1. proc 调试信息

在 Linux 操作系统中，/proc 文件系统是一个特殊的虚拟文件系统，提供了对系统内核和运行中进程的实时信息访问。本文将深入探讨 /proc 文件系统的作用、结构以及如何使用它来获取系统信息。

首先我们可以通过 man proc 直接查看相关介绍。

```
PROC(5)                                    Linux Programmer's Manual                                    PROC(5)

NAME
       proc - process information pseudo-filesystem

DESCRIPTION
       The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures.  It is commonly mounted at /proc.  Typically, it is
       mounted automatically by the system, but it can also be mounted manually using a command such as:

           mount -t proc proc /proc

       Most of the files in the proc filesystem are read-only, but some files are writable, allowing kernel variables to be changed.
```

1.1. proc 调试节点

在 Linux 操作系统中，/proc 目录中的调试节点（debugging nodes）提供了一种机制，允许开发人员访问和调试内核的运行时信息。这些调试节点可以用于观察内核的状态、执行跟踪和性能分析，以帮助解决问题和优化系统。下面介绍一些常见的 /proc 调试节点以及它们的作用：

```
[root@imx6ull:/]# ls /proc/
1            12           135          22           318          78           92           cpuinfo      keys         stat
10           121          136          23           321          79           93           crypto       kmsg         swaps
100          122          14           231          322          8            94           device-tree  kpagecount   sys
101          123          15           24           323          80           95           devices      kpageflags   sysrq-trigger
104          124          16           25           324          81           96           diskstats    loadavg      sysvipc
106          125          17           26           330          82           97           driver       locks        thread-self
107          126          170          269          448          83           98           execdomains  meminfo      timer_list
108          127          173          27           451          84           99           fb           misc         tty
11           128          18           273          457          85           asound       filesystems  modules      uptime
110          129          184          28           458          86           buddyinfo    fs           mounts       version
111          13           185          288          5            87           bus          interrupts   mtd          vmallocinfo
112          130          186          290          7            88           cgroups      iomem        net          vmstat
113          131          19           3            74           89           cmdline      ioports      pagetypeinfo zoneinfo
114          132          199          301          75           9            config.gz    irq          partitions
115          133          2            303          76           90           consoles     kallsyms     self
116          134          21           306          77           91           cpu          key-users    softirqs
```

1.1.1. CPU 信息查询

/proc/cpuinfo 该文件包含了当前系统 CPU 的参数信息，包括 CPU 的核心数量、每个核心的工作频率、缓存大小、字宽、地址线长度等。比如，在 imx6 开发板上，cat /proc/cpuinfo 显示的部分信息如下：

```
[root@imx6ull:/proc]# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 3.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : Freescale i.MX6 UltraLite (Device Tree)
Revision        : 0000
Serial          : 0000000000000000
```

- processor：指示处理器的编号，从 0 开始递增。
- model name：提供了关于 CPU 型号的更详细的信息，通常包括制造商、型号和一些特定的功能。
- bogomips：一个估算值，表示在一秒钟内 CPU 可以执行的虚拟指令数量。

1.1.2. 内存信息查询

/proc/meminfo 这个文件显示的是系统中当前的内存状态信息，如物理内存总容量、已使用内存、空闲内存、共享内存、交换内存大小，等等。

```
[root@imx6ull:/]# cat /proc/meminfo
MemTotal:         503392 kB
MemFree:          309032 kB
MemAvailable:     376528 kB
Buffers:            4216 kB
Cached:            67984 kB
SwapCached:            0 kB
Active:            44724 kB
Inactive:          53404 kB
Active(anon):      26744 kB
Inactive(anon):     1064 kB
Active(file):      17980 kB
Inactive(file):    52340 kB
Unevictable:           0 kB
Mlocked:               0 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         503392 kB
LowFree:          309032 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:         25952 kB
Mapped:            37536 kB
Shmem:              1880 kB
Slab:              12228 kB
SReclaimable:       4032 kB
SUnreclaim:         8196 kB
KernelStack:         840 kB
PageTables:          948 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      251696 kB
Committed_AS:      40956 kB
VmallocTotal:    1556480 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
CmaTotal:         327680 kB
CmaFree:          252872 kB
```

- MemTotal：显示系统中物理内存的总量。
- MemFree：显示当前未被使用的物理内存量。
- MemAvailable：显示近似的可用内存量，考虑了一些保留的内存和缓存。
- Buffers：显示用于缓存数据块的内存量。
- Cached：显示用于缓存文件数据的内存量。
- SwapCached：显示被交换到磁盘交换空间的内存中，仍然保留在缓存中的部分。
- Active：显示活跃的内存页面数量。
- Inactive：显示非活跃的内存页面数量。
- SwapTotal：显示交换空间的总量。
- SwapFree：显示当前可用于交换的空闲空间量。

- Dirty：显示等待被写回到磁盘的脏数据页的数量。
- Writeback：显示正在被写回到磁盘的脏数据页的数量。
- AnonPages：显示不属于任何文件的匿名内存页的数量。
- Mapped：显示映射到文件的内存页的数量。
- Shmem：显示共享内存的总量，包括 Tmpfs 文件系统中的共享内存。
- Slab：显示内核 Slab 分配器使用的内存量。
- SReclaimable：显示可回收的 Slab 内存的数量。
- SUnreclaim：显示不可回收的 Slab 内存的数量。
- KernelStack：显示内核栈的总内存量。
- PageTables：显示用于页表的内存量。

1.1.3. 符号表

该文件是一个符号表，包含了内核的所有全局变量和函数在内存中的地址。

```
[root@imx6ull:/]# cat /proc/kallsyms
80008000 T stext
80008000 T _text
8000808c t __create_page_tables
80008138 t __turn_mmu_on_loc
80008144 t __fixup_smp
800081ac t __fixup_smp_on_up
800081d0 t __fixup_pv_table
80008224 t __vet_atags
80100000 T _stext
80100000 T __turn_mmu_on
80100000 T __idmap_text_start
80100020 T cpu_resume_mmu
80100020 t __turn_mmu_on_end
80100044 T cpu_ca17_reset
80100044 T cpu_ca8_reset
80100044 T cpu_ca9mp_reset
80100044 T cpu_v7_reset
80100058 T __idmap_text_end
80101000 T asm_do_IRQ
80101000 T __exception_text_start
80101000 T __hyp_idmap_text_end
80101000 T __hyp_idmap_text_start
80101014 T do_undefinstr
80101214 T handle_fiq_as_nmi
801012c4 T do_IPI
801012c8 T do_DataAbort
80101384 T do_PrefetchAbort
80101424 t tzic_handle_irq
```

这是一个帮助内核开发者调试内核而加入的文件，在 Linux 系统崩溃时产生的 Opps 信息中，函数调用堆栈中显示出来的函数名，就是在这个文件的帮助下生成的。

1.1.4. 中断信息查询

该文件包含了系统记录的在每个 CPU 上处理的各类中断的计数信息。

```
[root@imx6ull:/]# cat /proc/interrupts
           CPU0
 16:     459829       GPC  55 Level     i.MX Timer Tick
 19:         22       GPC  33 Level     2010000.ecspi
 20:       1593       GPC  26 Level     2020000.serial
 21:          0       GPC  98 Level     sai
 22:          0       GPC  50 Level     2034000.asrc
 40:          0       GPC   4 Level     20cc000.snvs:snvs-powerkey
 41:     194588       GPC 120 Level     20b4000.ethernet
 42:          0       GPC 121 Level     20b4000.ethernet
 43:          0       GPC  80 Level     20bc000.wdog
 44:          0       GPC  49 Level     imx_thermal
 49:          0       GPC  19 Level     rtc alarm
 55:          0       GPC   2 Level     sdma
 60:          1       GPC  43 Level     2184000.usb
 61:       1605       GPC  42 Level     2184200.usb
 62:          0       GPC 118 Level     2188000.ethernet
 63:          0       GPC 119 Level     2188000.ethernet
 64:          0       GPC  22 Level     mmc0
 65:       4827       GPC  23 Level     mmc1
 66:          1       GPC 100 Level     2198000.adc
 67:          0       GPC  36 Level     21a0000.i2c
 68:        291       GPC  37 Level     21a4000.i2c
 70:          3       GPC   5 Level     21c8000.lcdif
 71:          0       GPC   8 Level     pxp-dmaengine-legacy
 72:          0       GPC  18 Level     pxp-dmaengine-std
 73:          0       GPC  28 Level     21ec000.serial
 74:          0       GPC  17 Level     21fc000.serial
 75:          0       GPC  46 Level     dcp-vmi-irq
 76:          0       GPC  47 Level     dcp-irq
 78:          2       GPC   6 Level     imx-rng
 80:          0  gpio-mxc   1 Edge      inv_mpu
 97:          0  gpio-mxc  18 Edge      SII902x_det
 98:          0  gpio-mxc  19 Edge      2190000.usdhc cd
189:          0  gpio-mxc  14 Edge      User2 Button
208:          0  gpio-mxc   1 Edge      User1 Button
IPI0:         0  CPU wakeup interrupts
IPI1:         0  Timer broadcast interrupts
IPI2:         0  Rescheduling interrupts
IPI3:         0  Function call interrupts
IPI4:         0  CPU stop interrupts
IPI5:    252364  IRQ work interrupts
IPI6:         0  completion interrupts
Err:          0
```

1.1.5. 系统平均负载查询

这个文件显示系统在过去一段时间的平均负载，一个真实的输出如下所示：

```
[root@imx6ull:/]# cat /proc/loadavg
0.00 0.00 0.00 1/105 456
```

从左到右每一列分别显示了：

- 过去一分钟的系统平均负载
- 过去五分钟的系统平均负载
- 过去十五分钟的系统平均负载
- 采样时刻运行队列的任务数/系统中活跃的总任务数
- 采样时刻占用最大的线程 ID

这个文件的输出中的前三个值，经常被用来观察系统负载的发展趋势。如果前面的值比后面的值小，说明系统的负载在减轻；反之，说明系统负载开始呈现出上升的趋势。

1.1.6. 网络设备查询

/proc/net 是一个虚拟文件夹，在 Linux 系统中，它包含了许多与网络相关的信息，可以通过查看其中的文件来获取有关网络配置、连接、统计等方面的信息。以下是一些可能包含在 /proc/net 文件夹中的文件和相关信息：

```
[root@imx6ull:/proc/net]# ls
anycast6        dev_snmp6       if_inet6        llc             pppoe           rt6_stats       softnet_stat    unix
arp             fib_trie        igmp6           mcfilter6       protocols       rt_cache        stat            vlan
bnep            fib_triestat    ip6_flowlabel   netfilter       ptype           rtl8723bu       tcp             wireless
can             hci             ip_tables_matches netlink       raw             sco             tcp6
can-bcm         hidp            ip_tables_names netstat         raw6            snmp            udp
connector       hostap          ip_tables_targets nfsfs         rfcomm          snmp6           udp6
dev             icmp            ipv6_route      packet          route           sockstat        udplite
dev_mcast       icmp6           l2cap           pnp             rpc             sockstat6       udplite6
```

- /proc/net/dev：这个文件提供了每个网络接口的详细统计信息，包括接收和发送的数据包数量、错误、丢失等。
- /proc/net/arp：这个文件列出了系统上的 ARP 缓存表，显示了 IP 地址和对应的 MAC 地址。
- /proc/net/route：这个文件显示了系统的路由表，包括目标网络、网关、接口等信息。
- /proc/net/tcp 和 /proc/net/tcp6：这些文件列出了当前 TCP 连接的详细信息，包括本地地址和端口、远程地址和端口、连接状态等。
- /proc/net/udp 和 /proc/net/udp6：这些文件列出了当前 UDP 连接的详细信息，类似于 TCP 连接文件。
- /proc/net/raw 和 /proc/net/raw6：这些文件提供了关于原始套接字的信息，可以查看当前原始套接字连接的详细信息。
- /proc/net/snmp：这个文件提供了 SNMP（Simple Network Management Protocol）的统计信息，包括接口、IP、ICMP、TCP、UDP 等方面的统计数据。
- /proc/net/ip_conntrack 和 /proc/net/nf_conntrack：这些文件显示了连接跟踪器（Connection Tracking）的信息，可以查看当前的网络连接状态。
- /proc/net/ip_vs：这个文件提供了 IP Virtual Server（IPVS）的信息，包括负载均衡池、转发规则等。

这只是 /proc/net 文件夹中可能包含的一些文件和信息。通过查看这些文件，你可以了解系统的网络配置、连接状态、统计数据等，这对于网络故障排除、性能调优以及监控网络活动非常有用。注意，这些信息可能会因为系统配置和活动的变化而发生变化。

1.1.7. 内核版本信息

```
[root@imx6ull:/proc/net]# cat /proc/version
Linux version 4.9.88 (book@li-virtual-machine) (gcc version 6.2.1 20161016 (Linaro GCC 6.2-2016.11) ) #1 SMP PREEMPT Wed Apr 22 15:53:26 CST 2020
```

1.1.8. /proc/PID/

在 Linux 操作系统中，/proc/PID 文件夹（其中的 PID 应该被替换为进程的实际进程 ID）包含了有关特定进程的详细信息。通过查看这个文件夹中的文件，你可以获取有关进程的各种统计数据、配置信息、状态等。以下是一些可能包含在 /proc/PID 文件夹中的文件和相关信息：

```
[root@imx6ull:/proc/1]# ls
auxv            comm            exe             map_files       mounts          oom_adj         personality     statm           timerslack_ns
cgroup          coredump_filter fd              maps            mountstats      oom_score       root            status          wchan
clear_refs      cwd             fdinfo          mem             net             oom_score_adj   smaps           syscall
cmdline         environ         limits          mountinfo       ns              pagemap         stat            task
```

- /proc/PID/cmdline：这个文件包含了启动进程的完整命令行参数。
- /proc/PID/cwd：这个文件是一个符号链接，指向进程的当前工作目录。
- /proc/PID/environ：这个文件包含了进程的环境变量列表。
- /proc/PID/exe：这个文件是一个符号链接，指向进程的可执行文件。
- /proc/PID/status：这个文件包含了进程的状态信息，包括进程 ID、父进程 ID、CPU 使用情况、内存使用情况等。
- /proc/PID/stat：这个文件包含了进程的状态信息，以一行文本的形式显示了进程的各种统计数据，如 CPU 时间、状态、优先级等。
- /proc/PID/io：这个文件包含了进程的输入输出统计信息，如读写字节数、读写操作次数等。
- /proc/PID/limits：这个文件列出了与进程相关的资源限制。
- /proc/PID/maps：这个文件显示了进程的内存映射信息，包括各个内存区段的起始地址、权限等。
- /proc/PID/net：这个文件夹包含有关进程网络连接的信息，包括打开的套接字等。
- /proc/PID/task：这个文件夹包含了与进程相关的所有线程的子文件夹。

1.2. proc 相关接口

在 Linux 内核编程中，/proc 文件系统提供了一种在运行时向用户空间提供信息的方式。这些信息通常以文件形式呈现，可以在 /proc 文件系统的目录结构中访问。下面是一些 /proc 文件系统相关的函数接口和概念的介绍：

1.2.1. proc_dir_entry 结构体

proc_dir_entry 结构体是 Linux 内核中表示 /proc 文件系统中文件节点的数据结构。通过这个结构体，可以定义文件节点的属性和操作。

```
struct proc_dir_entry {
    unsigned int low_ino;
    umode_t mode;
```

```
    nlink_t nlink;
    kuid_t uid;
    kgid_t gid;
    loff_t size;
    const struct inode_operations *proc_iops;
    const struct file_operations *proc_fops;
    struct proc_dir_entry *parent;
    struct rb_node subdir;
    struct rb_root subdir_tree;
    void *data;
    atomic_t count;
    atomic_t in_use;
    struct completion *pde_unload_completion;
    struct list_head pde_openers;
};
```

在 Linux 内核中，proc_fs.h 头文件定义了操作 /proc 文件系统的相关 API。这些 API 允许内核模块在 /proc 目录下创建虚拟文件，以向用户空间提供信息。下面是一些 proc_fs.h 中常见 API 的简要介绍：

```
/*
 * The proc filesystem constants/structures
 */
#ifndef _LINUX_PROC_FS_H
#define _LINUX_PROC_FS_H

#include <linux/types.h>
#include <linux/fs.h>

struct proc_dir_entry;

#ifdef CONFIG_PROC_FS

extern void proc_root_init(void);
extern void proc_flush_task(struct task_struct *);

extern struct proc_dir_entry *proc_symlink(const char *,
                struct proc_dir_entry *, const char *);
extern struct proc_dir_entry *proc_mkdir(const char *, struct proc_dir_entry *);
extern struct proc_dir_entry *proc_mkdir_data(const char *, umode_t,
                                              struct proc_dir_entry *, void *);
extern struct proc_dir_entry *proc_mkdir_mode(const char *, umode_t,
                                              struct proc_dir_entry *);

extern struct proc_dir_entry *proc_create_data(const char *, umode_t,
                                               struct proc_dir_entry *,
                                               const struct file_operations *,
                                               void *);
```

1.2.2. create_proc_entry / proc_create 接口

create_proc_read_entry 是 Linux 内核中用于创建 /proc 文件系统中读取操作的文件节点的函数接口。然而，需要注意的是，create_proc_read_entry 函数在较新的内核版本中已经被弃用，并建议使用更先进的 proc_create 函数替代。

```
static inline struct proc_dir_entry *proc_create(
        const char *name, umode_t mode, struct proc_dir_entry *parent,
        const struct file_operations *proc_fops)
{
        return proc_create_data(name, mode, parent, proc_fops, NULL);
}
```

- name：虚拟文件的名称。这是一个字符串，将作为 /proc 目录下的文件名。
- mode：虚拟文件的权限模式。这指定了用户、组和其他用户的访问权限。
- parent：父目录的 proc_dir_entry。可以指定虚拟文件要创建在 /proc 下的哪个子目录。
- proc_fops：一个指向文件操作结构（struct file_operations）的指针。这个结构包含了虚拟文件的操作函数，如读取和写入。

1.2.3. remove_proc_entry / proc_remove 接口

remove_proc_entry 和 proc_remove 是 Linux 内核中用于操作 /proc 文件系统中的文件节点的函数接口。

remove_proc_entry 函数：

```
void remove_proc_entry(const char *name, struct proc_dir_entry *parent);
```

remove_proc_entry 函数用于从 /proc 文件系统中移除一个指定的文件节点。它接受两个参数：

- name：要移除的文件节点的名称字符串。
- parent：父级目录的 proc_dir_entry 结构，指示要从哪个目录中移除文件节点。

proc_remove 函数：

```
void proc_remove(struct proc_dir_entry *entry);
```

proc_remove 函数也用于从 /proc 文件系统中移除一个指定的文件节点，但与 remove_proc_entry 不同，它只需要一个参数：

- entry：要移除的 proc_dir_entry 结构。

proc_remove 与 remove_proc_entry 的功能基本相同，都是用于删除 /proc 文件系统中的文件节点。在内核中，proc_remove 实际上是调用了 remove_proc_entry 函数，因此它们的效果是一样的。