**03-文件模块**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **所有者** | **修改时间** | **创建时间** |
| [04-驱动模块](https://cqbk0n8o1h.feishu.cn/wiki/U5siwACK0iAzW3k8cUrcwrdznac) | 李2017 | 2024-05-14 08:51 | 2024-05-14 08:49 |
| [05-磁盘操作](https://cqbk0n8o1h.feishu.cn/wiki/SSG5wYExmitY4RkDHiRc0NI9nmc) | 李2017 | 2024-05-14 08:50 | 2024-05-14 08:49 |

**文件模块**

inotifytools

1. **系统文件**

1.1 **linux目录**

|  |  |
| --- | --- |
| 目录 | 目录放置的内容 |
| bin | 存放系统命令的目录，如命令 cat， cp， mkdir |
| boot | 存放开机启动过程所需的内容，如开机管理程序 grub2 |
| dev | 所有设备文件的目录（如声卡、硬盘、光驱） |
| etc | 系统的主要配置文件 |
| home | 用户家目录数据的存放目录 |
| lib | 存放 sbin 和 bin 目录下命令所需的库文件 |
| lib32/lib64 | 存放二进制函数库，支持 32 位/64 位 |
| lost+fond | 在 EXT3/4 系统中，当系统意外崩溃或意外关机时，会产生一些碎片文件在这个目录  下面，系统启动 fcsk 工具会检查这个目录，并修复已损坏的文件。 |
| media | 用于挂载光盘，软盘和 DVD 等设备 |
| mnt | 同 media 作用一样，用于临时挂载存储设备 |
| opt | 第三方软件安装存放目录。 |
| proc | 进程及内核信息存放目录，不占用硬盘空间。 |
| root | root 用户的家目录 |
| run | 是一个临时文件系统，存储系统启动以来的信息。当系统重启时，这个目录下的文件  应该被删掉或清除。 |
| sbin | root 用户使用的命令存放目录 |
| srv | 一些网络服务所需要的数据文件 |
| sys | 同 proc 目录，用于记录 CPU 与系统硬件的相关信息 |
| tmp | 程序运行时产生的临时文件存放目录 |
| usr | 系统存放程序的目录，类似于在 windows 下的文件夹 programefiles |
| var | 存放内容常变动的文件目录，如系统日志文件 |

根目录

/sys： sysfs 文件系统，用于在用户空间访问内核信息。

/proc： proc 文件系统，提供有关进程和系统状态的信息。

/dev： devtmpfs 文件系统，用于管理设备节点。

/dev/pts ：devpts 文件系统，提供伪终端设备的支持。

/run： tmpfs 文件系统，用于存储临时文件和运行时数据。

/： ext4 文件系统，根文件系统。

/sys/kernel/security： securityfs 文件系统，用于内核安全模块。

/dev/shm： tmpfs 文件系统，用于共享内存段。

/run/lock： tmpfs 文件系统，用于存储锁文件。

/sys/fs/cgroup： tmpfs 文件系统，用于控制组（cgroup）层次结构。

/sys/fs/cgroup/unified： cgroup2 文件系统，用于统一的 cgroup 实现。

/sys/fs/cgroup/systemd： cgroup 文件系统，用于 Systemd cgroup 实现。

/sys/fs/pstore： pstore 文件系统，用于内核存储错误信息。

/sys/fs/cgroup/perf\_event： cgroup 文件系统，用于 perf\_event cgroup 实现。

/sys/fs/cgroup/cpu,cpuacct： cgroup 文件系统，用于 CPU 和 CPUACCT cgroup 实现。

/sys/fs/cgroup/freezer： cgroup 文件系统，用于冻结进程的 cgroup 实现。

/sys/fs/cgroup/cpuset： cgroup 文件系统，用于 cpuset cgroup 实现。

/sys/fs/cgroup/rdma： cgroup 文件系统，用于 RDMA cgroup 实现。

/sys/fs/cgroup/blkio： cgroup 文件系统，用于块 I/O cgroup 实现。

/sys/fs/cgroup/hugetlb： cgroup 文件系统，用于巨页 cgroup 实现。

/sys/fs/cgroup/devices： cgroup 文件系统，用于设备 cgroup 实现。

/sys/fs/cgroup/net\_cls,net\_prio： cgroup 文件系统，用于网络类和网络优先级 cgroup 实现。

/sys/fs/cgroup/memory： cgroup 文件系统，用于内存 cgroup 实现。

/sys/fs/cgroup/pids： cgroup 文件系统，用于进程 ID cgroup 实现。

/proc/sys/fs/binfmt\_misc： binfmt\_misc 文件系统，用于在 Linux 内核中注册二进制格式的解释器。

/dev/hugepages： hugetlbfs 文件系统，用于巨页的支持。

/sys/kernel/debug： debugfs 文件系统，用于内核调试接口。

/dev/mqueue： mqueue 文件系统，用于消息队列。

/sys/kernel/config： configfs 文件系统，用于内核配置项。

/sys/fs/fuse/connections： fusectl 文件系统，用于控制 FUSE 文件系统连接。

/snap： 各个 Snap 包的 squashfs 文件系统挂载点。

/mnt/hgfs： vmhgfs-fuse 文件系统，用于 VMware 共享文件夹的支持。

/run/vmblock-fuse： vmware-vmblock 文件系统，用于 VMware vmblock 的支持。

/run/user/1000： tmpfs 文件系统，用于用户运行时数据。

/run/user/1000/gvfs： fuse.gvfsd-fuse 文件系统，用于 GVFS 的支持。

1.2 **ext文件**

|  |
| --- |
| C++ e2fsck命令 |

* -p：自动修复文件系统，不询问用户。
* -y：自动回答“是”（yes）所有问题。
* -c：检查坏道，并标记不良扇区。

2. **虚拟文件**

procfs，debugfs和sysfs

2.1 **procfs**

文件系统是一个伪文件系统，它为内核数据结构提供接口。

最初就是用来跟内核交互的唯一方式，用来获取cpu、内存、设备驱动、进程等各种信息，proc下文件基本都是只读的。

/proc/cmdline 启动时传递给kernel的参数信息（就是bootargs信息）

/proc/cpuinfo cpu的信息

/proc/crypto 内核使用的所有已安装的加密密码及细节

/proc/devices 已经加载的设备并分类

/proc/dma 已注册使用的ISA DMA频道列表

/proc/execdomains Linux 内核当前支持的execution domains

/proc/fb 帧缓冲设备列表，包括数量和控制它的驱动

/proc/filesystems 内核当前支持的文件系统类型

/proc/interrupts x86架构中的每个IRQ中断数

/proc/iomem 每个物理设备当前在系统内存中的映射

/proc/ioports 一个设备的输入输出所使用的注册端口范围

/proc/kcore 代表系统的物理内存，存储为核心文件格式，里边显示的是字节数，等于RAM大小加上4kb

/proc/kmsg 记录内核生成的信息，可以通过/sbin/klogd或/bin/dmesg来处理

/proc/loadavg 根据过去一段时间内CPU和IO的状态得出的负载状态，与uptime命令有关

/proc/locks 内核锁住的文件列表

/proc/mdstat 多硬盘，RAID配置信息(md=multiple disks)

/proc/meminfo RAM使用的相关信息

/proc/misc 其他的主要设备(设备号为10)上注册的驱动

/proc/modules 所有加载到内核的模块列表

/proc/mounts 系统中使用的所有挂载

/proc/partitions 分区中的块分配信息

/proc/pci 系统中的PCI设备列表

/proc/slabinfo 系统中所有活动的 slab 缓存信息

/proc/stat 所有的CPU活动信息

/proc/uptime 系统已经运行了多久

/proc/swaps 交换空间的使用情况

/proc/version Linux内核版本和gcc版本

/proc/bus 系统总线(Bus)信息，例如pci/usb等

/proc/driver 驱动信息

/proc/fs 文件系统信息

/proc/ide ide设备信息

/proc/irq 中断请求设备信息

/proc/net 网卡设备信息

/proc/scsi scsi设备信息

/proc/tty tty设备信息

/proc/net/dev 显示网络适配器及统计信息

/proc/vmstat 虚拟内存统计信息

/proc/vmcore 内核panic时的内存映像

/proc/diskstats 取得磁盘信息

/proc/schedstat kernel调度器的统计信息

/proc/zoneinfo 显示内存空间的统计信息，对分析虚拟内存行为很有用

/proc/self 链接到当前正在运行的进程

以下是/proc目录中进程pid的信息

/proc/pid/cmdline 进程启动命令

/proc/pid/cwd 链接到进程当前工作目录

/proc/pid/environ 进程环境变量列表

/proc/pid/exe 链接到进程的执行命令文件

/proc/pid/fd 包含进程相关的所有的文件描述符 （ls /proc/pid/fd | wc -l 查看某个进程打开多少FD）

/proc/pid/maps 与进程相关的内存映射信息

/proc/pid/mem 指代进程持有的内存，不可读

/proc/pid/root 链接到进程的根目录

/proc/pid/stat 进程的状态

/proc/pid/statm 进程使用的内存的状态

/proc/pid/status 进程状态信息，比stat/statm更具可读性

2.2 **sysfs**

导出内核对象，挂载在/sys目录，为设备驱动服务的

/sys/block

代表着系统中当前被发现的所有块设备

/sys/bus

一般来说每个子目录(总线类型)下包含两个子目录，一个是devices，另一个是drivers；其中devices下是这个总线类型下的所有设备，这些设备都是符号链接，它们分别指向真正的设备(/sys/devices/…下)；而drivers下是所有注册在这个总线上的驱动，每个driver子目录下 是一些可以观察和修改的driver参数。 (它也是构成linux统一设备模型的一部分)

/sys/class

该目录下包含所有注册在kernel里面的设备类型，每个设备类型表达具有一种功能的设备。每个设备类型子目录下是具体设备的符号链接，这些链接指向/sys/devices/…下的具体设备。设备类型和设备并没有一一对应的关系，一个物理设备可能具备多种设备类型；一个设备类型只表达具有一种功能的设备，比如：系统所有输入设备都会出现在/sys/class/input之下，而不论它们是以何种总线连接到系统的。(/sys/class也是构成linux统一设备模型的一部分)

/sys/dev

该目录下有字符设备(block)和块设备(char)两个子目录，里面全是以主次设备号(major:minor)命名的链接文件，链接到/sys/devices

/sys/devices

该目录下是全局设备结构体系，包含所有被发现的注册在各种总线上的各种物理设备

/sys/firmware

该目录下包含对固件对象(firmware object)和属性进行操作和观察的接口，即这里是系统加载固件机制的对用户空间的接口.(关于固件有专用于固件加载的一套API)

/sys/fs

该目录用来描述系统中所有的文件系统，包括文件系统本身和按照文件系统分类存放的已挂载点。

/sys/kernel

该目录下存放的是内核中所有可调整的参数

/sys/module

该目录包含所有被载入Kernel的模块，无论这些模块是以内联(inlined)方式编译到内核映像文件中还是编译为外模块(.ko文件)

/sys/power

该目录下有几个属性文件可以用于控制整个机器的电源状态，如向其中写入控制命令让机器关机/重启等等。

2.3 **debugfs**

它是内核开发人员向用户空间提供信息的一种简单方法。

在调试linux驱动的时候，可以用debugfs来调试，debugfs类似字符设备驱动一样，甚至更简单，不需要主设备号次设备号等等，只需要实现一个file\_operations，然后通过debugfs\_create\_file就可以在debugfs中建立一个文件结点，就像字符设备驱动那样，只需要对这个文件结点进行open就可以进行read、write、ioctl，等等操作，这些操作对应到我们在驱动里为debugfs准备的file\_operations。

内核支持DEBUGFS，使能宏CONFIG\_DEBUG\_FS，在中内核配置

3. **日志管理**

3.1 **syslog日志**

**配置**

可能在 /usr/adm 或 /var/log目录

通过 /etc/syslog.conf文件或者 /etc/syslog-ng/syslog-ng.conf文件检查配置

用户尝试登录失败记录

/var/log/auth.log

**内容**

/var/log/syslog：这是系统的主要日志文件，记录了系统各个组件和服务的消息。

/var/log/messages：类似于 syslog 文件，用于记录系统的各种消息。

/var/log/auth.log：记录与认证和授权相关的消息，例如用户登录和权限变更。

/var/log/kern.log：记录内核级别的消息，包括内核模块加载、硬件故障等。

/var/log/cron.log：记录定时任务（cron jobs）的执行情况。

/var/log/mail.log：记录邮件系统的日志消息。

4. **inotify**

inotify 是 Linux 内核提供的一种文件系统事件监测机制，它可以用来监测文件系统的变化，包括文件或目录的创建、删除、修改、移动等操作，以及对文件的读取权限。inotify 可以让用户空间程序实时地监测和响应这些文件系统事件。

inotify 提供了一组系统调用和一个文件描述符，用于监听一个或多个文件或目录。它使用文件描述符来通知用户空间程序关心的事件。当文件或目录上发生关心的事件时，内核会将相关信息写入文件描述符，用户空间程序可以读取这些信息来获取发生的事件以及相关的文件信息。

inotify 的主要优点包括：

实时性：inotify 可以实时监测文件系统的变化，不需要用户空间程序进行轮询操作，从而大大降低了资源消耗。

灵活性：用户可以选择监听单个文件、单个目录，或者多个文件和目录的事件。

可扩展性：inotify 可以同时监测大量文件和目录，适用于处理大规模的文件系统监测需求。

inotify 的基本工作原理如下：

用户空间程序使用 inotify\_init() 系统调用创建一个 inotify 实例，获取一个文件描述符。

用户空间程序使用 inotify\_add\_watch() 系统调用将文件或目录添加到 inotify 实例中，指定要监听的事件类型。

内核监测到文件系统事件发生时，将相关信息写入到相应的文件描述符中。

用户空间程序通过读取文件描述符来获取事件信息，并根据事件类型进行相应的处理。

常见的 inotify 事件类型包括：

IN\_ACCESS：文件被访问（读取）。

IN\_MODIFY：文件被修改。

IN\_CREATE：文件或目录被创建。

IN\_DELETE：文件或目录被删除。

IN\_MOVE：文件或目录被移动或重命名。

IN\_OPEN：文件被打开。

...