ULK Linux Kernel 2.6.11 Note 笔记

基于 ULK Linux Kernel 2.6.11 学习

ElegantLaTeX Program

Victory won't come to us unless we go to it.

作者：Miao

时间：July 11, 2023

邮箱: chenmiao.ku@gmail.com

版本: 0.10

目　　录

1　绪论 **3**

1.1　操作系统基本概念 3

1.2　多用户系统 3

1.3　用户和组 3

1.4　进程 4

1.5　内核体系结构 4

1.6　Unix 文件系统概述 5

1.6.1　文件 5

1.6.2　硬链接和软连接 5

1.6.3　文件类型 6

1.6.4　文件描述符与索引节点 6

1.6.5　访问权限和文件模式 7

第 1 章 绪论

1.1 操作系统基本概念

任何计算机系统都包含一个名为操作系统的基本程序集合。在这个集合中，最重要的程序称为内核 (kernel)。启动后，内核中包含了系统运行所必不可少的很多核心过程 (procedure)，和其他一些不太重要的实用程序。

系统根本的样子和能力还是由内核决定，内核也为操作系统中所有事情提供了主要功能，并决定高层软件的很多特性。

操作系统必须完成两个主要目标：

1) 与硬件部分交互，为包含在硬件平台上的所有底层可编程部件提供服务
2) 为运行在计算机系统上的应用程序提供执行环境

类 *Unix* 系统把与计算机物理组织相关的所有底层细节都对用户运行的程序隐藏起来，硬件为 *CPU* 引入了至少两种不同的执行模式：非特权模式 *(*用户态 *(User Mode)* 和特权模式 *(Kernel Mode))*。

1.2 多用户系统

多用户系统 *(multiuser system)* 就是一台能并发和独立地执行分别属于两个或多个用户的若干应用程序的计算机。

并发 (concurrently) 意味着几个应用程序能够同时处于活动状态并竞争各种资源。独立 (independently) 意味着每个应用程序能够执行自己的任务，而无需考虑其他用户的应用程序在做什么。

多用户操作系统必须包含：

1) 核实用户身份的认证机制
2) 防止有错误的用户程序妨碍其他应用程序在系统中运行的保护机制
3) 防止有恶意的用户程序干涉或窥视其他用户的活动的保护机制
4) 限制分配给每个用户的资源数的记账机制

1.3 用户和组

操作系统必须保证用户空间的私有部分仅仅对于其拥有者是可见的。

所有的用户由一个唯一的数字来表示，这个数字叫用户标识符 (User ID, UID)。

为了和其他用户有选择地共享资料，每个用户是一个或多个用户组的一名成员，组由唯一的用户组标识符 (user group ID) 标识。每个文件也恰好与一个组相对应。

任何类 *Unix* 操作系统都有一个特殊的用户，*root(超级用户 (superuser))*。系统管理员能够通过 root 账号登陆，值得一提的是：*root* 几乎无所不能，其能访问系统中的每一个文件，能干涉每一个正在执行的用户程序。

1.4 进程

所有的操作系统都有一种基本的抽象：进程 *(process)*。一个进程可以定义为：”程序运行时的一个实例”，或者一个运行程序的”执行上下文”。

传统的操作系统中，一个进程在地址空间中 *(address space)* 执行一个单独的指令序列。现代操作系统允许具有多个执行流的进程，也就是在相同的地址空间可执行多个指令序列。

允许进程并发活动的系统称为多道程序系统 *(multiprogramming)* 或多处理系统 *(multiprocessing)*。值得注意的是：几个进程能够并发地执行同一个程序，而同一个进程能顺序的执行几个程序。

调度程序 (scheduler) 的部分决定哪个进程能执行，一些操作系统只允许有非抢占式 (nonpreemptable) 进程，也就是说，只有当进程自愿放弃时，调度程序才能被调用。但是，多用户系统中的进程必须是抢占式的 *(preemptable)*。

1.5 内核体系结构

大部分 Unix 内核都是单块结构：每一个内核层都被集成到整个内核程序中，并代表当前进程在内核态下运行。

微内核 *(microkernel)* 操作系统只需要内核有一个很小的函数集 *(*几个同步原语[1]，一个简单的调度程序和进程间通信*)*。

Linux 内核提供了模块 (module) 用于达到微内核理论上的很多优点且不影响性能。模块是一个目标文件，其代码可以在运行时链接到内核或从内核解除链接。这种目标代码通常由一组函数组成，用来实现文件系统、驱动程序或其他内核上层功能。

使用模块的主要优点：

1) 模块化方法

   任何模块都能在运行时被链接或解除链接。这要求程序员提出良定义的软件接口以访问由模块处理的数据结构

2) 平台无关性

   即使模块依赖于某些特殊的硬件特点，但它不依赖于某个固定的硬件平台

[1]原语（primitive）是计算机科学中的一个概念，它指的是一组基本的操作或指令，可以直接在计算机硬件上执行。原语通常是由计算机硬件提供的，用于支持高级编程语言或操作系统的功能

3) 节省内存使用

当需要模块时，就链接；不需要时，则解除

4) 无性能损失

模块的目标代码一旦被链接进内核，起作用与静态链接的内核的目标代码完全对等。因此无需显式的进行消息传递[1]

1.6 Unix 文件系统概述

1.6.1 文件

Unix 文件是以字节序列组成的信息载体 (container)，内核不解释文件的内容。

从用户的观点来看，文件被组织在一个树结构的命名空间内：

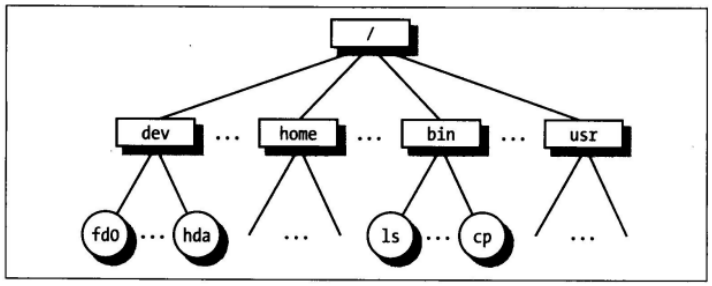

图 1.1: 目录树结构

除叶节点外，所有节点都表示目录名。目录节点包含它下面文件及目录的所有信息。

Unix 每个进程都有一个当前工作目录，属于进程执行上下文 (execution context)，标识出进程所用的当前目录。

路径名 (pathname) 由斜杠及一列指向文件的目录名交替组成。如果第一个字符是斜杠，那么就是所谓的绝对路径；否则就是所谓的相对路径。

当标识文件名时，用符号"." 和".." 分别标识当前工作目录和父目录。

1.6.2 硬链接和软连接

包含在目录中的文件名就是一个文件的硬链接 (hard link)，或简称连接 (link)。

使用 Unix 命令：

```
1 $ ln P1 P2
```

[1]模块被链接或解除时，都有一定的性能下降。但是在微内核中也是如此

用来创建一个新的硬链接，即为由路径 P1 标识的文件创建一个路径名为 P2 的硬链接。

硬链接有两方面的限制：

1) 不允许给目录创建硬链接，这可能使得目录树编程环形图从而无法通过名字定位一个文件
2) 只有在同一文件系统中的文件之间才能创建链接。

为了克服限制，引入软链接 *(soft link)[*也称符号链接 *(symbolic link)]*，符号链接是短文件，这些文件包含另一个文件的任意一个路径名。

值得注意的是：路径名可以指向位于任意一个文件系统的任意文件或目录 *(*哪怕它不存在*)*。

Unix 命令：

```
1 $ ln -s P1 P2
```

创建一个路径名为 P2 的新软连接，P2 指向路径名 P1。当执行命令时，文件系统抽取 P2 的目录部分，并在此创建 P2 的符号链接属性的新项。因此，任何对 P2 的引用都可以自动被转换为指向 P1 的引用。

1.6.3 文件类型

Unix 命令文件可以是以下类型:

- 普通文件 (regular file)
- 目录
- 符号链接
- 面向块的设备文件 (block-oriented device file)
- 面向字符的设备文件 (character-oriented device file)
- 管道 (pipe) 和命名管道 (named pipe)(也叫 FIFO)
- 套接字 (socket)

1.6.4 文件描述符与索引节点

除了设备文件和特殊文件系统外，每个文件都由字符序列组成。文件内容不包括任何控制信息，如文件长度或文件结束符 *(end-of-file, EOF)*。

文件系统处理文件需要的所有信息都包含在一个名为索引节点 *(inode)* 的数据结构中，文件系统用索引节点来标识文件。

索引节点 (inode) 至少包括：

- 文件类型
- 与文件相关的硬链接个数

- 以字节为单位的文件长度
- 设备标识符 (即包含文件的设备的标识符)
- 文件系统中标识的索引节点号
- 文件拥有者的 UID
- 文件的用户组 ID
- 几个时间戳 (改变时间、最后访问时间、最后修改时间)
- 访问权限和文件模式

1.6.5 访问权限和文件模式

文件的潜在用户分为三种类型：

- 文件所有者
- 同组用户 (不含所有者)
- 其他用户

同时，拥有三种类型的访问权限————读、写以及执行。因此就有九种组合不同的二进制来标记，还有三种额外的标记

- suid

  进程执行一个文件时通常保持进程拥有者的 UID，若设置 suid，进程就可以获取该文件拥有者的 UID

- sgid

  进程执行一个文件时保持进程组的用户组 ID，若设置 sgid，进程就可以获得该文件用户组 ID

- sticky

  设置 sticky 标志位相当于向内核发出请求，当程序结束后仍保留在内存[1]

[1]该标记已经过时，已被其他方法取代