

# 《Orange' S: 一个操作系统的实现》

## 目 录

### 上 篇

#### 第 1 章 马上动手写一个最小的“操作系统” 2

- 1.1 准备工作 2
- 1.2 十分钟完成的操作系统 3
- 1.3 引导扇区 4
- 1.4 代码解释 4
- 1.5 水面下的冰山 6
- 1.6 回顾 7

#### 第 2 章 搭建你的工作环境 8

- 2.1 虚拟计算机 Bochs 8
  - 2.1.1 Bochs 初体验 8
  - 2.1.2 Bochs 的安装 9
  - 2.1.3 Bochs 的使用 10
  - 2.1.4 用 Bochs 调试操作系统 12
- 2.2 QEMU 15
- 2.3 平台之争: Windows 还是\*nix 16
- 2.4 GNU/Linux 下的开发环境 20
- 2.5 Windows 下的开发环境 22
- 2.6 总结 23

#### 第 3 章 保护模式 (Protect Mode) 25

- 3.1 认识保护模式 25
  - 3.1.1 保护模式的运行环境 29
  - 3.1.2 GDT (Global Descriptor Table) 31
  - 3.1.3 实模式到保护模式, 不一般的 jmp 33
  - 3.1.4 描述符属性 35
- 3.2 保护模式进阶 38
  - 3.2.1 海阔凭鱼跃 38
  - 3.2.2 LDT (Local Descriptor Table) 44
  - 3.2.3 特权级概述 48
  - 3.2.4 特权级转移 51
  - 3.2.5 关于“保护”二字的一点思考 65
- 3.3 页式存储 65
  - 3.3.1 分页机制概述 66
  - 3.3.2 编写代码启动分页机制 67
  - 3.3.3 PDE 和 PTE 68
  - 3.3.4 cr3 71
  - 3.3.5 回头看代码 72
  - 3.3.6 克勤克俭用内存 73
  - 3.3.7 进一步体会分页机制 81
- 3.4 中断和异常 87
  - 3.4.1 中断和异常机制 87

3.4.2	外部中断	90
3.4.3	编程操作 8259A	91
3.4.4	建立 IDT	94
3.4.5	实现一个中断	95
3.4.6	时钟中断试验	96
3.4.7	几点额外说明	98
3.5	保护模式下的 I/O	100
3.5.1	IOPL	100
3.5.2	I/O 许可位图 (I/O Permission Bitmap)	100
3.6	保护模式小结	101
第 4 章	让操作系统走进保护模式	102
4.1	突破 512 字节的限制	102
4.1.1	FAT12	103
4.1.2	DOS 可以识别的引导盘	108
4.1.3	一个最简单的 Loader	108
4.1.4	加载 Loader 入内存	109
4.1.5	向 Loader 交出控制权	116
4.1.6	整理 boot.asm	116
4.2	保护模式下的“操作系统”	117
第 5 章	内核雏形	119
5.1	在 Linux 下用汇编写 Hello World	119
5.2	再进一步, 汇编和 C 同步使用	120
5.3	ELF (Executable and Linkable Format)	123
5.4	从 Loader 到内核	127
5.4.1	用 Loader 加载 ELF	127
5.4.2	跳入保护模式	131
5.4.3	重新放置内核	137
5.4.4	向内核交出控制权	142
5.5	扩充内核	143
5.5.1	切换堆栈和 GDT	144
5.5.2	整理我们的文件夹	148
5.5.3	Makefile	149
5.5.4	添加中断处理	155
5.5.5	两点说明	168
5.6	小结	169
第 6 章	进程	171
6.1	迟到的进程	171
6.2	概述	171
6.2.1	进程介绍	172
6.2.2	未雨绸缪——形成进程的必要考虑	172
6.2.3	参考的代码	173
6.3	最简单的进程	174
6.3.1	简单进程的关键技术预测	175
6.3.2	第一步——ring0→ring1	178

6.3.3	第二步——丰富中断处理程序	189
6.4	多进程	200
6.4.1	添加一个进程体	200
6.4.2	相关的变量和宏	200
6.4.3	进程表初始化代码扩充	202
6.4.4	LDT	203
6.4.5	修改中断处理程序	203
6.4.6	添加一个任务的步骤总结	206
6.4.7	号外：Minix 的中断处理	207
6.4.8	代码回顾与整理	212
6.5	系统调用	220
6.5.1	实现一个简单的系统调用	222
6.5.2	get_ticks 的应用	227
6.6	进程调度	232
6.6.1	避免对称——进程的节奏感	232
6.6.2	优先级调度总结	240
第 7 章	输入/输出系统	242
7.1	键盘	242
7.1.1	从中断开始——键盘初体验	242
7.1.2	AT、PS/2 键盘	243
7.1.3	键盘敲击的过程	244
7.1.4	用数组表示扫描码	248
7.1.5	键盘输入缓冲区	251
7.1.6	用新加的任务处理键盘操作	253
7.1.7	解析扫描码	254
7.2	显示器	263
7.2.1	初识 TTY	264
7.2.2	基本概念	264
7.2.3	寄存器	267
7.3	TTY 任务	270
7.3.1	TTY 任务框架的搭建	272
7.3.2	多控制台	277
7.3.3	完善键盘处理	281
7.3.4	TTY 任务总结	288
7.4	区分任务和用户进程	289
7.5	printf	291
7.5.1	为进程指定 TTY	292
7.5.2	printf() 的实现	292
7.5.3	系统调用 write()	294
7.5.4	使用 printf()	296
下 篇		
第 8 章	进程间通信	300
8.1	微内核还是宏内核	300
8.1.1	Linux 的系统调用	302

8.1.2	Minix 的系统调用	303
8.1.3	我们的选择	305
8.2	IPC	306
8.3	实现 IPC	306
8.3.1	assert() 和 panic()	309
8.3.2	msg_send() 和 msg_receive()	313
8.3.3	增加消息机制之后的进程调度	321
8.4	使用 IPC 来替换系统调用 get_ticks	322
8.5	总结	324
第 9 章	文件系统	325
9.1	硬盘简介	325
9.2	硬盘操作的 I/O 端口	326
9.3	硬盘驱动程序	327
9.4	文件系统	337
9.5	硬盘分区表	338
9.6	设备号	344
9.7	用代码遍历所有分区	347
9.8	完善硬盘驱动程序	352
9.9	在硬盘上制作一个文件系统	355
9.9.1	文件系统涉及的数据结构	356
9.9.2	编码建立文件系统	358
9.10	创建文件	366
9.10.1	Linux 下的文件操作	366
9.10.2	文件描述符 (file descriptor)	367
9.10.3	open()	369
9.11	创建文件所涉及的其他函数	377
9.11.1	strip_path()	377
9.11.2	search_file()	378
9.11.3	get_inode() 和 sync_inode()	379
9.11.4	init_fs()	381
9.11.5	read_super_block() 和 get_super_block()	382
9.12	关闭文件	383
9.13	查看已创建的文件	384
9.14	打开文件	386
9.15	读写文件	387
9.16	测试文件读写	390
9.17	文件系统调试	393
9.18	删除文件	395
9.19	插曲: 奇怪的异常	401
9.20	为文件系统添加系统调用的步骤	403
9.21	将 TTY 纳入文件系统	404
9.22	改造 printf	411
9.23	总结	413
第 10 章	内存管理	414

10.1	fork	414
10.1.1	认识 fork	414
10.1.2	fork 前要做的工作（为 fork 所做的准备）	417
10.1.3	fork() 库函数	421
10.1.4	MM	421
10.1.5	运行	427
10.2	exit 和 wait	427
10.3	exec	432
10.3.1	认识 exec	433
10.3.2	为自己的操作系统编写应用程序	434
10.3.3	“安装”应用程序	436
10.3.4	实现 exec	442
10.4	简单的 shell	447
10.5	总结	449
第 11 章	尾声	451
11.1	让 mkfs() 只执行一次	451
11.2	从硬盘引导	455
11.2.1	编写硬盘引导扇区和硬盘版 loader	455
11.2.2	“安装” hdboot.bin 和 hdldr.bin	461
11.2.3	grub	461
11.2.4	小结	463
11.3	将 OS 安装到真实的计算机	465
11.3.1	准备工作	465
11.3.2	安装 Linux	466
11.3.3	编译源代码	466
11.3.4	开始安装	467
11.4	总结	467
参考文献		470