# Lab # 2

# Information about hardware, Kernel/distribution information Redirection and Pipes, Filters and Basic Linux Tools

**I. Information about hardware:**

## For CPU information

## $ cat /proc/cpuinfo

/proc/cpuinfo is a short, read-only, [plain text f](http://www.linfo.org/plain_text.html)ile that contains [information a](http://www.linfo.org/information.html)bout the CPUs (central processing units) on a computer. If a computer contains two or more CPUs, the information about each is separated by a blank line. The information about each processor consists of a set of lines (18 on Red Hat 9), each of which contains a key word followed by a separator (consisting of spaces, a colon and then another space) and then by a value. Each key word can be any combination of [alphan](http://www.linfo.org/alphabet.html)umeric, underscore and space [characters.](http://www.linfo.org/character.html)

Only a few of the lines contain information that is meaningful to most users. Among them are processor (the value of which is zero for single-processor systems), vendor\_id (the value of which is GenuineIntel in the case of an Intel processor), cpu family, model\_name, cpu MHz (processor speed in millions of cycles per second), cache size (the amount of high speed cache [memory b](http://www.linfo.org/memory.html)uilt into the processor) and fpu (the value of which is yes if the processor contains a floating point unit).

The flags field shows which features are available in the CPU; this information is of interest only to users with an in-depth knowledge of processors. The final item, bogomips, is a [Linux-](http://www.linfo.org/linuxdef.html)specific measurement of the CPU's speed in MIPS (millions of instructions per second) while not performing any actual work; it is not an accurate indicator of overall processor speed and is only used for some testing purposes.

## $ cat /proc/meminfo

MemTotal — Total amount of physical RAM, in kilobytes.

MemFree — The amount of physical RAM, in kilobytes, left unused by the system.

Buffers — The amount of physical RAM, in kilobytes, used for file buffers.

Cached — The amount of physical RAM, in kilobytes, used as cache memory.

SwapCached — The amount of swap, in kilobytes, used as cache memory.

Active — The total amount of buffer or page cache memory, in kilobytes, that is in active use. This is memory that has been recently used and is usually not reclaimed for other purposes. Inactive — The total amount of buffer or page cache memory, in kilobytes, that are free and available. This is memory that has not been recently used and can be reclaimed for other purposes.

HighTotal and HighFree — The total and free amount of memory, in kilobytes, that is not directly mapped into kernel space. The HighTotal value can vary based on the type of kernel used.

LowTotal and LowFree — The total and free amount of memory, in kilobytes, that is directly mapped into kernel space. The LowTotal value can vary based on the type of kernel used. SwapTotal — The total amount of swap available, in kilobytes. SwapFree — The total amount of swap free, in kilobytes.

Dirty — The total amount of memory, in kilobytes, waiting to be written back to the disk.

Writeback — The total amount of memory, in kilobytes, actively being written back to the disk. Mapped — The total amount of memory, in kilobytes, which have been used to map devices, files, or libraries using the mmap command.

Slab — The total amount of memory, in kilobytes, used by the kernel to cache data structures for its own use.

Committed\_AS — The total amount of memory, in kilobytes, estimated to complete the workload. This value represents the worst case scenario value, and also includes swap memory. PageTables — The total amount of memory, in kilobytes, dedicated to the lowest page table level.

VMallocTotal — The total amount of memory, in kilobytes, of total allocated virtual address space.

VMallocUsed — The total amount of memory, in kilobytes, of used virtual address space. VMallocChunk — The largest contiguous block of memory, in kilobytes, of available virtual address space.

HugePages\_Total — The total number of hugepages for the system. The number is derived by dividing Hugepagesize by the megabytes set aside for hugepages specified in /proc/sys/vm/hugetlb\_pool. This statistic only appears on the x86, Itanium, and AMD64 architectures.

HugePages\_Free — The total number of hugepages available for the system. This statistic only appears on the x86, Itanium, and AMD64 architectures.

Hugepagesize — The size for each hugepages unit in kilobytes. By default, the value is 4096 KB on uniprocessor kernels for 32 bit architectures. For SMP, hugemem kernels, and AMD64, the default is 2048 KB. For Itanium architectures, the default is 262144 KB. This statistic only appears on the x86, Itanium, and AMD64 architectures

## $ free

Displays the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel. The shared memory column should be ignored; it is obsolete.

### $ lspci

lspci - list all PCI devices lspci is a utility for displaying information about all PCI buses in the system and all devices connected to them.

To make use of all the features of this program, you need to have Linux kernel 2.1.82 or newer which supports the /proc/bus/pci interface. With older kernels, the PCI utilities have to use direct hardware access which is available only to root and it suffers from numerous race conditions and other problems.

## $ dmesg

dmesg (display message or driver message) is a command on most [Linux-](http://en.wikipedia.org/wiki/Linux) and [Unix-](http://en.wikipedia.org/wiki/Unix)based operating systems that prints the message buffer of the [kernel.](http://en.wikipedia.org/wiki/Kernel_(computer_science))

**$ dmesg | grep –i eth**

In root

## $ dmidecode

dmidecode command reads the system DMI(Desktop Management Interface) table to display hardware and BIOS information of the server. Apart from getting current configuration of the system, you can also get information about maximum supported configuration of the system using dmidecode. For example, dmidecode gives both the current RAM on the system and the maximum RAM supported by the system.

## II. Kernel/distribution information

### $ uname –r

-r, --kernel-release/ version print the kernel release 2.6.32-220.17.1.el6.i686

$ uname –a

List the complete information

### $ cat /proc/cpuinfo | grep flags

To check processor support 64 bit or 32 bit

$ cat /etc/redhat-release

[finding out linux release version](http://linux.dsplabs.com.au/cat-etc-release-finding-out-release-version-of-a-linux-distribution-suse-fedora-ubuntu-p35/)

## III. stdin, stdout, and stderr

The shell (and almost every other Linux command) takes input from stdin (stream 0) and sends output to stdout (stream 1) and error messages to stderr (stream 2).

**a) output redirection**

**> stdout**

stdout can be redirected with a greater than sign. While scanning the line, the shell will see the > sign and will clear the file.

$ echo It is cold today!

It is cold today!

$ echo It is cold today! > winter.txt

$ cat winter.txt

It is cold today!

### >> append

Use >> to append output to a file.

$ echo It is cold today! > winter.txt

$ cat winter.txt

It is cold today!

**$** echo Where is the summer ? >> winter.txt

$ cat winter .txt

It is cold today!

Where is the summer ?

**b) error redirection**

**2> stderr**

Redirecting stderr is done with 2>. This can be very useful to prevent error messages from cluttering your screen. The screenshot below shows redirection of stdout to a file, and stderr to /dev/null. Writing 1> is the same as >.

$ find / > allfiles.txt 2> /dev/null

### 2>&1

To redirect both stdout and stderr to the same file, use 2>&1.

$ find / > allfiles\_and\_errors.txt 2>&1

Note that the order of redirections is significant. For example, the command ls > dirlist 2>&1

directs both standard output (file descriptor 1) and standard error (file descriptor 2) to the file dirlist, while the command

ls 2>&1 > dirlist

directs only the standard output to file dirlist, because the standard error made a copy of the standard output before the standard output was redirected to dirlist.

**c) input redirection**

**< stdin**

Redirecting stdin is done with **<** (short for 0<).

#### $ cat < text.txt

one

two

#### $ tr 'onetw' 'ONEZZ' < text.txt

ONE ZZO

## IV. Pipes

One of the most powerful advantages of Linux is the use of pipes. A pipe takes stdout from the previous command and sends it as stdin to the next command. All commands in a pipe run simultaneously.

### | vertical bar

Consider the following example.

#### $ ls /etc > etcfiles.txt

#### $ tail -4 etcfiles.txt

X11xdg xml xpdf

This can be written in one command line using a pipe.

$ ls /etc | tail -4

X11 xdg xml xpdf

The pipe is represented by a vertical bar | between two commands.

### multiple pipes

One command line can use multiple pipes. All commands in the pipe can run at the same time.

$ ls /etc | tail -4 | tac

xpdf

xml

xdg

X11

## V. filters

### grep

The grep filter is famous among Linux users. The most common use of grep is to filter lines of text containing (or not containing) a certain string.

$ cat cricket.txt

[Younus Khan, P](http://en.wikipedia.org/wiki/Younus_Khan)ak

[Umar Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

[Junaid Khan, P](http://en.wikipedia.org/wiki/Junaid_Khan)ak

[Shane Watson, A](http://en.wikipedia.org/wiki/Shane_Watson)us

$ cat cricket.txt | grep Khan

[Younus Khan, P](http://en.wikipedia.org/wiki/Younus_Khan)ak [Junaid Khan, P](http://en.wikipedia.org/wiki/Junaid_Khan)ak

You can write this without the cat.

$ grep Khan cricket.txt

[Younus Khan, P](http://en.wikipedia.org/wiki/Younus_Khan)ak

[Junaid Khan, P](http://en.wikipedia.org/wiki/Junaid_Khan)ak

#### grep -i

One of the most useful options of grep is grep -i which filters in a case insensitive way. $ grep PAK cricket.txt

[Umar Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

$ grep -i Pak cricket.txt

[Younus Khan, P](http://en.wikipedia.org/wiki/Younus_Khan)ak

[Umar Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

[Junaid Khan, P](http://en.wikipedia.org/wiki/Junaid_Khan)ak

#### grep -v

Another very useful option is grep -v which outputs lines not matching the string. $ grep -v Aus cricket.txt

[Younus Khan, P](http://en.wikipedia.org/wiki/Younus_Khan)ak

[Umar Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

[Junaid Khan, P](http://en.wikipedia.org/wiki/Junaid_Khan)ak

#### tr

You can translate characters with tr. The screenshot shows the translation of all occurrences of e to E.

$ cat cricket.txt | tr 'a' 'A'

[Younus KhAn, P](http://en.wikipedia.org/wiki/Younus_Khan)Ak

[UmAr Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

[JunAid KhAn, P](http://en.wikipedia.org/wiki/Junaid_Khan)Ak

[ShAne WAtson, A](http://en.wikipedia.org/wiki/Shane_Watson)us

Here we set all letters to uppercase by defining two ranges.

$ cat cricket.txt | tr 'a-z' 'A-Z'

[YOUNUS KHAN, P](http://en.wikipedia.org/wiki/Younus_Khan)AK

[UMAR GUL, P](http://en.wikipedia.org/wiki/Umar_Gul)AK

[JUNAID KHAN, P](http://en.wikipedia.org/wiki/Junaid_Khan)AK

[SHANE WATSO](http://en.wikipedia.org/wiki/Shane_Watson)[N, A](http://en.wikipedia.org/wiki/Shane_Watson)US

**tr –d** to delete characters.

$ cat cricket.txt | tr -d a

[Younus Kh](http://en.wikipedia.org/wiki/Younus_Khan)[n, P](http://en.wikipedia.org/wiki/Younus_Khan)k

[Umr Gul, P](http://en.wikipedia.org/wiki/Umar_Gul)K

[Junid Khn, P](http://en.wikipedia.org/wiki/Junaid_Khan)k

[Shne Wtson, u](http://en.wikipedia.org/wiki/Shane_Watson)s

### wc

Counting words, lines and characters is easy with wc.

$ wc tennis.txt

4 15 100 tennis.txt

$ wc -l tennis.txt tennis.txt

$ wc -w tennis.txt

15 tennis.txt

$ wc -c tennis.txt

100 tennis.txt

### sort

The sort filter will be used for alphabetical sort.

$ cat names.txt

Bilal

Kashif

Omar

Hassan Hassan

$ sort names.txt

Bilal

Hassan

Hassan

Kashif Omar

### uniq

With uniq you can remove duplicates from a sorted list.

$ cat names.txt

Bilal

Kashif

Omar

Hassan

Hassan

$ sort names.txt

Bilal

Hassan

Hassan

Kashif Omar

$ sort names.txt | uniq

Bilal

Hassan

Kashif Omar

**uniq –c**

uniq can also count occurrences with the -c option.

$ sort names.txt |uniq –c

1Bilal

2Hassan

1Kashif 1Omar

## VI. Basic Linux tools

### find

Find all files in /etc and put the list in etcfiles.txt find /etc > etcfiles.txt

Find all files of the entire system and put the list in allfiles.txt find / > allfiles.txt

### time

The time command can display how long it takes to execute a command. The date command takes only a little time.

$ time date

Sat Apr 17 13:08:272010

Real 0m0.014s

user 0m0.008s

sys 0m0.006s

This bzip2 command compresses a file and uses a lot of cpu time.

$ time bzip2 text.txt

real 0m2.368s

user 0m0.847s

sys 0m0.539s

### gzip – gunzip

Users never have enough disk space, so compression comes in handy. The gzip command can make files take up less space.

$ ls -lh text.txt

-rw-rw-r-- 1 root root 6.4M Apr 17 13:11 text.txt

$ gzip text.txt

$ ls -lh text.txt.gz

-rw-rw-r-- 1 root root 760K Apr 17 13:11 text.txt.gz

You can get the original back with gunzip.

$ gunzip text.txt.gz

$ ls -lh text.txt

-rw-rw-r-- 1 root root 6.4M Apr 17 13:11 text.txt

### zcat

Text files that are compressed with gzip can be viewed with zcat.

$ head -4 text.txt

/

/opt

/opt/VBoxGuestAdditions-3.1.6

/opt/VBoxGuestAdditions-3.1.6/routines.sh

$ gzip text.txt

$ zcat text.txt.gz | head -4

/

/opt

/opt/VBoxGuestAdditions-3.1.6

/opt/VBoxGuestAdditions-3.1.6/routines.sh

### bzip2 - bunzip2

Files can also be compressed with bzip2 which takes a little more time than gzip, but compresses better.

$ bzip2 text.txt

$ ls -lh text.txt.bz2

-rw-rw-r-- 1 root root 569K Apr 17 13:11 text.txt.bz2 Files can be uncompressed again with bunzip2. $ bunzip2 text.txt.bz2

$ ls -lh text.txt

-rw-rw-r-- 1 paul paul 6.4M Apr 17 13:11 text.txt

### bzcat

And in the same way bzcat can display files compressed with bzip2.

$ bzip2 text.txt

$ bzcat text.txt.bz2 | head -4

/

/opt

/opt/VBoxGuestAdditions-3.1.6

/opt/VBoxGuestAdditions-3.1.6/routines.sh

## VII. PROGRAM FOR SIMULATION OF GREP UNIX COMMANDS

### ALGORITHM

STEP1: Start the program

STEP2: Declare the variablesfline[max], count=0, occurrences=0 and pointers \*fp, \*newline**.**

STEP 3: Open the file in read mode.

STEP4: In while loop checkfgets(fline,max,fp)!=NULL STEP 5: Increment count value.

STEP 6: Check newline=strchr(fline, „\n‟)

STEP 7: print the count,fline value and increment the occurrence value.

STEP 8: Stop the program

**PROGRAM:**

#include<stdio.h>

#include<string.h> #define max 1024 void usage()

{

printf(“usage:\t. /a.out filename word \n “);

}

int main(int argc, char \*argv[])

{

FILE \*fp; char fline[max]; char \*newline; int count=0; int occurrences=0; if(argc!=3) { usage(); exit(1); } if(!(fp=fopen(argv[1],”r”)))

{ printf(“grep: couldnot open file : %s \n”,argv[1]); exit(1); }

while(fgets(fline,max,fp)!=NULL)

{ count++; if(newline=strchr(fline, „\n‟)) \*newline=‟\0‟; if(strstr(fline,argv[2])!=NULL)

{ printf(“%s: %d %s \n”, argv[1],count, fline); occurrences++;

}

}

}

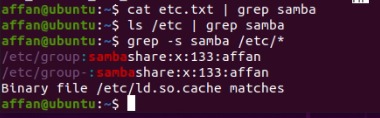
**Lab Tasks**

1. Use ls to output the contents of the /etc/ directory to a file called etc.txt.

Text

Description automatically generated

1. Make a list of all filenames in /etc that contain the string samba



1. Make a sorted list of all files in /etc that contain the case insensitive string samba.

A screenshot of a computer

Description automatically generated with medium confidence

1. Write a line that removes all non-letters from a stream.

Text

Description automatically generated

1. Write a line that receives a text file, and outputs all words on a separate line.

Text

Description automatically generated

Text

Description automatically generated