование: Машины состояний 478

Глава 37

37 Переработанный проект будильника 501

Глава 38

38 Студенческий проект: Расширенный оконный контроллер 514

Глава 39

39 Альтернативные техники кодирования машин состояния 524

Глава 40

40 Сложно: последовательная связь 526

Глава 41

41 Цифровой радиоканал 597

Глава 42

42 Введение в I2C 617

Глава 43

43 Студенческий проект: Таймер полива теплицы 631

Глава 44

44 Проект Велосипедного аудиоусилителя 642

Глава 45

45 Графические LCD 648

Глава 46

46 Проект Отслеживания температуры GLCD 660

Глава 47

47 Прерывания 672

Глава 48

48 Таймеры/Счётчики 692

Глава 49

49 Проект скроллинга графического LED дисплея: массивы и таймеры 698

Глава 50

50 Проект медицинского прибора: реализация таймера 709

Глава 51

Проект часов на 7-сегментном индикаторе реализация на сдвоенном таймере

Глава 52

52 ИС драйвера дисплея MAX 7219/7221 739

Глава 53

53 Подключение через мобильную связь: ADH8066 744

Глава 54

54 Передача данных через Internet 778

Глава 55

55 Задание: математика в реальном мире 816

Глава 56

56 Цветной графический LCD на основе SSD1928 825

Глава 57

57 Светофор: помощь и решение 865

Глава 58

58 Компьютерное программирование: низкоуровневые детали 869

Глава 59

59 USB-программатор: USBASP 876

Глава 60

60 Программатор USBTinyISP 877

Глава 61

61 Программирование на Си и AVR 881

Глава 62

62 Объектно-Ориентированное Программирование (ООП) на C_{+}^{+} и AVR 929

Глава 63

63 Современные (2014) отладочные платы на AVR 953

Глава 64

64 Eagle: создание собственной библиотеки 970

Глава 65

65 Практические методы 979

Глава 66

66 ЧПУ 990

Глава 67

67 Индекс 1008

Часть II

Основы электроники

Здесь идет список ссылок на онлайн лекции в edX, Coursera, и т.п.

Глава 68

Линейные схемы на пассивных элементах, основы электротехники

Глава 69

Симуляция и расчет схем в ngSPICE

Глава 70

KiCAD

[1]

[2]

KiCad — распространяемый по лицензии GNU GPL программный комплекс САПР EDA с открытыми исходными текстами, предназначенный для разработки электрических схем и печатных плат.

Кроссплатформенность компонентов KiCad обеспечивается использованием библиотеки wxWidgets. Поддерживаются операционные системы Linux, Windows NT 5.x, FreeBSD и Solaris.

Разработчик — Жан-Пьер Шарра (фр. Jean-Pierre Charras), исследователь в LIS (фр. Laboratoire des Images et des Signaux — Лаборатория Изображений и Сигналов) и преподаватель электроники и обработки изображений в фр. IUT de Saint Martin d'Hères (Франция).

[3]

Этот раздел познакомит Вас с основами использования системы KiCad. Он содержит информацию о всех шагах создания простой печатной платы: от рисования электрической схемы до печати готового рисунка платы. Вам будут представлены различные возможности KiCad и предложены эффективные пути решения различных задач.

Руководство пользователя, поставляемое вместе с KiCad, содержит значительно больше информации, чем этот урок. Ознакомтесь с ним, чтобы узнать больше об использовании программы.

70.1 Установка под ⊞Windows

⊞ + R http://www.kicad-pcb.org/ Download ⊞

⊞ + R http://kicad.nosoftware.cz/ **KiCad_testing201x.xx.xx-BZRxxxx_Win_full_version.exe**

[1] копипаста http://teholabs.com/knowledge/kicad.html

[2] копипаста http://ru.wikibooks.org/wiki/KiCad

[3] копипаста http://ru.wikibooks.org/wiki/KiCad/Miniurok

Installer Language > English > Ok в русифицированном инсталляторе кривые шрифты
KiCAD 20xx.xx.xx Setup > Next
Лицензия > Agree
Components > ☒ все > Next
Location > **C:/KiCad** > Install
Completing Setup > ☐Wings3D > Finish

70.2 Установка под Linux

```
root# aptitude install kicad-doc-ru kicad
```

+++ ~/.blackbox/menu

```
1 [submenu] (CAD)
2 [exec] (KiCAD) {kicad}
```

Для добавления библиотек, поставляемых с этой книгой, сделайте git checkout:

```
user:~$ git clone --depth=1 -o gh https://github.com/ponyatov/Azbuka.git Azbuka
```

kicad > eeschema > Настройки > Библиотека
Пользовательские пути поиска > Добавить > **/home/user/Azbuka/kicad/lib**
Файлы библиотек > Добавить > R,L,C,SPICE,DA_POWER,.. > Открыть > OK

Для проверки работы библиотек можете открыть проект
kicad > Файл > Открыть > **/Azbuka/bcollis/led1/led1.pro**
или сразу схему
eeschema > Файл > Открыть > **/Azbuka/bcollis/led1/led1.sch**

70.3 Создание проекта в менеджере проектов kicad

В верхней части панели менеджера проектов kicad имеются большие кнопки запуска компонентов KiCad:

- EeSchema — Редактор принципиальных схем
- CvPcb — Программа редактирования падстеков (отверстий и площадок)
- Pcbnew — Редактор печатных плат
- GerbView — Программа просмотра фотошаблонов в формате Gerber
- Bitmap2Component — Создание компонента из черно-белого изображения (например логотипа)
- PcbCalculator — Калькулятор для печатных плат
- PageLayout — редактор формата листа схемы

Каждая кнопка запускает соответствующую программу. Мы будем использовать эти программы по мере изучения.

Лучше всего для каждого проекта использовать раздельные папки; в противном случае система может сбиться с толку, если файлы из разных проектов будут лежать в одной папке. Проделайте следующие шаги:

1. Создайте папку проекта **D:/ARM/SpindleDriver**
2. Запустите программу KiCad
3. Создайте проект (project)
   - На панели инструментов KiCad выберите левую иконку с подсказкой Начать новый проект, используйте команду меню Файл > Новый > Пустой или сочетание клавиш Ctrl + N.
   - В диалоге Создать новый проект выберите созданную папку выберите только что созданную папку **D:/ARM/SpindleDriver** и введите имя проекта **SpindleDriver** и нажмите Сохранить.
   - Если папка проекта содержит какие-то файлы, будет выведено окно выбора: создать подпапку с именем проекта Yes, или записать файл проекта в указанную папку как есть No. Нажмите No.
   - Сохраните проект кнопкой Сохранить текущий проект, Файл > Сохранить или Ctrl + S.
   - В папке появится файл **SpindleDriver.pro**, содержащий установки вашего проекта. Файл имеет тектовый формат, поэтому при необходимости его можно открыть в любом редакторе и вручную аккуратно подправить, например скорректировать настройки зазоров печатной платы.

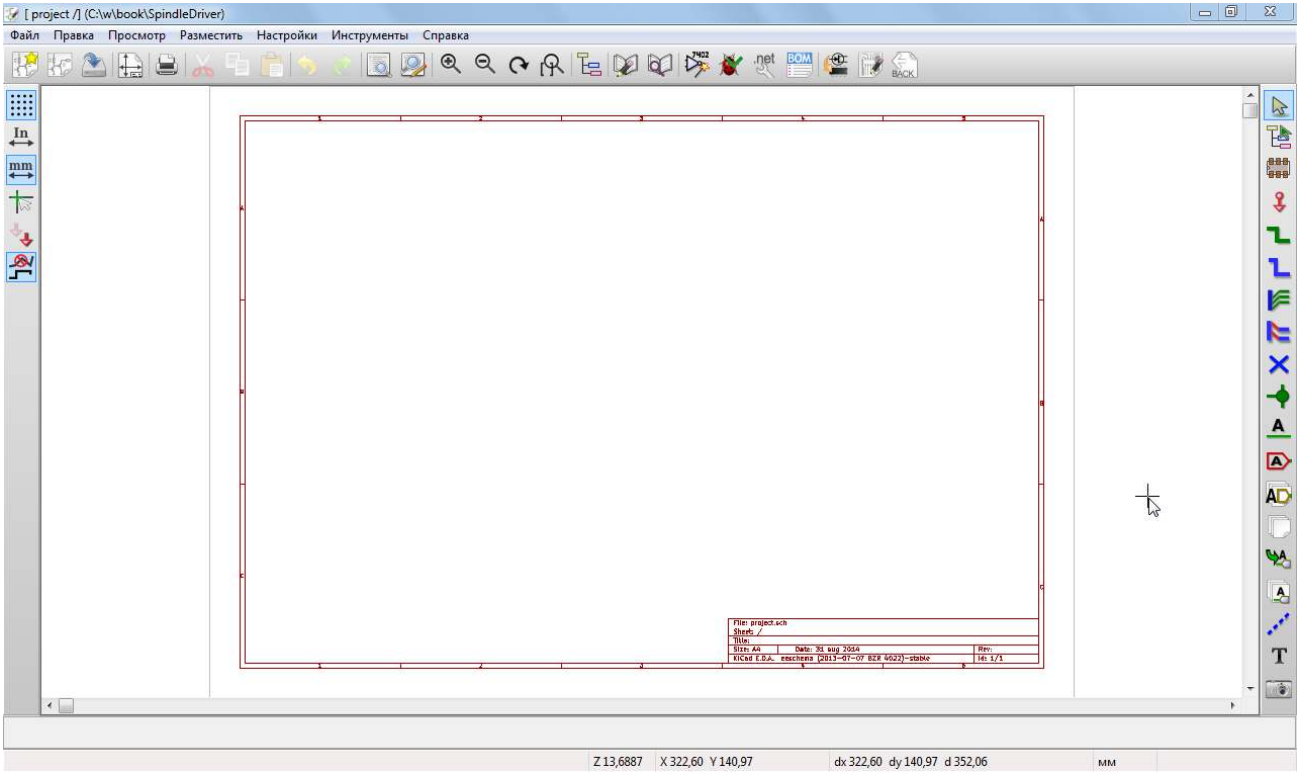
70.4 Создание принципиальной схемы в eeschema (часть 1)

Запустите редактор принципиальных схем, нажав на панели KiCad большую кнопку .

При первом запуске eeschema стартует с новым проектом и показывает предупреждение, что файла схемы еще нет. Просто нажмите OK.

Если вас не устраивает черный фон рабочец области или цвета элементов схемы, поменяйте настроки цветов Настройки > Цвета.

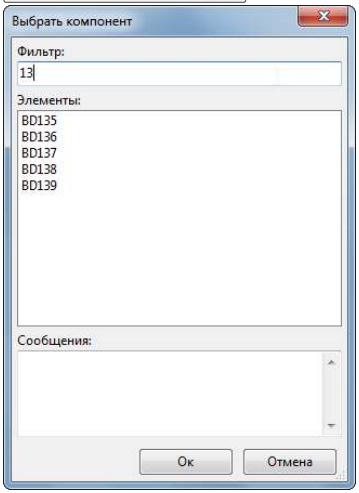
На правом краю окна редактора схем есть вертикальная панель инструментов, которые мы и будем использовать для рисования схемы. Этими инструментами можно выбирать объекты, размещать компоненты, вводить связи и т.д.



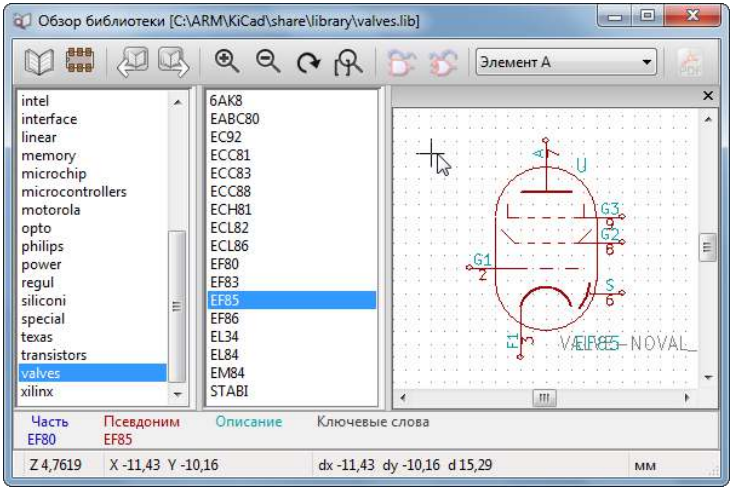
Завершение работы инструмента: вы можете выбрать другой инструмент из правой инструментальной панели или же указать Отложить инструмент по правому клику мышки ▷.

70.5 Инструмент Добавить компоненты

- На правой панели нажмите кнопку Разместить компонент . Курсор изменится со стрелки на карандаш. Удобнее использовать сочетание клавиш Shift + A. Кликните в поле схемы чтобы начать размещение компонента. Появится диалог Выбор компонента. Вы можете выбрать компонент несколькими путями:
- 1. Если вы знаете точное имя копонента, введите его в поле Имя, а затем нажмите Enter или OK.
  2. Если вы знаете имя только приблизительно, в поле Имя введите шаблон для поиска, например, *BD*, затем нажмите Enter или OK. Вы увидите окно Выбрать компонент со списком найденных компонентов.



3. Вы можете искать компонент по ключевому слову, введя его в поле Имя, затем кликнув Поиск по ключевому слову. Однако на данный момент качество библиотек все еще низкое, и немногие компоненты имеют ключевые слова, поэтому эта возможность полезна косвенно.
4. Можно выбрать недавно использованные компоненты из Списка истории.
5. Кнопка Список всех вызывает диалог, в котором можно выбрать сначала библиотеку 74xx, а затем ее компонент 74HCT04.
6. Кнопка Выбор просмотром вызывает Обзор библиотек, позволяя просмотреть библиотеки и находящиеся в них условные графические изображения.



Вы также можете вызвать обозреватель библиотек кнопкой

Просмотр библиотек и компонентов

Выбрав элемент ◁◁, вставьте символ в нужное место схемы ◁. Позже вы сможете переместить его если нужно. Зеркальное отражение компонента можно произвести следующим образом:

- Поместите курсор на компоненте.
- По ▷ выберите Ориентация компонента » Отражение.
- Без использования контекстного меню — наведите мышь на компонент и нажмите кнопку X или Y.

70.6 eeschema: редактор электрических схем

обеспечивает:

- создание однолистовых и иерархических схем,
- проверку их корректности ERC (контроль электрических правил),
- создание списка электрических цепей netlist для редактора топологии платы pcbnew или для spice-моделирования схемы,
- доступ к документации на используемые в схеме электронные компоненты (datasheet).

70.7 Библиотеки компонентов

[4]

В составе KiCad поставляются библиотеки электронных компонентов (обычных и поверхностно монтируемых SMD). Для многих библиотечных компонентов есть 3D-модели, созданные в Wings3D.

[4] копипаста http://habrahabr.ru/post/197582/

Но как только вы начинаете работать со свежеустановленным KiCADом, тут же обнаруживается, что библиотечные компоненты или не подходят[5], или нужных компонентов попросту нет в библиотеках.

Модель компонента в САПР EDA обычно состоит из следующих частей:

70.8 gerbview: просмотр фотошаблонов

позволяет просматривать Gerber-файлы перед передачей печатных плат в производство.

70.9 Программа Wings3D для создания 3D моделей

Эта программа может вам пригодиться если вы планируете создавать 3D модели для PCB элементов.

Архив и файлы документации (Linux и Windows) в папке kicad/wings3d.

Взгляните на домашнюю страницу Wings3D чтобы узнать подробнее о программе.

pcbnew использует файлы в формате wrml (.wrl) экспортируемые из Wings3D (родной формат Wings3D - это .wings).

70.9.1 Установка Wings3D под ⊞Windows

⊞ + R http://www.wings3d.com/ » Downloads » Stable Release » ⊞Windows(32/64b)
filewings-n.n.n.exe
Compononets » ☒QuickLaunch » Next
Location » **C:/Program Files/wings3d_n.n.n** » Next » Install » Close

[5] например не соответствуют ГОСТ или стандартам предприятия

Глава 71

Простейшие полупроводниковые элементы

71.1 Оптоэлектроника

71.2 Схемы на биполярных транзисорах

71.3 Схемы на на полевых транзисорах

Глава 72

Операционные усилители

Глава 73

Источники питания

73.1 Батарейное питание

73.2 Линейные стабилизаторы

73.3 Импульсные преобразователи на ШИМ-контроллерах

73.4 Цепи защиты и гашения кондуктивных помех

Глава 74

Цифровая электроника

Глава 75

Компьютерные интерфейсы

75.1 Поколение 90x: COM, LPT, ISA

75.1.1 Резервный программатор AVR “пять проводков”

75.2 Сеть CAN

75.3 Интерфейсные модули USB

75.3.1 Универсальный высокоскоростной конвертер FTDI FT2232H

75.3.2 JTAG-адаптер

75.3.3 Отладочный модуль CAN

Глава 76

ПЛИС

Глава 77

Датчики

Глава 78

Электропривод и исполнительные устройства

Часть III

Основы конструирования РЭС

Глава 79

Пакеты моделирования на основе OpenFOAM

Глава 80

Обеспечение теплового режима

Глава 81

Электромагнитная совместимость

81.1 Кондуктивные помехи

81.2 Компоновочные модели и оптимизация кабельной сети

Часть IV

Технология РЭС

Глава 82

Инструменты и оборудование

82.1 Паяльное оборудование

82.1.1 Паяльник

Паяльник — обязателен дешевый сетевой мощностью не менее 20 Вт, типа ЭПСН-25/220. Ограничитель мощности или регулятор температуры легко собрать самостоятельно.

Для сборки электроники хорошо также иметь маленький монтажный 12 В 8 Вт от паяльной станции ZD-927 (∼100 р), без самой станции. Если не жалко 500 р, берите станцию ZD-927 целиком, внутри простейший регулятор мощности, и вам не понадобится источник питания на 12 В, который вы еще не сделали.

Паяльник ЭПСН-25/220

Паяльник 220В 25Вт, СВЕТОЗАР, SV-55310-25 230 р.

Паяльник 220В 25Вт ZD-721N 175 р.

Паяльник для станции ZD-927 12 В 8 Вт 85 р.

82.1.2 Паяльная станция

Из всего разнообразия для хоббита оптимальным являются паяльные станции Lukey 702/853D (3000+ р). Для работы или регулярного хобби паяльная станция с феном, а может даже и встроенным источником питания, вещь незаменимая, и не такая уж дорогая.

Паяльная станция ZD-927 520 р.

Паяльная станция LUKEY 702 3100 р.

Паяльная станция LUKEY 853D с источником питания 5200 р.

82.2 JTAG-адаптер

82.3 Отладочные платы

Прежде чем начать работать с отдельными МК, устанавливая их на плату собственной разработки, для быстрого старта используют отладочные платы[1]

82.3.1 Arduino /Atmel Mega AVR8/

82.3.2 Cortex-Mx

82.3.3 CubieBoard /Cortex-A8 AllWinner A10/

82.3.4 Raspberry Pi /ARM11 BCM3032/

82.3.5 BlackSwift /MIPS/

82.3.6 VoCore /MIPS/

82.4 Радиомонтажный инструмент

Пара надфилей, заточной камень на дрель, комплект сверел и несколько листов наждачки.

[1] development board, demo board

82.4.1 Pro'sKit

Отдельного обзора заслуживает инструмент и наборы Pro'sKit

PK-5308BM универсальный набор инструментов

1PK-616B Набор инструментов для электроники профессиональный

1PK-813B Набор базовых инструментов для электроники

По личному опыту: в 1PK-813B не хватает

- мелкого мультиметра,
- стриппера 1PK-3001E,
- микрокусачек типа 8PK-30D,
- канифоли,
- ножа,
- настроечную отвертку заменить индикаторной.

Инструмент до 1000 В

Для электромонтажных работ обязательно приобретите комплект высоковольтного инструмента до 1000 В:

РМ-911 Пассатижи 1 кВ

РМ-917 Кусачки (бокорезы) 1 кВ

Хранение

103-132D Кассетница для деталей и компонентов

SB-3428SB Портативная кассетница для саморезов и т.п.

Радиомонтаж

8PK-30D Кусачки миниатюрные

1PK-709 Длинногубцы-кусачки

1PK-055S Длинногубцы изогнутые

1PK-29 Круглогубцы

1PK-101T Пинцет прямой

1PK-3001E Клещи для зачистки проводов прецизионные (стриппер)

PD-374 Тиски на струбцине

Прочие

Попалась интересная недорогая отвертка: аиксация четкая, исполнение очень неплохое, позволяет добраться до узких мест. Из минусов: ручка похоже не цельнометаллическая, при изломе есть риск распороть руку.

82.5 Измерительное оборудование

82.5.1 Мультиметр

Мультиметр — обязателен, без него работать невозможно[2]. Для совсем начинающего больше всего подойдет M32082.5.1 с автодиапазоном, когда освоитесь возьмете вторым прибором что-то из крупных серий M89x/MY6x с измерением температуры[3] или “рыльцеметр”82.5.5 (RLC).

[2] или собирать замену на паре измерительных головок тока/напряжения, и делителях

[3] иногда нужно для измерения температуры корпусов элементов, радиаторов, растворов если возитесь с электрохимией

Mastech M838

Простой, компактный, дешевый, с измерением температуры

Mastech M300

Простой, очень компактный, дешевый, в чехле очень удачно умещается в набор инструментов.

Mastech M320

То же что и M300 82.5.1, но с автодиапазоном, т.е. не требует переключения диапзонов измерения вручную. На любителя, возможно удобен для совсем начинающих, но слишком медленен если требуется измерение меняющегося тока/напряжения.

82.5.2 Осциллограф

82.5.3 Логический анализатор

82.5.4 Генератор сигналов

82.5.5 Рыльцеметр RLC

82.6 Электроинструмент

82.6.1 Дрель

Дрель ударная сетевая Praktyl-R PID13D01 400 Вт (!)395 р.

Дрель безударная сетевая Интерскол Д-11/530ЭР (с БЗП) 1120 р.

Дрель — одноразовая китайчатина от 400 р. Продаются уже брендированные на Леруа Мерлен, наклейка «PID13D01 Ударная дрель 400 Вт, 13 мм». Скорость регулируется глубиной нажатия курка, крутилка

на курке ограничивает глубину механически, фиксатор держит скорость близко к минимальной, запаха горелой пластмассы через несколько минут работы на холостом ходу нет.

По надежности рекомедуется Интерскол 1100+р. Надежность Интерскола — не «китай», классика ДУ-580ЭР работает в хвост и гриву, используется криворукими студентами, лежит в подвале в пыли от точила, и никаких вопросов даже со щетками.

Если не планируете много сверлить бетон, **берите дрель без ударного механизма**: отсутствуют лишние продольные перемещения, что может быть важно при использовании в качестве шпинделя сверлильного станка, и механизации других технологических поделок.

У шуруповерта нет 43 мм шейки для фиксации, поэтому как средство электропривода он практически бесполезен, и нужен собственно для заворачивания большого количества саморезов. Хотя наличие ограничителя крутящего момента и малые габариты удобны при сверлении и сборке поделок.

Имея некоторое количество поделочного материала, кривые руки и особенно доступ к станочному оборудованию, можно сколкозить некоторое подобие настольных станочков /стр.140/ для механизации некоторых работ, используя дрель в качестве привода.

Главным элементом такой оснастки — зажим на шейку дрели 43 мм. Особых требований по его точности и качеству нет, т.к. сама шейка обычно пластиковая, и никакой доводки по круглости и параллельности оси инструмента не проходит.

makita

МСУЭ-16-2РЭ

82.6.2 Лобзик

Praktyl 350 Вт 356 р.

Makita 4329 2260 р.

Лобзик полезен при разделке стеклотекстолита, и изготовлении технологической мебели (стеллажи, рабочие столы и т.п.).

82.6.3 Ж двигатель

Если у вас возникло желание механизировать изготовление механических деталей, а свободного доступа к настоящему станочному оборудованию нет, есть смысл рассмотреть изготовление самодельной механизированной оснастки типа /стр.140/, или даже самодельных станочков. В этом случае надо рассмотреть применения универсального привода.

Первый кандидат на место универсального электропривода достается той самой дрели, не забываем об обязательном наличии 43 мм монтажной шейки. Достоинство дрели как привода — прямое подключение к сети, встроенный редуктор, есть модели с простой регулировкой оборотов, есть резьба и отверстие под винт на валу, в комплекте есть патрон для зажима мелких деталей в точилке[4].

Ограниченно доставаемые двигатели от стиральных машин, отличаются мощностю и оборотистостью, особенно от старых моделей. Часто доступны сразу с готовым шкивом на валу, который иногда проще использовать, чем снять.

Автозапчасти: привод печки Камаза, двигатель постоянного тока 24 В 50 Вт

Новые асинхронные двигатели АИРЕ 56 В2/В4 (3000/1500 об.) с заводским конденсатором, подключается к сети ∼220 В, цена от 2500 р. С ростом размеров и мощности цена резко повышается. Следует обратить внимание на возможность монтажа на дополнительный фланцевый подшипниковый щит, (?) с моделями АИРЕ 80.

Для самодельных серлилок и микроинструмента хороши китайские воздушные шпиндели постоянного тока с цанговыми патронами ER11. Требуют источник питания постоянного тока 9÷48 В. В магазинах не попадались, необходима прямая покупка с AliExpress[5] по почте.

---

[4] БЗП удобен, патрон с ключем дает лучший зажим и возможно точнее

[5] пользуйтесь английской версией — переводная жуткое УГ

Жвигатель Вятка-Автомат 19?? г.

Двигатель печки Камаза

АИРЕ 56 В2, 0.2 КВт

Воздушный шпиндель с цангой ER11

Съемные фрезерные шпиндели, поставляются отдельно или в комплекте с насадкой ручного фрезера по дереву. Лучшие, со стальной шейкой — Kress, активно применяются хобби-ЧПУшниками. Попроще и сильно дешевле делал Интерскол, иногда попадается noname. Недостаток как универсального привода — они высокоскоростные, возникают проблемы с понижающими передачами. Применение — приводной высокоскоростной инструмент: боры, фрезы по дереву, микроинструмент для граверов (микродиски, шарошки). Цанга 8 мм. Для некоторых моделей бывают наборы цанг на мелкий инструмент.

KRESS 530/800/1050 FM(E)
5600+ р.

Интерскол ФМ-30/750
/снят с производства/

Интерскол ФМ-55/1000 Э
5050 р.

Глава 83

Трассировка плат и подготовка производства в KiCAD

83.1 Технология ЛУТ (Лазерный УТюг)

83.2 Технология фоторезиста

83.3 Формат Gerber и подготовка промышленного производства

Глава 84

FreeCAD

[1] В среде специалистов ряда отраслей известна проблема создания полноценной САПР в рамках OpenSource, и хотя FreeCAD ещё не является кандидатом на такую «полноценность», этот продукт может рассматриваться как одна из попыток создания базы для решения этой проблемы. Разработчик FreeCAD Юрген Ригель, работающий в корпорации DaimlerChrysler, позиционирует свою программу как первый бесплатный инструмент проектирования механики (сравнивая свой продукт с такими развитыми проприетарными системами как CATIA версий 4 и 5, SolidWorks), созданный на основе библиотеки **Open CASCADE**. Цель программы — предоставить базовый инструментарий этой библиотеки в интерактивном режиме.

Следует отметить, что имеет место ещё один программный продукт имеющий название freeCAD, его разработчик — Aik-Siong Koh, и он не связан с FreeCAD'ом Юргена Ригеля.

[2] FreeCAD — CAD/CAE приложение трёхмерного параметрического моделирования. Оно в основном сделано для механического проектирования, но также может быть использовано для любых других случаев, в которых вам нужно точно моделировать трёхмерные объекты с контролем над историей моделирования.

FreeCAD все еще находится в ранней стадии разработки, так что, хотя он уже предлагает Вам большой (и растущий) список функций, многого еще не хватает, особенно если сравнивать его с коммерческими решениями, и вы можете не найти его достаточно развитым для использования в производственной среде. Тем не менее, есть быстрорастущее сообщество пользователей-энтузиастов, и вы уже можете найти много примеров качественных проектов, разработанных с FreeCAD.

Как и все проекты с открытым исходным кодом, проект FreeCAD не единственый способ работы обеспеченный Вам его разработчиками. Это во многом зависит от роста его сообществу пользователей и разработчиком, доработки функций и стабилизации кода (да здравствует исправление ошибок!). Так

[1] копипаста https://ru.wikipedia.org/wiki/FreeCAD_(Juergen_Riegel%27s)
[2] копипаста http://www.freecadweb.org/wiki/index.php?title=Getting_started

что не забывайте об этом, когда начинаете использовать FreeCAD, если вам он нравится, вы можете непосредственно влиять и помочь проекту!

84.1 Установка под ⊞Windows

⊞ + R » http://www.freecadweb.org/ » Download » ⊞Windows » **FreeCAD 0.14** » **. . . _setup.exe**
FreeCAD 0.14.3700_x86_setup » FreeCAD 0.14 License » I agree
Distination Folder » **C:/FreeCAD** » Install » Completed » Close
⊞ » Программы » FreeCAD 0.14 » FreeCAD » ▷ » Закрепить в панели задач

FreeCAD 0.14
FreeCAD © Juergen Riegel, Werner Mayer, Yorik van Havre 2001-2011
Версия 0.14
Редакция 3700 (Git)
Дата выпуска 2014/07/13 11:34:36
Операционная система Windows 7
Word size 32-bit
Branch releases/FreeCAD-0-14
Hash 32f5aae0a64333ec8d5d160dbc46e690510c8fe1
Лицензия ...
Скопировать в буфер обмена
OK

84.2 Чертеж

84.3 Эскиз

84.4 Деталь

84.5 Сборка

84.6 Автогенерация конструкторской докуметации

84.7 Скрипты и пользовательские расширения

Часть V

Станки

На основе [3]
© Joe Martin, illustration by Craig Libuse
Tabletop Machining
A basic approach to making small parts on miniature machine tools

© Джо Мартин, иллюстрации Craig Libuse
Настольные станки
Основной подход к изготовлению мелких деталей на миниатюрных настольных станках

[3] копипаста http://rutracker.org/forum/viewtopic.php?t=3126529

Глава 85

A special note to engineers reading this book

Глава 86

Machining for engineen and engineering for machinists

At first glance the subtitle on the cover of this book could be a bit deceiving. What does tabletop machining have do with engineering you may ask? Compare it to a book that has been written about the ocean. The seas could he described from the perspective of a young man who has just sailed around the world in a twenty-five foot sailboat or by a merchant seaman who has spent his career aboard a giant ocean liner. Each would have an entirely different view of what the ocean was all about. In a stenn, the chap in the small boat would write ahout surviving broken masts and mountainous seas while the merchant seaman might write about seasick passengers. I believe you would learn more ahout the ocean from the young man in the small boat, because in a sense he was more involved in his subject. He was not just on it, he was in it.

Глава 87

Navigating the seas of machining

The ocean in this case is the world of machining. The craftsman using tabletop machine tools is like the sailor in a small boat, while the professional machinist with his big CNC shop tools is like the world-traveling seaman. The process of producing complex, accurate parts cannot be described by looking in the window of a quarter million dollar CNC machine. It would be like a merchant seaman working in the engine room trying to describe a stonn in the Atlantic Ocean by telling you how much extra fuel the ship used. The professional's view of the subject may be so cluttered with details that it is difficult to sort the things you really need to know to sail in rough seas or make good parts. I t is the craftsman working with small tools, turning the cranks by hand, who will have the most to tell you about the real world of working with metal.

Глава 88

looking at engineering from the craftsman's perspedive

With the aid of computers, parts can easily be drawn that can't be built. CAD prvgrams allow a designer to put a perfect .0001 "radius on the inside comer of a pocket cut in tool steel. Hopefully after reading this book you will not ask a toolmaker to do it, but if you do, you'll at least know it is going to cost a great deal of money to try. Working with metal is far more difficult than one would imagine. A false impression is gained by looking at the beautiful yet inexpensive machined parts that we deal with daily. They have been produced in very large quantities, and that five-dollar part you may consider a "ripoff' could easi ly cost five hundred dollars if you had to manufacture just one. New engineers will often think a toolmaker is a failure when the seemingly simple part they design ends up costing a thousand dollars to make. Most engineers wi ll eventually have to deal with the craftsman who tum their ideas into reality, and in reading this book I would hope you come away with a new perspective of what is really involved in producing a machined part or a product. An alternate subtitle for

the book might have been "Things they should have taught you in engineering school but didn't". This book might be considered your textbook for a course called "Reality WI".

Глава 89

Seeing produdion from the point of view of both the engineer and machinist

My perspective on machining could be considered unique because, in order to survive, I have had to deal with every aspect of product design from engineering to prototyping to tooling to manufacturing to sales. In this book I have tried to pass along the logic I used to solve the associated problems. Understanding how a craftsman thinks and works is an essential part of getting projects done. Unless you are willing to build your designs yourself, you are going to have to learn how to deal with the craftsman who will actually build them. The more you know about their methods. personalities and unique problems, the better your chances are for success. Smooth sailing.

© Joe Martin

Глава 90

About the Joe Martin

Joe Martin worked in the construction trades after graduating from high school, but his real love was always building and fl ying radio controlled model airplanes. When he decided to turn his hobby into a business and start his own company making components for the radio control industry, he had to learn about machining and toolmaking on his own. He simply couldn afford to hire anyone else to set up the tools and make the molds. He has designed and taken to market numerous products and owned several companies over the years. He began his association with Sherline Products as an importer of Australi an-built lathes in the early 1970's. Since then, Joe's company has grown to become the sole manufacturer and worldwide distributor of Sherline machine tools.

Joe was one of the founders of the sport ofFonnula One model aircraft competition as well as one of its early champions. His competitive nature seems to find its way into whatever form of fun he pursues. He has been a winner in sports from model airplane competition to ocean sai lboat racing and, most recently, automobile racing.

Never one to be a spectator in life, he has tried and mastered many skills. In this book, he passes on to you some of his hard-won knowledge about machining. His down-to-earth s ty le is not hi gh ly polished. In fact, if you could say that life has put a finish on him, it would probably be described as ground or honed .. . ve ry company making components for the radio control industry, he had to learn about machining and toolmaking on his own. He simply couldn afford to hire anyone else to set up the tools and make the molds. He has designed and taken to market numerous products and owned several companies over the years. He began his association with Sherline Products as an importer of Australi an-built lathes in the early 1970's. Since then, Joe's company has grown to become the sole manufacturer and worldwide distributor of Sherline machine tools. Joe was one of the founders of the sport ofFonnula One model aircraft competition as well as one of its early champions. His competitive nature seems to find its way into whatever form of fun he pursues. He has been a winner in sports from model airplane competition to ocean sai lboat racing and, most recently, automobile racing. Never one to be a spectator in life, he has tried and mastered many skills. In this book, he passes on to acc urate but not slick. I think his heartfe lt love of good too ls and miniature machining will be apparent to all who read this book. Working with him these past 25 years is certainly an experience I would not have wanted to miss.

© Craig Libuse

Глава 91

Dedication

[1]

Carl Hammons, my friend and business partner for thirty years. died September 11 , 1997 as I was writing this book. We shared thousands of lunches and coffee breaks over the years we worked together, and much of the knowledge I have passed on in this book came from Carl. Carl and I shared the rare distinction of having been partners not just once, but twice. We both played different roles in putting together the product line, and without him it just isn't going to be as much fun.

When we joined forces for the second time. we had an agreement that eliminated any need to financially justify the purchase of a new piece of equipment. We would buy machines that interested us and find a job for them later. The laser engraver was a perfect example of thi s, but now we couldn't get along without it. It may

[1] The photo COII/positioll ahol'e ix ajoillt effort. The photo a/Carl II'OS taken by his wife Barbara. The photo o/Swall Lake. MOil/alia. a /m'odle spot oj Carl's, was takeu byfrieud WaYl1e Arll/s/rOllg. The two images were composed ill PIIO/OShopl by artist £loille lolli/IS

seem contrary to smart business practice, but that's the way we did it. I have no regrets, for we were always the happiest when we were confronted with a new set of technical problems. Therefore. I dedicate this book to Carl Hammons: my business partner, my friend.

I should also credit the English teachers in the Cranston, Rhode Island school system for forcing a not-so-willing student enrolled in the "boys general class"to learn enough about our language to dare to take on the task of expressing difficult concepts in simple words. I graduated in 1953. You, the reader, will be the ultimate judge of their (and my) success in this undertaking.

© Joe Martin

Глава 92

Правила техники безопасности

Глава 93

Предисловие

Глава 94

Введение

Глава 95

Галерея фотографий проектов

Глава 96

Профиль Мастера — Скотти Хьюитт... 24

Глава 97

РАЗДЕЛ I — О станочной обработке

97.1 1. Получение информации об обработке..... 27

97.2 2. Вам нужен токарный станок. фрезерный станок или оба ? 29

97.3 3. Материалы для металлообработки... 39

97.4 4. Процессы для металлообработки

97.5 5. Использование ручных инструментов и абразивов 57

Часть VI

SECTION 2-LATHE OPERATIONS

Глава 98

Craftsman Profile-Jerry Kieffer 112

Глава 99

Jerry Kieffer's Flying Pendulum Clock 114

Глава 100

I. Lathe work holding 115

Глава 101

2. Lathe operating instnlclions 121

Глава 102

3. Tail slock lools and operations 141

Глава 103

4. Ri ser blocks 145

Глава 104

5. Supporting long or thin work 149

Глава 105

6. Gelling started in thread cutting. 157

Глава 106

7. Knurled fini shes. . . . .
167

Глава 107

8. Watchmaking and clockmaking tools 171

Глава 108

9. Milling operations on a lathe . . . .
... 177

Часть VII

SECTION 3-MIII1NG OPERATIONS

Глава 109

Craftsman Profile- Augie Hiscano 180

Глава 110

I. Holding parts for milling. 183

Глава 111

2. Mill operating in structions.. 191

Глава 112

3. Squaring up a block 205

Глава 113

4. The rotary table and index ing attachment 209

Глава 114

5. Gears and Geartrains 2 19

Глава 115

6. Accessories for milling

Глава 116

• Horizontal milling conversion 235

Глава 117

• Rotary column attachmcnt 237

Часть VIII

SECTION 4-OTHER MACHINING TOPICS

Глава 118

Craft sman Profiles-Dan Lutz and Paul White . 242

Глава 119

I. Setting up a small workshop 247

Глава 120

2. Lathe and mill alignment and adj ustments .. 25 1

Глава 121

3. Enginering drawings 259

Глава 122

4. Frequently asked questions 265

Глава 123

5. Making a bus iness o ut of a hobby .. .Joe Martin's

Глава 124

and Sherline's story 273

Глава 125

6. Using CNC in a home shop 309

Часть IX

SECTION S-PROJECTS AND RESOURCES

Глава 126

Cra ftsman Profi le-Bob Bres lauer 3 16

Глава 127

Machini st's tips 3 18

Глава 128

I. Plans and projects you can build 3 19

Глава 129

1. Miniature Tap Handle ... a beginning

Глава 130

project you can use in your shop 32
1

Глава 131

2. Mill vise "sofe jaws. 325

Глава 132

3. Lay ing out a circul ar hole pattern

Глава 133

fordrilling, a handy skill to learn. 327

Глава 134

4. "Millie"... a small oscillating steam

Глава 135

engine by Ed Warren, a simple project

Глава 136

from the pages of Modeltec magazine) 329

Глава 137

5. Ordering plans for the Little Ange l

Глава 138

hit ’n miss engine ... an advanced

Глава 139

machining project by Bob Shores 333

Глава 140

2. Contests and awards for tabletop machini sts . . 335

Глава 141

3. Exploded views and part number li sting

Глава 142

Model 4000 and 4400 Lathes 340

Глава 143

Model 5000 and 5400 Milling machines . . . 341

Глава 144

Model 2000 8-Direction Mill Column 342

Глава 145

Part number listing 343

Глава 146

4. A simple RPM gage for your latheor mill 345

Глава 147

Harold Cli sby and the first Sherline lathe 346

Глава 148

S. Index 347

Глава 149

Conversion faclors 350

Часть X

Станочное оборудование

Самый распространенные станки — сверлильные, т.к. имеют самую простую конструкцию, и минимальную стоимость. Предназначены для самой частой операции: изготовления перпендикулярных круглых дырок в различных материалах, топовые модели имеют также функцию нарезения резьбы.

Наиболее многочисленную группу металлорежущих станков составляют токарные станки, используются в механических, инструментальных и ремонтных цехах заводов, а также в ремонтных мастерских в основном для обработки деталей, имеющих форму тел вращения. При использовании соответствующей оснастки позволяют растачивать отверстия в призматических (прямоугольных) деталях, и фрезеровать небольшие детали. Самый ходовой тип детали — тело вращения с наружними и внутренними резьбами: валики, втулки, оси, болты, винты, шпильки, кольца, шайбы и т.д.

К основным размерам, характеризующим токарный станок, относятся

- наибольший допустимый диаметр обрабатываемой заготовки,
- высота центров над станиной и
- расстояние между центрами.
- Часто обращают внимание на диаметр проходного отверстия шпинделя, определяющий максимальный диаметр длинномерных заготовок, что важно при изготовлении мелких партий деталей или нарезке резьб на трубах.

Значительную часть среди металлорежущих станков составляют фрезерные станки. Наибольшее распространение имеют консольно-фрезерные. Предназначены для выполнения различных фрезерных работ цилиндрическими, дисковыми, фасонными и другими фрезами, можно фрезеровать плоскости, пазы, фасонные поверхности, и т.д. Кроме этого, универсальные консольно-фрезерные станки с поворотным столом или делительной головкой позволяют фрезеровать различного рода винтовые канавки и зубья зубчатых колес.

Основными размерами фрезерных станков, по которым можно определить возможность установки и обработки конкретных заготовок с определенными габаритами, являются размеры рабочей поверхности

стола (длина и ширина) и рабочий ход стола/рабочая зона в продольном, поперечном и вертикальном направлениях.

Глава 150

Маркировка моделей станков производства СССР

станок-"чугунок", простой, дубовый, надежный (потому что ненадежные давно сломались), дешевый, но требует помещение с силовым полом, 3х-фазное питание, и кучу времени на поиск запчастей для восстановления по металлобазам и развалам. В диком виде пока что встречается в школах и других

Для станков, выпускавшихся в СССР, принята единая система классификации и условных обозначений, основанная на присвоении каждому станку особого шифра (номера). Станки каждой группы подразделяются на девять типов. Внутри каждого типа металлорежущие станки могут отличаться друг от друга конструктивными особенностями. Эти особенности, а также некоторые другие характеристики и отражаются в шифре (номере) станка.

```
<группа>[<буква1>]<тип><характеризующий размер>[<буква2>][М][Фn]
```

Кроме цифр, в условные обозначения модели станка часто входят буквы. Если буква1 стоит между первой и второй цифрами, то это означает, что конструкция станка подверглась усовершенствованию по сравнению с прежней моделью.

Если буква2 стоит в конце номера станка, то это говорит об изменении основной, «базовой» модели станка. Часто буква2 задает класс точности:

Н нормальная (часто не указывается)

П повышенная

В высокая

А особо высокая

С особо-точный станок

Для станков с ЧПУ:

М наличие магазина инструментов,

Ф1 станки с цифровой индикацией и преднабором координат,

Ф2 с позиционными и прямоугольными системами,

Ф3 с контурными системами,

Ф4 с универсальной системой для позиционной и контурной обработки. Эти шифры пишутся в конце номера модели.

группа/тип:

1 токарные;
  1
  2
  3 револьверные;
  4
  5
  6 токарно-винторезные;
2 сверлильные и расточные;
  1 вертикально-сверлильные,
  2 одношпиндельные полуавтоматы,
  3 многошпиндельные полуавтоматы,
  4 координатно-расточные,
  5 радиально-сверлильные,
  6 горизонтально-расточные,
  7 алмазно-расточные,
  8 горизонтально-сверлильные,
  9 разные сверлильные.
3 шлифовальные, полировальные, доводочные и заточные;
4 специальные;
5 зубо- и резьбообрабатывающие;
6 фрезерные;
7 разрезные;
8 строгальные, долбежные, протяжные;
9 прочие

Характеризующий размер:

токарные

высота оси шпинделя над станиной,

задает максимально возможный радиус обрабатываемой заготовки

Пример: 1А616: станок токарно(1)-винторезный(6), модификация(А), высота центров над станиной (16)0 мм.

Глава 151

1А616: станок токарно-винторезный

151.1 Назначение и области применения

151.2 Распаковка и транспортировка

151.3 Фундамент станка, монтаж и установка

151.4 Подготовка станка к первоначальному пуску

151.5 Паспортные данные

151.6 Описание основных узлов

151.7 Смазка

151.8 Первоначальный пуск

151.9 Указания по технике безопасности

151.10 Настройка

151.11 Регулирование

151.12 Ведомость комплектации

Глава 152

Основы ЧПУ и цифрового производства

152.1 CAM-пакеты для FreeCAD

Часть XI

Основы теории систем автоматического управления

Глава 153

Математический аппарат

153.1 Передаточная функция

153.2 Устойчивость САУ

153.3 Сети Петри

153.4 Автоматы Маркова

Глава 154

Релейное управление

Глава 155

Пропорциональные САУ

Глава 156

ПИДn-регуляторы

Часть XII

Разработка ПО для встраиваемых систем

Глава 157

Архитектура программных систем

157.1 Литература

[**proghant**] Программист-прагматик. Путь от подмастерья к мастеру Хант Э., Томас Д., Лори/Питер, 2004, 2007

157.2 Ортогональность

[1][**proghant**]

[1] копипаста http://ru.wikibooks.org/wiki/%D0%9E%D1%80%D1%82%D0%BE%D0%B3%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C

Ортогональность очень важна, если вы хотите создавать системы, которые легко поддаются проектированию, сборке, тестированию и расширению. Однако этому принципу редко обучают непосредственно. Часто он является лишь скрытым достоинством других разнообразных методик, которые вы изучаете. Это неправильно. Как только вы научитесь непосредственно применять принципы ортогональности, вы сразу заметите, как улучшилось качество создаваемых вами систем.

157.2.1 Что такое ортогональность?

Термин "ортогональность" заимствован из геометрии. Две линии являются ортогональными, если они пересекаются под прямым углом, например, оси координат на графике. В терминах векторной алгебры две такие линии перемещения являются независимыми. Если двигаться параллельно оси X вдоль одной из линий, то проекция движущейся точки на другую линию не меняется. Этот термин был введен в информатике для обозначения некой разновидности независимости или несвязанности. В грамотно спроектированной системе программа базы данных будет ортогональной к интерфейсу пользователя: вы можете менять интерфейс пользователя без воздействия на базу данных и менять местами базы данных, не меняя интерфейса. Перед тем как рассмотреть преимущества ортогональных систем, познакомимся с неортогональной системой.

157.2.2 Неортогональная система

Предположим, вы находитесь в экскурсионном вертолете, совершающем полет над Гранд-Каньоном, когда пилот, который совершил ошибку, наевшись рыбы за обедом внезапно вскрикивает и теряет сознание. По счастливой случайности это происходит, когда вы парите на высоте 30 метров. Вы догадываетесь, что рычаг управления общим шагом несущего винта обеспечивает подъем машины, так что, если его слегка опустить, вертолет начнет плавно снижаться. Однако когда вы пытаетесь сделать это, то осознае-

те, что жизнь — не такая уж простая штука. Вертолет клюет носом, и вас начинает вращать по спирали влево.

Внезапно вы понимаете, что управляете системой, в которой каждое воздействие имеет побочные эффекты. При нажатии на левый рычаг вам придется сделать уравновешивающее движение назад правым рычагом и нажать на правую педаль. Но при этом каждое из этих действий вновь повлияет на все органы управления. Неожиданно вам приходится жонглировать невероятно сложной системой, в которой любое изменение влияет на все остальные управляющие воздействия. Вы испытываете феноменальную нагрузку: ваши руки и ноги находятся в постоянном движении, пытаясь уравновесить все взаимодействующие силы. Органы управления вертолетом определенно не являются ортогональными.

157.2.3 Преимущества ортогональности

Как показывает пример с вертолетом, неортогональные системы сложнее изменять и контролировать. Если составляющие системы отличаются высокой степенью взаимозависимости, то невозможно устранить какую-либо неисправность лишь на локальном уровне.

Исключайте взаимодействие между объектами, не относящимися друг к другу

Мы хотим спроектировать компоненты, которые являются самодостаточным независимыми, с единственным, четким назначением. Когда компоненты изолированы друг от друга, вы уверены, что можно изменить один из них, не заботясь об остальных. Пока внешние интерфейсы этого компонента остаются неизменными можете быть спокойны, что не создадите проблем, которые распространятся по всей системе. С созданием ортогональных систем у вас появятся два больших преимущества: увеличение производительности и снижение риска.

157.2.4 Увеличение производительности

- Изменения в системе локализуются, поэтому периоды разработки и тестирования сократятся. Легче написать относительно небольшие, самодостаточные компоненты, чем один большой программный модуль. Простые компоненты могут быть спроектированы, запрограммированы, протестированы и затем забыты — не нужно непрерывно менять существующий текст по мере того, как к нему добавляются новые фрагменты.
- Ортогональный подход также способствует многократному использованию компонентов. Если компоненты имеют определенную, четкую сферу ответственности, они могут комбинироваться с новыми компонентами способами, которые не предполагались при их первоначальной реализации. Чем меньше связанность в системах, тем легче их перенастроить и провести их обратное проектирование.
- При комбинировании ортогональных компонентов происходит заметное увеличение производительности. Предположим, что один компонент способен осуществлять M, а второй — N различных операций. Если эти компоненты ортогональны и комбинируются, то в сумме они способны осуществить $M \times N$ различных операций. Но если два компонента не являются ортогональными, они будут перекрываться, и результат их действия будет меньшим по сравнении с ортогональными компонентами. Вы получаете большее количество функциональных возможностей в пересчете на единичное усилие, если комбинирует между собой ортогональные компоненты.

157.2.5 Снижение риска

Ортогональный подход приводит к снижению уровня риска, присущего любой разработке.

- Ошибочные фрагменты текста программы изолируются. Если модуль содержит ошибку, то вероятность ее распространения на всю систему уменьшается. Кроме того, ошибочный фрагмент может быть извлечен и заменен новым (исправленным).
- Конечный продукт (система) становится менее хрупким. Проблемы, появляющиеся при внесении небольших изменений и устранении недочетов на определенном участке, не проходят дальше этого участка.
- Ортогональная система способствует повышению качества тестирования, поскольку облегчается проектирование и тестирование отдельных ее компонентов.
- Вы не будете слишком сильно привязаны к определенному субподрядчику, программному продукту или платформе, поскольку интерфейсы между компонентами, производимыми фирмами-субподрядчиками, не будут играть главенствующей роли в проекте.

157.2.6 Проектные группы

Приходилось ли вам замечать, насколько эффективно работают проектные команды, все члены которых знают, что делать, и полностью отдают себя делу, тогда как в других командах сотрудники постоянно препираются между собой и не собираются ни в чем уступать друг другу? Зачастую это не что иное, как проблема ортогональности. Если команды организованы с большим числом перекрытий, то сотрудники путают свои должностные обязанности. Для любого изменения необходимо собирать всю команду, поскольку оно, может быть, затронет каждого. Как разбить команду на группы с четкими обязанностями и минимальным перекрытием? На этот вопрос нет простого ответа. В некоторой степени это зависит от проекта и вашего анализа областей, которые в перспективе могут измениться. Это же зависит от людей, находящихся в вашем распоряжении.

Мы предпочитаем отделять инфраструктуру от приложения. Каждому из основных инфраструктурных компонентов[2] приписывается только ему принадлежащая группа. Подобным образом производится разделение функциональных возможностей приложения. После этого мы изучаем людей, которые имеются в нашем распоряжении на данный момент[3], и сообразно этому корректируем состав групп. Вы можете неформально определить уровень ортогональности структуры проектной команды. Для этого просто посмотрите, скольких людей нужно привлечь к обсуждению каждого изменения, требуемого со стороны. Чем больше эта цифра, тем ниже уровень ортогональности группы. Отсюда ясно, что ортогональная команда работает более эффективно.[4]

157.2.7 Проектирование

Большинство разработчиков знакомо с потребностью в проектировании ортогональных систем, хотя они наверняка используют термины "модульный "компонентно-ориентированный"и "многоуровневый"для описания конкретного процесса. Системы должны быть скомпонованы из набора взаимодействующих модулей каждый из которых реализует функциональные возможности независимо от других. Иногда эти компоненты объединены в уровни, каждый из которых обеспечивает некий уровень абстракции. Данный многоуровневый подход является мощным методом проектирования ортогональных систем. Поскольку на каждом уровне используются только абстракции, обеспеченные на низших уровнях, можно легко изменить основные реализации, не затрагивая самой программы. Иерархическое представление также уменьшает риск появления неконтролируемых зависимостей между модулями. Существует простой тест на ортогональность проектирования. Как только вы составили схему компонентов, спросите себя: "Сколько

[2] база данных, интерфейс связи, промежуточное программное обеспечение и т.д.

[3] или планируем их появление в будущем

[4] Высказав это, мы тем самым поощряем стремление сотрудников более мелких подразделений постоянно общаться друг с другом.

модулей подвергнутся воздействию, если я резко изменю требования по конкретной функции?"В ортогональной системе ответ должен быть "один". Перемещение кнопки на панели графического интерфейса пользователя не должно требовать внесения изменений в схему базы данных. Добавление контекстно-зависимой справки не должно изменить подсистему выставления счетов. Рассмотрим сложную систему контроля и управления отопительной установкой. Первоначально требовалось наличие графического интерфейса, но затем требования были изменены, с тем чтобы добавить систему речевого ответа и управления установкой при помощи телефона с тональным набором. В ортогонально спроектированной системе для этого вам пришлось бы изменить только модули, связанные с интерфейсом пользователя, а основная логика управления предприятием остается неизменной. На самом деле, если вы тщательно структурируете систему, то у вас должна быть возможность поддержки обоих интерфейсов при наличии одной и той же программной базы. Стоит спросить себя, как защитить вашу конструкцию от изменений в окружающем мире, например, вы пользуетесь номером телефона в качестве идентификатора заказчика. Что произойдет, если телефонная станция изменит коды междугородней связи? Не полагайтесь на свойства предметов, которыми не можете управлять.

157.2.8 Инструментарии и библиотеки

Будьте внимательным, чтобы сохранить ортогональность вашей системы при введении инструментариев и библиотек, произведенных фирмами-субподрядчиками. Проявите мудрость при выборе технологии. Однажды авторы работали над проектом, в котором требовалось, чтобы некий фрагмент программы на языке Java выполнялся автономно — на сервере и в удаленном режиме — на клиентской машине. В этом случае возможными вариантами распределения классов были технологии RMI и CORBA. Если удаленный доступ к классу обеспечивался при помощи RMI, то в этом случае каждое обращение к удаленному методу в этом классе могло бы привести к генерации исключения, означающей, что эта наивная реализация потребовала бы от нас обработки этого исключения всякий раз при использовании удален-

ных классов. В данном случае использование RMI явно не ортогонально: программа, обращающаяся к удаленным классам, не должна зависеть от их физического расположения. Альтернативный способ — технология CORBA — не налагает подобного ограничения: мы можем написать программу, для которой не имеет значения, где физически находятся классы. Когда вы используете инструментарий (или даже библиотеку, созданную другими разработчиками), вначале спросите себя, не заставит ли он внести в вашу программу изменения, которых там быть не должно. Если схема долговременного хранения объекта прозрачна, то она ортогональна. Если же при этом требуется создание объектов или обращение к ним каким-либо особым образом, то она неортогональна. Отделение этих подробностей от вашей программы дает дополнительное преимущество, связанное с возможностью смены субподрядчиков в будущем. Интересным примером ортогональности является система Enterprise Java Beans. В большинстве диалоговых систем обработки запросов прикладная программа должна обозначать начало и окончание каждой транзакции. В системе EJB эта информация выражена описательно в виде метаданных вне любых программ. Та же прикладная программа может работать в различных транзакционных средах без каких-либо изменений. Вероятно, это станет прообразом многих операционных сред будущего. Другой интересной проверкой на ортогональность является технология Aspect Oriented Programming — исследовательский проект фирмы Xerox Parc. Технология AOP позволяет выразить в одном-единственном месте линию поведения, которая в противном случае была бы распределена по всему исходному тексту программы. Например, журнальные сообщения обычно генерируются путем явных обращений к некоторой функции записи в журнал по всему исходному тексту. Используя технологию AOP, вы реализуете процедуру записи в журнал ортогонально к записываемым данным. Используя версию AOP для языка Java можно записать сообщение журнала при входе в любой метод класса Fred, запрограммировав аспект:

```
aspect Trace {
advise * Fred.*(..) {
static before {
```

```
Log.write("-> Entering + thisJoinPoint.methodName); }
}
}
```

При вплетении этого аспекта в текст вашей программы будут генерироваться трассировочные сообщения. Если этого не сделать, не будет и сообщений. В обоих случаях исходный текст остается неизменным.

157.2.9 Написание текста программы

Всякий раз, когда вы пишете программу, вы подвергаетесь риску снижения уровня ортогональности вашего приложения. Если вы постоянно не отслеживаете не только то, что вы делаете, но и весь контекст приложения, то существует опасность неумышленного дублирования функциональных возможностей в некотором другом модуле или выражения существующих знаний дважды. Есть ряд методик, которые можно использовать для поддержки ортогональности:

- Сохраните вашу программу "несвязанной". Напишите "скромную" программу — модули, которые не раскрывают ничего лишнего для других модулей и не полагаются на их внедрение. Попробуйте применить закон Деметера , который обсуждается в разделе "Несвязанности и закон Деметера". При надобности изменения состояния объекта это должен делать сам объект. В таком случае программа остается изолированной от реализации другой программы, а вероятность того, что система останется ортогональной, увеличивается.
- Избегайте глобальных данных. Всякий раз, когда ваша программа ссылается на глобальные данные, она привязывается к другим компонентам, использующим эти данные. Даже глобальные переменные, которые вы собираетесь использовать только для чтения, могут вызвать проблемы (например, если вам нужно срочно изменить программу, сделав ее многопоточной). Вообще программа станет проще в понимании и сопровождении, если вы явно перешлете любой требуемый контекст в

ваши модули. В объектно-ориентированных приложениях контекст часто пересылается как параметр к конструктора! объектов. В другой программе вы можете создать конструкции, содержащие контекст, и обходить ссылки на них.

- Подобные функции. Зачастую вы сталкиваетесь с набором функций, похожих друг на друга; возможно, они используют общий фрагмент в начале и конце программы, но в ее середине каждая пользуется своим алгоритмом. Дублированная программа является признаком структурных проблем. Для того чтобы составить программу лучше, следует обратить внимание на шаблон Strategy в книге "Design Patterns". Пусть постоянное критическое отношение к вашей программе войдет у вас в привычку. Ищите любые возможности реорганизации для усовершенствования ее конструкции и повышения уровня ортогональности. Этот процесс называется реорганизацией.

157.2.10 Тестирование

Систему, спроектированную и реализованную ортогональным образом, намного проще тестировать. Поскольку взаимодействие между компонентами системы формализовано и ограничено, большая часть тестирования может осуществляться на уровне отдельных модулей. Это хорошо, поскольку подобное тестирование значительно легче поддается спецификации и выполнению, чем интеграционное тестирование. Мы предлагаем, чтобы каждый модуль был снабжен своим собственным встроенным тестом, и эти тесты выполнялись автоматически как часть обычной процедуры сборки. Процедура сборки модульного теста сама по себе является интересным тестом на ортогональность. Что требуется, чтобы собрать и скомпоновать тест модуля? Должны ли вы задействовать большую часть системы только для того, чтобы скомпилировать или скомпоновать тест? В этом случае модуль очень хорошо связан с оставшейся частью системы. Момент устранения ошибки также подходит для оценки ортогональности системы в целом. Когда вы сталкиваетесь с проблемой, оцените, насколько локален процесс ее устранения. Нужно изменить лишь один модуль, или изменения должны происходить по всей системе? Когда вы меняете

что-либо, устраняются ли при этом ошибки или происходит загадочное появление новых? Это удачный момент для внедрения автоматизации. Если вы применяете систему управления исходным текстом, комментируйте устранение ошибок, когда вы осуществляете возвращение модуля в библиотеку после тестирования. Затем вы можете генерировать ежемесячные отчеты, где анализируются тенденции в ряде исходных файлов, в которых производилось устранение ошибок.

157.2.11 Документация

Что удивительно, ортогональность применима и к документации. Координатами являются содержание и представление. Если документация действительно ортогональна, вы можете существенно изменить внешний вид, не изменяя содержания. Современные текстовые процессоры содержат стили и макрокоманды, которые помогают в этом.

157.2.12 Жизнь в условиях ортогональности

Ортогональность тесно связана с принципом DRY (анг. «don't repeat yourself» — «не повторяй самого себя»). DRY — это базовый принцип заявленный в книге «Программист-прагматик» Эндрю Ханта и Дэйва Томаса (The Pragmatic Programmer). Авторы расматривали этот принцип в контексте баз данных, тестовых планов, проектирования программных систем, а также документирования систем[1] Используя этот принцип, можно свести к минимуму дублирование в пределах системы, а при помощи ортогональности уменьшить взаимозависимость между компонентами системы.

Звучит неуклюже, но если вы используете принцип ортогональности в тесной связи с принципом DRY, вы обнаружите, что разрабатываемые вами системы становятся более гибкими, более понятными и более простыми в отладке, тестировании и сопровождении. Когда вы присоединяетесь к проекту, в котором люди ведут отчаянную борьбу за внесение изменений, а каждое изменение приводит к появлению четырех

новых проблем, вспомните кошмар с вертолетом. Вероятно, проект сконструирован и запрограммирован неортогонально.

Глава 158

Вспомогательные скрипты на языке Python

Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще ассоциируют именно со змеёй, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте `http://www.python.org` (до выхода версии 2.5) изображают змеиные головы.

Python[1] — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

Python удобно применять для написания различных вспомогательных скриптов. Часто его используют при разработке сложных программных систем для написания первых версий. В процессе работы над большими программами часто перерабатываются большие объемы кода, поэтому для ускорения разработки требуется максимально высокоуровневый язык. После того как архитектура программы стабилизируется, узким местом становится производительность, и программу переписывают на более низкоуровневом компилируемом языке, чаще всего C_{+}^{+}.

Написание программ упрощают:

- **объектно-ориентированное программирование** облегчает разработку программ, позволяет переопределить стандартные операторы для пользовательских типов данных, упрощая синтаксис
- **динамическая типизация** не требуется заранее упределять переменные, они создаются простым присваиванием
- **обработка исключений** для секции кода можно определить обработчик ошибок
- **высокоуровневые структуры данных** — списки, словари (набор элементов ключ:значение), очереди
- богатая стандартная библиотека и множество дополнительных библиотек на все случаи

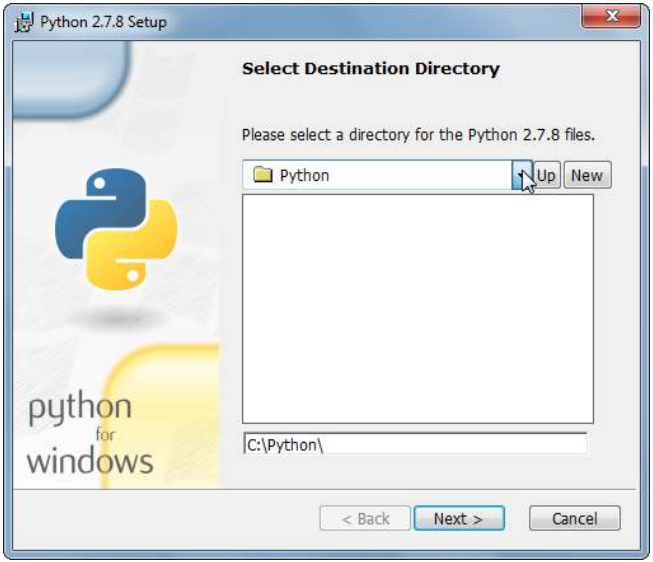
158.1 Установка под ⊞Windows

⊞ + R » http://www.python.org » Downloads » Python 2.7.8

python-2.7.8.msi » Setup » for all users/for me

[1] в оригинале читается **па́йтон**, но давно русифицировался как **пито́н**

Destination Directory » **C:/Python/** » Next



Customize » Python » Add python.exe to PATH » Next » Finish

158.2 Запуск

Из командной строки: ⊞ + R » cmd » python

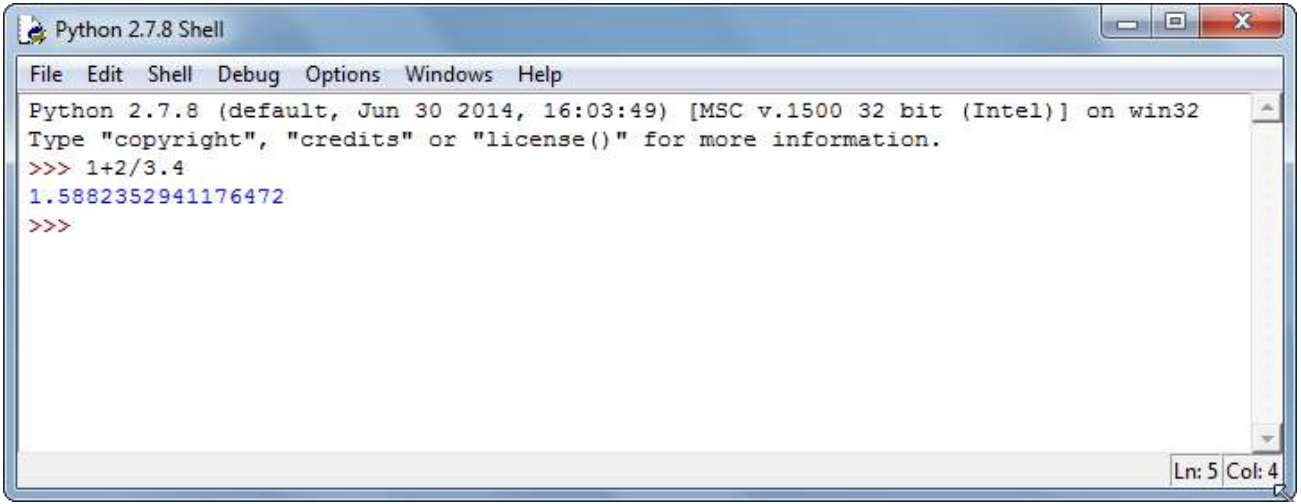
```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\dmitry>python
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+2/3.4
1.5882352941176472
>>> _
```

Простейшая среда IDLE[2]:

⊞ Программы Python 2.7 IDLE (Python GUI)
Панель задач IDLE ▷ Закрепить в панели задач ◁

[2] на GUI-библиотеке Tkinter, идущей в комплекте



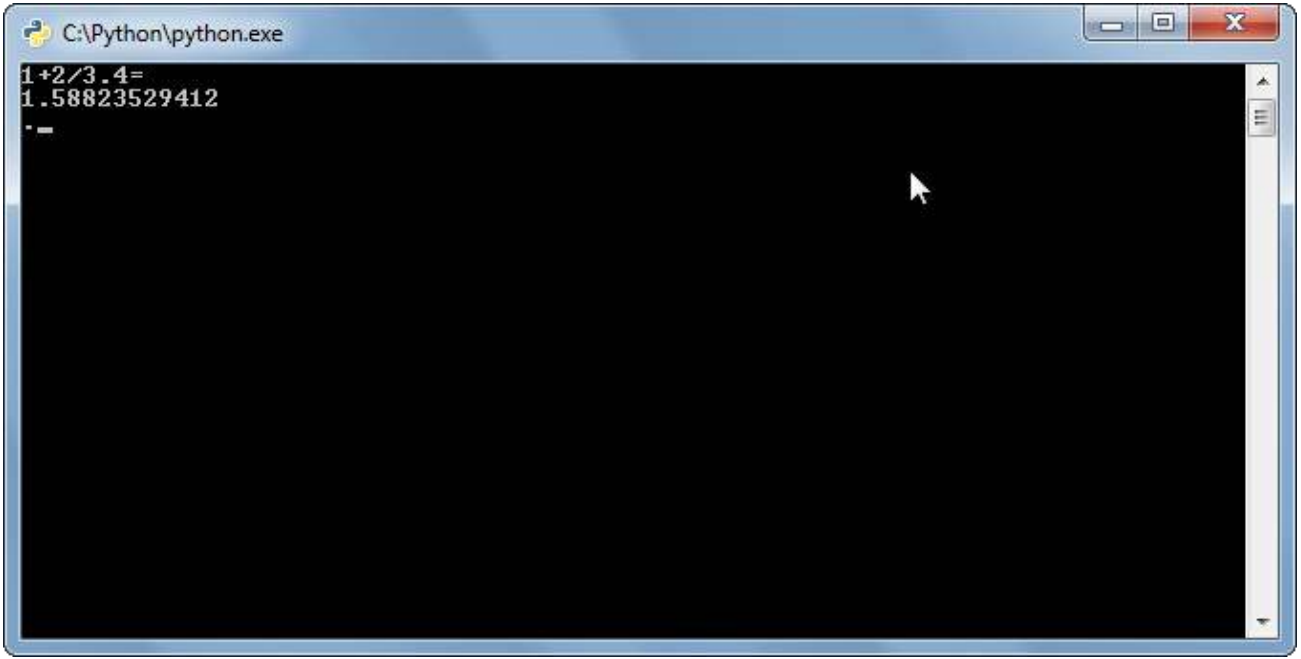
◁◁ по файлу скрипта:

⊞+R » notepad **/tmp/py.py**

/tmp/py.py

```
1 print "1+2/3.4="
2 print 1+2/3.4
3
4 raw_input('.')
```

⊞+R » /tmp/py.py



Открытием файла скрипта в IDLE:

⊞ + R » **/tmp/**
py.py » ▷ » Edit with IDLE
меню » Run » Run Module » F5

Python 2.7.8: py.py - C:\w\Azbuka\python\install\py.py

File Edit Format Run Options Windows Help

```
print "1+2/3.4="
print 1+2/3.4

raw_input('.')
```

Ln: 6 Col: 0

Python 2.7.8 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
1+2/3.4=
1.58823529412
.
>>>
```

Ln: 8 Col: 4

158.3 Дополнительные материалы

[**pyotkidach**] Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач, Язык программирования Python
[**pythink**] Аллен Дауни Думать на языке Python: Думать как компьютерный специалист

Глава 159

Make: управление сборкой проектов

Глава 160

VCS: системы контроля версий

160.1 CVS

160.2 Subversion

160.3 Git

160.3.1 GitHub

Глава 161

Основы Си и C_+^+

161.0.2 Установка MinGW (win32)

161.1 Особенности C_+^+ в embedded

Глава 162

LLVM и разработка собственных компиляторов

Глава 163

Лексический и синтаксический анализ

Очень часто в практике возникает необходимость работы с данными в текстовых форматах — plain text файлы, в которых в каком-либо формате[1] описаны данные. И от вас требуется реализовать разбор такого файла, выделяя элементы данных, чтобы в дальнейшем после их преобразования например записать текстовый файл в другом формате.

В таком виде хранятся результаты расчетных программ, работащих в пакетном режиме, данные с измерительных систем, поток данных с приемников GPS[2], очень популярный мета-формат XML со всеми его частными случаями типа HTML, XLIFF??, OpenDocument, тексты программ для станков с ЧПУ,...

С некоторыми хинтами точно так же можно работать и с бинарными файлами, преобразовав их сначала в текстовую форму (в простейшем случае просто сделав hex dump).

[1] на языке разметки, или DDL [D]ata [D]efinition [L]anguage

[2] протокол NMEA 0183

В некоторых случаях необходимо написание трансляторов форматов (текстовых) данных, или даже интерпретаторов/компиляторов языков программирования.

Все эти техники с использованием стандартных утилит flex и bison будут кратко описаны в этой главе. Подробнее эти техники рассмотрены в книгах163.3, особенно стоит отметить талмуд **DragonBook**163.3.

163.1 Лексер и лексический анализ, утилита flex

Лексер — программа или ее часть, которая

1. получает на вход исходные данные в виде сплошного потока одиночных сиволов,
2. группирует символы согласно набору правил, и
3. отдает на выходе символы, уже сгруппированные в лексе́мы или токе́ны.

Цель лексера — подготовить последовательность лексем для входа другой программы.

Лексический анализ — процесс программного разбора входной последовательности символов[3] с целью получения на выходе последовательности групп символов — токенов, имеющих собственное смысловое значение[4]. Как правило, лексический анализ производится в соответствии набора правил определённого формального, искуственного или компьютерного языка.

[3] например, такой как исходный код на одном из языков программирования

[4] подобно группировке букв в слово

Язык, а точнее его грамматика, задаёт определённый набор лексем, которые могут встретиться на входе лексера, и набор правил, по которым их следует группировать.

Традиционно принято организовывать процесс лексического анализа, рассматривая входную последовательность символов как поток одиночных символов. При такой организации лексер самостоятельно управляет выборкой отдельных символов из входного потока.

Распознавание лексем с учетом грамматики обычно производится путём их идентификации согласно идентификаторам токенов, определяемых грамматикой языка. При этом любая последовательность символов входного потока (лексема), которая согласно грамматике не может быть идентифицирована как токен языка, обычно рассматривается как специальный токен-ошибка.

Каждый выделенный токен можно представить в виде парной структуры, содержащей

1. идентификатор токена и
2. саму последовательность символов лексемы, выделенной из входного потока[5].

Рассморим обработку текстового файла: описания простой графической фигуры в текством формате IGES: универсальном формате обмена данными для САПР. IGES-файл состоит из 80-символьных ASCII-записей[6]. Текстовые строки представлены в «Холлерит»-формате – число символов в строке, плюс латинская буква [H] и сама строка, например — «4HSLOT». Рассмотрим очень короткий IGES-файл 1987 года, включающий в себя лишь сущности пары точек (POINT, тип 116), пары полуокружностей (CIRCULAR ARC, тип 100) и двух линий (LINE, тип 110). Можно его считать эскизом паза под шпонку на валу, или шаблоном некруглого монтажного отверстия.

[5] запись строки, числа и т. д.

[6] длина записи произошла из эры перфокарт, типовая ширина вывода терминала или принтера

Файл разделен на 5 секций, обозначенных буквами (S, G, D, P, или T) в колонке 73. Характеристики и геометрическая информация для каждой сущности поделены между двумя секциями; одна — в формате фиксированной длины[7], другая в составной записи, с фиксированной точкой[8]. При отображении мы должны видеть две желтые точки, одна из которых в начале координат [0,0,0], две красных полуокружности и две зеленые линии.

sample.iges

```
                                                                        S      1
1H,,1H;,4HSLOT,37H$1$DUA2:[IGESLIB.BDRAFT.B2I]SLOT.IGS;,                G      1
17HBravo3  BravoDRAFT,31HBravo3->IGES  V3.002  (02-Oct-87),32,38,6,38,15,  G      2
4HSLOT,1.,1,4HINCH,8,0.08,13H871006.192927,1.E-06,6.,                   G      3
31HD. A. Harrod, Tel. 313/995-6333,24HAPPLICON - Ann Arbor, MI,4,0;     G      4
     116       1       0       1       0       0       0       0      1D      1
     116       1       5       1       0                              0D      2
     116       2       0       1       0       0       0       0      1D      3
     116       1       5       1       0                              0D      4
     100       3       0       1       0       0       0       0      1D      5
     100       1       2       1       0                              0D      6
     100       4       0       1       0       0       0       0      1D      7
     100       1       2       1       0                              0D      8
     110       5       0       1       0       0       0       0      1D      9
     110       1       3       1       0                              0D     10
     110       6       0       1       0       0       0       0      1D     11
     110       1       3       1       0                              0D     12
```

[7] [D]irectory [E]ntry секция
[8] [P]arameter [D]ata секция

```
18 116,0.,0.,0.,0,0,0;                                  1P      1
19 116,5.,0.,0.,0,0,0;                                  3P      2
20 100,0.,0.,0.,0.,1.,0.,-1.,0,0;                       5P      3
21 100,0.,5.,0.,5.,-1.,5.,1.,0,0;                       7P      4
22 110,0.,-1.,0.,5.,-1.,0.,0,0;                         9P      5
23 110,0.,1.,0.,5.,1.,0.,0,0;                          11P      6
24 S        1G        4D        12P        6                T      1
```

163.2 Компилятор Паскаля

163.3 Дополнительная литература

[**dragonbook**] **Книга Дракона**: Ахо, Сети, Ульман Принципы построения компиляторов.

Habr: Компиляция. 1: лексер
Habr: Компиляция. 2: грамматики
Habr: Компиляция. 3: бизон
Habr: Компиляция. 4: игрушечный ЯП
Habr: Компиляция. 5: нисходящий разбор
Habr: Компиляция. $5\frac{1}{2}$: llvm как back-end
Habr: Компиляция. 6: промежуточный код

Глава 164

Сборка кросс-компилятора GNU toolchain

Часть XIII

Микроконтроллеры Cortex-Mx

Глава 165

Отладочные платы

165.1 STM32DISCOVERY /Cortex-M3 STM32F103/

165.2 STM32F4DISCOVERY /Cortex-M4 STM32F407/

Часть XIV

Периферия

Часть XV

Встраиваемый emLinux

Linux для встраиваемых систем[1] — популярный метод быстрого создания комплекса ПО для больших сложных приложений, работающих на достаточно мощном железе, особенно предполагающих интенсивное использование сетевых технологий.

За счет использования уже существующей и очень большой базы исходных текстов ядра, библиотек и программ для Linux, бесплатно доступных в т.ч. и для коммерческих приложений, можно на порядки сократить стоимость разработки собственных программных компонентов, и при этом получить готовую команду бесплатных стронних разработчиков, уже знакомых с созданием ПО для Linux.

Из недостатков можно отметить:

- Отсутствие полноценной поддержки режима жесткого реального времени;
- Тяжелое ядро;
  - Поддерживаются только мощные семейства процессоров[2];
  - Значительные требования по объему RAM и общей производительности;
- Дремучесть техспециалистов, контуженных ТурбоПаскалем и Windowsом;

Для сборки emLinux-системы используется метод кросс-компиляции, когда используется кросс-тулчейн, компилирующий весь комплект ПО для компьютера с другой архитектурой. Типичный пример — сборка ПО на ПК с процессором Intel i7 для Raspberry Pi или планшета на процессоре AllWinner/Tegra/. . . .

emLinux очень широко применяется на рынке мобильных устройств[3], и устройств интенсивно использующих сетевые протоколы (роутеры, медиацентры).

В качестве примера применения рассмотрим относительно простое приложение: многофункциональные настенные часы с синхронизацией времени через Internet, с будильником, медиапроигрывателем, блэкджеком и плюшками.

---

[1] будем называть его emLinux

[2] 32-бит, необходим блок MMU

[3] в т.ч. является основой Android

165.3 Загрузчик syslinux

Самый простой и удобный загрузчик для i386-систем, ставится на флешку из под Windows, работает с FAT-разделами, поддерживает загрузку с флешек, CDROM и по сети.

http://www.syslinux.org/

165.3.1 Закачка

.zip с бинарной сборкой syslinux:
https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.zip

memtest86+ — полезная утилита для тестирования RAM:
http://www.memtest.org/download/5.01/memtest86+-5.01.zip

Если планируете устанавливать рабочую станцию для сборки azLinux с флешки, нужно скачать полные или netinst установочные **.iso**-образы:
Установочный образ Debian Linux i386:
http://cdimage.debian.org/debian-cd/7.7.0/i386/iso-cd/debian-7.7.0-i386-netinst.iso
Установочный образ Debian Linux amd64:
http://cdimage.debian.org/debian-cd/7.7.0/amd64/iso-cd/debian-7.7.0-amd64-netinst.iso
Сборка HDD-инсталлятора i386:
http://http.us.debian.org/debian/dists/wheezy/main/installer-i386/current/images/hd-media/initrd.gz
http://http.us.debian.org/debian/dists/wheezy/main/installer-i386/current/images/hd-media/vmlinuz

Сборка HDD-инсталлятора amd64:
http://http://http.us.debian.org/debian/dists/wheezy/main/installer-amd64/current/images/hd-media initrd.gz
http://http://http.us.debian.org/debian/dists/wheezy/main/installer-amd64/current/images/hd-media vmlinuz

165.3.2 Установка под ⊞Windows на флешку

Распакуйте файлы из **.zip**

/bios/win32/syslinux.exe	консольный инсталлятор
/bios/com32/menu/menu.c32	модуль текстового меню
/bios/com32/menu/vesamenu.c32	модуль графического меню (VESA)
	служебные библиотеки syslinux
/bios/com32/libutil/libutil.c32	
/bios/com32/lib/libcom32.c32	

Для установки syslinux на флешку с FAT, на которую назначена буква **F:**, выполните батник:

/syslinux/syslinuxusb.bat

```
syslinux -i -m -a -d syslinux F:
pause
```

-i	install	установить
-m	MBR	в MBR
-a	active	сделать раздел активным
-d	directory	в каталог **syslinux**

Если ставите Debian, распакуйте из **debian-netinst.iso** в **F:/Debian/**

debian-7.7.0-amd64-netinst.iso	**.iso**-образ установочного CD-ROM
debian-7.7.0-i386-netinst.iso	**.iso**-образ установочного CD-ROM
.iso/install.amd/vmlinuz	ядро amd64 (x64)
.iso/install.amd/initrd.gz	ramdisk с инсталляром
.iso/install.386/vmlinuz	ядро i386 (x32)
.iso/install.386/initrd.gz	ramdisk с инсталляром

165.3.3 syslinux.cfg

syslinux настривается текстовым файлом **syslinux.cfg**.

/syslinux/syslinux.cfg

```
UI vesamenu.c32
MENU RESOLUTION 640 480
MENU TITLE azLinux
MENU BACKGROUND /syslinux/splash640x480.png

DEFAULT azmicro
LABEL azmicro
MENU LABEL azLinux micro
KERNEL /azLinux/micro.kernel
INITRD /azLinux/micro.initrd
APPEND vga=none

```

```
13 LABEL azclock
14 MENU LABEL azLinux clock
15 KERNEL /azLinux/clock.kernel
16 INITRD /azLinux/clock.initrd
17 APPEND vga=ask
18
19 LABEL memtest
20 MENU LABEL memtest86+
21 KERNEL /syslinux/memtest.krn
22
23 LABEL debian64
24 MENU LABEL Debian GNU/Linux 7.7.0-amd64-netinst
25 KERNEL /Debian/amd64/vmlinuz
26 INITRD /Debian/amd64/initrd.gz
27 APPEND vga=none
28
29 LABEL debian32
30 MENU LABEL Debian GNU/Linux 7.7.0-i386-netinst
31 KERNEL /Debian/i386/vmlinuz
32 INITRD /Debian/i386/initrd.gz
33 APPEND vga=none
```

В примере показана реализация с использованием графического VESA меню. Для использования более надежного текстового меню замените на `UI menu.c32`.

Обратите внимание на возможность включения нестандартных видеорежимов используя **[VESA]MENU RESOLUTION**: этот финт нужен для включения графики на ASUS EeePC 701: режим 800×480 недоступен для включения через параметр ядра **vga=**, поэтому приходится использовать возможности syslinux.

Глава 166

azLinux

Чтобы разобраться как можно собрать встраивамый Linux, в этом разделе описан набор Make-файлов и файлов конфигурации для сборки минимального emLinux. Это обрезанный форк проекта Cross Linux, подробнее описанного в разделе ??. Ограничено количество поддерживаемого железа, упрощены конфигурационные файлы, минимизировано количество библиотек и программных пакетов.

Изначально идея создания этой системы появилась из желания заменить тухлую связку x86/DOS/TurboPascal на что-то

- более переносимое: на энергоэффективное ARM/MIPS-железо, в т.ч. (типа)отечественного производства,
- стабильное: с полноценной многозадачностью, защитой памяти и данных, и
- позволяющее использовать максимум возможностей аппаратуры: большая RAM, ECC, gcc-оптимизированный 32/64-битный код, USB, CAN, Ethernet, WiFi, разнообразные носители данных, аппаратный watchdog.

- Также большой интерес представляют десятки готовые библиотек сжатия и кодирования данных, численных методов, ЦОС, и обработки изображений, а также
- множество готовых программ, доступных в исходных кодах[1] для выполнения различных полезных функций: сетевые серверы, символьная математика, обработка данных,...
- Еще одна ключевая фича — способность Linux полностью загружаться в RAM с **любых** носителей, в т.ч. заблокированных на запись. Это важно для случаев, когда возможны внезапные выключения питания: вся система работает в RAM-диске, а корректность записи данных на изменяемые носители можно гибко контролировать программно. При запуске после аварийного выключения никаких проверок файловых систем не требуется, ОС стартует сразу, а проверку/починку разделов данных возможно выполнять в фоновом режиме.
- Время запуска системы — на x86 удалось экспериментально получить время запуска 0.2 сек от загрузки ядра до начала выполнения пользовательского кода. Используя модульное ядро, возможно выполнить критический к времени запуска пользовательский код до инициализации USB, сети, внешних носителей данных и тяжелых сервисов.

Почему не BuildRoot Эта система сборки создавалась как максимально облегченный пакет для решения узких задач, и для освоения технологии кросс-компиляции. Предполагается что функциональное наполнение не будет развиваться шире набора:

- ядро (реального времени)
- μlibc
- урезанная командная оболочка (busybox)
- несколько прикладных библиотек поддержки (сжатие, кодирование, базовая графика)
- пользовательский узкоспециализированный код на Си/C_+^+

[1] для использования как есть, изучения принципов работы и модификации под собственные нужды

Расширять функционал, добавляя libQt, X Window, Apache, MySQL,..., Gnome/KDE и т.д. не планируется в принципе — это система для решения узких прикладных задач на аппаратуре с минимальными ресурсами[2]. Интерактивная работа с пользователем также не предполагается, доступна только командная консоль[3], и очень ограниченные графические и мультимедийные возможности. Если ваши хотелки выходят за этот функционал, рекомендую сразу уходить на использование широко известной системы кросс-сборки Linux-систем под названием **BuildRoot??**.

166.1 Требования к системе сборки (BUILD-хост)

Требования жесткие — 2х-ядерный процессор, 2+Гб RAM, для 4+ Гб RAM нужен 64х-битный дистрибутив Linux (рекомедую Debian), и естественно никаких виртуалок. Возможна установка системы на флешку, в этом случае требования к RAM еще более ужесточаются — потребуется каталоги с временными файлами смонтировать как **tmpfs**:

добавить в **/etc/fstab**

```
1 tmpfs  /home/user/Azbuka/azlin/tmp    tmpfs    auto,uid=user,gid=user 0 0
2 tmpfs  /home/user/Azbuka/azlin/src    tmpfs    auto,uid=user,gid=user 0 0
3 tmpfs  /home/user/.ccache             tmpfs    auto,uid=user,gid=user 0 0
```

Можно попытаться сделать билд-сервер и на худшем железе, но будьте готовы к тормозам или внезапному окончанию памяти — ресурсоемка сборка тяжелых библиотек типа libQt или крупных пакетов типа gcc.

[2] особенно интересны процессорные модули в DIMM форм-факторе, только CPU, RAM, NAND и GPIO гребенка

[3] ее можно считать сервисным режимом

Вы можете попробовать поставить Linux на виртуалку, на флешку, и на жесткий диск (если найдете место) и оценить возможности этих вариантов на сборке пакета gcc0. При сборке с флешки на ноутбуке с 2Гб RAM мне для сборки gcc0 пришлось временно размонтировать **cross/src**, сделать **./mk.rc && make gcc ramclean**, а потом примонтировать **tmpfs** опять на **src**.

Сборка под MinGW/Cygwin совершенно неживая. Если совсем никак без винды — используйте виртуалки, и будьте готовы ждать.

166.2 Понятие пакет

Прежде чем продолжить, введем понятие пакет. В azLinux пакетом называется одна или несколько частей скриптов сборки, обозначаемых именем. В чем-то это похоже на бинарные пакеты обычных дистрибутивов Linux — чтобы добавить в систему какой-то функционал, мы устанавливаем бинарный пакет. Но есть и отличие: пакет дистрибутива это реальный архивный файл, содержащий в себе файлы программ, данных; в azLinux пакет — виртуальная штука с именем.

Просматривая файлы в каталоге **mk/**, легко найти имена пакетов по шаблону:

```
.PHONY: somename
somename: [зависимые файлы]
    [команда1]
    ...
```

Если вы запустите команду:

```
cd ~/az ; ./mk.rc && make somename
```

запустится сборка пакета somename.

Но не нужно забывать, что кроме этой секции в **.mk**, существуют зависимости между файлами, при работе команд сборки динамически создаются и изменяются файлы, иногда что-то скачивается из Interneta — все эти процессы тоже входят в пакет.

Часть пакетов не связана со сборкой программ, а выполняют служебные функции, поэтому для них правильнее будет фраза запуск пакета.

Все действия выполняются с помощью команды make. Обратите особое внимание на то, что **Makefile** собирается скриптом mk.rc из частей в каталоге **mk/**, поэтому если вы что-то меняете в скриптах, не забудьте сначала запустить ./mk.rc.

mk.rc

```
#!/bin/sh
cat \
    mk/head.mk \
    mk/dirs.mk \
    mk/versions.mk \
    mk/packages.mk \
    mk/commands.mk \
    mk/clean.mk \
    mk/gz.mk \
    mk/src.mk \
    mk/cfg.mk \
    mk/cross.mk \
    mk/core.mk \
    mk/kernel.mk \
```

```
15     mk/ulibc.mk \
16     mk/busybox.mk \
17     mk/user.mk \
18     mk/libs.mk \
19     mk/apps.mk \
20     mk/emu.mk \
21     mk/root.mk \
22     mk/boot.mk \
23 > Makefile
```

166.3 Клонирование проекта azLinux

При необходимости вносить правки[4] работайте с вашим собственным форком на GitHub.

Получите клон пакета из репозитория:

```
cd ~ ; git clone --depth=1 -o gh https://github.com/user/azlin az
```

При необходимости обновитесь:

```
cd ~/az ; git pull
```

[4] что естественно — вам потребуется добавлять свои пакеты и поддержку железа

166.4 Общий порядок сборки

Каждый пакет собирается командой:

```
./mk.rc && make [HW=rpi] [APP=clock] <package>
```

1. dirs создание дерева каталогов 166.6
2. gz закачка архивов исходников 166.7
3. tc сборка кросс-компилятора 166.11.2
   (a) binutils ассемблер, линкер и утилиты 166.11.2
   (b) cclibs библиотеки для сборки gcc 166.11.2
   (c) gcc0 сборка минимального кросс-компилятора Си 166.11.2
4. core сборка основной системы 166.11.3
   (a) kernel ядро Linux 166.11.3
   (b) ulibc библиотека uClibc 166.11.3
   (c) busybox набор утилит busybox 166.11.3
   (d) gcc пересборка полного кросс-компилятора Си/C_{+}^{+}166.11.3
5. libs сборка библиотек **${LIBS}** 166.11.4
6. apps сборка прикладных пакетов **${APPS}** 166.11.5
7. user сборка пользовательского кода 166.11.6
8. root формирование корневой файловой системы 166.11.7
9. boot сборка загрузчика 166.11.8 syslinux/grub/uboot
10. emu запуск собранной системы в эмуляторе 166.12
11. netboot сетевая загрузка 166.13
12. firmware прошивка на устройство 166.14

166.5 Фиксация переменных

Если вам требуется собрать систему со значениями переменных, отличающихся от тех, которые прописаны в make-файлах, при запуске всех пакетов нужно указывать трюбуемые значения в командной строке. Это позволить легко выбрать нужный вам вариант сборки.

```
./mk.rc && make HW=rpi APP=clock distclean dirs tc core libs apps boot root
```

При таком указании все переназначения для этих переменных игнорируются, поэтому возможны некоторые сложности с указанием например опций оптимизации.

166.6 dirs: Создание дерева каталогов

После загрузки или обновления запустите пакет dirs:

```
$ ./mk.rc && make dirs
mkdir -p /home/user/Azbuka/azlin/gz /home/user/Azbuka/azlin/src
/home/user/Azbuka/azlin/tmp /home/user/Azbuka/azlin/x86_64-linux-gnu
/home/user/Azbuka/azlin/qemu386-clock /home/user/Azbuka/azlin/qemu386-clock/boot

$ ls -la
итого 72
-rwxr-xr-x 1 user user   121 Дек  8 11:43 mk.rc
drwxr-xr-x 2 user user  4096 Дек  8 11:43 mk
drwxr-xr-x 2 user user  4096 Дек  8 11:43 app
```

```
drwxr-xr-x 2 user user  4096 Дек  8 11:43 hw
drwxr-xr-x 2 user user  4096 Дек  8 11:43 cpu
drwxr-xr-x 2 user user  4096 Дек  8 11:43 arch
drwxr-xr-x 2 user user  4096 Дек  8 13:26 gz
drwxr-xr-x 2 user user  4096 Дек  8 11:43 app
drwxr-xr-x 3 user user  4096 Дек  8 13:26 qemu386micro
drwxr-xr-x 2 user user  4096 Дек  8 13:26 qemu386micro.cross
-rw-r--r-- 1 user user   147 Дек  8 11:43 README.md
-rw-r--r-- 1 user user 17699 Дек  8 13:25 azlin.tex
drwxr-xr-x 2 user user  4096 Дек  8 13:26 src
drwxr-xr-x 2 user user  4096 Дек  8 13:26 tmp
-rw-r--r-- 1 user user  2087 Дек  8 13:26 Makefile
```

Пакет dirs прописан в файле

mk/dirs.mk

```
# directories processing
# sw sources mirror archive in .tar.[GZ]
GZ = $(PWD)/gz
# [S]ou[RC]e codes unpacked
SRC = $(PWD)/src
# [T]e[MP] build dirs
TMP = $(PWD)/tmp
# build/target triplets
BUILD = $(shell gcc -dumpmachine)
# target root filesystem
```

```
11 ROOT = $(PWD)/$(HW)$(APP)
12 # cross-compiler [T]ool[C]hain
13 TC = $(ROOT).cross
14 BOOT = $(ROOT)/boot
15 ETC = $(ROOT)/etc
16 USR = $(ROOT)/usr
17 USRBIN = $(USR)/bin
18 USRLIB = $(USR)/lib
19 DIRS = $(GZ) $(SRC) $(TMP) $(TC) $(ROOT) $(BOOT) $(USR) $(USRBIN) $(USRLIB)
20 .PHONY: dirs
21 dirs:
22 ____mkdir -p $(DIRS)
```

Встроенная переменная **PWD** содержит полное имя каталога, из которого был запущен make.

Каталог зеркала архивов исхоных текстов программ

GZ

```
1 GZ = $(PWD)/gz
```

Каталог распаковки исходных текстов: некоторые пакеты должна собираться в дереве исходников

SRC

```
1 SRC = $(PWD)/src
```

Каталог out-of-tree сборки: остальные пакеты умеют собираться вне дерева исходников, если хватает RAM этот каталог удобно монировать как **tmpfs**

TMP

```
TMP = $(PWD)/tmp
```

Переменная **BUILD** задает триплет системы, на которой вы собираете: x86_64-linux-gnu, i686-linux-gnu или что-то подобное. Для получения триплета используется подстановка строки, выдаваемой запуском gcc.

BUILD

```
BUILD = $(shell gcc -dumpmachine)
```

Каталог в который собирается кросс-компилятор. Используется триплет рабочей Linux-системы.

TC

```
TC = $(ROOT).cross
TC = $(ROOT)/etc
```

Каталог целевой **rootfs**. Отдельно прописан загрузочный каталог, в который будет записываться собранное ядро, образ initrd, бинарники и конфиги загрузчика. Имя **ROOT** создается из двух переменных, описанных далее: имя аппаратной платформы **HW**166.9 и имени приложения **APP**166.8.

ROOT/BOOT

```
ROOT = $(PWD)/$(HW)$(APP)
BOOT = $(ROOT)/boot
```

Список всех рабочих каталогов в одной переменной:

DIRS

```
DIRS = $(GZ) $(SRC) $(TMP) $(TC) $(ROOT) $(BOOT) $(USR) $(USRBIN) $(USRLIB)
```

166.7 gz: Загрузка архивов исходников

mk/gz.mk

```
# download sw packages sources to gz/

.PHONY: gz
gz:
	make gz_cc
	make gz_core
	make gz_libs

.PHONY: gz_cc
gz_cc:
	$(WGET) http://ftp.gnu.org/gnu/binutils/$(BINUTILS).tar.bz2
	$(WGET) http://gcc.skazkaforyou.com/releases/$(GCC)/$(GCC).tar.bz2
	$(WGET) ftp://ftp.gmplib.org/pub/gmp/$(GMP).tar.bz2
	$(WGET) http://www.mpfr.org/mpfr-current/$(MPFR).tar.bz2
	$(WGET) http://www.multiprecision.org/mpc/download/$(MPC).tar.gz

.PHONY: gz_core
gz_core:
	$(WGET) https://www.kernel.org/pub/linux/kernel/v3.x/$(KERNEL).tar.xz
	$(WGET) http://www.uclibc.org/downloads/$(ULIBC).tar.xz
	$(WGET) http://busybox.net/downloads/$(BUSYBOX).tar.bz2

.PHONY: gz_libs
```

```
gz_libs:
	$(WGET) https://www.libsdl.org/release/$(SDL).tar.gz
```

166.8 APP: Приложение

Приложение — короткое кодовое название вашего варианта сборки системы в целом.

Приложение задается в файле **mk/head.mk** через переменную **APP**, доступные значения:

1. micro: минимальная версия системы, только командная консоль
2. clock: простые Linux-powered электронные часы

Значение переменной **APP** по умолчанию задано в **mk/head.mk**:

APP @ hw/head.mk

```
APP = micro
```

Если вам нужно собрать другое приложение, вы можете переопределить значение из командной строки при запуске всех пакетов:

```
./mk.rc && make APP=micro distclean dirs tc core libs apps
```

В **app/${APP}.mk** в переменных задаются:

- **LIBS**: набор используемых библиотек
- **PACKS**: набор используемых программных пакетов 166.11

app/micro.mk

```
# app: micro
```

```
LIBS =
APPS = hello
```

app/clock.mk

```
# app: clock
LIBS = sdl
APPS = hello
```

166.9 HW: Поддерживаемое железо

Конфигурация целевого железа задается в файле **mk/head.mk** через переменную **HW**, доступные значения приведены в таблице:

HW	CPU	ARCH	RAM	HD	SD	USB	Eth	WiFi	GPIO
qemu386	i486sx	i386	32M+	☐ IDE		☐	ne2k		
eeepc701	CeleronM	i386	512M+	☐ SSD	☐ SD	☒	A??	☐ AR2425	
gac1037	Celeron1037U	x86_64	1G+	☐ SATA		☒	2×RTL8111		
qemuARM		arm	32M+						
cubie1	AllWinnerA10	armhf	1G		☐ μSD	☒			
rpi	BCM2835	armel	512M		☐ SD	☒			
qemuMIPS		mips	32M+						
mr3020	AR7240	mips	32M	4M		☒		☐ AR9331	
vocore	RT5350	mips	32M	8M		☐		☐ SoC	
bswift		AR9331	mips	64M	16M	☐			20+

mk/head.mk

```
# [H]ard [W]are: qemu386 qemuARM qemuMIPS cubie1 rpi ...
HW = qemu386
# [APP]lication: micro clock
APP = micro
# load extra hw definitions: ARCH CPU ...
include hw/$(HW).mk
# load extra defs for CPU setted in $(HW).mk
include cpu/$(CPU).mk
# load extra defs for ARCHitecture setted in $(CPU).mk
include arch/$(ARCH).mk
# load extra app defs: LIBS APPS ...
include app/$(APP).mk
```

166.9.1 i386

Персоналки с архитектурой **i386** — самое сложное семейство с точки зрения поддержки. Комбинации процессоров, материнских плат и плат расширений дают сотни вариантов конфигураций, в т.ч. десятки моделей компьютеров в формате PC/104 и промышленных панелей. Особенно доставляет тот факт, что 95% периферии имеет закрытые бинарные драйвера только для ⊞Windows, поэтому будьте аккуратны с выбором железа.

qemu386: эмулятор QEMU

hw/qemu386.mk

```
# QEMU emulator: i386 mode
CPU = i486sx
QEMU_CFG = -m 32M -net none
```

eeepc701: ASUS Eee PC 701

hw/eeepc701.mk

```
# ASUS Eee PC 701
CPU = CeleronM
```

gac1037: Gigabyte GA-C1037UN-EU rev.2

CPU	Celeron 1037U
RAM	2 × DDR3, max 16G, 2 канала
чипсет	Intel NM70
сеть	2 × Realtek® GbE 1Gb (rtl8111)
HDD	2 × SATA2 (3Gb/s), 1 × SATA3 (6Gb/s), 1 × eSATA
USB	6 × USB3.0
видео	IntelGMA, выходы на VGA D-Sub и HDMI 1.4
аудио	Realtek ALC887 (HDA)
PCI	×1

В качестве варианта 64-битной платформы взята портативная материнка для неттопов: Gigabyte GA-C1037UN-EU. На ней возможны два варианта сборки:

1. x86_64/x64: нативный `CPU=Celeron1037u`
2. i686/x32: режим соместимости со старыми процессорами `CPU=i686`

http://www.ixbt.com/news/hard/index.shtml?17/33/31

На текущий момент эта материнка — оптимальный вариант для офисного, рабочего неигрового компьютера, или базы для изготовления мобильной рабочей станции в миникейсе: комплект из GA-C1037UN-EU и блока питания стоит порядка 5 тыс.руб, при этом возможна установка до 2×8G RAM, что пока недоступно на дешевых ноутбуках. Но — ОТВРАТИТЕЛЬНЫЙ радиатор на мосте NM70, нагрев до 70^oC, в обязательном порядке делать сквозную продувку корпуса **до включения материнки**.

материнская плата	GA-C1037UN-EU rev.2
CPU	Celeron 1037U (впаян)
охлаждение	отвратительное, обязательны кулера на все чипы и сквозная продувка корпуса,
RAM	1×8G DDR3 (в планах 2×8G)
HDD	нет, используется флешка 8G USB3 Transcend JF750 (возможно SATA SSD)
TFT	китайский 3.5" автомониторчик + конвертер HDMI2RCA на случай посмотреть видеовывод, обычно везде где я использую эту поделку, есть VGA монитор или хотя бы большой телевизор
электропитание	БП Hipro HPE-350W + автоинвертор батарея не требуется, под боком всегда есть 220 или 12 В по необходимости в транспорт грузится пара заряженных автоаккумуляторов

hw/gac1037.mk

```
1 # Gigabyte GA-C1037UN-EU
2 CPU = Celeron1037U
```

166.9.2 ARM

qemuARM: эмулятор QEMU

hw/qemuARM.mk

---

cubie1: Cubie Board v.1

rpi: Raspberry Pi model B

166.9.3 MIPS

qemuMIPS: эмулятор QEMU

hw/qemuMIPS.mk

---

mr3020: роутер MR3020

mr3020

vocore: VoCore

vocore

bswift: BlackSwift

bswift

166.10 CPU: Конфигурации процессоров

Настройки на процессор задаются в файле **cpu/${CPU}.mk**.

ARCH	архитектура целевой системы, используется при конфигурировании ядра
TARGET	триплет целевой системы, параметр задает тип целевой системы при сборке кросс-компилятора и используется во всех скриптах configure при сборке остальных пакетов
CFG_CPU	параметры при сборке кросс-компилятора
CPU_FLAGS	параметры gcc для оптимизации кода

166.10.1 i386

cpu/i486sx.mk

```
# i486sx cpu
# target arch is Intel x86 (32 bit)
ARCH = i386
# target triplet for embedded linux
TARGET = i486-linux-uclibc
# target CPU configure params
CFG_CPU = --disable-mmx --disable-3dnow --disable-sse
```

cpu/CeleronM.mk

```
# Intel Celeron M ULV 353
ARCH = i386
TARGET = celeronm-linux-uclibc
```

cpu/Celeron1037U.mk

```
# Intel Celeron 1037U
ARCH = i386
TARGET = celeronm-linux-uclibc
```

166.10.2 ARM

166.10.3 MIPS

cpu/AR7240.mk

```
# Atheros AR7240 CPU (400Mhz)
ARCH = mips
TARGET = mips-linux-uclibc
```

cpu/RT5350.mk

```
# Ralink RT5350 360MHz
ARCH = mips
TARGET = mips-linux-uclibc
```

166.11 Пакеты

166.11.1 versions.mk: Версии пакетов

tc: кросс-компилятор

mk/versions.mk

```
BINUTILS_VER = 2.24
GMP_VER = 5.1.3
MPFR_VER = 3.1.2
MPC_VER = 1.0.2
GCC_VER = 4.9.1
```

core: ядро

mk/versions.mk

```
KERNEL_VER = 3.17.6
ULIBC_VER = 0.9.33.2
BUSYBOX_VER = 1.22.1
```

boot: загрузчики

mk/versions.mk

libs: библиотеки

mk/versions.mk

```
SDL_VER = 1.2.15
```

166.11.2 tc: сборка кросс-компилятора

```

CFG_BINUTILS = --target=$(TARGET) $(CFG_ARCH) $(CFG_CPU) \
    --with-sysroot=$(ROOT) \
    --with-native-system-header-dir=/include \
    --enable-lto

CFG_CCLIBS = --disable-shared \
    --with-gmp=$(TC) --with-mpfr=$(TC) --with-mpc=$(TC)

CFG_GMP = $(CFG_CCLIBS)
CFG_MPFR = $(CFG_CCLIBS)
CFG_MPC = $(CFG_CCLIBS)

CFG_GCC0 = $(CFG_BINUTILS) $(CFG_CCLIBS) \
    --disable-shared --disable-threads \
    --without-headers --with-newlib \
```

```
17 ____--enable-languages="c"
18
19 CFG_GCC = $(CFG_BINUTILS) $(CFG_CCLIBS) \
20 ____--enable-threads --enable-libgomp \
21 ____--enable-languages="c,c++"
22
23 .PHONY: tc
24 tc: binutils cclibs gcc0
25
26 .PHONY: binutils
27 binutils: $(SRC)/$(BINUTILS)/README
28 ____rm -rf $(TMP)/$(BINUTILS) && mkdir $(TMP)/$(BINUTILS) &&\
29 ____cd $(TMP)/$(BINUTILS) &&\
30 ____$(SRC)/$(BINUTILS)/$(BCFG) $(CFG_BINUTILS) &&\
31 ____$(MAKE) && make install-strip
32
33 .PHONY: cclibs
34 cclibs: gmp mpfr mpc
35
36 .PHONY: gmp
37 gmp: $(SRC)/$(GMP)/README
38 ____rm -rf $(TMP)/$(GMP) && mkdir $(TMP)/$(GMP) &&\
39 ____cd $(TMP)/$(GMP) &&\
40 ____$(SRC)/$(GMP)/$(BCFG) $(CFG_GMP) &&\
41 ____$(MAKE) && make install-strip
42
```

```
43 .PHONY: mpfr
44 mpfr: $(SRC)/$(MPFR)/README
45 ____rm -rf $(TMP)/$(MPFR) && mkdir $(TMP)/$(MPFR) &&\
46 ____cd $(TMP)/$(MPFR) &&\
47 ____$(SRC)/$(MPFR)/$(BCFG) $(CFG_MPFR) &&\
48 ____$(MAKE) && make install-strip
49
50 .PHONY: mpc
51 mpc: $(SRC)/$(MPC)/README
52 ____rm -rf $(TMP)/$(MPC) && mkdir $(TMP)/$(MPC) &&\
53 ____cd $(TMP)/$(MPC) &&\
54 ____$(SRC)/$(MPC)/$(BCFG) $(CFG_MPC) &&\
55 ____$(MAKE) && make install-strip
56
57 .PHONY: gcc0
58 gcc0: $(SRC)/$(GCC)/README
59 ____rm -rf $(TMP)/$(GCC) && mkdir $(TMP)/$(GCC) &&\
60 ____cd $(TMP)/$(GCC) &&\
61 ____$(SRC)/$(GCC)/$(BCFG) $(CFG_GCC0)
62 ____cd $(TMP)/$(GCC) && $(MAKE) all-gcc
63 ____cd $(TMP)/$(GCC) && $(MAKE) install-gcc
64 ____cd $(TMP)/$(GCC) && $(MAKE) all-target-libgcc
65 ____cd $(TMP)/$(GCC) && $(MAKE) install-target-libgcc
66
67 .PHONY: gcc
68 gcc: $(SRC)/$(GCC)/README
```

```
    rm -rf $(TMP)/$(GCC) && mkdir $(TMP)/$(GCC) &&\
    cd $(TMP)/$(GCC) &&\
    $(SRC)/$(GCC)/$(BCFG) $(CFG_GCC)
    cd $(TMP)/$(GCC) && $(MAKE) all-gcc
    cd $(TMP)/$(GCC) && $(MAKE) install-gcc
    cd $(TMP)/$(GCC) && $(MAKE) all-target-libgcc
    cd $(TMP)/$(GCC) && $(MAKE) install-target-libgcc
```

binutils: ассемблер, линкер и утилиты

–target=$(TARGET)	триплет целевой платформы
$(CFG_ARCH)	параметры архитектуры
$(CFG_CPU)	параметры процессора
–program-prefix	префикс <prefix>-(asld|..)|
–with-sysroot	каталог в котором находятся хедеры и библиотеки
–with-native-system-header-dir=/include	каталог с хедерами
–enable-lto	[L]ink[T]ime [O]ptimization

arch/i386.mk

```
CFG_ARCH = --disable-multilib
```

–disable-multilib выключить смешанный 32/64-битный режим

arch/arm.mk

```
CFG_ARCH = --enable-interwork
```

–enable-interwork разрешить смешанный код ARM/THUMB

cpu/i486sx.mk

```
# i486sx cpu
# target arch is Intel x86 (32 bit)
ARCH = i386
# target triplet for embedded linux
TARGET = i486-linux-uclibc
# target CPU configure params
CFG_CPU = --disable-mmx --disable-3dnow --disable-sse
```

cclibs: библиотеки для сборки gcc

gmp mpfr mpc

gcc0: сборка минимального кросс-компилятора Си

При сборке gcc0/gcc отключайте ccache: кэш при разовых сборках не работает, но при монтировании как tmpfs потребляет слишком много RAM:

```
./mk.rc && make CCACHE= gcc0
```

gcc: пересборка полного кросс-компилятора Си/C_+^+

Пакет собирается после сборки core.

166.11.3 core: сборка основной системы

mk/core.mk

```
.PHONY: core
core: kernel ulibc busybox gcc
```

kernel: ядро Linux

mk/kernel.mk

```
CFG_KERNEL = ARCH=$(ARCH) INSTALL_HDR_PATH=$(ROOT)

.PHONY: kernel
kernel: $(SRC)/$(KERNEL)/README
    # 1
    cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) distclean
    cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) allnoconfig
    # 2
    cat kernel/all >> $(SRC)/$(KERNEL)/.config
```

```
10 ____cat kernel/arch/$(ARCH) >> $(SRC)/$(KERNEL)/.config
11 ____cat kernel/cpu/$(CPU) >> $(SRC)/$(KERNEL)/.config
12 ____cat kernel/hw/$(HW) >> $(SRC)/$(KERNEL)/.config
13 ____cat kernel/app/$(APP) >> $(SRC)/$(KERNEL)/.config
14 ____# 3
15 ____echo "CONFIG_CROSS_COMPILE=\"$(TARGET)-\"" >> $(SRC)/$(KERNEL)/.config
16 ____echo "CONFIG_LOCALVERSION=\"-$(HW)$(APP)\"" >> $(SRC)/$(KERNEL)/.config
17 ____echo "CONFIG_DEFAULT_HOSTNAME=\"$(HW)$(APP)\"" >> $(SRC)/$(KERNEL)/.config
18 ____# 4
19 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) menuconfig
20 ____# 5
21 ____cd $(SRC)/$(KERNEL) && $(MAKE) $(CFG_KERNEL)
22 ____# 6
23 ____make kernel-$(ARCH)-fix
24 ____cp $(SRC)/$(KERNEL)/arch/x86/boot/zImage $(BOOT)/$(HW)$(APP).kernel
25 ____# 7
26 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) headers_install
27
28 .PHONY: kernel-i386-fix
29 kernel-i386-fix:
30 ____cp $(SRC)/$(KERNEL)/arch/x86/boot/bzImage \
31 ____ ____$(SRC)/$(KERNEL)/arch/x86/boot/zImage
```

ARCH архитектура: **src/linux-x.x.x/arch/***

INSTALL_HDR_PATH путь установки хедеров ядра

1. подготовка к сборке с пустым конфигом **src/linux-x.x.x/.config**

2. накатываем на пустой **.config** файлы-модификаторы, содержащие переопределения переменных конфигурации:
   - **all** универсальные параметры ядра для всех платформ
   - **arch** параметры, специфичные для архитектуры
   - **cpu** параметры, адаптирующие ядро к конкретному процессору (эмулятор FPU, возможности управления тактовой частотой и потреблением, поддержка многоядерности и т.п.)
   - **hw** параметры чипсета и периферии материнской платы (модули ядра = драйвера)
   - **app** параметры, в основном отключаемые для конкретного приложения (отключение графики, режим реального времени, спец.параметры)
3. установка параметров кросс-компиляции и имени сборки
4. запуск интерактивного меню конфигурирования, выйти с сохранением
5. сборка ядра
6. копирование готового файла ядра в **${BOOT}**
7. генерация хедеров в **${ROOT}**

Все параметры ядра идут с префиксом **CONFIG_**:

- core.mk
  - **CROSS_COMPILE** префикс кросс-компилятора `<pfx>-(gcc|as|ld..)`
  - **LOCALVERSION** суффикс ядра `linux-x.x.x-suffix`
  - **DEFAULT_HOSTNAME** имя компьютера **<host>.<domain>**
- all для всех платформ
  - **NOHIGHMEM=y** сборка в режиме ≤ 1 Gb RAM
  - **KERNEL_GZIP=y** ядро сжимать алгоритмом gzip[5]
  - корневая файловая система в initrd/ramdisk

[5] важно для загрузчиков на не-i386 системах

    * **BLK_DEV_INITRD=y** rootfs в RAM(initrd)
    * **PROC_FS=y** файловая подсистема /**proc**
    * **SYSFS=y** файловая подсистема /**sys**
    * **DEVTMPFS=y** файловая подсистема /**dev** (интерфейсы драйверов устройств)
    * **DEVTMPFS_MOUNT=y** автоматически монтировать devfs
  - режим реального времени[6] ??
    * **PREEMPT=y** вытесняющая многозадачность в режиме ядра
    * **HZ_1000=y** системный таймер 1 KHz
  - исполняемые форматы файлов
    * **BINFMT_ELF=y** бинарные файлы в формате ELF
    * **BINFMT_SCRIPT=y** файлы скриптов с `#!/bin/sh` в заголовке
    * **COREDUMP=n** не писать корки при сбоях
  - поддержка консоли tty
    * **TTY=y** последовательные порты и командная консоль
    * **UNIX98_PTYS=n** псевдотерминалы UNIX98
    * **LEGACY_PTYS=n** псевдотерминалы UNIX/BSD
  - клавиатура
    * **INPUT_KEYBOARD=y** ввод с аппаратной клавиатуры
  - мышь
    * **INPUT_MOUSE=y**
    * **INPUT_MOUSEDEV=y**
    * **INPUT_MOUSEDEV_PSAUX=n** не создавать `/dev/psaux`

[6] повышенная или гарантированная отзывчивость системы на внешние события

  - USB HID: устройства ввода без драйверов (клавиатура, мышь, джойстик, самодельные кнопки) включение поддержки USB на i386 не требуется
    * **HID=y** [H]uman [I]nterface [D]evice
    * **HID_GENERIC=y** универсальный драйвер USB HID Class
  - Графический режим FrameBuffer
    * **FB=y**
    * **FRAMEBUFFER_CONSOLE=y** командная консоль
    * **LOGO=y** вывод пингвина при запуске
    * **LOGO_LINUX_MONO=y** выводить только черно-белый вариант
    * **LOGO_LINUX_VGA16=n**
    * **LOGO_LINUX_CLUT224=n**
  - Отладка и printk лог ядра
    * **DEBUG_KERNEL=y** включение отладочных функций ядра
    * **EARLY_PRINTK=y** вывод пусковой части ядра
    * **PRINTK=y** вывод главного ситемного лога ядра printk
    * **PRINTK_TIME=y** выводить метки времени (для отладки времени запуска)
    * **SCHED_DEBUG=n** не отлаживать планировщик
    * **DEBUG_PREEMPT=n** не отлаживать вытесняющий режим
- arch i386
  - **X86_GENERIC=y** универсальная оптимизация для всех i386-процессоров
  - UART/COM/RS232 последоваельные порты
    * **SERIAL_8250=y** UART 8250/16550
    * **SERIAL_8250_CONSOLE=y** командандная консоль на COM-портах
  - **VGA_CONSOLE=y** командандная консоль на VGA 80×25
  - FrameBuffer

        * **FB_VESA=y** универсальный видеодрайвер VESA 2.0+
    - клава
        * **KEYBOARD_ATKBD=y** клавиатура PC AT / PS2
    - мышь
        * **MOUSE_PS2=y** мышь PS/2
        * **MOUSE_SERIAL=y** (старая) мышь на RS232
    - **NVRAM=y** доступ к памяти CMOS
    - отладка
        * **MAGIC_SYSRQ=y** волшебная кнопка Alt+SysRq
        * **X86_VERBOSE_BOOTUP=y** доп.сообщения при пуске ядра
- cpu i386 **i486sx**
    - **M486=y**
    - **MATH_EMULATION=y** включить программную эмуляцию мат.сопроцессора (FPU) в ядре
- hw qemu386 **qemu386**
    - **MATH_EMULATION=n** выключить эмуляцию FPU: QEMU запускается на процессорах Pentium и старше, которые всегда имеют аппаратный модуль плавающей точки
    - **HZ_1000=n** в режиме эмуляции быстрый таймер не имеет смысла,
    - **HZ_100=y** переключаемся на 100 Hz
- app micro
    - **FB=n** выключаем поддерку FrameBuffer,..
    - **INPUT_MOUSE=n** мыши

ulibc: **библиотека uClibc**

mk/ulibc.mk

```
CFG_ULIBC = CROSS=$(TARGET)- ARCH=$(ARCH) PREFIX=$(ROOT)
#   UCLIBC_EXTRA_CFLAGS=""

.PHONY: ulibc
ulibc: $(SRC)/$(ULIBC)/README
    # 1
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) distclean
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) allnoconfig
    # 2
    cat ulibc/all >> $(SRC)/$(ULIBC)/.config
    cat ulibc/arch/$(ARCH) >> $(SRC)/$(ULIBC)/.config
    cat ulibc/cpu/$(CPU) >> $(SRC)/$(ULIBC)/.config
    cat ulibc/app/$(APP) >> $(SRC)/$(ULIBC)/.config
    # 3
    echo "CROSS_COMPILER_PREFIX=\"$(TARGET)-\"" >> $(SRC)/$(ULIBC)/.config
    echo "KERNEL_HEADERS=\"$(ROOT)/include\"" >> $(SRC)/$(ULIBC)/.config
    # 4
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) menuconfig
    # 5
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC)
    # 6
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) install
    # 7
    cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) install_utils
    # 8
```

```
26 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) hostutils
27 ____cp $(SRC)/$(ULIBC)/utils/ldd.host        $(TC)/bin/$(TARGET)-ldd
28 ____cp $(SRC)/$(ULIBC)/utils/ldconfig.host $(TC)/bin/$(TARGET)-ldconfig
29 ____cp $(SRC)/$(ULIBC)/utils/getconf.host   $(TC)/bin/$(TARGET)-getconf
30 ____# 9 (in root package)
31 ____# $(LDCONFIG) -v -r $(ROOT)
```

1. подготовка к сборке с пустым конфигом **src/uClibc-x.x.x/.config**
2. накатываем на пустой **.config** файлы-модификаторы, содержащие переопределения переменных конфигурации:
   - **all** универсальные параметры ulibc для всех платформ
   - **arch** параметры, специфичные для архитектуры
   - **cpu** параметры, адаптирующие ulibc к конкретному процессору (набор команд и оптимизация)
   - **app** параметры, в основном отключаемые для конкретного приложения (отключение shared библиотек и неиспользуемых групп функций)
3. установка параметров кросс-компиляции и пути к хедерам ядра
4. запуск интерактивного меню конфигурирования, выйти с сохранением
5. сборка ulibc
6. инсталляция в ROOT
7. инсталляция утилит для работы с shared библиотеками на целевой системе
8. инсталляция утилит для **BUILD**-системы
9. создание **/etc/ld.so.cache** (выполняется в пакете root)

- – **ARCH_HAS_MMU=y** всегда включено
  – **ARCH_USE_MMU=y** всегда включено
  – **UCLIBC_HAS_FPU=y** всегда включено: используется эмулятор в ядре

  - **UCLIBC_HAS_FLOATS=y** функции плавающей точки
  - **DO_C99_MATH=n** функции float math C99
  - **UCLIBC_CTOR_DTOR=y** конструктор/декструктор
  - **UCLIBC_HAS_LFS=y** большие файлы
- разделяемые shared библиотеки
  - **HAVE_SHARED=y**
  - **LDSO_CACHE_SUPPORT=y /etc/ld.co.conf**
  - **LDSO_PRELOAD_ENV_SUPPORT=n** переменная **LD_PRELOAD** содержит список библиотек, загружаемых первыми
  - **LDSO_LD_LIBRARY_PATH=n** отключить переменную **LD_LIBRARY_PATH** со списком каталогов поиска shared библиотек
- многопоточные threads программы
  - **LINUXTHREADS_NEW=y** новый вариант потоков
- параметры установки
  - **DOSTRIP=y**
  - **RUNTIME_PREFIX=**
  - **DEVEL_PREFIX=**
- опции необходимые busybox
  - **UCLIBC_HAS_CTYPE_TABLES=y**
  - **UCLIBC_HAS_NETWORK_SUPPORT=y**
  - **UCLIBC_HAS_FNMATCH=y**
  - **UCLIBC_HAS_GNU_GETOPT=y**
  - **UCLIBC_HAS_REGEX=y**
  - **UCLIBC_SUSV3_LEGACY=y**
  - •

ulibc/all

```
ARCH_HAS_MMU=y
ARCH_USE_MMU=y
UCLIBC_HAS_FPU=y
UCLIBC_HAS_FLOATS=y
#DO_C99_MATH=y

HAVE_SHARED=y
LDSO_CACHE_SUPPORT=y
LDSO_PRELOAD_ENV_SUPPORT=n
LDSO_LD_LIBRARY_PATH=n

LINUXTHREADS_NEW=y
UCLIBC_CTOR_DTOR=y

UCLIBC_LINUX_SPECIFIC=y
UCLIBC_LINUX_MODULE_26=n
UCLIBC_HAS_LFS=y

UCLIBC_HAS_LFS=y
UCLIBC_HAS_REALTIME=y
UCLIBC_HAS_STRING_GENERIC_OPT=y
UCLIBC_HAS_STRING_ARCH_OPT=y

DOSTRIP=y
RUNTIME_PREFIX=""
```

```
26 DEVEL_PREFIX=""
27
28 # BB needs
29 UCLIBC_HAS_CTYPE_TABLES=y
30 UCLIBC_HAS_NETWORK_SUPPORT=y
31 UCLIBC_HAS_FNMATCH=y
32 UCLIBC_HAS_GNU_GETOPT=y
33 UCLIBC_HAS_REGEX=y
34 UCLIBC_SUSV3_LEGACY=y
35
36 ## Python needs
37
38 #DO_C99_MATH=y
39 #UCLIBC_HAS_LONG_DOUBLE_MATH=n
40 #UCLIBC_HAS_WCHAR=y
41 #UCLIBC_SUSV4_LEGACY=y
42 #UCLIBC_HAS_PTY=y
43 #UNIX98PTY_ONLY=n
44 #UCLIBC_HAS_LIBUTIL=y
45
46 #DOMULTI=y
47 #UCLIBC_HAS_IPV4=y
48 #UCLIBC_HAS_COMPAT_RES_STATE=n
49 #UCLIBC_HAS_SYSLOG=y
50 ## ALSA needs
51 #UCLIBC_HAS_BSD_ERR=y
```

ulibc/arch/i386

```
TARGET_i386=y
```

ulibc/cpu/i486sx

```
CONFIG_486=y
```

gcc: пересборка полного gcc

После сборки ulibc необходимо еще раз пересобрать gcc с полными настройками.

busybox: набор утилит busybox

mk/busybox.mk

```
CFG_BUSYBOX = \
    CONFIG_PREFIX=$(ROOT) \
    CROSS_COMPILE=$(TARGET)- \
    SYSROOT=$(ROOT)

.PHONY: busybox
busybox: $(SRC)/$(BUSYBOX)/README
    # 1
    cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) distclean
    cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) allnoconfig
    # 2
```

```
12 ____cp app/$(APP).bb $(SRC)/$(BUSYBOX)/.config
13 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) menuconfig
14 ____cp $(SRC)/$(BUSYBOX)/.config app/$(APP).bb
15 ____# 3
16 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) install
```

1. подготовка к сборке с пустым конфигом
2. интерактивное меню конфигурирования с сохранением конфига в **app/${APP}.bb**
3. сборка и инсталляция в **ROOT**

Особенность busybox — скрипты конфигурации не умеют отрабатывать переписывание переменных конфигурации, поэтому приходиться держать в **app/*.bb** настройки для каждого приложения полностью. Если вы создаете свое приложение, скопируйте **app/micro.bb** в качестве базовой версии вашего **app/userapp.bb**.

166.11.4 libs: сборка библиотек ${LIBS}

166.11.5 apps: сборка прикладных пакетов ${APPS}

166.11.6 user: сборка пользовательского кода

166.11.7 root: формирование корневой файловой системы

mk/root.mk

```
1 ROOTREX = "./(boot|include|lib/.*\.[ao])"
2
```

```
.PHONY: root
root:
	# 1
	rm -rf $(ETC) ; cp -r etc $(ROOT)/
	cp README.md $(ETC)/
	chmod +x $(ETC)/init.d/*
	# 2
	$(LDCONFIG) -v -r $(ROOT)
	# 3
	ln -fs /sbin/init $(ROOT)/init
	# 4
	cp -r share $(ROOT)/
	# 5
	cd $(ROOT) && find . | egrep -v $(ROOTREX) | \
		cpio -o -H newc > $(BOOT)/$(HW)$(APP).cpio
	cat $(BOOT)/$(HW)$(APP).cpio | gzip -9 > $(BOOT)/$(HW)$(APP).rootfs
```

1. пересоздание **/etc/**
2. генерация кеша динамических библиотек **/etc/ld.so.cache**
3. **/sbin/init** → **/init**
4. **/share/**
5. создание initrd используя фильтрацию файлов регулярным выражением в переменной **ROOTREX**

/etc/inittab

```
::sysinit:/etc/init.d/rcS
::shutdown:/etc/init.d/rcD
::ctrlaltdel:/sbin/reboot
```

```
tty1::askfirst:/bin/sh
tty2::askfirst:/bin/sh
tty3::askfirst:/bin/sh
tty4::askfirst:/bin/sh
#ttyS1::respawn:/bin/sh
```

/etc/init.d/rcS

```
#!/bin/sh
# system dirs
mkdir /tmp
mkdir /proc ; mount -t proc  proc  /proc
mkdir /sys  ; mount -t sysfs sysfs /sys
# hotplug device manager (automount, firmware boot, config)
mdev -s && echo /sbin/mdev > /proc/sys/kernel/hotplug
# about
uname -a
cat /etc/README.md
```

1. создание системных каталогов
2. запуск демона mdev, обслуживающего **devfs**: создание/удаление устройств в **/dev**, автомонтирование, загрузка прошивок,..
3. вывод **README**

/etc/init.d/rcD

```
#!/bin/sh
sync
```

```
umount -a
```

166.11.8 boot: сборка загрузчика syslinux/grub/uboot

mk/boot.mk

syslinux

grub

uboot

166.12 emu: запуск собранной системы в эмуляторе

Для отладки кода, не связанного жестко с железом, для которого допускается выполнение в эмуляторе, используется QEMU.

mk/emu.mk

```
include app/$(APP).qemu

.PHONY: emu
emu: $(BOOT)/$(HW)$(APP).kernel $(BOOT)/$(HW)$(APP).rootfs
	qemu-system-$(ARCH) $(QEMU_CFG) \
	-kernel $(BOOT)/$(HW)$(APP).kernel \
```

```
7 ____-initrd $(BOOT)/$(HW)$(APP).rootfs \
8 ____-append "$(QEMU_VGA)"
```

hw/qemu386.mk

```
1 # QEMU emulator: i386 mode
2 CPU = i486sx
3 QEMU_CFG = -m 32M -net none
```

Переменная `QEMU_CFG` задает параметры запуска QEMU:

-m	объем RAM на эмулируемой системе
-net none	отключить сеть и iPXE boot
-kernel	прямая загрузка ядра Linux из файла
-initrd	образ корневой файловой системы (**initrd**)
-append	параметры ядра:
vga=ask	запрос списка видеорежимов при загрузке и выбор режима
vga=none	std. VGA text 80×25, FrameBuffer отключен
vga=0x315	VESA 800×600×24bit
vga=0x317	VESA 1024×768×16bit

```
Mode 0x0301: 640x480 (+640), 8 bits
Mode 0x0303: 800x600 (+800), 8 bits
Mode 0x0305: 1024x768 (+1024), 8 bits
Mode 0x0311: 640x480 (+1280), 16 bits
Mode 0x0312: 640x480 (+2560), 24 bits
Mode 0x0314: 800x600 (+1600), 16 bits
```

```
Mode 0x0315: 800x600 (+3200), 24 bits
Mode 0x0317: 1024x768 (+2048), 16 bits
Mode 0x0318: 1024x768 (+4096), 24 bits
```

166.13 netboot: Сетевая загрузка

166.14 Прошивка на устройство

166.15 RT-патч

Глава 167

Библиотека libSDL2

Библиотека SDL предоставляет такие средства, как быстрый вывод 2D-графики, обработку ввода, проигрывание звука, вывод 3D через OpenGL и другие операции, причем делает это кросплатформенно. Список платформ обширный: Linux, Windows, Windows CE, BeOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX и QNX — и вдобавок есть неофициальные порты на другие системы.

Сама библиотека написана на Си и поддерживает C_{+}^{+}, однако есть биндинги к большинству популярных языков. Автор libsdl был нанят компанией Valve, программные продукты которой активно используют библиотеку. К тому же, теперь библиотека выходит под лицензией zlib, а не LGPLv2, как было раньше, и SDL 2.0 можно использовать в любых своих приложениях, в т.ч. коммерческих Скорее всего сделано это было для того, чтобы Valve смогла включить ее в Steam для Linux.

Глава 168

OpenGL

Глава 169

RealTime: режим жесткого реального времени в Linux

Глава 170

netboot: Сетевая загрузка

Глава 171

cross

Глава 172

BuildRoot

Глава 173

Особенности OpenWrt

173.1 Реализация microGUI

Глава 174

Приложения для X Window

Глава 175

Программирование сетевых приложений

Глава 176

Сборка кросс-компиляторя GNU мальтийским крестом

Часть XVI

IDE

IDE — Integrated Development Environment, интегрированная среда разработки.
Программный пакет, включающий

- средства управления проектом,
- отслеживание зависимостей между файлами (в т.ч. с анализом исходного текста программ на конструкции типа `#include`, `module`, `uses`),
- автозапуском компиляторов для изменившихся файлов,
- GUI для отладчиков (gdb),
- специализированный редактор plain text[1] файлов с
  - цветовой и шрифтовой подстветкой синтаксиса,
  - автодополнением: дописываются имена объектов программ, синтаксические конструкции и параметры функций,
  - автоформатированием: фрагмент текста переформатируется в соответствии с синтаксисом языка редактируемого файла, проставляются отступы в зависимости от вложенности синтаксических конструкций типа циклов и условных блоков)
  - выделением строк, на которые указывают сообщения об ошибках компиляторов,
  - маркеры точек останова отладчика
- отображение структуры программ, например деревья классов и структур данных
- контекстные справочники по используемым языкам программирования, автоматический вывод списка параметров при вводе имени функции
- отображение дизассемблерных листингов для компилируемых языков
- отображение браузера как вкладки или MDI окна
- отображение вывода статических анализаторов программ с кликабельными ссылками

[1] файлы не включающие непечатаемых символов и бинарных данных, которые можно причитать простым выводом на экран командами типа **type**, **cat**, **more**

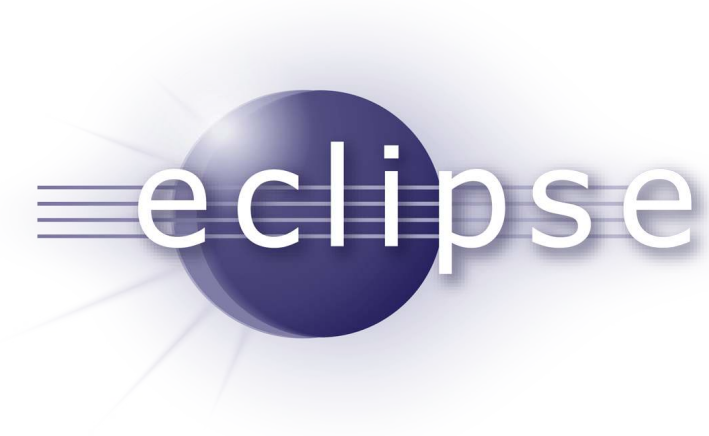
- вывод компиляторов и трансляторов с цветовым выделением и переход на ошибочную строку в редакторе при щелчке на ошибке
- ...

В этой книге рассмотрены три бесплатных мультиплатформенных OpenSource IDE, в порядке навороченности, универсальности, и требуемым ресурсам для работы самой среды:

1. ECLIPSE 177: самая навороченная и ресурсоемкая IDE, написана на Java, имеет десятки дополнительных модулей на все случаи, умеет работать со всеми распространенными языками программирования, жрет память, и требует современного компьютера минимум с 2+ Гб ОЗУ. Последний релиз ECLIPSE Luna работает заметно быстрее (особенно при запуске).
2. Code::Blocks 178: легкая среда для разработки на C/C_{+}^{+}, для других языков модет потребоваться написать свои модули или файлы описания синтаксиса
3. (g)Vim 179: самый легкий и портабельный универсальный текстовый редактор с расширенными функциями, работает на всех существующих платформах (кроме совсем уж embedded), использует минимум ресурсов, но требует некоторого обучения даже чтобы выйти из `vim` ☺

Глава 177

⊜eclipse



177.1 Установка eclipse под Windows

⊞ + R » https://eclipse.org/ » Download » Eclipse Luna release for » ⊞Windows
Качаем архив базовой системы: Eclipse IDE for Java Developers » ⊞Windows 32/64 Bit
Или сразу сборку CDT ECLIPSE: Eclipse IDE for C/C++ Developers » ⊞Windows 32/64 Bit

177.2 Установка eclipse под Linux

⊞ + R » https://eclipse.org/ » Download » Eclipse Luna release for » Linux
Качаем архив базовой системы: Eclipse IDE for Java Developers » Linux 32/64 Bit
Или сразу сборку CDT ECLIPSE: Eclipse IDE for C/C++ Developers » Linux 32/64 Bit

Пока качается, параллельно устанавливаем в систему Java-рантайм:

```
sudo aptitude install openjdk-7-jre
```

Распаковывем полученный архив **eclipse-java-luna-SR1-linux-gtk-x86_64.tar.gz** в **$HOME**:

```
cd ~
tar zx < Downloads/eclipse-java-luna-SR1-linux-gtk-x86_64.tar.gz
ls -la eclipse/eclipse
-rwxr-xr-x 1 user user 74675 Авг 13 16:12 eclipse/eclipse
```

Прописываем запуск ECLIPSE в ваш оконный менеджер или **.blackboxmenu** с параметром **-noSplash** для лечения глюка с запуском на x64-битных системах:

.blackbox.menu

177.3 Установка CDT

CDT — расширение ECLIPSE для разработки на Си/C_+^+, редактирования make-файлов. Это расширение критически важно для вашей работы, поэтому ставить его обязательно, или сразу качать сборку CDT ECLIPSE.

ECLIPSE > Help > Install New Software...
Work with > Add > Add repository
Name > CDT
Location > http://download.eclipse.org/tools/cdt/releases/8.5
OK
Work with > CDT
CDT Main Features > ☒ C/C++ Development Tools
CDT Optional Features
Парсер файлов исходников на диалекте C99: ☒ C99 LR Parser
Поддержка gcc в режиме кросс-компиляции: ☒ GCC Cross Compiler Support
Аппаратная отладка через gdb: ☒ GDB Hardware Debugging
Next > Next > Accept > Finish

177.4 Установка PyDev

PyDev — расширение для разработки на Python:

ECLIPSE > Help > Install New Software...
Work with > Add > Add repository

Name > PyDev
Location > http://pydev.org/updates
OK
Work with > PyDev
PyDev > ☒ PyDev for Eclipse
Next > Next > Accept > Finish > Certitificate > Restart Eclipse > Ok

177.5 Установка TeXlipse

Если планируете работать с документацией в формате LaTeX, установите расширение TeXlipse:

ECLIPSE > Help > Install New Software...
Work with > Add > Add repository
Name > TeXlipse
Location > http://texlipse.sourceforge.net/
OK
Work with > TeXlipse

Это расширение поддерживает подсветку синтаксиса, автодополнение, построение динамического оглавления, автокомпиляцию по сохранению, и несколько визардов создания проекта.

177.6 Редактирование файлов в формате XML и производных

Установите пакет ECLIPSE WST:

Help > Install New Software
Work with: > Luna - http://download.eclipse.org/releases/luna
Filter: > WST > Eclipse WST > Next > Next > Restart > OK

177.7 Проверка орфографии

[1]

То, что проверка орфографии очень удобная вещь вряд ли нужно объяснять. Есть конечно люди, которые не обращают на неё внимание, но это чаще всего из-за экономии времени и отсутствия удобных средств проверки.

Действительно, удобная автоматическая проверка орфографии есть в офисных пакетах, но мне сложно представить разработчика, который будет переносить комментарии в Word и обратно ☺.

Поэтому очень удобно иметь проверку правописания прямо в IDE. И ECLIPSE в этом смысле полностью соответствует ожиданиям.

Долго объяснять что к чему нет смысла. Проверка орфографии встроена в ECLIPSE и если вы пишите только на английском, то может быть не захотите ничего менять.

Кроме того, есть статья Aaron'a (en) в которой автор рассказывает о подключении дополнительных словарей и плагине **eSpell**.

Но русских словарей в дистрибутиве нет, а при подключении внешних есть нюансы. Поэтому мы максимально подробно рассмотрим подготовку и добавление русских словарей.

Первый вопрос. В каком виде должны быть словари и где их взять?

Тут всё просто. Формат словаря — обычный текстовый файл, в котором каждое слово начинается с новой строки. И нам вполне подойдут свободно распространяемые словари **aSpell**.

[1] копипаста http://www.simplecoding.org/proverka-orfografii-v-eclipse.html

Установка состоит из 4 шагов:

1. качаем aSpell и словари для нужных языков
   ⊞ + R » http://aspell.net/win32/
   Binaries » Full installer
   Precompiled dictionaries » English
   Precompiled dictionaries » Russian
2. устанавливаем сначала **aSpell**, потом отдельно каждый словарь
   Aspell-0-50-3-3-Setup.exe » Setup GNU Aspell » Next » License » Next
   Directory » **C:/GnuWin32/Aspell** » Next » Next
   Additional » Next » Install » Next » ☐ View manual » Finish
   Aspell-en-0.50-2-3.exe » Aspell English Dictionary » Next » License » Next
   Directory » **C:/GnuWin32/Aspell** » Next » Next » Install » Finish
   Aspell-ru-0.50-2-3.exe » Aspell Russian Dictionary » Next » License » Next
   Directory » **C:/GnuWin32/Aspell** » Next » Next » Install » Finish
3. делаем дамп словарей, перекодируем из koi8r в utf8 и объединяем
   ⊞ + R cmd

```
cd \GnuWin32\Aspell
bin\aspell dump master en > en.dict
bin\aspell dump master ru > ru.koi8
iconv -f koi8-r -t utf-8 < ru.koi8 > ru.dict
copy en.dict + ru.dict enru.dict
```

4. настраиваем spell-checker ECLIPSE
   ECLIPSE » Window » Preferences » Editors » Text editors » Spelling
   User defined dictionary » **C:/GnuWin32/Aspell/enru.dict**

Encoding > UTF-8
Apply > OK

Глава 178

Code::Blocks

Глава 179

(g)Vim

179.1 Установка под ⊞Windows

⊞ + R cmd » http://www.vim.org/ » Download » PC: MS-DOS and MS-Windows » **gvim74.exe**
Vim 7.4 Setup » This will install » Да
License » I'm Angry
Installation Options » ☒ Create .bat files » Next
Installation Folder » Install
Completed » Close
Do you want to see README » **Да**

Теперь можно настроить темную тему и выключение подстветки синтаксиса, по умолчанию после установки используется светлая тема и подстветка выключена:
меню » Правка » Настройка запуска

Переходим в конец файла и включаем режим вставки
Ctrl + Down Ins Enter Enter

```
syntax on
colorscheme pablo
```

Выходим в режим команд и принудительно сохраняем
Esc : w ! Enter Enter

Выходим из (g)Vim
Esc : q ! Enter

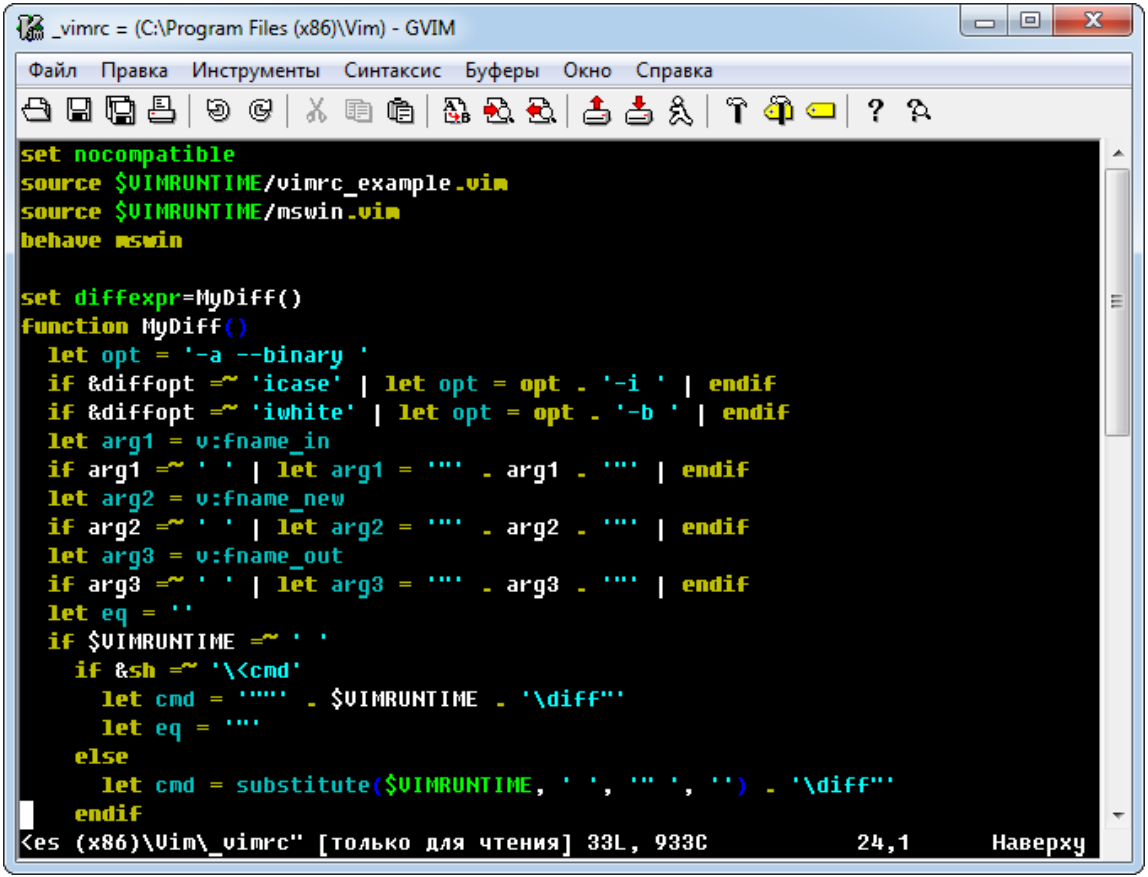
Если не получилось (под Windows 7):

⊞ + R cmd **/Program Files (x86)/Vim/**

Копируем файл **_vimrc** в любой каталог, например в **/tmp/**, затем ▷▷ Edit with Vim, и повторяем редактирование еще раз.

Затем копируем **_vimrc** обратно в **/Program Files (x86)/Vim/** с заменой.

Если теперь открыть на редактирование тот же файл, или любой другой текстовый, получим более удобный вид: для файлов известных типов будет работать подсветка синтаксиса.



179.2 Выход из (g)Vim

Esc : ! q Enter

179.2.1 Выход с автосохранением

Esc Shift + Z Shift + Z

179.3 Переход в режим редактирования

(g)Vim запускается в командном режиме, для перехода в режим редактирования используются следующие клавиатурные команды:

- Ins или i: включение режима вставки по текущему положению курсора
- Ins Ins или r: включение режима перезаписи поверх текста после курсора
- Shift + A: включение режима вставки в конец текущей строки

179.4 Переход в режим команд

Esc

179.5 Запись редактируемого файла

Esc : w Enter

Если выводится предупреждение типа “файл защищен от записи” или подобное, может сработать принудительная запись:

Esc : ! w Enter

179.6 Перезагрузка файла

Для перезагрузки возможно измененнного извне файла или отмены всех несохраненных изменений

Esc : e Enter

179.7 Отмена последних изменений (undo)

Esc u u . . .

Часть XVII

Замечания для авторов

179.8 Набор репозиториев на GitHub

https://github.com/ponyatov/Azbuka	основная репа
https://github.com/ponyatov/bib	библиографические базы данных
https://github.com/ponyatov/scratcher	журнал, используются некоторые материлы

Для работы с проектом сделайте собственный форк основной репы, библиографическую базу и журнал можете клонироввать напрямую. Создайте каталог и склонируйте репы:

```
D:
cd \
mkdir w
cd \w\
git clone --depth=1 -o gh git@github.com:username/Azbuka.git

git clone --depth=1 -o gh git@github.com:ponyatov/bib.git
git clone --depth=1 -o gh git@github.com:ponyatov/scratcher.git
```

179.9 Верстка в LaTeX

Было много вопросов по выбору языка разметки, и даже предложения некоторые материалы просто навордятить. Но все же используется LaTeX, т.к. это самая наворченная система подготовки больших изданий, широко распространенная (в узких кругах), и прежде всего как имеющая богатейший набор пакетов-расширений для всевозможных вывертов.

LaTeX не предназначен для верстки полноцвета, журнальной верстки или ручного таскания блоков по листу. LaTeX изначально был заточен на подготовку научно-технической и учебной многостраничной литературы **с логической разметкой**.

Как профессиональный инструмент, LaTeX требует обучения. Начерно навордятить на нем текст, накидав как попало картинок, и наляпав шрифтов[1] не получиться. И это хорошо.

Но — материалы на добавление принимаются в любых форматах, группой авторов, способных их доварить до нужного качества. Единственное ограничение: наличие бесплатных средств просмотра на трех основных платформах: ⊞Windows, Linux и MacOS, или в онлайне (Google Docs, M$ облака, и прочие сетевые болота).

Также приветствуетсся использование различных более простых языков разметки (SPHINX, Wiki, DocBook, .md, ...). Для первоначального сбора и группировки материала они проще для освоения, чаще всего рендер-движки для этих языков заточены под веб-редактиврование в т.ч. групповое, и хорошо подходят для простой по оформлению документации на программные пакеты, или как сборник ссылок на другие ресурсы.

Для включения таких материалов в основную верстку несложно написать TeX-транслятор (если нет сразу готового), который создаст .tex файлы нужного вида с минимумом ручной доработки.

[1] про существование стилей многие вордятники даже не слышали

Часть XVIII

Подготовка публикаций в LaTeX

Подробное описание этого слона см. [**Ivovsky**].

Внимание: LaTeX не предназначен для верстки полноцветных и журнальных изданий, его область — научно-техническая, учебная и художественная литература, и пакетная верстка материалов, сформированных автоматически (отчеты работы программ, графики и анализ экспериментальных данных, и т.п.)

[2]

LaTeX (по-русски произносится **латéх**) — наиболее популярный набор макрорасширений (или макропакет) системы компьютерной вёрстки TeX, который облегчает набор сложных документов. В типографском наборе форматируется как LaTeX.

Главная идея LaTeX состоит в том, что авторы должны думать о содержании, о том, что они пишут, не беспокоясь о конечном визуальном облике (печатный вариант, текст на экране монитора или что-то другое). Готовя свой документ, автор указывает логическую структуру текста (разбивая его на главы, разделы, таблицы, изображения), а LaTeX решает вопросы его отображения. Так содержание отделяется от оформления. Оформление при этом или определяется заранее (стандартное), или разрабатывается для конкретного документа.

В практическом смысле использование LaTeX позволяет (в порядке уменьшения важности):

- с помощью макросов и TeX-программирования реализовывать любые стили и сложную верстку, существует множество готовых пакетов для верстки графических химических формул, разнообразных схем, транскрипционных знаков, внезапно электронных схем, цветных листингов и т.п.
- автоматизировать работу с документами: пересобирать выходные файлы через Make, генерировать части документов с помощью своих скриптов[3]

[2] копипаста https://ru.wikipedia.org/wiki/LaTeX

[3] отчеты, стандартные формы, результаты работы любых программ

- получить выходой документ в .pdf .html .txt .PostScript .djvu ... с кликабельными ссылками, анимированными, а иногда и интерактивными элементами
- не использовать файлы документов в закрытом формате
- легко держать набор файлов в VCS
- не покупать текстовый процессор

Особенно важен пункт про сложную верстку: она всегда нужна в крупных технических публикациях, особенно в учебной литературе, или отчетных работах. Вам обязательно понадобиться вставлять графики экспериментальных данных, тематически специфичные схемы, листинги, выходные данные работы ваших программ и т.п.

Традиционно LaTeX любим математиками, и всеми кто готовит публикации с большим количеством формул и перекрестных ссылок: после небольшого обучения формулы вводятся с листа со скоростью набора текста, особенно если ваш редактор умеет автодополнение, и никакой мышиной возьни.

Естественно всякие чисто автоматические вещи типа автонумерации ссылок и формул, сборки оглавлений и индексов, цветовая подсветка синтаксиса в листингах программ, размещение плавающих иллюстраций и т.п. выполняются автоматически TeX-процессором в пакетном режиме, и на выходе получается красивый печатный или электронный (.pdf) документ.

Единственная область, не удобная в LaTeX-верстке — создание сложных таблиц. Для этого были созданы визуальные редакторы, позволяющие отрисовать структуру таблицы мышью, а затем заполнить готовый шаблон данными.

179.10 Установка MikTeX под ⊞Windows

⊞ + R » http://miktex.org/ » Download » Recommended » Basic MiKTeX Installer

179.11 Структура документа

179.11.1 Заголовочный файл или блок

179.11.2 Стили документа

179.11.3 Пакеты

179.11.4 Автор и название

179.11.5 Верстка титульных страниц

179.11.6 Оглавление

179.12 Верстка слайдов

179.13 Список литературы и цитирование

LaTeX умеет мощную подсистему управления цитированием и списками литературы. В простейшем случае, например при написании единственной статьи, раздел библиографии можно создать в том же документе, добавив в конец `thebibliography`:

```
\documentclass{article}

\input{header}
```

```
\author{Вася Пупкин}
\title{Пример статьи с цитатами}

\begin{document}
\maketitle

В статье используются книги: \cite{A} и \cite{B}

\begin{thebibliography}{99}

\bibitem{A} Книга A

\bibitem{B} Книга B

\end{thebibliography}
\end{document}
```

Но если вы регулярно работаете с документацией, или часто пишете статьи, возникает естественное желание вынести весь список литературы в отдельную базу данных, прописать авторов, названия, издательства и т.п. Это делается с помощью программы **biber** и пакета **biblatex**.

Пример использования этой системы вы легко найдете в исходниках этой книги:

- файл **header.tex** содержит секцию подключения пакета и подгрузки библиофайлов:

  ```
  % books bib management
  \usepackage{biblatex}
  \addbibresource{../bib/python.bib}
  ```

```
\addbibresource{../bib/eskd.bib}
...
```

- библиофайлы хранятся в **соседнем** репозитории **../bib**, склонированном с https://github.com/ponyatov/bib.
- порядок вызова **pdflatex** и **biber** см. **Makefile**

Для оформления библиографии в нужном стиле см. примеры [**bibiso**].

179.14 Команды секционирования: часть, глава, раздел,..

179.15 Таблицы

179.16 Формулы

179.17 Перекрестные ссылки и гипессылки

179.18 Листинги скриптов и текстовых данных

179.19 Подготовка иллюстраций

Подготовка иллюстраций — одна из самых геморных тех в создании документации, и ее верстке для бумажных и электронных изданий.

Предпочтение нужно отдавать векторным форматам, за исключеним фотоиллюстраций. В идеале скриншоты также хорошо бы переводить в векторыный формат, но пока инструмент для этого не найден, поэтому выходные файлы будут пухнуть в объеме.

Для подготовки векторных иллюстраций: схем, графиков, диаграмм и т.п. используйте пакеты, принимающие на вход программы на специализированном языке программрования, легко читаемым человеком. В этом случае у вас сохраниться отслеживать изменения, читая логи VCS.

Обратите внимание на возможность использования стилевых файлов на весь проект (для всех иллюстраций в книге например). Их использование даст профессиональный вид продукту, при этом сохраниться возможность взять и переформатировать 100500 схем в 10-томнике, поменяв шрифт, цвета, толщины линий, зазоры между элементами и т.п.

Пользуйтесь только относительными единицами размеров, и привязывайтесь к размерам шрифтов, это даст возможность использовать готовую иллюстрацию в нескольких проектах с разными размерами бумаги и наборами используемых шрифтов.

179.19.1 Графики GNUPLOT

Самый постой способ получит график простой аналитической функции или экспериментальных данных — воспользоваться утилитой GNUPLOT.

Оценить возможности можно вот по этому видео

Примеры на википедии

Примеры выложенные вместе с текстом на языке gnuplota

179.19.2 Схемы и графы в GraphViz

Для отрисовки графов и схем, легко к ним сводящихся, можно использовать пакет **GraphViz** и язык **Dot**.

179.19.3 PGF/TikZ

Сложные графики можно рисовать с помощью пакета **PGF/TikZ**, но для его работы нужна установленная LaTeX-система. Этот пакет предназначен прежде всего для набора и верстки изданий с множеством сложных схем.

179.19.4 GLE

GLE — универсальный язык описания векторных графических объектов с элементами языка программирования. Поддерживает вычисления, типовые конструкции программирования (циклы, условия, рекурсию).

- графики
- 3D графики
- диаграммы
- фракталы
- электронные схемы
- исчо

179.19.5 Xy-pic

179.20 Верстка электронных изданий

Для электронных изданий, предназначенных для чтения с различных экранов как компьютера, так и портативных устройств, существует ряд ограничений и рекомендаций, из-за особенностей экранов: малый размер, низкое разрешение, поддержка цвета (TFT vs e-Ink) и т.п.: [**ebooktex**]

Установка полей в .PDF:

```
\hypersetup{
pdftitle={Азбука халтурщика-ARMатурщика},
pdfauthor={ruOpenWrt, HackSpace Чебураторный завод, Bill Collis (part 1)}
}
```

Часть XIX

Символьная и численная математика

В практике любого инженера математика занимают главную роль. Без хорошего знания математики, причем практически всех областей, от школьной до дифференциального исчисления, работать в этой области практически невозможно.

Прежде всего свободное знание математики, физики, и химии необходимо для чтения любой литературы, если вам нужно разобраться в какой-либо прикладной области. Очень часто приходится реализовывать некоторые численные методы вычислений, выполняющиеся в вашем устройстве в реальном времени, для управления процессами, обработки сигналов с датчиков, принятия решений о включении исполнительных устройств и т.п. Ну и конечно вы не сможете создать само устройство, не понимая принципы его работы ☺. Это конечно не относится к различным простейшим устройствам типа таймеров или простой автоматики, но стоимость заказов такого типа $\to 0$.

Если вы хотите поднять или восстановить свой уровень знания базовых наук (а заодно и английского), удобно воспользоваться ресурсом `https://www.khanacademy.org/`: это знаменитая on-line академия **Khan Academy**, имеющая как набор видеолекций по базовым техническим наукам, так и большую батарею тестов для проверки ваших знаний. Не забывайте периодически проходить все тесты, чтобы поддерживать свои знания рабочими. Из недостатков — отвратнейшая реализация на мобильных устройствах, часть тестов просто не работает, а ввод ответов крайне неудобен из-за необходимости постоянно пользоваться (полно)экранной клавиатурой и переключения на числовой ввод.

На русском языке ресурсов такого класса к сожалению пока не попадалось. Кое-что есть кусочками, но по большей части только лекции в стиле «книжкой по башке», похоже навыков вводного обучения в России просто не существует. Если есть силы и желание, можете сами реализовать проект по созданию онлайн системы базового образования ☺.

В этом разделе собраны примеры проектов, требующие некоторых базовых знаний, а также рассмотрено использование OpenSource программ для вычислений и обработки данных.

Глава 180

Общие сведения о компьютерной математике

[1][2]

Для начала пару слов о том, что из себя представляют эти самые символьные или, как их еще называют, аналитические вычисления, в отличие от численных расчетов. Компьютеры, как известно, оперируют с числами, целыми и с плавающей запятой[3]. К примеру, решения уравнения $x^2 = 2x + 1$ можно получить как -0.41421356 и 2.41421356, а $3x = 1$ — как 0.33333333. А ведь хотелось бы увидеть

[1] копипаста http://maxima.sourceforge.net/ru/maxima-tarnavsky-1.html

[2] Тихон Тарнавский. Maxima — максимум свободы символьных вычислений

[3] на самом деле настоящую "плавучку" поддерживают только достаточно мощные процессоры, не хуже i486dx, встраиваемые не-DSP CPU/MCU аппаратно работают только с целыми числами: ± 127, $\pm 2^{16-1}$ и $\pm 2^{32-1}$ в зависимости от разрядности ядра 8- 16- или 32-бит

не приближенную цифровую запись, а точную величину, т. е. $1 \pm \sqrt{2}$ в первом случае и $1/3$ во втором. С этого простейшего примера и начинается разница между численными и символьными вычислениями.

Но кроме этого, есть еще задачи, которые вообще невозможно решить численно или наоборот аналитически.

Например, параметрические уравнения, где в виде решения нужно выразить неизвестное через параметр; или нахождение производной от функции; да практически любую достаточно общую задачу можно решить только в символьном виде.

Наоборот, для многих задач не существует точного аналитического решения, и приходится применять численные методы их решения.

В некоторых случаях нужно получение простого и быстрого приближенного решения — это может понадобится в системах управления, когда микроконтроллер не успевает за управляемым процессом, если пытается получить точное численное решение. При обработке сигналов например не требуется точное решение, достаточно результата, получаемого численными методами.

Для решения аналитических задач давно появились компьютерные программы, оперирующие любыми математическими объектами, от чисел любого типа, векторов и матриц до тензоров, от функций до интегро-дифференциальных уравнений и т. д. — они имеют общее название [C]omputer [A]lgebra [S]ystem.

Среди математического ПО для аналитических (символьных) вычислений наиболее широко известны коммерческие CAS-пакеты Maple, Mathematica и MathCAD. Для символьных вычислений предназначен пакет MatLab. Это очень мощные и очень дорогие инструменты для ученых и инженеров, позволяющие автоматизировать наиболее рутинную и требующую повышенного внимания часть работы, оперируя при этом аналитической записью данных, т. е. почти математическими формулами.

Такие программы можно назвать средой программирования, с той разницей, что в качестве элементов языка программирования выступают привычные человеку математические обозначения.

Для преподавателей, аспирантов, и студентов предоставляются академические более дешевые лицензии, но для хоббитов и коммерческого применения требуется покупка полной лицензии, имеющей

зачастую космическую стоимость. Неплохим вариантом может послужить использование бесплатного и свободного OpenSource программного обеспечения, описанного далее — пакетов **Maxima** и **Octave**.

С другой стороны, основное направление, кроме научных разработок, где такие программы востребованы — высшее образование; а использование для учебных нужд именно свободного ПО — реальная возможность и для ВУЗа, и для студентов и преподавателей иметь в своем распоряжении легальные копии такого ПО без больших, и даже каких-либо денежных затрат.

Глава 181

Пакет Maxima

Maxima – пакет CAS: символьной математики, но также включает функционал численных вычислений и визуализации.

181.1 Установка Maxima под ⊞Windows

⊞ + R › http://maxima.sourceforge.net/ru/ › Загрузка
Maxima-Windows › **5.34.1-Windows** › **maxima-5.34.1.exe**
maxima-5.34.1.exe › Установка › Язык › **English** › OK › Next
License › I accept › Next
Install folder › по умолчанию › Next
Components › ☐ Language packs › Next › Next

☒ wxMaxima ☐ XMaxima Next Install Next Finish
⊞ Maxima-5.34.1 wxMaxima ▷ Закрепить в панели задач

Дополнительная документация: http://maxima.sourceforge.net/ru/documentation.html

PDF для книги Ильина В.А., Силаев П.К. Система аналитических вычислений Maxima для физиков-теоретиков [**maxphis**] получена из файла **.ps** с помощью сервиса http://ps2pdf.com/convert.htm.

181.2 Калькулятор

Часть XX

Куча

В этот раздел собраны все материалы, не вошедшие в основную часть потому что слишком сложны для начинающих, не попадают не в один раздел по тематике, или не вписались по каким-то другим параметрам.

Все новые материалы также сначала попадают сюда, а потом принимается решение об их переносе в основную часть.

Часто сюда пишут статьи те, кто принимает участие в создании книги эпизодически, или те, у кого нет достаточно времени заниматься их подготовкой.

Предметный указатель

azLinux
- настройка
  - приложение, 299
- пакет
  - dirs, 296
- переменная
  - APP, 299
  - BOOT, 297
  - BUILD, 297
  - DIRS, 297
  - GZ, 296
  - ROOT, 297
  - SRC, 296
  - TC, 297
  - TMP, 297
- приложение
  - clock, 300
  - micro, 300
- скрипт
  - mk.rc, 292
- железо, 300
  - AR7240, 307
  - ASUS Eee PC 701, 302
  - BlackSwift, 306
  - Celeron1037U, 307
  - CeleronM, 307
  - Gigabyte GA-C1037UN-EU, 304
  - i386, 301
  - i486sx, 306
  - QEMU, 302, 305
  - RT5350, 307
  - VoCore, 305