

## 目 录

### 第 1 章 简介(1)

#### 1.1 简介

- 1.1.1 简史
- 1.1.2 创始之初
- 1.1.3 繁衍
- 1.1.4 BSD
- 1.1.5 System V
- 1.1.6 商业化
- 1.1.7 Mach
- 1.1.8 标准
- 1.1.9 OSF 和 UI
- 1.1.10 SVR4 及其之后

#### 1.2 演变动力

- 1.2.1 功能
- 1.2.2 网络
- 1.2.3 性能
- 1.2.4 硬件变化
- 1.2.5 改进质量
- 1.2.6 模式变化
- 1.2.7 其他应用领域
- 1.2.8 简洁就是美
- 1.2.9 灵活性

#### 1.3 回顾与展望

- 1.3.1 UNIX 好在哪里
- 1.3.2 UNIX 的误区在哪儿

#### 1.4 本书的范围

#### 1.5 参考文献

### 第 2 章 进程与内核(17)

#### 2.1 简介

#### 2.2 模式、空间和上下文

#### 2.3 进程抽象

- 2.3.1 进程状态
- 2.3.2 进程上下文
- 2.3.3 用户凭证
- 2.3.4 u 区和 proc 结构

#### 2.4 内核态下运行

- 2.4.1 系统调用接口
- 2.4.2 中断处理

#### 2.5 同步

- 2.5.1 阻塞操作
- 2.5.2 中断
- 2.5.3 多处理器
- 2.6 进程调度
- 2.7 信号
- 2.8 新进程和程序
  - 2.8.1 fork 和 exec
  - 2.8.2 进程创建
  - 2.8.3 fork 优化
  - 2.8.4 执行一个新程序
  - 2.8.5 进程终止
  - 2.8.6 等待进程终止
  - 2.8.7 僵尸(Zombie)进程
- 2.9 小结
- 2.10 练习
- 2.11 参考文献

### 第3章 线程和轻量级进程(41)

- 3.1 简介
  - 3.1.1 动机
  - 3.1.2 多线程和多处理器
  - 3.1.3 并发和并行
- 3.2 基本抽象概念
  - 3.2.1 内核线程
  - 3.2.2 轻量级进程
  - 3.2.3 用户线程
- 3.3 轻量级进程设计——要考虑的问题
  - 3.3.1 fork 的语义
  - 3.3.2 其他的系统调用
  - 3.3.3 信号传递和处理
  - 3.3.4 可视性
  - 3.3.5 堆栈增长
- 3.4 用户级线程库
  - 3.4.1 编程接口
  - 3.4.2 线程库的实现
- 3.5 调度器调用
- 3.6 Solaris 和 SVR4 的多线程处理
  - 3.6.1 内核线程
  - 3.6.2 轻量级进程的实现
  - 3.6.3 用户线程
  - 3.6.4 用户线程的实现
  - 3.6.5 中断处理
  - 3.6.6 系统调用处理
- 3.7 Mach 中的线程

- 3.7.1 Mach 的抽象概念——任务和线程
- 3.7.2 Mach 的 C-threads
- 3.8 Digital UNIX
  - 3.8.1 UNIX 接口
  - 3.8.2 系统调用和信号
  - 3.8.3 pthreads 线程库
- 3.9 Mach 3.0 的续体
  - 3.9.1 编程模型
  - 3.9.2 使用续体
  - 3.9.3 优化
  - 3.9.4 分析
- 3.10 小结
- 3.11 练习
- 3.12 参考文献

## 第 4 章 信号和会话管理(72)

- 4.1 简介
- 4.2 信号生成和处理
  - 4.2.1 信号处理
  - 4.2.2 信号生成
  - 4.2.3 典型情景
  - 4.2.4 睡眠和信号
- 4.3 不可靠信号
- 4.4 可靠的信号
  - 4.4.1 主要特性
  - 4.4.2 SVR3 的实现
  - 4.4.3 BSD 信号管理
- 4.5 SVR4 信号机制
- 4.6 信号机制的实现
  - 4.6.1 信号生成
  - 4.6.2 信号传递和处理
- 4.7 异常
- 4.8 Mach 中的异常处理
  - 4.8.1 异常端口
  - 4.8.2 错误处理
  - 4.8.3 调试器的交互
  - 4.8.4 分析
- 4.9 进程组和终端管理
  - 4.9.1 基本概念
  - 4.9.2 SVR3 模型
  - 4.9.3 局限性
  - 4.9.4 4.3BSD 中的进程组和终端
  - 4.9.5 缺点
- 4.10 SVR4 会话的体系结构

- 4.10.1 目的(动机)
- 4.10.2 会话和进程组
- 4.10.3 数据结构
- 4.10.4 控制终端
- 4.10.5 4.4BSD 中会话的实现
- 4.11 小结
- 4.12 练习
- 4.13 参考文献

## 第 5 章 进程调度(98)

- 5.1 简介
- 5.2 时钟中断处理
  - 5.2.1 调出链表
  - 5.2.2 报警
- 5.3 调度器的目标
- 5.4 传统的 UNIX 调度
  - 5.4.1 进程优先级
  - 5.4.2 调度器的实现
  - 5.4.3 运行队列管理
  - 5.4.4 分析
- 5.5 SVR4 的调度器
  - 5.5.1 类无关层
  - 5.5.2 调度类的接口
  - 5.5.3 分时类
  - 5.5.4 实时类
  - 5.5.5 系统调用 `priocntl`
  - 5.5.6 分析
- 5.6 Solaris2.x 调度的改善
  - 5.6.1 抢占式内核
  - 5.6.2 多处理器的支持
  - 5.6.3 隐式调度
  - 5.6.4 优先级逆转
  - 5.6.5 优先级继承的实现
  - 5.6.6 优先继承的局限性
  - 5.6.7 Turnstiles
  - 5.6.8 分析
- 5.7 mach 中的调度
  - 5.7.1 多处理器的支持
- 5.8 Digital UNIX 的实时调度器
  - 5.8.1 多处理器支持
- 5.9 其他的一些调度实现
  - 5.9.1 fair share 调度
  - 5.9.2 最终期限驱动调度
  - 5.9.3 三级(Three-Level)调度器

- 5.10 小结
- 5.11 练习
- 5.12 参考文献

## 第 6 章 进程间通信(130)

- 6.1 简介
- 6.2 通用 IPC 方法
  - 6.2.1 信号
  - 6.2.2 管道
  - 6.2.3 SVR4 的管道
  - 6.2.4 进程跟踪
- 6.3 System V 的进程间通信
  - 6.3.1 公共元素
  - 6.3.2 信号量
  - 6.3.3 消息队列
  - 6.3.4 共享内存
  - 6.3.5 讨论
- 6.4 Mach IPC
  - 6.4.1 基本概念
- 6.5 消息
  - 6.5.1 消息的数据结构
  - 6.5.2 消息传递接口
- 6.6 端口
  - 6.6.1 端口名字空间
  - 6.6.2 端口数据结构
  - 6.6.3 端口变换
- 6.7 消息传递
  - 6.7.1 端口权力的传递
  - 6.7.2 脱机内存
  - 6.7.3 控制流
  - 6.7.4 通知
- 6.8 端口操作
  - 6.8.1 释放一个端口
  - 6.8.2 备份端口
  - 6.8.3 端口集合
  - 6.8.4 端口的添加
- 6.9 扩展性
- 6.10 Mach 3.0 的改进
  - 6.10.1 一次发送权
  - 6.10.2 Mach 3.0 的通知
  - 6.10.3 发送权的用户引用记数
- 6.11 讨论
- 6.12 小结
- 6.13 练习

## 6.14 参考文献

## 第 7 章 同步和多处理器(164)

### 7.1 简介

### 7.2 传统 UNIX 内核中的同步

#### 7.2.1 中断屏蔽

#### 7.2.2 睡眠和唤醒

#### 7.2.3 传统方法的局限性

### 7.3 多处理器系统

#### 7.3.1 内存模型

#### 7.3.2 同步支持

#### 7.3.3 软件体系结构

### 7.4 多处理器同步问题

#### 7.4.1 唤醒丢失问题

#### 7.4.2 巨群问题

### 7.5 信号灯

#### 7.5.1 提供互斥访问的信号灯

#### 7.5.2 使用的信号灯的事件等待

#### 7.5.3 用于控制可计数资源的信号灯

#### 7.5.4 信号灯的缺点

#### 7.5.5 护卫

### 7.6 自旋锁

#### 7.6.1 自旋锁的使用

### 7.7 条件变量

#### 7.7.1 实现问题

#### 7.7.2 事件

#### 7.7.3 阻塞锁

### 7.8 读写锁

#### 7.8.1 设计考虑

#### 7.8.2 实现

### 7.9 引用计数

### 7.10 其他考虑

#### 7.10.1 死锁避免

#### 7.10.2 递归锁

#### 7.10.3 阻塞还是自旋

#### 7.10.4 锁什么

#### 7.10.5 粒度和持续时间

### 7.11 例子分析

#### 7.11.1 SVR 4.2/MP

#### 7.11.2 Digital UNIX

#### 7.11.3 其他实现

### 7.12 小结

### 7.13 练习

### 7.14 参考文献

## 第 8 章 文件系统接口和框架(191)

- 8.1 简介
- 8.2 文件的用户接口
  - 8.2.1 文件和目录
  - 8.2.2 文件属性
  - 8.2.3 文件描述符
  - 8.2.4 文件 I/O
  - 8.2.5 分散-聚集 I/O(Scatter - Gather I/O)
  - 8.2.6 文件加锁
- 8.3 文件系统
  - 8.3.1 逻辑磁盘
- 8.4 特殊文件
  - 8.4.1 符号链接
  - 8.4.2 管道和 FIFO
- 8.5 文件系统框架
- 8.6 vnode/vfs 体系结构
  - 8.6.1 目标
  - 8.6.2 设备 1 门的经验
  - 8.6.3 vnode/vfs 接口概述
- 8.7 实现概述
  - 8.7.1 目标
  - 8.7.2 v 节点和打开文件
  - 8.7.3 v 节点
  - 8.7.4 v 节点引用计数
  - 8.7.5 vfs 对象
- 8.8 文件系统相关对象
  - 8.8.1 每个文件的私有数据
  - 8.8.2 vnodeops 向量
  - 8.8.3 vfs 层中的文件系统相关部分
- 8.9 安装一个文件系统
  - 8.9.1 虚拟文件系统转换
  - 8.9.2 mount 的实现
  - 8.9.3 VFS-MOUNT 处理
- 8.10 对文件的操作
  - 8.10.1 路径名遍历
  - 8.10.2 目录查找缓存
  - 8.10.3 VOP\_LOOKUP 操作
  - 8.10.4 打开文件
  - 8.10.5 文件 I/O
  - 8.10.6 文件属性
  - 8.10.7 用户凭证
- 8.11 分析
  - 8.11.1 SVR4 实现的缺点

- 8.11.2 4.4BSD 模型
- 8.11.3 OSF/1 方法
- 8.12 小结
- 8.13 练习
- 8.14 参考文献

## 第 9 章 文件系统实现(227)

- 9.1 简介
- 9.2 System V 文件系统(s5fs)
  - 9.2.1 目录
  - 9.2.2 i 节点
  - 9.2.3 超级块
- 9.3 s5fs 内核组织
  - 9.3.1 内存 i 节点
  - 9.3.2 i 节点查找
  - 9.3.3 文件 I/O
  - 9.3.4 i 节点的分配与回收
- 9.4 对 s5fs 的分析
- 9.5 伯克利快速文件系统(FFS)
- 9.6 硬盘结构
- 9.7 磁盘组织
  - 9.7.1 块和碎片
  - 9.7.2 分配策略
- 9.8 FFS 的增强功能
- 9.9 分析
- 9.10 临时文件系统
  - 9.10.1 内存文件系统
  - 9.10.2 tmpfs 文件系统
- 9.11 特殊目的文件系统
  - 9.11.1 specfs 文件系统
  - 9.11.2 /proc 文件系统
  - 9.11.3 处理器文件系统
  - 9.11.4 半透明文件系统
- 9.12 以往的磁盘缓存
  - 9.12.1 基本操作
  - 9.12.2 缓冲区头结构
  - 9.12.3 优点
  - 9.12.4 缺点
  - 9.12.5 保证文件系统的一致性
- 9.13 小结
- 9.14 练习
- 9.15 参考文献

## 第 10 章 分布式文件系统(255)



- 10.1 简介
- 10.2 分布式文件系统的一般特征
  - 10.2.1 设计考虑
- 10.3 网络文件系统(NFS)
  - 10.3.1 用户透视
  - 10.3.2 设计目标
  - 10.3.3 NFS 组成
  - 10.3.4 无状态
- 10.4 协议族
  - 10.4.1 扩展数据表示(XDR)
  - 10.4.2 远程过程调用(RPC)
- 10.5 NFS 实现
  - 10.5.1 控制流
  - 10.5.2 文件句柄
  - 10.5.3 mount 操作
  - 10.5.4 路径名查找
- 10.6 UNIX 语义
  - 10.6.1 打开文件权限
  - 10.6.2 删除打开文件
  - 10.6.3 读和写
- 10.7 NFS 性能
  - 10.7.1 性能瓶颈
  - 10.7.2 客户端高速缓存
  - 10.7.3 延迟写
  - 10.7.4 重传高速缓存
- 10.8 专用 NFS 服务器
  - 10.8.1 Auspex 功能性多处理器结构
  - 10.8.2 IBM 的 HA-NFS 服务器
- 10.9 NFS 安全性
  - 10.9.1 NFS 访问控制
  - 10.9.2 UID 重新映射
  - 10.9.3 root 重新映射
- 10.10 NFSv3
- 10.11 远程文件共享(RFS)文件系统
- 10.12 RFS 结构
  - 10.12.1 远程消息协议
  - 10.12.2 有状态操作
- 10.13 RFS 实现
  - 10.13.1 远程安装
  - 10.13.2 RFS 客户和服务端
  - 10.13.3 崩溃恢复
  - 10.13.4 其他问题
- 10.14 客户端高速缓存
  - 10.14.1 高速缓存一致性

- 10.15 Andrew 文件系统
  - 10.15.1 可扩展的结构
  - 10.15.2 存储和名字空间组织
  - 10.15.3 会话语义
- 10.16 AFS 实现
  - 10.16.1 缓存以及一致性
  - 10.16.2 路径名查找
  - 10.16.3 安全
- 10.17 AFS 的缺陷
- 10.18 DCE 分布式文件系统(DCE DFS)
  - 10.18.1 DFS 体系结构
  - 10.18.2 高速缓冲区一致性
  - 10.18.3 令牌管理器
  - 10.18.4 其他 DFS 服务
  - 10.18.5 分析
- 10.19 小结
- 10.20 练习
- 10.21 参考文献

## 第 11 章 高级文件系统(298)

- 11.1 简介
- 11.2 传统文件系统的局限
  - 11.2.1 FFS 磁盘布局
  - 11.2.2 写的主导性
  - 11.2.3 元数据更新
  - 11.2.4 崩溃恢复
- 11.3 文件系统成簇(Sun-FFS)
- 11.4 日志方法
  - 11.4.1 基本特征
- 11.5 日志结构文件系统
- 11.6 4.4BSD 日志文件系统
  - 11.6.1 写日志
  - 11.6.2 数据检索
  - 11.6.3 崩溃恢复
  - 11.6.4 清除进程
  - 11.6.5 分析
- 11.7 元数据日志
  - 11.7.1 正常操作
  - 11.7.2 日志的一致
  - 11.7.3 崩溃恢复
  - 11.7.4 分析
- 11.8 Episode 文件系统
  - 11.8.1 基本抽象

- 11.8.2 结构
- 11.8.3 记日志
- 11.8.4 其他特性
- 11.9 监视器(watchdog)
  - 11.9.1 目录监视器
  - 11.9.2 消息通道
  - 11.9.3 应用
- 11.10 4.4BSD 端口文件系统
  - 11.10.1 使用端口(portals)
- 11.11 堆栈式文件系统层
  - 11.11.1 框架和接口
  - 11.11.2 Sun Soft 原型
- 11.12 4.4BSD 文件系统接口
  - 11.12.1 Nullfs 和 Union Mount 文件系统
- 11.13 小结
- 11.14 练习
- 11.15 参考文献

## 第 12 章 内核内存管理(328)

- 12.1 简介
- 12.2 功能需求
  - 12.2.1 评估标准
- 12.3 资源映射图分配器
  - 12.3.1 分析
- 12.4 简单 2 次幂空闲表
  - 12.4.1 分析
- 12.5 McKusick-Karels 分配器
  - 12.5.1 分析
- 12.6 伙伴系统
  - 12.6.1 分析
- 12.7 SVR4 Lazy 伙伴算法
  - 12.7.1 Lazy 合并
  - 12.7.2 SVR4 实现细节
- 12.8 Mach/1 的 zone 分配器
  - 12.8.1 垃圾收集
  - 12.8.2 分析
- 12.9 多处理器的分层分配器
  - 12.9.1 分析
- 12.10 Solaris 2.4 的 Slab 分配器
  - 12.10.1 对象复用
  - 12.10.2 硬件 Cache 利用率
  - 12.10.3 分配器 footprint
  - 12.10.4 设计与接口
  - 12.10.5 实现

- 12.10.6 分析
- 12.11 小结
- 12.12 练习
- 12.13 参考文献

## 第 13 章 虚存(352)

- 13.1 简介
  - 13.1.1 内存管理的石器时代
- 13.2 分页
  - 13.2.1 功能需求
  - 13.2.2 虚拟地址空间
  - 13.2.3 页面初始访问
  - 13.2.4 交换区
  - 13.2.5 转换映射图
  - 13.2.6 页面替换策略
- 13.3 硬件需求
  - 13.3.1 MMU 缓存
  - 13.3.2 Intel 80x86
  - 13.3.3 IBM RS/6000
  - 13.3.4 MIPS R3000
- 13.4 4.3BSD 实例研究
  - 13.4.1 物理内存
  - 13.4.2 地址空间
  - 13.4.3 页面在哪里
  - 13.4.4 交换区
- 13.5 4.3BSD 内存管理操作
  - 13.5.1 创建进程
  - 13.5.2 页面失效处理
  - 13.5.3 空闲页面链表
  - 13.5.4 交换
- 13.6 分析
- 13.7 练习
- 13.8 文献

## 第 14 章 SVR4 VM 体系结构(382)

- 14.1 动机
- 14.2 内存映射文件
  - 14.2.1 mmap 及相关系统用
- 14.3 VM 设计原理
- 14.4 基本抽象概念
  - 14.4.1 物理内存
  - 14.4.2 地址空间
  - 14.4.3 地址映射
  - 14.4.4 匿名页面

- 14.4.5 硬件地址转换
- 14.5 段驱动程序
  - 14.5.1 seg - vn
  - 14.5.2 seg - map
  - 14.5.3 seg - dev
  - 14.5.4 seg-kmem
  - 14.5.5 seg-kp
- 14.6 交换层
- 14.7 VM 操作
  - 14.7.1 创建一个新映射
  - 14.7.2 匿名页面处理
  - 14.7.3 创建进程
  - 14.7.4 共享匿名页面
  - 14.7.5 页面失效处理
  - 14.7.6 共享内存
  - 14.7.7 其他部件
- 14.8 与 v 节点子系统的交互
  - 14.8.1 v 节点接口变化
  - 14.8.2 统一的文件访问
  - 14.8.3 其他问题
- 14.9 Solaris 中的虚拟交换空间
  - 14.9.1 扩展交换空间
  - 14.9.2 虚交换管理
  - 14.9.3 讨论
- 14.10 分析
- 14.11 性能改进
  - 14.11.1 高失效率原因
  - 14.11.2 SVR4 对 SunOS VM 实现的改进
  - 14.11.3 结果与讨论
- 14.12 小结
- 14.13 练习
- 14.14 参考文献

## 第 15 章 进一步关于内存管理的主题(413)

- 15.1 简介
- 15.2 Mach 的内存管理设计
  - 15.2.1 设计目标
  - 15.2.2 编程接口
  - 15.2.3 基本抽象概念
- 15.3 共享内存设施
  - 15.3.1 copy-on-write 共享
  - 15.3.2 读写共享
- 15.4 内存对象和 Pager
  - 15.4.1 内存对象初始化

- 15.4.2 内核与 pager 间的接口
- 15.4.3 内核与 pager 交互
- 15.5 外部 pager 和内部 pager
  - 15.5.1 一个网络共享内存服务器
- 15.6 页面替换
- 15.7 分析
- 15.8 4.4BSD 的内存管理
- 15.9 快表(TLB)一致性
  - 15.9.1 单处理机上的 TLB 一致性
  - 15.9.2 多处理机问题
- 15.10 Mach 的 TLB 击落算法
  - 15.10.1 同步和死锁避免
  - 15.10.2 讨论
- 15.11 SVR4 和 SVR4.2 UNIX 中的 TLB 一致性
  - 15.11.1 SVR4/MP
  - 15.11.2 SVR4.2/MP
  - 15.11.3 Lazy 击落算法
  - 15.11.4 立即击落
  - 15.11.5 讨论
- 15.12 其他 TLB 一致性算法
- 15.13 虚地址缓存
  - 15.13.1 映射变化
  - 15.13.2 地址别名
  - 15.13.3 DMA 操作
  - 15.13.4 维护缓存一致性
  - 15.13.5 分析
- 15.14 练习
- 15.15 参考文献

## 第 16 章 设备驱动程序 I/O(446)

- 16.1 简介
- 16.2 概述
  - 16.2.1 硬件配置
  - 16.2.2 设备中断
- 16.3 设备驱动程序框架
  - 16.3.1 设备和驱动程序分类
  - 16.3.2 调用驱动程序代码
  - 16.3.3 设备开关表
  - 16.3.4 驱动程序入口点
- 16.4 I/O 子系统
  - 16.4.1 主、次设备号
  - 16.4.2 设备文件
  - 16.4.3 specfs 文件系统
  - 16.4.4 公共 snode

- 16.4.5 设备克隆
- 16.4.6 字符设备 I/O
- 16.5 poll 系统调用
  - 16.5.1 poll 的实现
  - 16.5.2 4.3BSD select 系统调用
- 16.6 块 I/O
  - 16.6.1 buf 结构
  - 16.6.2 与 v 节点的交互
  - 16.6.3 设备访问方法
  - 16.6.4 到块设备的 raw I/O
- 16.7 DDI/DKI 说明
  - 16.7.1 建议
  - 16.7.2 第三部分函数
  - 16.7.3 其他部分
- 16.8 新的 SVR4 版本
  - 16.8.1 多处理器可靠驱动程序
  - 16.8.2 SVR4.1/ES 的变化
  - 16.8.3 动态加载和卸载
- 16.9 发展趋势
- 16.10 小结
- 16.11 练习
- 16.12 参考文献

## 第 17 章 流(477)

- 17.1 目的
- 17.2 概述
- 17.3 消息和队列
  - 17.3.1 消息
  - 17.3.2 虚拟拷贝
  - 17.3.3 消息类型
  - 17.3.4 队列和模块
- 17.4 流 I/O
  - 17.4.1 STREAMS 调度程序
  - 17.4.2 优先带(Priority Bands)
  - 17.4.3 流量控制
  - 17.4.4 驱动程序尾
  - 17.4.5 流头
- 17.5 配置和设置
  - 17.5.1 配置一个模块或驱动程序
  - 17.5.2 打开流
  - 17.5.3 插入(Pushing)模块
  - 17.5.1 克隆设备
- 17.6 STREAMS ioctl
  - 17.6.1 STR ioctl 处理

- 17.6.2 透明 ioctl
- 17.7 内存分配
  - 17.7.1 扩展 STREAMS 缓冲区
- 17.8 多路复用
  - 17.8.1 上部多路复用器
  - 17.8.2 下部多路复用器
  - 17.8.3 链接流
  - 17.8.4 数据流
  - 17.8.5 普通链接和持久链接
- 17.9 FIFO 和管道
  - 17.9.1 STREAMS FIFO
  - 17.9.2 STREAMS 管道
- 17.10 网络接口
  - 17.10.1 传输供应者接口(TPI)
  - 17.10.2 传输层接口(TLI)
  - 17.10.3 sockets
  - 17.10.4 SVR4 socket 实现
- 17.11 小结
- 17.12 练习
- 17.13 参考文献