soln 1:-

we can perform direct calculation on terminal by just installing bc utility in it.it gives us everything, we expect from scientific,

finacial, or evem simple calculator. bc utility can be scripted from the command line if needed.this allows us to use in shell script.

in case we need to do more complex math.

bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

Arithmetic operations are the most basic in any kind of programming language. Linux or Unix operating system provides the bc command and expr command for doing arithmetic calculations. You can use these commands in bash or shell script also for evaluating arithmetic expressions.

The bc command supports the following features:

1.Arithmetic operators

2.Increment or Decrement operators

3.Assignment operators

3.Comparison or Relational operators

4.Logical or Boolean operators

5.Math functions

6.Conditional statements

7.Iterative statements

```
user@user-VirtualBox:~$ a=12

user@user-VirtualBox:~$ b=3

user@user-VirtualBox:~$ echo "$a+$b" | bc

15

user@user-VirtualBox:~$ echo "$a*$b" | bc

36

user@user-VirtualBox:~$ echo "$a/$b" | bc
```

4

user@user-VirtualBox:~$ echo "$a-$b" | bc

9

user@user-VirtualBox:~$ echo "$a%$b" | bc

0

user@user-VirtualBox:~$ echo "4+5" | bc

9

user@user-VirtualBox:~$ echo "12+3" | bc

15

user@user-VirtualBox:~$ echo "12*3" | bc

36

user@user-VirtualBox:~$ echo "12/3" | bc

4

user@user-VirtualBox:~$ echo "12-3" | bc

9

user@user-VirtualBox:~$ echo "12%3" | bc

0

Input: $ echo "for(i=1; i<=10; i++) {i;}" | bc

Output:

1

2

3

4

5

6

7

8

9

10

Input: $ echo "i=1;while(i<=10) {i; i+=1}" | bc

Output:

1

2

3

4

5

6

7

8

9

10

limit set:

```
user@user-VirtualBox:~$ y=23.4000
user@user-VirtualBox:~$ printf "%.2f" "$y"
23.40
user@user-VirtualBox:~$ z=55.4
user@user-VirtualBox:~$ printf "%.6f" "$z"
55.400000
user@user-VirtualBox:~$ a=12345
user@user-VirtualBox:~$ printf "%3d" "$a"
```

```
    12345
    user@user-VirtualBox:~$ a=65
    user@user-VirtualBox:~$ printf "%7d" "$a"
    65
    user@user-VirtualBox:~$
```

---

soln2:-Program of binary search tree using shell script:-

Binary search using shell script

```
echo “Enter the limit:”

read n

echo “Enter the numbers”

for(( i=0 ;i<n; i++ ))

do

read m

a[i]=$m

done
```

```
for(( i=1; i<n; i++ ))

do

for(( j=0; j<n-i; j++))

do

if [ ${a[$j]} -gt ${a[$j+1]} ]

then

t=${a[$j]}

a[$j]=${a[$j+1]}

a[$j+1]=$t

fi

done

done

echo “Sorted array is”
```

```
for(( i=0; i<n; i++ ))

do

echo “${a[$i]}”

done

echo “Enter the element to be searched :”

read s

l=0

c=0

u=$(($n-1))

while [ $l -le $u ]

do

mid=$(((( $l+$u ))/2 ))

if [ $s -eq ${a[$mid]} ]
```

```
then

c=1

break

elif [ $s -lt ${a[$mid]} ]

then

u=$(($mid-1))

else

l=$(($mid+1))

fi

done

if [ $c -eq 1 ]

then

echo “Element found at position $(($mid+1))”
```

else

echo “Element not found”

----------*----------
*----------

soln3:-

1.Memory Usage

On linux, there are commands for almost everything, because the gui might not be always available. When working on servers only shell access is available and everything has to be done from these commands. So today we shall be checking the commands that can be used to check memory usage on a linux system. Memory include RAM and swap.

It is often important to check memory usage and memory used per process on servers so that resources do not fall short and users are able to access the server. For example a website. If you are running a webserver, then the server must have enough memory to serve the visitors to the site. If not, the site would become very slow or even go down when there is a traffic spike, simply because memory would fall short. Its just like what happens on your desktop PC.

1. free command

The free command is the most simple and easy to use command to check memory usage on linux. Here is a quick example

$ free -m

	total	used	free	shared	buffers	cached
Mem:	7976	6459	1517	0	865	2248

-/+ buffers/cache: 3344 4631

Swap: 1951 0 1951

The m option displays all data in MBs. The total os 7976 MB is the total amount of RAM installed on the system, that is 8GB. The used column shows the amount of RAM that has been used by linux, in this case around 6.4 GB. The output is pretty self explanatory. The catch over here is the cached and buffers column. The second line tells that 4.6 GB is free. This is the free memory in first line added with the buffers and cached amount of memory.

Linux has the habit of caching lots of things for faster performance, so that memory can be freed and used if needed.

2.

5 commands to check memory usage on Linux

By Silver Moon | October 26, 2013

32 Comments

Memory Usage

On linux, there are commands for almost everything, because the gui might not be always available. When working on servers only shell access is available and everything has to be done from these commands. So today we shall be checking the commands that can be used to check memory usage on a linux system. Memory include RAM and swap.

It is often important to check memory usage and memory used per process on servers so that resources do not fall short and users are able to access the server. For example a website. If you are running a webserver, then the server must have enough memory to serve the visitors to the site. If not, the site would become very slow or even go down when there is a traffic spike, simply because memory would fall short. Its just like what happens on your desktop PC.

1. free command

The free command is the most simple and easy to use command to check memory usage on linux. Here is a quick example

```
$ free -m
             total       used       free     shared    buffers     cached
Mem:          7976       6459       1517          0        865       2248
-/+ buffers/cache:       3344       4631
Swap:         1951          0       1951
```

The m option displays all data in MBs. The total os 7976 MB is the total amount of RAM installed on the system, that is 8GB. The used column shows the amount of RAM that has been used by linux, in this case around 6.4 GB. The output is pretty self explanatory. The catch over here is the cached and buffers column. The second line tells that 4.6 GB is free. This is the free memory in first line added with the buffers and cached amount of memory.

Linux has the habit of caching lots of things for faster performance, so that memory can be freed and used if needed.

The last line is the swap memory, which in this case is lying entirely free.

2. /proc/meminfo

The next way to check memory usage is to read the /proc/meminfo file. Know that the /proc file system does not contain real files. They are rather virtual files that contain dynamic information about the kernel and the system.

3. vmstat

The vmstat command with the s option, lays out the memory usage statistics much like the proc command.

4. top command

The top command is generally used to check memory and cpu usage per process. However it also reports total memory usage and can be used to monitor the total RAM usage. The header on output has the required information.

5. htop

Similar to the top command, the htop command also shows memory usage along with various other details

---
*---
*---

soln 4:-

```
user@user-VirtualBox:~$ cd parent
user@user-VirtualBox:~/parent$ cd child
user@user-VirtualBox:~/parent/child$ ls
michel  watson  white
user@user-VirtualBox:~/parent/child$ ls --file-type *.txt
ls: cannot access '*.txt': No such file or directory
user@user-VirtualBox:~/parent/child$ cat >> y.txt
bfwhwhbgiuser@user-VirtualBox:~/parent/child$ ls
```

michel watson white y.txt

user@user-VirtualBox:~/parent/child$ ls --file-type *.txt

y.txt

user@user-VirtualBox:~/parent/child$ cat >> a.txt

qbgdjkduser@user-VirtualBox:~/parent/child$ cat >> b.cpp

akfhioauser@user-VirtualBox:~/parent/child$ cat >> c.cpp

hifgiouser@user-VirtualBox:~/parent/child$ cat >> d.exe

whfiwhfo

user@user-VirtualBox:~/parent/child$ cat >> e.exe

bihgqiuuser@user-VirtualBox:~/parent/child$ ls --file-type *.txt

a.txt y.txt

user@user-VirtualBox:~/parent/child$ ls --file-type *.exe

d.exe e.exe

user@user-VirtualBox:~/parent/child$ ls --file-type *.cpp

b.cpp c.cpp

user@use

---

soln 5:-

user@user-VirtualBox:~/parent/child$ cd ../..

user@user-VirtualBox:~$ mkdir nitt

user@user-VirtualBox:~$ cd nitt

user@user-VirtualBox:~/nitt$ mkdir mca

user@user-VirtualBox:~/nitt$ cd mca

user@user-VirtualBox:~/nitt/mca$ cd ../..

user@user-VirtualBox:~$

---

soln 6:-

Linux is an operating system like iOS and Windows. The popularity of the Linux OS has been increasing very rapidly and more smart devices with Linux OS is being developed nowadays. The biggest reasons behind the enormous increase in the popularity of Linux is considered the high tech security system of the Linux. Linux is an open source operating system whose code can be easily read out by the users, but still, it is the more secure operating system when compared to the other OS(s).

Though Linux is very simple but still very secure operating system, which protects the important files from the attack of viruses and malware. So, if you are wondering how Linux is more secure than the giant operating systems, like iOS, Windows, and Android, then to better understand this, look at few advantages of Linux security.

How Linux security overpowers other OS?

1. A perk of Accounts.

In the operating system such as Windows, users have full admin access to the accounts of software. When the virus strikes in this system and then within few seconds it corrupts the whole system. In short, all the files are in danger due to the open access, but in the Linux, very low access is given to the users. Thus the viruses can’t attack the whole system and they only attack few files, and other system works without any issue.

2.Strong Community.

Windows and other operating systems are more vulnerabilities to the type of social engineering Ltd compared to Linux. Amateur users can easily expose to the viruses in other OS by opening one email. But this is not the case in the Linux and user needs full execution right before opening any new attachment. Thus web developers and testers prefer this system as it saves them from the vulnerabilities.

3.IPtables.

A high tech security of IPtables is used by the Linux to enhance the security circle of the system. This firewall that allows you to create a more secure environment for the execution of any command or access the network.

4.Different working environment.

Linux system operates in the different environment such as such as Linux Mint, Debian, Ubuntu, Gentoo, Arch, and many others. The division and segmented working environment protect from the attack of the virus. On the other hand, Windows isn’t much divided operating system and thus it is more exposed to the threats.

5.Recording in Linux.

A Proper log is established in the Linux of the timing and it can be viewed later on easily. If someone tries to enter safe system files, these system gaps can be viewed by the system administrator. Also, the disk to fail attempts are presented to read for later on.

6.Fewer users.

The number of users in the Linux operating system is lesser than the iOS or Windows user. Thus fewer people are using Linux system makes it more secure as compared to the overly crowded operating system Windows.

Conclusion:

Well, saying that the Linux is 100% secure operating system is impossible as no OS is fully secured. To make the systems fully secure further software's are required, but still, Linux has few features which make it more secure than the other operating software.

----------*----------
*----------

soln 7:-

```
$gedit x.cpp
        #include<stdio.h>
        int main()
        {       int num,sum=0,i=1;
                printf("Enter no n");
                scanf("%d",&num);
                if(num<0)
                num=num-(num*2);
                while(num!=0)
        {
                sum+=(num%8)*i;
                num=num/8;
                i*=10;

        }
```

```
                printf("%d",sum);
                return 0;
        }
```

```
$gcc -o x x.cpp
```

---

soln 8:-

```
user@user-VirtualBox:~$ passwd
Changing password for user.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
user@user-VirtualBox:~$ passwd
Changing password for user.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
You must choose a longer password
Enter new UNIX password:
Retype new UNIX password:
Sorry, passwords do not match
```

passwd: Authentication token manipulation error

passwd: password unchanged

user@user-VirtualBox:~$ passwd

Changing password for user.

(current) UNIX password:

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

user@user-VirtualBox:~$ AA

---

soln 9:-

1.$find . -name '*.c' -or -name '*.cpp'

2.umask u+w

sets the mask so that when files are created, they will have permissions which allow write permission for the user (file owner). The rest of the file's permissions would be unchanged from the operating system default.

Multiple changes can be specified by separating multiple sets of symbolic notation with commas (but not spaces!). For example:

umask u-x,g=r,o+w

This command will set the mask so that when subsequent files are created, they will have permissions that:

prohibit the execute permission from being set for the file's owner (user), while leaving the rest of the owner permissions unchanged;

enable read permission for the group, while prohibiting write and execute permission for the group;

enable write permission for others, while leaving the rest of the other permissions unchanged.

Note that if you use the equals operator ("="), any permissions not specified will be specifically prohibited. For example, the command

umask a=

Will set the file creation mask so that new files are inaccessible to everyone.

3.user@user-VirtualBox:~$ cd parent

user@user-VirtualBox:~/parent$ cd child

user@user-VirtualBox:~/parent/child$ ls

a.txt b.cpp c.cpp d.exe e.exe michel watson white y.txt

user@user-VirtualBox:~/parent/child$ cd ..

user@user-VirtualBox:~/parent$ rmdir -r child

rmdir: invalid option -- 'r'

Try 'rmdir --help' for more information.

user@user-VirtualBox:~/parent$ cd ..

user@user-VirtualBox:~$ rmdir -r parent

```
rmdir: invalid option -- 'r'
Try 'rmdir --help' for more information.
user@user-VirtualBox:~$ pwd
/home/user
user@user-VirtualBox:~$ rmdir -r /home/user/parent
rmdir: invalid option -- 'r'
Try 'rmdir --help' for more information.
user@user-VirtualBox:~$ rm -r parent
user@user-VirtualBox:~$
```
