Linux系统中进程调度策略

葛　君[1,2],郑凤婷[2]

(1.软件工程国家重点实验室(武汉大学),湖北 武汉,430072; 2.商丘职业技术学院,河南 商丘,476000)

摘　要:Linux是一个多用户多任务的操作系统.Linux中实现了对多个进程公平、高效的调度,并不是采用单一的调度策略,而是几种调度策略有机地综合应用.

关键词:进程调度;优先级;时间片轮转;实时进程

在任何一种操作系统中,进程调度一直是一个核心问题,进程调度策略的选择对整个系统性能有至关重要的影响.一个好的调度算法应该考虑很多方面:公平、有效、响应时间、周转时间、系统吞吐量等等,但这些因素之间又是相互矛盾的,最终的取舍根据系统要达到的目标而定.本文以Linux操作系统为例,分析其进程调度策略,以期对进程调度过程有更深层次的认识.

1　Linux的进程调度

Linux支持多进程,进程控制块PCB(Process Control Block)是系统中最为重要的数据结构之一,用来存放进程所必需的各种信息.PCB用结构task_struct来表示,包括进程的类型、进程状态、优先级、时钟信息等.

Linux系统中,进程调度操作由schedule()函数执行.这是一个只在内核态运行的函数,函数代码为所有进程共享.

2　Linux进程调度时机

Linux的进程调度时机与现代操作系统中的调度时机基本一致,为了判断是否可以执行内核的进程调度程序来调度进程,Linux中设置了进程调度标志need_resched,当标志为1时,可执行调度程序.通常,Linux调度时机分以下两种情况:(1)主动调度:指显式调用schedule()函数明确释放CPU,引起新一轮调度.一般发生在当前进程状态改变,如:进程终止、进程睡眠、进程对某些信号处理过程中等.(2)被动调度:指不显示调用schedule()函数,只是PCB中的need_resched进程调度标志,该域置位为1将引起新的进程调度,而每当中断处理和系统调用返回时,核心调度程序都会主动查询need_resched的状态(若置位,则主动调用schedule()函数).一般发生在新的进程产生时、某个进程优先级改变时、某个进程等待的资源可用被唤醒时、当前进程时间片用完等.

3　Linux进程调度策略

一般来说,不同用途的操作系统的调度策略是不同的.Linux进程调度是将优先级调度、时间片轮转法调度、先进先出调度综合起来应用.Linux系统中,不同类型的进程调度策略也不一样.

3.1　与进程调度相关的数据结构

每个进程都是一个动态的个体,其生命周期依次定义的数据结构为:TASK_RUNNING,TASK_INTERRUPTIBLE,TASK_UNINTERRUPTIBLE,TASK_ZOMBIE和TASK_STOPPED.一个进程在其生存期间,状态会发生多次变化.

与其数据结构相对应的即是 Linux进程的状态,分别是:运行态、等待态、暂停态和僵死态.其中等待态又分为可中断睡眠和不可中断睡眠两种.状态转换图如图 1所示.

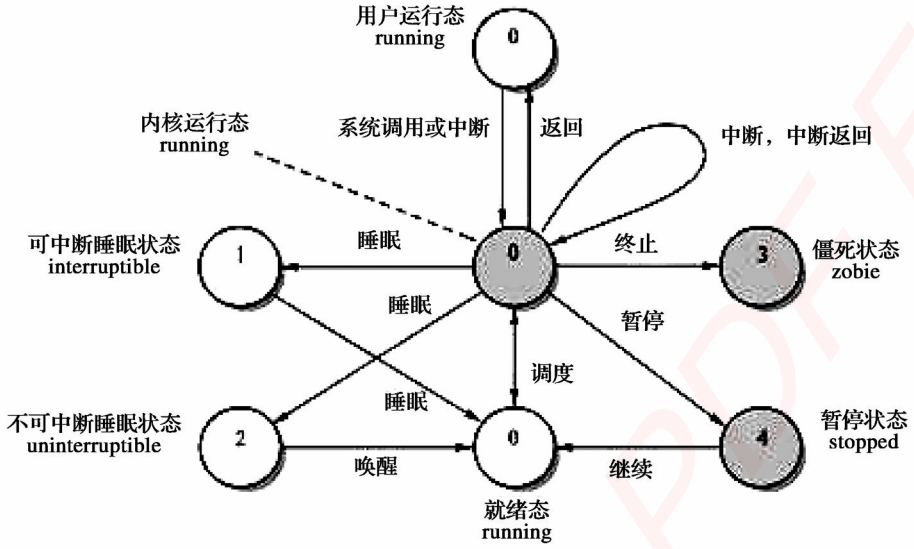

图 1 Linux进程的状态及其转换

其中:

TASK_RUNNING:进程正在运行处于运行状态或者将要运行处于就绪状态.

TASK_INTERRUPTIBLE:可中断睡眠,可以通过资源有效或信号唤醒.

TASK_UNINTERRUPTIBLE:不可中断睡眠,只能通过资源有效唤醒.

TASK_STOPPED:暂停状态,通过其他进程的信号才能被唤醒.

TASK_ZOBIE:僵死状态,大部分资源已经释放,进程终止前的一个过渡状态.

3.2 进程状态及其转换过程的描述

进程创建时的状态为不可打断睡眠,在 do_fork()结束前被父进程唤醒后,变为执行状态.处于执行状态的进程被移到 run_queue就绪任务队列中等待调度,适当时候由 schedule()按调度算法选中,获得 CPU.若采用轮转法,即时,由时钟中断触发 timer_interrupt(),其内部调用 schedule(),引起新一轮调度,当前进程的状态仍处于执行状态,因而把当前进程挂到 run - queue队尾[1]119 - 135.

获得 CPU且正在运行的进程若申请不到某资源,则调用 sleep - on()或 interruptible - sleep - on ()睡眠,其 task_struct进程控制块挂到相应资源的 wait_queue等待队列.如果调用 sleep_on(),则其状态变为不可打断睡眠,如果调用 interruptible_sleep_on(),则其状态变为可打断睡眠. sleep_on ()或 interruptible - sleep _on ()将调用 schedule()函数把睡眠进程释放.

3.3 进程分类和相应的进程调度策略

Linux系统中,为了高效地调度进程,将进程分成两类:实时进程和普通进程 (又称非实时进程或一般进程).实时进程的优先级要高于其他进程.如果一个实时进程处于可执行状态,它将先得到执行.实时进程又有两种策略:时间片轮转和先进先出.在时间片轮转策略中,每个可执行实时进程轮流执行一个时间片,而先进先出策略每个进程按各自在运行队列中的顺序执行且顺序不能变化[2].

在 Linux中,进程调度策略共定义了 3种:

(1) SCHED_FIFO:用于实时进程的调度.

(2) SCHED_RR:用于实时进程的调度.

(3) SCHED_OTHER:用于非实时进程的调度.

Linux系统中的每个进程用 task_struct结构来描述.进程调度的依据是 task_struct结构中的 policy、priority、counter和 rt_priority. PCB中设置 Policy数据项,其值用于反映针对不同类型的进程而采用的调度策略. SCHED_RR和 SCHED_FIFO用于实时进程,分别表示轮转调度策略和先进先出调度策略; SCHED_OTHER表示普通进程,也按照轮转调度策略处理.这三类调度策略均基于优先级. PCB中设置 Priority数据项,其值

为普通进程的调度优先级. 普通进程的可用时间片的初始值即为该值,该值通过系统调用是可以改变的. PCB中设置 rt_priority数据项,其值是实时进程专用的调度优先级,实时进程的可用时间片的初始值即为该值,该优先级也可以用系统调用来修改. PCB中设置 counter数据项,用于进程可用时间片时值的计数,初始值为 rt _priority或 Priority,进程启动后该值随时钟周期递减.

Linux确定进程调度策略时,将所应考虑的各种因素均反映到了进程的优先权中,即将进程调度时的各种原则要求统一到了优先权这一尺度上,各进程调度策略均以优先权为依据. 它的计算方法如下:

实时进程的优先权 weight = 1000 + rt_priority;

普通进程的优先权 weight = counter; counter初值为 priority.

可见,普通进程的优先权随着其进程占用的 CPU时间越长,优先权越低 (因 counter是时间片的时值计数器,随着进程使用 CPU的时间增长而递减). 另外,实时进程和普通进程使用轮转调度时,其时间片的初始值均取各自进程的优先级,这样就将进程的时间片与优先级对应起来了,优先级越高时间片越长[3]62 - 87.

4 结语

通过对 Linux进程调度策略的简单分析,可以看出多进程的管理是一种非常复杂的并发程序设计. 每个进程的状态不仅由其自身决定,而且还要受诸多外在因素的影响. 而在此基础上的进程调度,为了保证操作系统的稳定性、提高效率和增加灵活性,还必须采用很多方法. 这些都是值得我们去研究和探讨的.

参考文献:

[1]刘振鹏,李亚平,王 煜,等. 操作系统 [M]. 北京:中国铁道出版社, 2003.

[2]赵明富,李太福,陈鸿雁,等. Linux嵌入式系统的实时性分析 [J]. 电脑知识与技术, 2003, 29(18): 53 - 55.

[3]马季兰. 操作系统原理与 Linux系统 [M]北京:人民邮电出版社, 1999.

.

Linux Systems in the Process of Scheduling Strategy

GE Jun[1,2], ZHENG Feng - ting[2]

(1. *State Key Lab of Software Engineering (Wuhan University), Wuhan* 430072, *China;*

2. *Shangqiu Volcational and Technical College, Shangqiu* 476000, *China*)

Abstract: Linux operating system processes are important concepts, Linux in the process of implementation of a number of fair and efficient scheduling, scheduling is not a single strategy, but several scheduling strategy organically combine applications.

Key words: process scheduling; priority; round - robin time slice; real - time process

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发

23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版

30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

PowerPC:

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计

4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计

22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析

5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现