

Computer Operating Systems, Practice Session 2

Booting Sequence and /proc File System

Resul Tugay (tugayr@itu.edu.tr)

February 14, 2018

Today

Computer Operating Systems, PS 2

PC Booting Sequence

Master Boot Record - MBR

Preloading Sectors

Linux /proc directory

When you press the power button...

- ▶ The system which starts the PC after the power button is pressed is called the boot loader (e.g. **B**asic **I**nput **O**utput **S**ystem) - BIOS)
- ▶ BIOS is a series of information which is stored on a ROM

Initial Processes

Initial Processes...

- ▶ *Power Good Signal* (which is typically +5V) is the signal that is generated when the power supply reaches its required operating conditions
- ▶ CPU is ready for operating. The first place to look up is the BIOS ROM for the start up program. Typically, the ROM ends with the memory space including the *jump* command
- ▶ The first operation the BIOS performs is to check the system: a process called Power On Self Test (POST). The hardware is checked for any potential malfunction before the system starts.
- ▶ The graphics card is started via searching for its BIOS.

BIOS controls

ROMs of the remaining peripherals is searched for a BIOS.

- ▶ Typically, the BIOSes of the IDE/ATA hard drives are found and executed.
- ▶ If any other peripheral has a BIOS, then, similarly, it is also executed.

Startup Screen

BIOS visualizes its startup screen. This startup screen has the following information:

- ▶ BIOS producer and version number
- ▶ BIOS date
- ▶ Keys to enter the BIOS Setup
- ▶ System logo
- ▶ BIOS serial number
- ▶ <http://www.wimsbios.com/> (an online BIOS scan)

BIOS tests

- ▶ BIOS performs many further tests on system like memory count test.
- ▶ The user is informed on any errors encountered at this point.
- ▶ *"Keyboard error, think F1 to continue..."*

Permanent system information

- ▶ After previous operations, BIOS reads the system date, system time and peripherals from the CMOS memory on the mainboard.
- ▶ CMOS integrated circuits require very low power, thus they are able to store their memories for very extended periods with a standard battery. In PCs, CMOS integrated circuits are typically used for storing the data like date and time, which need to be unaffected from power failures.
- ▶ By reading the information stored in the CMOS, the PC learns which hard drives are connected and in which order they should be checked for a proper startup sequence. Therefore, it is able to start the operating system properly.

MBR

- ▶ If the booting will be performed using a hard drive, Cylinder 0, Head 0, Sector 1 which is called *Master Boot Record* is read.
- ▶ At this point, BIOS is disengaged.
- ▶ In order to load the OS, system copies the first 512 bytes of the first hard drive into the memory and executes the code existing at the beginning of this section. Information included is related to the further booting operations. That is why it is called as MBR.

PC Booting Sequence

Up to this point, booting operations are independent of the installed operating system and are same for all PCs.

Master Boot Record - MBR

- ▶ The organization of the MBR has a very standard structure irrespective of the type of the installed operating system:
 - ▶ First portion of 446 bytes are reserved for the program code.
 - ▶ Latter 64 bytes includes a partition table containing 4 partitions.
 - ▶ Last 2 bytes includes a special number (magic number AA55). An MBR having a different number is not validated by BIOS and any operating system.

Structure of a classical generic MBR

Address		Description	Size in bytes
Hex	Dec		
+000h	+0	Bootstrap code area	446
+1BEh	+446	Partition entry #1	16
+1CEh	+462	Partition entry #2	16
+1DEh	+478	Partition entry #3	16
+1EEh	+494	Partition entry #4	16
+1FEh	+510	55h	2
+1FFh	+511	AAh	
Total size: $446 + 4 \cdot 16 + 2$			512

- ▶ Program, starts booting sequence by looking at the partition table and deciding which partition to be used for the startup. Then, program transfer the flow control to the specified partitions preloading sector (boot sector).

Place of preloading sectors

- ▶ Preloading sectors are the first sectors of the hard discs (a.k.a. boot sectors). They provide a space (512 bytes) for the code to start the operating system in that portion. Additionally, they include some basic information on the file system.
- ▶ A valid preloading sector (likewise in MBR) includes a special number stored in last 2 bytes (AA55).

As a summary..

- ▶ When we supply enough power to the PC, CPU executes codes on BIOS(ROM). First operation is POST process. Then, BIOS checks whether there is an OS or not on hard drive, or disc. Sequence order is predefined.
- ▶ If BIOS finds an MBR on the selected device, it transforms MBC from the device to the memory, after this operation MBC seizes the control of the PC. Later on, there are two things that can happen:
 - ▶ There is an OS in this portion, MBC loads this OS to memory
 - ▶ MBC encounters a bootloader such as , NTLDR, LILO or GRUB and user selects an OS from the menu and processes continue..

Linux Boot Loader

In Linux, different boot loaders can be written to different preloading sectors.

- ▶ LILO (Linux Loader) - GRUB (Grand Unified Boot Loader)
 - ▶ Is responsible for the loading of the system and conveying the control to the kernel
 - ▶ Supports many operating systems and file systems
- ▶ LILO (Linux Loader) - GRUB (Grand Unified Boot Loader) differences
 - ▶ LILO, does not provide interactive command interface like GRUB
 - ▶ LILO does not support booting from network: GRUB does
 - ▶ In LILO, with an erroneous modification in the config file, MBR with an improper configuration may cause the system to be un-bootable. In GRUB, on the occurrence of such condition, system passes to the interactive command interface.

Functions of the kernel and /proc

- ▶ Linux kernel has two basic functionalities:
 - ▶ Control the access to the hardware
 - ▶ Determine when and how the processes will interact with these entities
- ▶ /proc folder contains files about the current status of the kernel.
- ▶ Information about hardware and active processes can be retrieved from files under /proc directory.
- ▶ /proc folder is on the virtual file system.
- ▶ In virtual file systems, information is kept in memory: do not take any place in discs.
- ▶ In virtual file systems, files act and seem like usual files.

Contents of the /proc directory

```

musty@musty-VirtualBox: /proc
File Edit Tabs Help
musty@musty-VirtualBox:~$ cd /proc/
musty@musty-VirtualBox:/proc$ ls
1      1433  1541  1674  268  5    926      interrupts      sched_debug
10     1434  1542  1687  27   50   929      iomem           schedstat
1053   1438  1547  17    281  51   979      ioports        scsi
1064   1439  1549  1705  29   52   980      irq            self
1067   1444  1560  1737  3    582  997      kallsyms       slabinfo
1087   1445  1565  1739  30   59   acpi      kcore          softirqs
11     1451  1572  1740  31   6    asound     keys           stat
1163   1452  1578  1753  360  611  buddyinfo key-users      swaps
1195   1458  1580  1759  4    621  bus       kmsg           sys
12     1459  1582  1784  423  654  cgroups  kpagecount     sysrq-trigger
120    1469  1591  18    43   7    cmdline      kpageflags     sysvipc
121    1471  1596  1882  44   72  consoles    loadavg        thread-self
122    1473  1597  19    446  73  cpuinfo      locks          timer_list
123    1482  16    2    45   744  crypto      mdstat         timer_stats
1237   1494  1600  20    46   758  devices     meminfo        tty
1247   15    1602  21    469  770  diskstats   misc           uptime
13     1508  1621  22    47   8    dma          modules        version
132    1509  1632  23    474  82   driver      mounts         version_signature
133    1513  1635  24    476  9    execdomains mtrr           vmallocinfo
1348   1522  1652  25    48   911  fb           net            vmstat
1360   1531  1661  26    486  920  filesystems pagetypeinfo  zoneinfo
14     1539  1665  264   49   925  fs           partitions

```


Properties of the files under /proc directory

- ▶ Files under /proc folder are updated continuously. Therefore:
 - ▶ Most of them always have size of 0 bytes.
 - ▶ The date and settings for the last access records of most of them reflect the current date and time.
- ▶ Most of the files are accessible to only 'root'.
- ▶ Files under /proc folder include many information about the system. Like:
 - ▶ uptime, version, kcore (displays a value given in bytes representing the size of the physical memory (RAM) used plus 4 KB)...
 - ▶ `cat /proc/cpuinfo`

Accessing CPU information

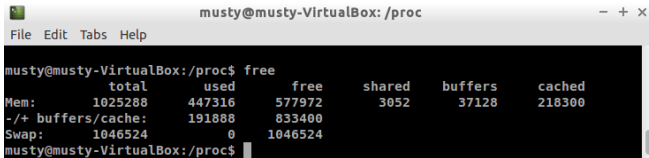
```
musty@musty-VirtualBox: /proc
File Edit Tabs Help
musty@musty-VirtualBox: /proc$ cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 58
model name     : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
stepping       : 9
microcode      : 0x19
cpu MHz        : 3392.294
cache size     : 8192 KB
physical id    : 0
siblings       : 1
core id        : 0
cpu cores      : 1
apicid         : 0
initial apicid : 0
fdiv_bug       : no
f00f_bug       : no
coma_bug       : no
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 rdtscp constant_tsc xtopology nonstop_tsc pni mo
nitor ssse3 sse4_1 sse4_2 hypervisor lahf_lm
bugs           :
bogomips       : 6784.58
clflush size   : 64
cache alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:

musty@musty-VirtualBox: /proc$
```

Monitoring memory space

- ▶ Some files under /proc are hard to read with naked eye. Therefore, we use auxiliary commands:
- ▶ In example: `free` gives information about memory space:
 - ▶ Swap space
 - ▶ Free and used portions of the physical memory
 - ▶ Buffers and cache consumed by the kernel

free command



```
musty@musty-VirtualBox: /proc
File Edit Tabs Help

musty@musty-VirtualBox:/proc$ free
              total        used        free      shared    buffers     cached
Mem:      1025288      447316      577972         3052       37128      218300
-/+ buffers/cache:      191888      833400
Swap:      1046524           0      1046524
musty@musty-VirtualBox:/proc$
```

top command

```

musty@musty-VirtualBox: /proc
File Edit Tabs Help

musty@musty-VirtualBox:/proc$ top

top - 14:06:15 up 12 min, 2 users, load average: 0.07, 0.21, 0.16
Tasks: 131 total, 1 running, 130 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 4.1 sy, 0.0 ni, 95.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1025288 total, 446548 used, 578740 free, 37144 buffers
top - 14:06:29 up 12 min, 2 users, load average: 0.05, 0.20, 0.16
Tasks: 131 total, 1 running, 130 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 5.4 sy, 0.0 ni, 93.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1025288 total, 446896 used, 578392 free, 37144 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free, 218412 cached Mem

  PID USER      PR  NI   VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 1064 root        20   0 101000 43912 17712 S   3.6   4.3   0:15.66 Xorg
 1737 musty     20   0 188524 26580 22788 S   2.0   2.6   0:04.43 lxterminal
 1459 musty     20   0 19128 3004 2680 S   0.7   0.3   0:05.30 VBoxClient
 1957 musty     20   0 8016 2904 2580 R   0.7   0.3   0:00.33 top
 1237 root        20   0 26236 2420 2048 S   0.3   0.2   0:00.92 VBoxService
    1 root        20   0 4436 3628 2580 S   0.0   0.4   0:02.94 init
    2 root        20   0 0 0 0 S   0.0   0.0   0:00.01 kthreadd
    3 root        20   0 0 0 0 S   0.0   0.0   0:00.14 ksoftirqd/0
    5 root        0 -20 0 0 0 S   0.0   0.0   0:00.00 kworker/0:0H
    7 root        20   0 0 0 0 S   0.0   0.0   0:00.86 rcu_sched
    8 root        20   0 0 0 0 S   0.0   0.0   0:00.00 rcu_bh
    9 root        rt   0 0 0 0 S   0.0   0.0   0:00.00 migration/0
   10 root        rt   0 0 0 0 S   0.0   0.0   0:00.06 watchdog/0
   11 root        0 -20 0 0 0 S   0.0   0.0   0:00.00 khelper
   12 root        20   0 0 0 0 S   0.0   0.0   0:00.00 kdevtmpfs
   13 root        0 -20 0 0 0 S   0.0   0.0   0:00.00 netns

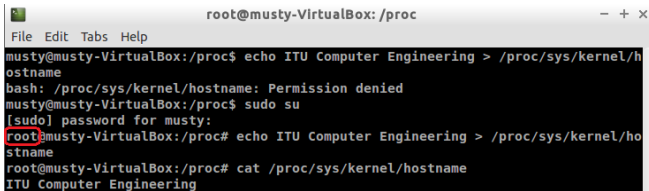
```

- ▶ PR: Priority level
- ▶ NI: Nice parameter, used in scheduling (Negative values - higher priority)
- ▶ VIRT: Virtual memory space used by the process
- ▶ SHR: How much virtual memory can be shared
- ▶ RES: Usage of the physical memory

Writing into the files under `/proc`

- ▶ Most of the time, these files are read-only.
- ▶ Some of them may be modified in order to configure some kernel parameters.
- ▶ Since the files are virtual, shell commands are needed for performing the modifications.

Writing to a file under /proc via echo command



```
root@musty-VirtualBox: /proc
File Edit Tabs Help
musty@musty-VirtualBox:/proc$ echo ITU Computer Engineering > /proc/sys/kernel/hostname
bash: /proc/sys/kernel/hostname: Permission denied
musty@musty-VirtualBox:/proc$ sudo su
[sudo] password for musty:
root@musty-VirtualBox:/proc# echo ITU Computer Engineering > /proc/sys/kernel/hostname
root@musty-VirtualBox:/proc# cat /proc/sys/kernel/hostname
ITU Computer Engineering
```

Process folders under /proc

```

root@musty-VirtualBox: /proc/929
File Edit Tabs Help
root@musty-VirtualBox:/proc# ls
1 1434 1565 1739 25 486 979 ioports scsi
10 1444 1572 1740 26 49 980 irq self
1053 1445 1578 1753 264 5 997 kallsyms slabinfo
1064 1451 1580 1759 268 51 acpi kcore softirqs
1067 1452 1582 1784 27 582 asound keys stat
1087 1456 1591 18 281 611 buddyinfo key-users swaps
11 1459 1596 19 29 621 bus kmsg sys
1163 1473 1597 1907 3 654 cgroups kpagecount sysrq-trigger
1195 1482 16 1908 30 7 cmdline kpageflags sysvipc
12 1494 1600 1916 31 72 consoles loadavg thread-self
120 15 1602 1951 360 73 cpuinfo locks timer_list
121 1508 1621 1963 423 744 crypto mdstat timer_stats
122 1509 1632 1964 43 758 devices meminfo tty
1237 1513 1635 1965 44 770 diskstats misc uptime
1247 1522 1652 1976 446 8 dma modules version
13 1531 1661 1994 45 82 driver mounts version_signature
132 1539 1665 2 46 9 execdomains mtrr vmallocinfo
133 1541 1674 20 469 911 fb net vfstat
1348 1542 1687 21 47 920 filesystems pagetypeinfo zoneinfo
1360 1547 17 22 474 925 fs partitions
14 1549 1705 23 476 926 interrupts sched_debug
1433 1560 1737 24 48 929 iomem schedstat

root@musty-VirtualBox:/proc# cd 929
root@musty-VirtualBox:/proc/929# ls
attr cpuset limits net root statm
autogroup cwd loginuid ns sched status
auxv environ map_files oom_adj schedstat syscall
cgroup exe maps oom_score sessionid task
clear_refs fd mem oom_score_adj setgroups timers
cmdline fdinfo mountinfo pagemap smaps uid_map
comm gid_map mounts personality stack wchan
coredump_filter io mountstats projid_map stat

root@musty-VirtualBox:/proc/929#

```

- ▶ Each working process has a folder under /proc.

References

- ▶ <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-proc.html>
- ▶ <http://www.kernelnewbies.org/documents/kdoc/procfs-guide/lkprocfsguide.html>
- ▶ <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/s1-proc-topfiles.html>
- ▶ <http://www.belgeler.org/>