

深入详解Linux内核中（内核和伪文件系统）



Linux嵌入式

3 人赞同了该文章

收起

一.内核简介

1、内核设计流派

- 单内核设计：把所有功能集成于同一个程序（Linux）
- 微内核设计：每种功能使用一个单独的子系统实现（Windows、Solaris）

2、内核功能

内核的主要功能包括进程管理、内存管理、网络管理、驱动程序、文件系统、安全功能。

3、内核特点

Linux 虽是单内核设计，但充分借鉴了微内核体系的设计的优点；为内核引入了模块化机制。

- **支持模块化**（.ko：kernel object，内核对象）

例如：文件系统、硬件驱动、网络协议等。

- **支持模块运行时动态装载或卸载**

二.内核组成部分

内核组成部分包括：内核核心、内核模块和辅助文件系统（非必须）。内核模块与内核核心版本一定要严格匹配。

下面就对内核组成部分进行介绍。

1、内核核心

内核核心（kernel），一般为 bzImage，通常位于 /boot 目录，名称为 vmlinuz-VERSION-release。

系统启动后，内核文件已经加载完成，即使你对内核文件进行修改，也不会对当前系统产生任何影响，除非重启或者打补丁。（CentOS 4 以后的版本支持打补丁进行滚动升级）

2、内核模块

简介

设计流派

功能

特点

组成部分

核心

模块

disk

信息获取

模块

iod

dinfo

dprobe

mod

isk 文件管理

initrd

提供的伪文件系统

下

oc

c/sys

通过/proc查看top中...

内核模块（kernel object，内核对象），一般放置于 /lib/modules/VERSION-release/。

有如下特点：

- 不管是32位系统还是64位系统，都是放在lib目录下
- 便于第三方厂商编写硬件驱动模块
- 支持动态装载和卸载
- 自己编译内核时，有以下三种选择：

```
[ ]: N      # 不要此内核模块，为空
[M]: Module # 编译为内核模块，用时再进行编译，只占用硬盘空间，不占用内存空间
[*]: Y      # 编译进内核核心，
```

```
[root@LeeMumu ~]# ls /lib/modules/3.10.0-957.21.3.el7.x86_64/
build          modules.builtin      modules.modesetting  source
extra          modules.builtin.bin  modules.networking   updates
kernel         modules.dep          modules.order        vdso
modules.alias  modules.dep.bin      modules.softdep      weak-updates
modules.alias.bin modules.devname      modules.symbols
modules.block  modules.drm          modules.symbols.bin
```

3、ramdisk

ramdisk 是辅助性文件，并非必须，这取决于内核是否能直接驱动 rootfs 所在的设备。

非必须的原因是因为如果能把硬盘驱动程序加载到内核里，就不需要创建临时文件系统。

ramdisk 可以理解为一个简装版的根文件系统。

ramdisk 可加载的驱动包括：

- 目标设备驱动，例如 SCSI 设备的驱动
- 逻辑设备驱动，例如 LVM 设备的驱动
- 文件系统，例如 xfs 文件系统

更多Linux内核源码高阶知识请加开发交流Q群【318652197】获取，进群免费获取相关资料，免费观看公开课技术分享，入群不亏,快来加入我们吧~ 前100名进群领取，额外赠送一份价值699的内核资料包（含视频教程、电子书、实战项目及代码）

20201021 深入理解Nginx模块与架构解析	20210519 剖析Linux内核CFS调度器	20210908 剖析linux内核protocolsockets_buff
20201027 高性能服务器《IO复用技术》详解	20210522 剖析Linux内核SMP负载均衡	20210911 剖析linux内核-内核互斥技术精髓
20201028 服务器通讯必备知识详解	20210601 剖析Linux内核ARM异常处理	20210915 剖析Linux内核虚拟文件系统架构
20201107 深度解析《大厂经典面试题》	20210606 剖析Linux内核物理内存管理	20210916 剖析Linux内核源码IPC机制
20201108 高性能稳定服务器《两招绝杀》	20210612 剖析Linux内核页高速缓存及回写	20210917 剖析Linux内核高速缓存精髓
20201112 高性能计算服务器：libevent和定时器	20210622 剖析Linux内核IPv4协议	20210918 剖析Linux内核内存分配与回收
20201121 面试大厂必考《预处理及内存管理》	20210624 Linux高性能服务器模型选择	20210920 剖析Linux内核设备驱动架构
20210310 剖析Linux内核进程调度与切换	20210630 剖析Linux内核内存管理	20210921 剖析Linux内核Ext2_3文件系统
20210312 剖析Linux内核调度策略	20210703 剖析Linux内核分配页机制	20210922 剖析Linux内核入门必背篇
20210318 剖析Linux内核锁及进程间通信	20210708 剖析Linux内核高级路由选择	20210923 剖析Linux内核缓存及中断处理
20210326 剖析Linux内核缓存和刷新机制	20210722 剖析Linux内核蓝牙系统架构	20210925 剖析Linux内核异常与中断处理
20210331 剖析Linux内核时钟机制及调度算法	20210809 剖析Linux内核CPU缓存技术	20210927 剖析Linux内核中断后部处理机制
20210407 剖析Linux内核地址映射机制	20210815 剖析Linux内核内存回收	20210930 剖析Qt开发入门必背第一讲
20210413 剖析Linux内核源码数据同步	20210818 剖析Linux内核页表缓存	20211004 剖析Linux内核后半部处理机制
20210417 剖析Linux内核四大核心框架	20210820 剖析Linux内核MMU机制	20211006 剖析Linux内核进程调度与切换
20210507 剖析Linux内核源码中断机制--	20210823 90分钟搞定DPDK技术精髓	20211009 剖析Qt跨平台GUI原理机制
20210514 剖析Linux内核并发与同步	20210901 剖析Linux内核网络协议栈	课程配套资料 @Linux嵌入式
20210517 剖析Linux内核MMU机制	20210905 剖析容器配手Kubernetes设计架构	

资源免费领

全网最详Linux内核技术解析【附视频教程和源码资料】...

docs.qq.com/doc/DS25VznBWUXRbHBM

学习直通车

Linux内核源码/内存调优/文件系统/进程管理/设备驱动/网络协议栈-学习视频教程-腾讯课堂...

ke.qq.com/course/4032547?flowToken=1042207

三.内核信息获取

可使用命令 `uname` 进行内核信息获取，格式和选项如下：

```
uname: - print system information
命令格式: uname [OPTION]...
-r: 内核的release号，发行号
-n: 主机名
-v: 编译版本号
-a: 显示所有信息
```

```
[root@LeeMumu ~]# uname -r
3.10.0-957.21.3.el7.x86_64
[root@LeeMumu ~]# uname -n
LeeMumu
[root@LeeMumu ~]# uname -v
#1 SMP Tue Jun 18 16:35:19 UTC 2019
[root@LeeMumu ~]# uname -a
Linux LeeMumu 3.10.0-957.21.3.el7.x86_64 #1 SMP Tue Jun 18 16:35:19 UTC
2019 x86_64 x86_64 x86_64 GNU/Linux
```

四.内核模块

内核模块信息的获取和管理有如下命令：`lsmod`、`modinfo`、`modprobe`、`depmod`、`insmod` 和 `rmmod` 等。

1、lsmod

```
lsmod命令 - Show the status of modules in the Linux Kernel
# 显示的内核模块信息来自于 /proc/modules 文件
# Module      模块名称
# Size        模块大小
# Used by     被引用的次数
```

```
[root@LeeMumu ~]# lsmod
Module                Size  Used by
ip6t_rpfilter         12595  1
ipt_REJECT             12541  2
nf_reject_ipv4        13373  1 ipt_REJECT
ip6t_REJECT           12625  2
... ..
dm_log                18411  2 dm_region_hash,dm_mirror
dm_mod               124461  11 dm_log,dm_mirror
```

2、modinfo

显示模块的具体文件信息，主要是读取 `/lib/modules/kernel_VERSION/` 目录下的文件来获取信息的，即使是未安装的模块，也能显示其信息。

```
modinfo - Show information about a Linux Kernel module
```

命令格式如下：

```
modinfo [-F field] [-k kernel] [modulename | filename...]
-F field      # 仅显示指定字段的信息
-n           # 显示文件路径
```

内核模块的信息如下，各个选项含义如下：

```
[root@LeeMumu ~]# modinfo ext4
filename:          /lib/modules/3.10.0-327.el7.x86_64/kernel/fs/ext4/ext4.ko # 3
license:          GPL # 4
description:      Fourth Extended Filesystem # 5
author:          Remy Card, ... .. and others # 6
alias:           fs-ext4 # 7
alias:           ext3 # 8
alias:           fs-ext3 # 9
alias:           ext2 # 10
alias:           fs-ext2 # 11
rhelversion:     7.2 # 12
srcversion:      DB48BDADD011DE28724EB21
depends:          mbcache,jbd2 # 13
intree:         Y
vermagic:        3.10.0-327.el7.x86_64 SMP mod_unload modversions
signer:          CentOS Linux kernel signing key # 14
sig_key:         79:AD:.. .:94:29:6A:B3
sig_hashalgo:    sha256 # 15
```

3、modprobe

模块管理命令，可自动处理可载入模块，用于智能地向内核中加载模块或者从内核中移除模块

modprobe 可载入指定的个别模块，或是载入一组相依的模块。modprobe 会根据depmod 所产生的相依关系，决定要载入哪些模块。若在载入过程中发生错误，在 modprobe 会卸载整组的模块。

modprobe - Add and remove modules from the Linux Kernel

选项：

- a 或 --all: 载入全部的模块
- c 或 --show-conf: 显示所有模块的设置信息
- d 或 --debug: 使用排错模式
- r 或 --remove: 模块闲置不用时，即自动卸载模块
- t 或 --type: 指定模块类型
- v 或 --verbose: 执行时显示详细的信息
- V 或 --version: 显示版本信息
- help: 显示帮助

命令格式：modprobe [-r] module_name

模块的动态装载：modprobe module_name

模块的动态卸载：modprobe -r module_name

对正在使用的模块，千万不要进行卸载

```
[root@LeeMumu ~]# lsmod | grep pp
iTCO_vendor_support 13718 1 iTCO_wdt
[root@LeeMumu ~]# modprobe ppdev
[root@LeeMumu ~]# lsmod | grep pp
ppdev 17671 0
iTCO_vendor_support 13718 1 iTCO_wdt
parport 46395 2 ppdev,parport_pc
[root@LeeMumu ~]# modprobe -r ppdev
[root@LeeMumu ~]# lsmod | grep pp
iTCO_vendor_support 13718 1 iTCO_wdt
```

4、depmod

内核模块依赖关系文件的生成工具，可产生模块依赖的映射文件，在构建嵌入式系统时，需要由这个命令来生成相应的文件，由 modprobe 使用。

```
depmod - Generate modules.dep and map files.
```

5、insmod 和 rmmod

模块的装载和卸载的另一组命令。

与 modprobe 不同，在运行 insmod 时，此处需要指明完整的模块文件的文件路径，可用 modinfo 获取其文件路径，但是 insmod 不会自动安装依赖的模块文件，需要自己使用 modinfo 获取其依赖文件，然后进行安装。

```
insmod命令：
insmod [filename] [module options...]
      filename: 模块文件的文件路径
[root@LeeMumu ~]# modinfo -n btrfs
/lib/modules/3.10.0-327.el7.x86_64/kernel/fs/btrfs/btrfs.ko

也可以直接使用命令调用文件路径就行
[root@LeeMumu ~]# insmod 'modinfo -n btrfs'

rmmod命令：
rmmod [module_name] 直接跟模块名称，不需要文件名称
```

五.ramdisk 文件管理

ramdisk 是内核中的特性之一，使用缓冲和缓存来加速对磁盘上的文件访问，并加载相应的硬件驱动。

```
ramdisk --> ramfs 提高速度
CentOS 5: initrd
           工具程序: mkinitrd
CentOS 6, 7: initramfs
           工具程序: mkinitrd, dracut
```

1、mkinitrd

为当前使用中的内核重新制作ramdisk文件。

```
# mkinitrd [OPTION...] [<initrd-image>] <kernel-version>
--with=<module>: 除了默认的模块之外需要装载至initramfs中的模块；
--preload=<module>: initramfs所提供的模块需要预先装载的模块；

注意：
# <kernel-version>: 不指定内核版本也可以，此命令会去探测现在系统中正在运行的内核版本
# 注意区分内核的发行版本和内核版本

示例：
~]# mkinitrd /boot/initramfs-$(uname -r).img $(uname -r)
```

2、dracut

比较底层的创建 ramdisk 的命令。

```
dracut - low-level tool for generating an initramfs image
命令格式：
# dracut [OPTION...] [<image> [<kernel version>]]

示例：
# dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

六.Linux 提供的伪文件系统

1、简介

Linux 内核提供了一种通过 /proc 文件系统，在运行时访问内核内部数据结构、改变内核设置的机制。proc 文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。

用户和应用程序可以通过 /proc 得到系统的信息，并可以改变内核的某些参数。由于系统的信息，如进程，是动态改变的，所以用户或应用程序读取 /proc 文件时，/proc 文件系统是动态从系统内核读出所需信息并提交的。

下面列出的这些文件或子文件夹，并不是都在你的系统中存在，这取决于你的内核配置和装载的模块。另外，在 /proc 下还有三个很重要的目录：net，scsi 和 sys。sys 目录是可写的，可以通过它来访问或修改内核的参数，而 net 和 scsi 则依赖于内核配置。例如，如果系统不支持scsi，则 scsi 目录不存在。

除了以上介绍的这些，还有的是一些以数字命名的目录，它们是进程目录。系统中当前运行的每一个进程都有对应的一个目录在 /proc 下，以进程的 PID 号为目录名，它们是读取进程信息的接口。而 self 目录则是读取进程本身的信息接口，是一个 link。

2、/proc

/proc 是内核状态和统计信息的输出接口，同时，还提供一个配置接口 /proc/sys。

目录内有两种类型的文件：

- 只读：信息输出；例如 /proc/#/* 输出某进程的相关信息
- 可写：可接受用户指定一个“新值”来实现对内核某功能或特性的配置，如 /proc/sys/ 目录。但是不可以直接修改，需要修改内核参数的配置文件来进行修改并重载（或重启），重载后，此参数就会发生变化。

```
/proc/sys/net/ipv4/ip_forward    相当于    net.ipv4.ip_forward
/proc/sys/kernel/hostname        相当于    kernel.hostname
```

可通过以下三种方式对内核配置进行更改：

- 使用 sysctl 命令：只针对当前运行内核有效，重启后失效
- 使用文件系统命令：如 echo、cat 等，只针对当前运行内核有效，重启后失效
- 修改相应配置文件：生效后一直有效

下面针对于这三张方式进行详细配置：

1)、sysctl

sysctl 命令专用于查看或设定 /proc/sys 目录下参数的值，能修改的就是 /proc/sys 目录下的参数。

```
sysctl - configure kernel parameters at runtime
```

命令格式：

```
sysctl [options] [variable[=value]]
```

查看：

```
# sysctl -a                # 查看所有的内核参数
# sysctl variable          # 查看指定变量的值
```

修改：

```
# sysctl -w variable=value # 等号两边不能有空格
```

示例：

```
# 通过以下命令可知道一共有1004个内核参数
[root@LeeMumu ~]# sysctl -a | wc -l
1004

修改内核转发参数：
[root@LeeMumu sys]# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
[root@LeeMumu sys]# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[root@LeeMumu sys]# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
[root@LeeMumu ~]# cat /proc/sys/net/ipv4/ip_forward
1
```

修改主机名称：

```
[root@jc ~]# uname -n
jc
[root@jc ~]# sysctl kernel.hostname
kernel.hostname = jc
[root@jc ~]# sysctl -w kernel.hostname=LeeMumu
kernel.hostname = LeeMumu
[root@jc ~]# sysctl kernel.hostname
kernel.hostname = LeeMumu
[root@jc ~]# uname -n
LeeMumu
[root@jc ~]# cat /proc/sys/kernel/hostname
LeeMumu
```

2)、文件系统命令

- 查看系统内核参数命令：cat
- 设定系统内核参数命令：echo

查看：

```
# cat /proc/sys/PATH/TO/SOME_KERNEL_FILE
```

设定：

```
# echo "VALUE" > /proc/sys/PATH/TO/SOME_KERNEL_FILE
```

示例：

```
[root@LeeMumu ipv4]# cat /proc/sys/net/ipv4/ip_forward
0
[root@LeeMumu ipv4]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@LeeMumu ipv4]# cat /proc/sys/net/ipv4/ip_forward
1
```

3)、配置文件

内核参数的配置文件路径，如下：

```
/etc/sysctl.conf
/etc/sysctl.d/*.conf
```

对这两个文件中的任何一个文件进行编辑，另外一个也会同时进行改变。如下：

```
[root@LeeMumu ~]# vi /etc/sysctl.conf
[root@LeeMumu ~]# grep "kernel.hostname*" /etc/sysctl.conf
kernel.hostname=LEEMUMU
[root@LeeMumu ~]# grep "kernel.hostname*" /etc/sysctl.d/99-sysctl.conf
kernel.hostname=LEEMUMU
```

修改配置后，如果让其立即生效，有如下两种方法：

```
# 重启系统，让内核重读配置文件
# 使用命令让其立即生效
sysctl -p [/PATH/TO/CONFIG_FILE]
```

修改主机名称

```
[root@LeeMumu ~]# uname -n                # 系统现有的主机名称，使用了3种方法查看
LeeMumu
[root@LeeMumu ~]# sysctl kernel.hostname
kernel.hostname = LeeMumu
[root@LeeMumu ~]# cat /proc/sys/kernel/hostname
LeeMumu
[root@LeeMumu ~]# vi /etc/sysctl.conf      # 对内核配置文件进行参数设定
[root@LeeMumu ~]# grep "kernel.hostname*" /etc/sysctl.conf
kernel.hostname=LEEMUMU
[root@LeeMumu ~]# sysctl -p                # 应用内核配置文件
kernel.hostname = LEEMUMU
[root@LeeMumu ~]# uname -n                # 确认配置已经生效
LEEMUMU
[root@LeeMumu ~]# sysctl kernel.hostname
kernel.hostname = LEEMUMU
[root@LeeMumu ~]# cat /proc/sys/kernel/hostname
LEEMUMU
```

开启转发功能

```
[root@neo ~]# vim /etc/sysctl.conf
net.ipv4.ip_forward = 1
[root@neo ~]# sysctl -p                    # 重读配置文件
net.ipv4.ip_forward = 1
[root@neo ~]# cat /proc/sys/net/ipv4/ip_forward
```

4、内核参数对应关系

内核配置文件（/etc/sysctl.conf、/etc/sysctl.d/*.conf）的相关参数名称与 /proc/sys 的目录路径的对应关系，如下：

```
ipv4转发参数：
net.ipv4.ip_forward          /proc/sys/net/ipv4/ip_forward
清除cache：
vm.drop_caches               /proc/sys/vm/drop_caches
主机名称：
kernel.hostname              /proc/sys/kernel/hostname
忽略所有ping操作：
net.ipv4.icmp_echo_ignore_all /proc/sys/net/ipv4/icmp_echo_ignore_all
```

5、内核常见参数说明

```
# 内核panic时，1秒后自动重启
kernel.panic = 1

# 允许更多的PIDs（减少滚动翻转问题）；may break some programs 32768
kernel.pid_max = 32768

# 内核所允许的最大共享内存段的大小（bytes）
kernel.shmmax = 4294967296

# 在任何给定时刻，系统上可以使用的共享内存的总量（pages）
kernel.shmall = 1073741824

# 设定程序core时生成的文件名格式
```



```
kernel.core_pattern = core_%e

# 当发生oom时, 自动转换为panic
vm.panic_on_oom = 1

# 表示强制Linux VM最低保留多少空闲内存 (Kbytes)
vm.min_free_kbytes = 1048576

# 该值高于100, 则将导致内核倾向于回收directory和inode cache
vm.vfs_cache_pressure = 250

# 表示系统进行交换行为的程度, 数值 (0-100) 越高, 越可能发生磁盘交换
vm.swappiness = 0

# 仅用10%做为系统cache
vm.dirty_ratio = 10

# 增加系统文件描述符限制 2^20-1
fs.file-max = 1048575

# 网络层优化
# listen() 的默认参数, 挂起请求的最大数量, 默认128
net.core.somaxconn = 1024

# 增加Linux自动调整TCP缓冲区限制
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

# 进入包的最大设备队列. 默认是300
net.core.netdev_max_backlog = 2000

# 开启SYN洪水攻击保护
net.ipv4.tcp_syncookies = 1

# 开启并记录欺骗, 源路由和重定向包
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# 处理无源路由的包
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0

# 开启反向路径过滤
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# 确保无人能修改路由表
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0

# 增加系统IP端口限制
net.ipv4.ip_local_port_range = 9000 65533

# TTL设置
net.ipv4.ip_default_ttl = 64

# 增加TCP最大缓冲区大小
net.ipv4.tcp_rmem = 4096 87380 8388608
net.ipv4.tcp_wmem = 4096 32768 8388608

# Tcp自动窗口
net.ipv4.tcp_window_scaling = 1

# 进入SYN包的最大请求队列. 默认1024
```

```
net.ipv4.tcp_max_syn_backlog = 8192

# 打开TIME-WAIT套接字重用功能, 对于存在大量连接的Web服务器非常有效。
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 0

# 表示是否启用以一种比超时重发更精确的方法(请参阅 RFC 1323)来启用对 RTT 的计算; 为了实现更
net.ipv4.tcp_timestamps = 0

# 表示本机向外发起TCP SYN连接超时重传的次数
net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_synack_retries = 2

# 减少处于FIN-WAIT-2连接状态的时间, 使系统可以处理更多的连接。
net.ipv4.tcp_fin_timeout = 10

# 减少TCP KeepAlive连接侦测的时间, 使系统可以处理更多的连接。
# 如果某个TCP连接在idle 300秒后, 内核才发起probe. 如果probe 2次(每次2秒)不成功, 内核才彻底放弃
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_keepalive_probes = 2
net.ipv4.tcp_keepalive_intvl = 2

# 系统所能处理不属于任何进程的TCP sockets最大数量
net.ipv4.tcp_max_orphans = 262144

# 系统同时保持TIME_WAIT套接字的最大数量, 如果超过这个数字, TIME_WAIT套接字将立刻被清除并打印
net.ipv4.tcp_max_tw_buckets = 20000

# arp_table的缓存限制优化
net.ipv4.neigh.default.gc_thresh1 = 128
net.ipv4.neigh.default.gc_thresh2 = 512
net.ipv4.neigh.default.gc_thresh3 = 4096
```

七、/proc/sys

/proc/sys 目录下的几个重要目录的含义及作用:

- sysfs: 输出内核识别出的各硬件设备的相关属性信息，也有内核对硬件特性的可设置参数；对这些参数的修改，即可定制硬件设备工作特性
- udev: 通过读取 /sys 目录下的硬件设备信息按需为各硬件设备创建设备文件；udev 是用户空间程序，不能直接读取硬盘文件；专用工具：devadmin, hotplug
- udev 为设备创建设备文件时，会读取其事先定义好的规则文件，一般在 /etc/udev/rules.d/ 目录下，以及 /usr/lib/udev/rules.d/ 目录下

八、如何通过/proc查看top中展示的进程状态

Linux系统上的/proc目录是一种文件系统，即proc文件系统。与其它常见的文件系统不同的是，/proc是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息，甚至可以通过更改其中某些文件来改变内核的运行状态。这个目录中包含每个进程的状态信息。

可通过创建死循环进行执行，然后来进行进程状态查看：

```
# 死循环脚本
#!/bin/bash
#
while true
do
echo "马哥教育「官网」-专业Linux云计算运维、SRE、Devops、容器云、Python、Go开发培训机构!"
done

# top命令查看脚本运行状态
```

```

11885 root      20   0  113180   1212   1028 R 100.0   0.0  10:25.56 sh

# 在/proc目录下查看进程状态
[root@node01 11885]# pwd
/proc/11885      #进程目录
[root@node01 11885]# ls
attr          clear_refs    cpuset        fd            limits        mem           net
autogroup     cmdline       cwd           fdinfo        loginuid      mountinfo     ns
auxv          comm          environ       gid_map       map_files     mounts        numa_maps
cgroup        coredump_filter exe           io            maps          mountstats    oom_adj
[root@node01 11885]#
# 各文件表示的含义
cmdline - 启动当前进程的完整命令，但僵尸进程目录中的此文件不包含任何信息
[root@node01 11885]# more cmdline
sh
cwd - 指向当前进程运行目录的一个符号链接
[root@node01 11885]# ls -l cwd
lrwxrwxrwx. 1 root root 0 1月 21 20:21 cwd -> /root/scripts
environ - 当前进程的环境变量列表，彼此间用空字符（NULL）隔开；变量用大写字母表示，其值用小写字母表示
exe - 指向启动当前进程的可执行文件（完整路径）的符号链接，通过/proc/N/exe可以启动当前进程的副本
fd - 这是个目录，包含当前进程打开的每一个文件的文件描述符（file descriptor），这些文件描述符的列表在/proc/N/fd/目录下
[root@node01 11885]# cd fd
[root@node01 fd]# ls -l
总用量 0
lrwx-----. 1 root root 64 1月 21 20:36 0 -> /dev/pts/0
lrwx-----. 1 root root 64 1月 21 20:36 1 -> /dev/pts/0
lrwx-----. 1 root root 64 1月 21 20:36 2 -> /dev/pts/0
lr-x-----. 1 root root 64 1月 21 20:36 255 -> /root/scripts/while.sh
limits - 当前进程所使用的每一个受限资源的软限制、硬限制和管理单元；此文件仅可由实际启动当前进程的进程访问
mem - 当前进程所占用的内存空间，由open、read和lseek等系统调用使用，不能被用户读取
stat - 当前进程的状态信息，包含一系统格式化后的数据列，可读性差，通常由ps命令使用
statm - 当前进程占用内存的状态信息，通常以“页面”（page）表示
status - 与stat所提供信息类似，但可读性较好，如下所示，每行表示一个属性信息；其详细介绍请参考man 8 status
[root@node01 11885]# more status
Name:   sh
Umask:  0022
State:  R (running)
Tgid:   11885
Ngid:   0
Pid:    11885
PPid:   7573
TracerPid: 0
Uid:    0   0   0   0
Gid:    0   0   0   0
FDSize: 256
Groups: 0
VmPeak: 113180 kB
VmSize: 113180 kB
VmLck:   0 kB
VmPin:   0 kB
VmHWM:   1212 kB
VmRSS:   1212 kB
RssAnon:  184 kB
RssFile:  1028 kB
RssShmem:   0 kB
VmData:   208 kB
VmStk:    132 kB
VmExe:    884 kB
VmLib:    2092 kB
VmPTE:     60 kB
VmSwap:    0 kB
Threads:  1
SigQ:    0/15077
SigPnd:  0000000000000000
ShdPnd:  0000000000000000
SigBlk:  0000000000000000
SigIgn:  0000000000000004
SigCgt:  000000000010000

```

```
CapInh: 0000000000000000
CapPrm: 0000001fffffffff
CapEff: 0000001fffffffff
CapBnd: 0000001fffffffff
CapAmb: 0000000000000000
Seccomp: 0
Cpus_allowed: 3
Cpus_allowed_list: 0-1
Mems_allowed: 00000000,00000000,00000000,00000000,00000000,00000000,00000000,
000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,0
Mems_allowed_list: 0
voluntary_ctxt_switches: 1
nonvoluntary_ctxt_switches: 152160
```

编辑于 2022-03-18 14:52

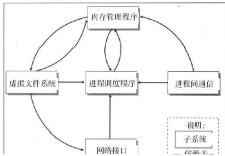
LinuxLinux 内核操作系统内核

写下你的评论...



还没有评论，发表第一个评论吧

推荐阅读



Linux内核系统篇——内核的5个重要子系统？

极致Linux内核



如何入门Linux内核？

量子孤岛发表于Linux...



Linux内核系统由哪些部分组成的

极致Linux内核

了解Linux内核的5

首先一张熟悉的图来该的基本体系结构：体用户（或应用程序）用户应用程序执行的地之下是内核空间，Lin于这里。Linux 内核可Linux内核园

