

## 目 录

### 硬 件 篇

- 第 1 章 处理器的基本概念 2
  - 1.1 区分微处理器与微控制器 2
  - 1.2 寄存器 2
  - 1.3 处理器是如何启动的 4
  - 1.4 输入与输出 4
  - 1.5 指令与数据 5
  - 1.6 中断 6
  - 1.7 字节序 8
  - 1.8 边界对齐 10
  - 1.9 程序断点和数据断点 15
  - 1.10 内存管理单元 16
  - 1.11 缓存 17
  - 1.12 小结 18
- 第 2 章 开发活动中的硬件问题 19
  - 2.1 两个案例 19
  - 2.2 案例的背后——信号完整性 19
  - 2.3 应对方法 21
  - 2.4 小结 21

### 工 具 篇

- 第 3 章 make, 开发环境全能管家 24
  - 3.1 从最简单的 Makefile 中了解规则 24
  - 3.2 创建基本的编译环境 29
    - 3.2.1 将规则运用于程序编译 30
    - 3.2.2 让 Makefile 更专业 34
  - 3.3 提高编译环境的实用性 48
    - 3.3.1 让编译环境更加有序 48
    - 3.3.2 提升依赖关系管理 51
  - 3.4 打造更专业的编译环境 67
    - 3.4.1 规划项目目录结构 68
    - 3.4.2 增进复用性 72
    - 3.4.3 支持头文件目录的指定 75
    - 3.4.4 实现库链接 77
    - 3.4.5 增强可使用性 82
    - 3.4.6 管理对库的依赖关系 84
    - 3.4.7 改善编译效率 87
    - 3.4.8 恰当地书写注释 89
  - 3.5 理解 make 的解析行为 90
  - 3.6 Makefile 的调试 91
  - 3.7 make 的常用选项 92
  - 3.8 活用 make 92
  - 3.9 小结 94
- 第 4 章 gcc, C 语言编译器 96

4.1	什么是交叉编译器	96
4.2	gcc 幕后工作揭示	97
4.3	实用的 gcc 选项	99
4.3.1	解决宏错误的好帮手	99
4.3.2	辅助编写汇编程序的好方法	100
4.3.3	获取系统头文件路径	101
4.3.4	产生映射文件	102
4.3.5	通过选项定义宏	102
4.3.6	生成依赖关系	103
4.3.7	指定链接库	104
第 5 章	binutils 工具集, 软件开发利器	107
5.1	addr2line, 指令地址翻译器	108
5.2	ar, 静态库生成器	111
5.3	nm, 符号显示器	113
5.4	objdump, 信息查看器	115
5.5	objcopy, 段剪辑器	119
5.6	ranlib, 库索引生成器	120
5.7	size, 段大小观察器	121
5.8	strings, 字符串窥视器	122
5.9	strip, 程序文件瘦身器	124
第 6 章	ld, 链接器	125
6.1	重定位的概念	125
6.2	链接脚本	126
6.2.1	段	128
6.2.2	符号	129
6.2.3	存储区域	130
6.2.4	常用命令	131
6.3	常用选项	137
6.3.1	指定程序的入口点	137
6.3.2	生成可重定位的中间文件	137
6.3.3	指定链接脚本	138
	练习与思考	138
第 7 章	gdb, 程序调试助手	139
7.1	启动和退出 gdb	139
7.2	获取帮助	140
7.3	调试程序	142
7.3.1	断点设置	142
7.3.2	控制程序运行	144
7.3.3	检查程序	147
7.3.4	提高调试效率	151
7.4	查看符号表	152
7.5	控制 gdb 的行为	153
	编程语言篇	
第 8 章	掌握必要的汇编知识	156

8.1 as 的语法	156
8.1.1 宏	157
8.1.2 汇编命令	157
8.1.3 符号和标签	157
8.1.4 汇编指令	158
8.2 嵌入汇编的语法	158
第9章 深入理解程序的结构	161
9.1 段	161
9.1.1 指令段	161
9.1.2 数据段	162
9.2 栈	166
9.3 堆	168
9.4 小结	169
第10章 ABI/EABI 规范, 缔造程序兼容合约	170
10.1 定义基本数据类型	171
10.2 规范字节对齐处理	171
10.3 分配寄存器的功能	173
10.4 规定栈帧结构	174
10.4.1 栈帧的含义和作用	175
10.4.2 函数参数的传递方法	182
10.4.3 函数返回值的返回方法	184
10.5 小结	187
练习与思考	187
第11章 混淆指针与数组所导致的问题	188
11.1 问题示例	188
11.2 问题分析	189
11.2.1 数组的内存模型	189
11.2.2 指针的内存模型	190
11.3 问题成因	191
11.4 预防措施	193
11.5 小结	194
第12章 volatile, 让我保持原样	195
设计篇	
第13章 设计, 软件质量之本	200
13.1 软件设计是什么	200
13.2 软件质量的概念	201
13.3 阻碍改善设计的常见观念	203
13.3.1 测试是替罪羊或救命稻草	203
13.3.2 资源永远不足	204
13.3.3 不改变就可以规避风险	204
13.4 如何提高设计能力	205
13.5 设计模式、设计原则和设计思想	206
13.6 放之四海皆适用的设计原则	207
13.6.1 以人为本	207

13.6.2	追求简单性	210
13.6.3	让模块善始善终	211
13.6.4	重视收集统计信息	212
13.6.5	借助命名传达设计意图	213
13.6.6	消除“审美告警”	215
13.6.7	通过机制解决问题	215
13.6.8	防止他人犯错	218
13.6.9	考虑可查错性	220
13.7	小结	221
第 14 章	模块管理，保障系统有序运行	222
14.1	管理参照系	222
14.2	设计思路	224
14.3	程序实现	226
14.3.1	引入模块标识	226
14.3.2	实现层与级的表达	226
14.3.3	系统状态和回调函数原型定义	228
14.3.4	模块注册	228
14.3.5	系统启动	230
14.3.6	系统关闭	232
14.4	module 示例程序	233
14.5	模块管理的一些思考	235
14.6	小结	235
练习与思考		235
第 15 章	错误管理，不可或缺的用户需求	236
15.1	表达错误的通用方法	236
15.1.1	错误码格式	237
15.1.2	定义方法	238
15.1.3	使用示例	239
15.1.4	提高可使用性	240
15.1.5	定义和使用错误码的准则	246
15.2	优化错误日志的输出	246
15.2.1	传统方法	246
15.2.2	更有效的方法	249
15.3	平台和框架层的错误处理	251
15.4	小结	251
第 16 章	目录结构管理，使项目进展更顺利	252
16.1	规划目录结构的意义	252
16.1.1	书架功能	252
16.1.2	意识引导	252
16.1.3	加速新手上手	253
16.2	出色目录结构的特点	253
16.3	一个示例	253
16.4	小结	254
第 17 章	平台与框架开发，高质量软件打造之路	255

17.1	区分系统库、平台和框架	255
17.1.1	系统库	255
17.1.2	平台	256
17.1.3	框架	256
17.2	本质和优点	257
17.3	确立架构模型	258
17.4	小结	259
第 18 章	可开发性设计，一种高效且经济的开发模式	260
18.1	可开发性问题一瞥	260
18.2	可开发性设计的内涵	261
18.3	引入设备抽象层	261
18.4	更复杂的设备抽象层	263
18.5	图形界面的可开发性设计	264
18.5.1	增强设备抽象层	264
18.5.2	提供可视化编辑环境	264
18.6	其他可开发性设计	264
18.7	小结	265
操作系统篇		
第 19 章	引导加载器，系统启航者	268
19.1	功能	268
19.2	文件存储布局	269
19.3	程序加载原理	270
19.4	优点	274
19.5	小结	274
练习与思考 275		
第 20 章	任务，软件基本调度单元	276
20.1	任务情景	278
20.1.1	情景内容	278
20.1.2	情景保存	279
20.1.3	情景恢复	281
20.1.4	情景切换	282
20.2	任务调度	286
20.2.1	调度算法	286
20.2.2	调度器	290
20.3	任务的生命周期	293
20.4	任务控制	295
20.4.1	任务创建	297
20.4.2	任务启动	306
20.4.3	任务删除	307
20.4.4	任务挂起	309
20.4.5	任务恢复	310
20.4.6	任务睡眠	311
20.5	竞争问题与中断控制	313
20.5.1	竞争问题的产生	314

20.5.2	通过中断控制解决竞争问题	315
20.5.3	中断控制的嵌套问题	316
20.6	任务与中断状态	317
20.7	任务栈溢出检测	318
20.8	滴答与空闲任务	320
20.9	多任务环境控制	323
20.10	任务模块管理	324
20.11	taskv1 示例程序	326
20.12	任务钩子函数	330
20.13	任务变量	334
20.13.1	taskv2 示例程序	334
20.13.2	原理	336
20.13.3	实现	337
20.14	其他概念与思考	340
20.14.1	抢占式任务与实时系统的关系	340
20.14.2	影响任务切换效率的因素	341
20.14.3	避免直接删除任务	341
20.14.4	小心多任务设计被滥用	342
20.15	小结	343
	练习与思考	343
	第 21 章 任务同步与通信, 实现协同工作	345
21.1	信号量	345
21.1.1	应用场合	345
21.1.2	程序实现	347
21.1.3	semaphore 示例程序	358
21.2	互斥锁	360
21.2.1	应用场合	361
21.2.2	程序实现	361
21.2.3	mutex 示例程序	365
21.2.4	优先级反转与继承	367
21.2.5	递归锁	375
21.3	事件	379
21.3.1	应用场合	379
21.3.2	程序实现	379
21.3.3	event 示例程序	384
21.4	消息队列	386
21.4.1	应用场合	386
21.4.2	程序实现	387
21.4.3	实现消息队列	390
21.4.4	queue 示例程序	396
21.4.5	使用指南	398
21.5	死锁及预防	399
21.6	小结	399
	练习与思考	400

第 22 章 内存管理，协调动态内存的使用	401
22.1 堆管理	401
22.1.1 heapv1 示例程序	401
22.1.2 程序实现	406
22.1.3 设计改进	416
22.1.4 支持内存泄漏检测	421
22.1.5 实现内存溢出检测	431
22.1.6 内存碎片问题	431
22.2 内存池管理	432
22.2.1 mpool 示例程序	432
22.2.2 程序实现	436
22.2.3 缓冲区泄漏检测	444
22.3 小结	444
练习与思考	444
第 23 章 设备管理，方便与外设交互	445
23.1 字符设备管理	445
23.2 中断管理	447
23.2.1 中断向量表	447
23.2.2 中断控制	448
23.2.3 中断状态管理	450
23.2.4 设备与中断	451
23.2.5 模块管理	451
23.3 实现设备管理	452
23.3.1 安装驱动程序	454
23.3.2 注册设备	455
23.3.3 打开设备	456
23.3.4 关闭设备	458
23.3.5 设备读写与控制	458
23.4 设备驱动程序实现	459
23.4.1 “滴答”设备	460
23.4.2 控制台设备	462
23.4.3 终止程序运行设备	464
23.5 驱动安装与设备注册	466
23.6 小结	468
练习与思考	468
第 24 章 定时器，程序闹钟	469
24.1 软件定时器分类	469
24.2 设计思路	469
24.3 中断回调定时器	470
24.3.1 程序实现	470
24.3.2 timerv1 示例程序	481
24.4 定时误差	484
24.5 提高遍历效率	484
24.6 改善实时性	489

24.6.1	实时性分析	490
24.6.2	改进实时性	491
24.7	任务回调定时器	494
24.7.1	程序实现	494
24.7.2	timerv3 示例程序	497
24.8	小结	498
	练习与思考	498
	第 25 章 ClearRTOS“实时”操作系统	499
25.1	设计原则	499
25.2	源程序目录管理	499
25.3	让 Makefile 体现概念	502
25.4	实现集中配置	503
25.5	改进与移植	504
	质量保证篇	
	第 26 章 质量保证导言	508
26.1	软件开发的特点	508
26.1.1	脑力密集型工作	508
26.1.2	实现不具唯一性	508
26.1.3	隐性成本高	509
26.1.4	忽视的细节很容易被放大	509
26.1.5	质量难以评估	509
26.2	保证质量的关键要素	509
26.2.1	完备的需求分析	510
26.2.2	高质量的设计	510
26.2.3	编程好习惯	510
26.2.4	充分的验证	510
26.2.5	必要的流程	511
26.2.6	合适的工具	512
26.2.7	言简意赅的文档	512
26.3	质量保证需要系统性的方法论	512
26.3.1	方法论 = 流程 + 工具	513
26.3.2	构建有效方法论的核心手段	516
26.4	走出质量困境的指导性思想	518
26.4.1	从管理者的角度	518
26.4.2	从工程师的角度	519
26.4.3	从组织的角度	519
26.5	小结	520
	第 27 章 编程好习惯，质量保证的基本条件	521
27.1	终生受用的编程好习惯	521
27.1.1	判断失败而非成功	521
27.1.2	采用 sizeof 减少内存操作失误	522
27.1.3	屏蔽编程语言特性	524
27.1.4	恰当使用 goto 语句	527
27.1.5	合理运用数组	529

27.1.6	以逆序方式释放资源	530
27.1.7	在模块对外接口中防范错误	531
27.1.8	避免出现魔数	532
27.1.9	利用编程语言特性提高效率	533
27.1.10	复用代码提高维护性	534
27.1.11	借助隐式初始化简化程序逻辑	536
27.1.12	青睐小粒度锁	538
27.1.13	精确包含头文件	539
27.1.14	让模块的对外头文件保持简洁	541
27.1.15	只暴露必要的变量和函数	542
27.1.16	清除编译器报告的所有警告	542
27.2	小结	543
第 28 章	单元测试，被忽视的质量保证方法	544
28.1	警惕单元测试无用论	544
28.2	一个简单但不完善的单元测试例子	545
28.3	构建单元测试框架	548
28.4	无缝整合单元测试	555
28.4.1	维护规则	557
28.4.2	目录规划	557
28.4.3	更改 Makefile	559
28.4.4	检查整合效果	566
28.5	几个实施问题	569
28.6	桩函数和打桩	570
28.7	错误注入，一种可测试性设计	571
28.8	平台开发与单元测试	576
28.9	被测行为的确定性	576
28.10	测试用例的有效性	577
28.11	小结	578
第 29 章	代码覆盖，单元测试效果的衡量指标	579
29.1	了解代码覆盖工具	580
29.2	无缝整合代码覆盖	584
29.2.1	更改 Makefile	584
29.2.2	检查整合效果	586
29.3	三个代码覆盖程度指标	587
29.4	小结	588
第 30 章	静态分析，防止将失误带给用户	589
30.1	认识静态分析工具	589
30.2	无缝整合静态分析	596
30.2.1	更改 Makefile	596
30.2.2	检查整合效果	601
30.3	小结	602
第 31 章	动态分析，使程序更健壮	603
31.1	结识动态分析工具	604

31.2	无缝整合动态分析	607
31.2.1	更改 Makefile	607
31.2.2	检查整合效果	609
31.3	小结	609
第 32 章	性能分析，让优化程序有的放矢	610
32.1	初探性能分析工具	610
32.2	无缝整合性能分析	612
32.2.1	更改 Makefile	613
32.2.2	检查整合效果	614
32.3	小结	615
第 33 章	qBench，一个开发高质软件的工作台	616
参考资料		618