Computer Operating Systems, Practice Session 1

Booting Sequence and /proc File System

G. Selda Uyanık (seldauyanik@itu.edu.tr)

Istanbul Technical University
34469 Maslak, İstanbul

12 February 2014

Today

Computer Operating Systems, Practice Session 1

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

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.

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.

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.

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.

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.

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.

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.

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..."*

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..."*

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.

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.

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.

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.

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.

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.

- 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).

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.

- 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).

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.

- 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).

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	*Partition table* (for primary partitions)	16
+1CEh	+462	Partition entry #2		16
+1DEh	+478	Partition entry #3		16
+1EEh	+494	Partition entry #4		16
+1FEh	+510	55h	*Boot signature*[nb 1]	2
+1FFh	+511	AAh		
			Total size: 446 + 4*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).

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	*Partition table* (for primary partitions)	16
+1CEh	+462	Partition entry #2		16
+1DEh	+478	Partition entry #3		16
+1EEh	+494	Partition entry #4		16
+1FEh	+510	55h	*Boot signature*[nb 1]	2
+1FFh	+511	AAh		
			Total size: 446 + 4*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 preloding sector (likewise in MBR) includes a special number stored in last 2 bytes (AA55).

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 preloding sector (likewise in MBR) includes a special number stored in last 2 bytes (AA55).

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.

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.

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.

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.

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.

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.

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.

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

root@koknar:/proc - Kabuk - Konsole

Oturum Düzenle Görüntüle Yer İmleri Ayarlar Yardım

```
[root@koknar root]# cd /proc/
[root@koknar proc]# ls
1     1742  2142  2511  2713  3          crypto       irq          pci
10    1932  2147  2514  2714  4          devices      kcore        scsi
110   1947  2151  2515  2715  5          diskstats    kmsg         self
151   1966  2154  2645  2716  6          dma          loadavg      slabinfo
152   1975  2157  2688  2719  7          driver       locks        stat
153   2     2159  2689  2720  8          execdomains  mdstat       swaps
1551  2006  2160  2691  2754  9          fb           meminfo      sys
1555  2037  2362  2692  2787  acpi       filesystems  misc         sysrq-trigger
1576  2059  2373  2694  2863  asound     fs           modules      sysvipc
1596  2069  2403  2703  2864  buddyinfo  ide          mounts       tty
1623  2078  2468  2704  2893  bus        interrupts   mtrr         uptime
166   2097  2506  2710  2894  cmdline    iomem        net          version
1730  2113  2509  2712  2911  cpuinfo    ioports      partitions   vmstat
[root@koknar proc]#
```

Kabuk

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`

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`

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

root@koknar:~ - Kabuk - Konsole <2>

Oturum Düzenle Görüntüle Yer İmleri Ayarlar Yardım

```
[root@koknar root]# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 2
model name      : Intel(R) Pentium(R) 4 CPU 2.60GHz
stepping        : 9
cpu MHz         : 2594.174
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe cid
bogomips        : 5144.57

[root@koknar root]#
```

Kabuk

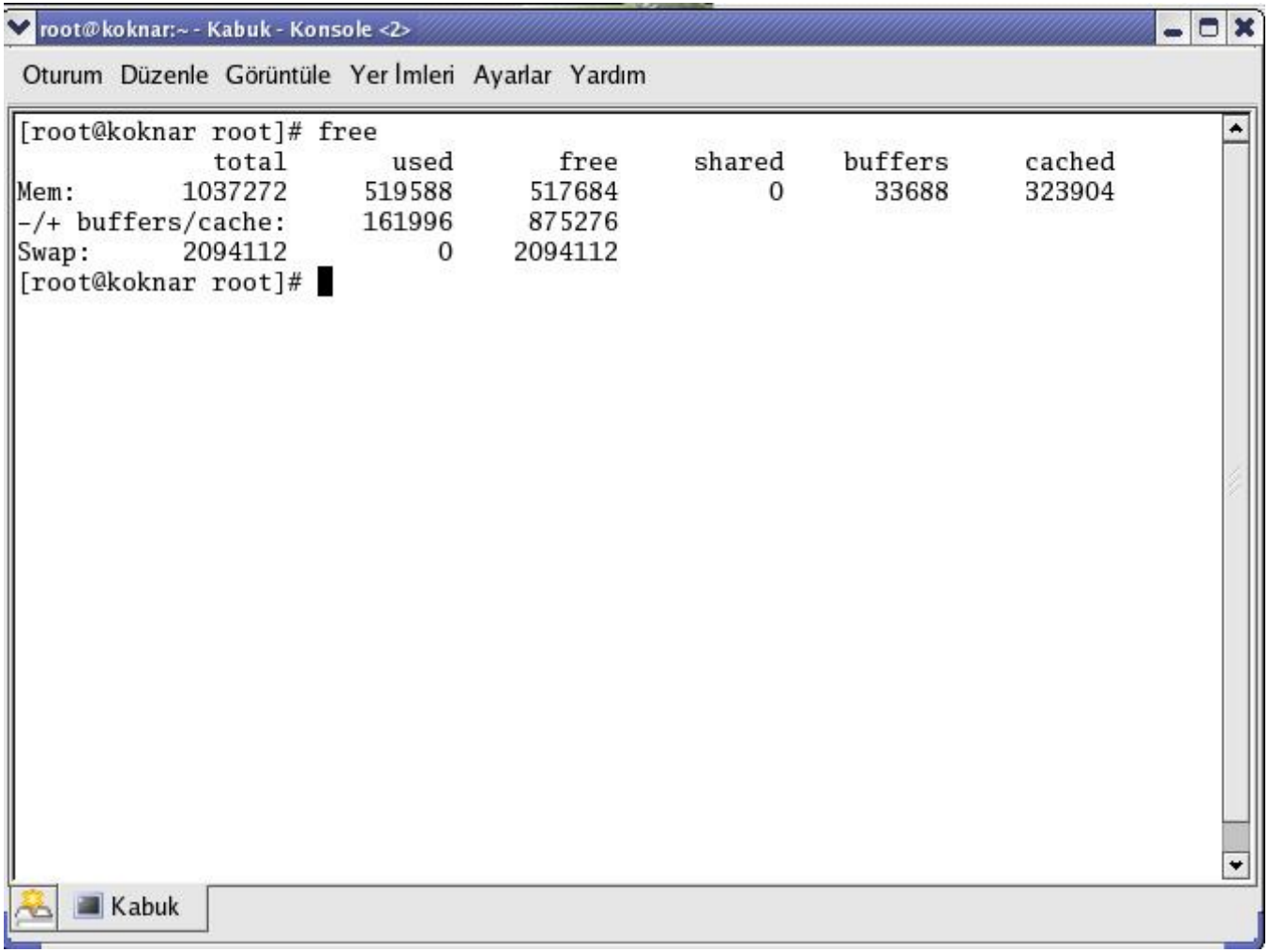
Monitoring memory space

- 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

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



top command

```
root@koknar:~ - Kabuk - Konsole <2>
Oturum  Düzenle  Görüntüle  Yer İmleri  Ayarlar  Yardım
top - 11:31:52 up 37 min,  3 users,  load average: 0.61, 0.33, 0.21
Tasks:  71 total,   1 running,  70 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.0% us,  0.7% sy,  0.0% ni, 96.3% id,  0.0% wa,  0.0% hi,  0.0% si
top - 11:32:11 up 37 min,  3 users,  load average: 0.50, 0.32, 0.21
Tasks:  71 total,   2 running,  69 sleeping,   0 stopped,   0 zombie
top - 11:32:20 up 37 min,  3 users,  load average: 0.42, 0.31, 0.21
Tasks:  71 total,   2 running,  69 sleeping,   0 stopped,   0 zombie
top - 11:34:57 up 40 min,  3 users,  load average: 0.16, 0.22, 0.18
Tasks:  71 total,   1 running,  70 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.7% us,  0.0% sy,  0.0% ni, 97.3% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1037272k total,   552004k used,   485268k free,    33896k buffers
Swap:  2094112k total,        0k used,  2094112k free,   323916k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2373 root      15   0  163m  15m 149m S  2.3  1.6   1:22.82 X
 3043 root      15   0 29804  14m  26m S  0.3  1.4   0:00.78 kdeinit
    1 root      16   0  2984  464 1316 S  0.0  0.0   0:05.50 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.01 events/0
    4 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 kblockd/0
    6 root      12 -10     0    0    0 S  0.0  0.0   0:00.00 khelper
    5 root      15   0     0    0    0 S  0.0  0.0   0:00.00 khubd
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
    8 root      15   0     0    0    0 S  0.0  0.0   0:00.01 pdflush
   10 root      12 -10     0    0    0 S  0.0  0.0   0:00.00 aio/0
Kabuk
```

- 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

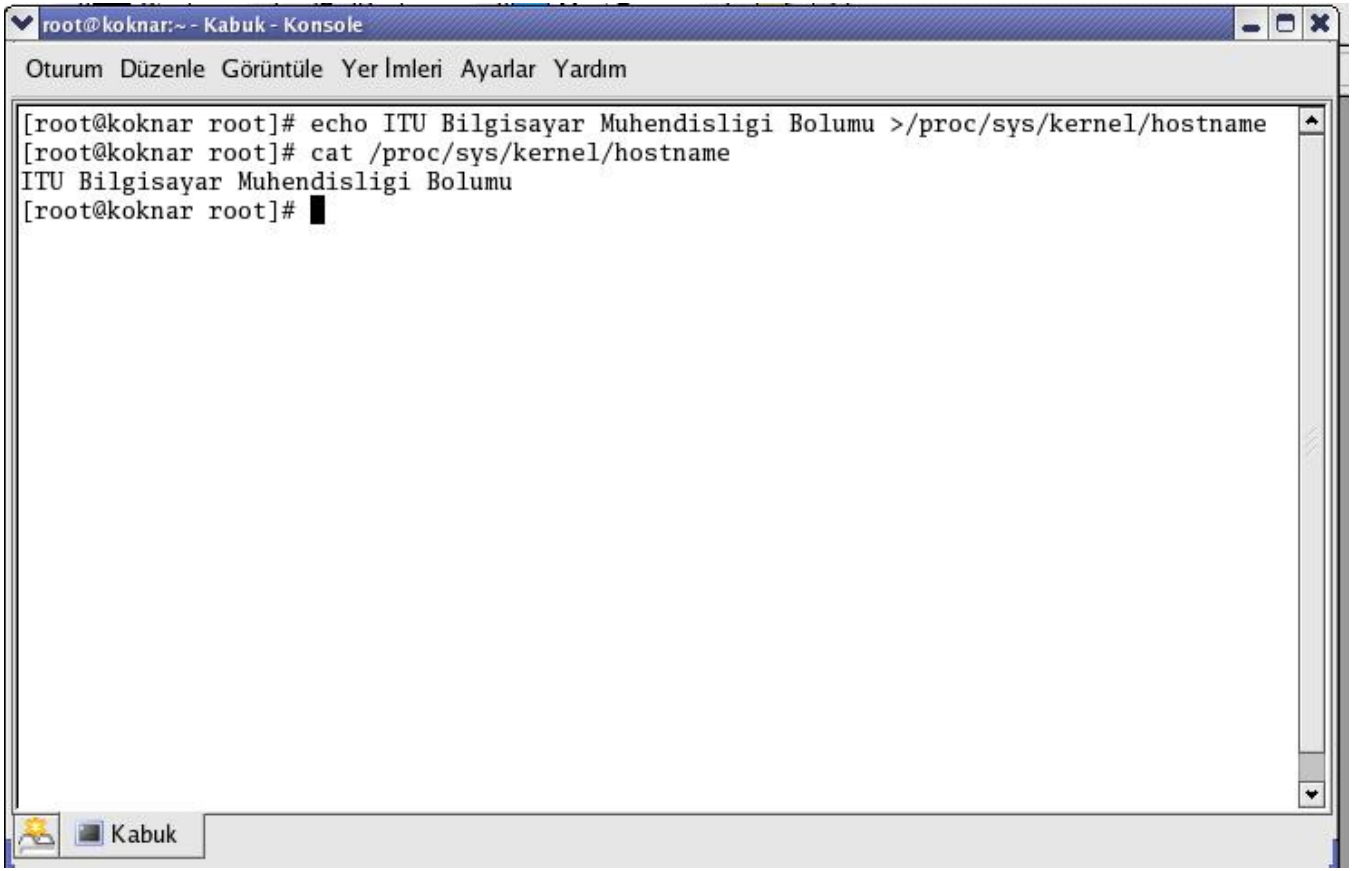
Writing into the files under /proc

- 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.

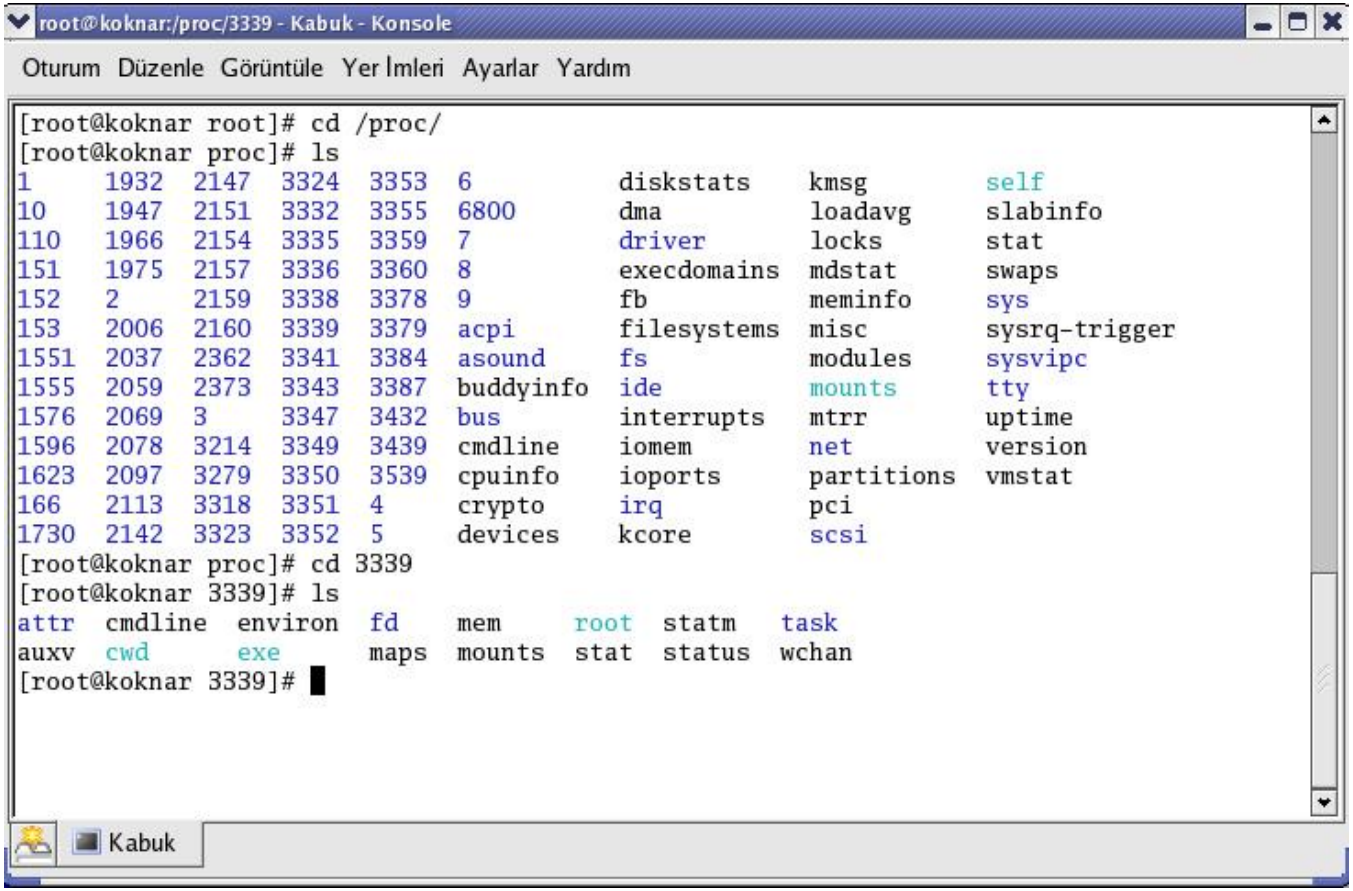
Writing into the files under /proc

- 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



Process folders under /proc



- 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/