**LINUX COMMANDS**

**Ex. No. 4**

**Date :**

# Basics Commands

* 1. echo SRM  to display the string SRM
  2. clear  to clear the screen
  3. date  to display the current date and time
  4. cal 2003  to display the calendar for the year 2003 cal 6 2003  to display the calendar for the June-2003
  5. passwd  to change password
  6. free –m  to view the size of RAM in MB free –g  to view the size of RAM in GB
  7. df –h  to view the disk space available and used.
  8. uptime  to view the system up time
  9. bc  to open a basic calculator
  10. ps  to view the current terminal running processes
  11. history  to get the history of all the past commands
  12. whoami  to know which user i am

# Working with Files

* 1. ls  list files in the present working directory

ls –l  list files with detailed information (long list) ls –a  list all files including the hidden files

ls –r root  list the directory recursively

ls –lh  list the current location content in human redable format ls –lt  to list the files based on modification time

ls –li  to view the inode number of files and directories lscpu  to view the system specifications

* 1. cat > f1  to create a file (Press ^d to finish typing)
  2. cat f1  display the content of the file f1
  3. wc f1  list no. of characters, words & lines of a file f1 wc –c f1  list only no. of characters of file f1

wc –w f1  list only no. of words of file f1

* 1. Output redirection

who > f2  the output of ‗who‘ will be redirected to file f2

* 1. cat >> f1  to append more into the file f1

# Piping

Syntax : Command1 | command2

Output of the command1 is transferred to the command2 as input. Finally output of the command2 will be displayed on the monitor.

ex. cat f1 | more  list the contents of file f1 screen by screen head –6 f1 |tail –2  prints the 5th & 6th lines of the file f1.

wc –l f1  list only no. of lines of file f1

* 1. cp f1 f2  copy file f1 into f2
  2. mv f1 f2  rename file f1 as f2
  3. rm f1  remove the file f1
  4. head –5 f1  list first 5 lines of the file f1 tail –5 f1  list last 5 lines of the file f1

# Working with Directories

* 1. mkdir elias  to create the directory elias
  2. cd elias  to change the directory as elias
  3. rmdir elias  to remove the directory elias
  4. pwd  to display the path of the present working directory
  5. cd  to go to the home directory cd ..  to go to the parent directory

cd -  to go to the previous working directory cd /  to go to the root directory

# File name substitution

* 1. ls f?  list files start with ‗f‘ and followed by any one character
  2. ls \*.c  list files with extension ‗c‘
  3. ls [gpy]et  list files whose first letter is any one of the character g, p

or y and followed by the word et

* 1. ls [a-d,l-m]ring  list files whose first letter is any one of the character

from a to d and l to m and followed by the word ring.

# I/O Redirection

* 1. Input redirection

wc –l < ex1  To find the number of lines of the file ‗ex1‘

# Environment variables

* 1. echo $HOME  display the path of the home directory
  2. echo $PS1  display the prompt string $
  3. echo $PS2  display the second prompt string ( > symbol by default )
  4. echo $LOGNAME  login name
  5. echo $PATH  list of pathname where the OS searches

for an executable file

# File Permission

-- chmod command is used to change the access permission of a file.

Method-1

Syntax : chmod [ugo] [+/-] [ rwxa ] filename

u : user, g : group, o : others

+ : Add permission - : Remove the permission r : read, w : write, x : execute, a : all permissions

ex. chmod ug+rw f1

adding ‗read & write‘ permissions of file f1 to both user and group members.

Method-2

Syntax : chmod octnum file1

The 3 digit octal number represents as follows

* first digit -- file permissions for the user
* second digit -- file permissions for the group
* third digit -- file permissions for others

Each digit is specified as the sum of following

4 – read permission, 2 – write permission, 1 – execute permission

ex. chmod 754 f1

it change the file permission for the file as follows

* read, write & execute permissions for the user ie; 4+2+1 = 7
* read, & execute permissions for the group members ie; 4+0+1 = 5
* only read permission for others ie; 4+0+0 = 4

QUESTIONS FOR PRACTICE:

**Q1.** Write a command to cut 5 to 8 characters of the file *f1*

$

**Q2.** Write a command to display user-id of all the users in your system.

$

**Q3.** Write a command to paste all the lines of the file *f1* into single line

$

**Q4.** Write a command to cut the first field of file *f1* and second field of file *f2* and paste intothe file *f3*.

$

**Q5.** Write a command to change all small case letters to capitals of file *f2*.

$

**Q6.** Write a command to replace all *tab* character in the file *f2* by **:**

# $

**Q7.** Write a command to check whether the user j*udith* is available in your system or not.(use grep)

$

**Q8.** Write a command to display the lines of the file *f1* starts with SRM.

$

**Q9.** Write a command to display the name of the files in the directory

*/etc/init.d* thatcontains the pattern *grep*.

$

**Q10.** Write a command to display the names of nologin users. (Hint: the command *nologin*

is specified in the last filed of the file /etc/passwd for nologin users)

$

**Q11.** Write a command to sort the file /etc/passwd in descending order

$

**Q12.** Write a command to sort the file /etc/passwd by user-id numerically. (Hint : user-id isin 3rd field)

$

**Q13.** Write a command to sort the file *f2* and write the output into the file

*f22*. Alsoeliminate duplicate lines.

$

**Q14.** Write a command to display the unique lines of the sorted file *f21*. Also display thenumber of occurrences of each line.

$

**Q15.** Write a command to display the lines that are common to the files *f1* and *f2*.

$

# 

**SYSTEM ADMIN COMMANDS**

**Ex. No. 5**

**Date:**

# INSTALLING SOFTWARE

**Procedure:**

* Open the Ubuntu software Center.
* To install any package, open the terminal (Ctrl + Alt + T) and type sudo apt-get install

<package name>.

* For instance, to get Chrome type sudo apt-get install chrome-browser.
* Likewise user can work package update, remove and reinstall the package using the following commands.

To update the package repositories

sudo apt-get update

To update installed software

sudo apt-get upgrade

To install a package/software

sudo apt-get install <package-name>

To remove a package from the system

sudo apt-get remove <package-name>

To reinstall a package

sudo apt-get install <package-name> --reinstall

To completely remove a software and it‘s dependent packages run the apt-get purge

sudo apt-get purge *<package-name>*

To remove all Debian (.deb) files those are no longer installed

---files in /var/cache/apt/archives

sudo apt-get autoclean

To empty whole cache files – to reduce the space consumption

sudo apt-get clean

To remove old dependent files and footprints installed by previous applications

sudo apt-get automove

To configure installed package

sudo dpkg –configure –a

To download but not install package

sudoapt-get download <package-name>

# MANAGING USERS

* Managing users is a critical aspect of server management.
* In Ubuntu, the root user is disabled for safety.
* Root access can be completed by using the sudo command by a user who is in the ―admin‖ group.
* When you create a user during installation, that user is added automatically to the admin group.

To add a user:

sudo adduser username

To disable a user:

sudo passwd -l username

To enable a user:

sudo passwd -u username

To delete a user:

sudo userdel –r username

To create a group:

sudo addgroup groupname

To delete a group:

sudo delgroup groupname

To create a user with group:

sudo adduser username groupname

To see the password expiry value for a user,

sudo chage -l username

To make changes:

sudo chage username

# GUI TOOL FOR USER MANAGEMENT

GUI Tool allow the admin to run the commands in terminal to manage users and groups.

To install a GUI add-on

sudo apt install gnome-system-tools

Once done, type

users-admin

# MANAGING THE FILE SYSTEM

A filesystem is a permanent storage for containing data. Any non-volatile storage device like hard disk, usb etc has a filesystem in place, on top of which data is stored. While installing Linux, you may opt for either EXT4 or EXT3 file system.

**Ext3 :** A journaling filesystem: logs changes in a journal to increase reliability in case of power failure or system crash.

EXT4: It is an advanced file syste. This file system supports 64-bit storage limits, columns up to 1 exabytes and you may store files up to 16 terabytes

Disk Partitions can be viewed by the command sudo fdisk -l

File system information are available in the file /etc/fstab

# MANGING THE NETWORK CONFIGURATION

Most networking is configured by editing two files:

* /etc/network/interfaces
  + Ethernet, TCP/IP, bridging
* /etc/resolv.conf
  + DNS

Other networking files:

* /etc/hosts
* /etc/dhcp3/dhcpd.conf

To test any host‘s connectivity

ping <ip-address>

To start/stop/restart/reload networking services

sudo /etc/init.d/mnetworking <function>

Note : <function> can be any one of stop or start or reload or restart

To list of all active network interface cards, including wireless and the loopback interface

sudo ifconfig

To display host Fully Qualified Domain Name

sudo hostname

To display arp table (ip to mac resolution)

sudo arp –a

To remove entry from arp table

sudo arp –d <user name>

To display or change network card settings, use ethtool

sudo ethtool eth0

To displays extensive status information when queried with the service iptables status command

sudo service iptables status

To start/stop services

sudo service iptables start/stop

# INSTALLING INTERNET SERVICES

Installing Apache server

sudo apt-get install apache2

Configuration file for Apache server

apache2.conf

Restart apache services after any configuration changes made

sudo /etc/init.d/mnetworking restart

Similarly all services can be installed, configured and restarted

# MANAGING BACKGROUND JOBS

To display jobs running in background

sudo jobs

To check the process id of background processes

sudo jobs –p

To bring a background job to the foreground

sudo fg

To start the Jobs suspended in background

sudo bg

**QUESTIONS FOR PRACTICE:**

Q1. Update the package repositories

Q2. Install the package ―simplescreenrecorder

Q3. Remove the package ―simplescreenrecorder‖

Q4. Create a user ‗elias‘. Login to the newly created user and exit.

Q5. Disable the user ‗elias‘, try to login and enable again.

Q6. Create a group ‗cse‘ and add the user ‗elias‘ in that group

Q7. List the account expiry information of the user ‗elias‘

Q8. Change the ‗Number of days warning before password expires‘ as 5 for the user ‗elias‘

Q9. Delete the user ‗elias‘ and then delete the group ‗cse‘

Q10. List the partitions available in your system

Q11. What are the file systems used in your system

Q12. Stop the networking service and then start the service

Q13. Check the connectivity of the host with IP address 127.0.0.1

Q14. Find the IP address of the localhost

Q15. Find the IP address of the DNS Server (name server)

Q16. Install mysql server

Q17. Restart mysql server

Q18. Check the configuration file for mysql server

Q19. Log on as root into mysql server

Q20. Create a new database for mysql server

.

**SIMPLE TASK AUTOMATION**

**Ex. No. 6**

**Date:**

**Crontab**

Linux Cron utility is an effective way to schedule a routine background job at a specific time and/or day on an on-going basis. User can use this to schedule activities, either as one- time events or as recurring tasks.

**Scheduling of Tasks (For Ubuntu)**

Step 1 : Open terminal and type the command crontab –e

Step 2 : Choose the editor. Better to select nano editor Step 3 : Edit the file based on the syntax given above Step 4 : Save and Exit the file

Step 5 : Start cron daemon using the following command

systemctl start cron

# Linux Crontab Format

MIN HOUR DOM MON DOW CMD

**Table: Crontab Fields and Allowed Ranges (Linux Crontab Syntax)**

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **Allowed Value** |
| MIN | Minute field | 0 to 59 |
| HOUR | Hour field | 0 to 23 |
| DOM | Day of Month | 1-31 |
| MON | Month field | 1-12 |
| DOW | Day Of Week | 0-6 |
| CMD | Command | Any command to be executed |

# Create a new crontab file, or edit an existing file

# crontab -e [username]

where *username* specifies the name of the user's account for which you want to create or edit a crontab file.

# Verify your crontab file changes

# crontab -l [username] **Install crontab** crontab -a filename **Edit the crontab**

#crontab -e **Display crontab** crontab -l

# Display the last edit the crontab file

crontab -v **Remove crontab** crontab -r

# Following are the syntax for cron

minute(s) hour(s) day(s) month(s) weekday(s) command(s) "Argument1" "Argument2" 1 \* 3 4 5 /path/to/command arg1 arg2

If you don't have parameter put star(\*) Commands:

1. **-l -** List or manage the task with crontab command
2. **-e -** edit crontab entry.
3. **-u -** To list scheduled jobs of a particular user called **tecmint** using.
4. **-r -** parameter will remove complete scheduled jobs without confirmation from crontab.
5. **-i -** prompt you confirmation from user before deleting user‘s crontab.

Allowed special character (\*, -, /, ?, #)

1. **Asterik(\*)** – Match all values in the field or any possible value.
2. **Hyphen(-)** – To define range.
3. **Slash (/)** – 1st field /10 meaning every ten minute or increment of range.
4. **Comma (,)** – To separate items.

# System Wide Cron Schedule

System administrator can use predefine cron directory as shown below.

1. /etc/cron.d
2. /etc/cron.daily
3. /etc/cron.hourly
4. /etc/cron.monthly
5. /etc/cron.weekly

# To Schedule a Job for Specific Time

The below jobs delete empty files and directory from **/tmp** at **12:30** am daily. User need to mention user name to perform crontab command.

In below example **root** user is performing cron job.

# crontab –e

30 0 \* \* \* root find /tmp -type f -empty –delete

# Special Strings for Common Schedule

|  |  |
| --- | --- |
| **Strings** | **Meanings** |
| @reboot | Command will run when the system reboot. |
| @daily | Once per day or may use @midnight. |
| @weekly | Once per week. |
| @yearly | Once per year.user can use @annually keyword also. |

**Multiple Commands with Double ampersand (&&)**

To run the command1 and command2 daily

**#** crontab -e

@daily <command1> && <command2>

# Disable Email Notification.

By default cron send mail to user account executing cronjob. If user want to disable using **>/dev/null 2>&1** option at the end of the file will redirect all the output of the cron results under **/dev/null**.

[root@tecmint ~]# crontab -e

\* \* \* \* \* >/dev/null 2>&1

# Scheduling a Job for a Specific Time

The basic usage of cron is to execute a job in a specific time as shown below. This will execute the full backup shell script (full-backup) on **10th June 08:30 AM**.

The below time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20. 30 08 10 06 \* /home/username/full-backup

* **30** – 30th Minute
* **08** – 08 AM
* **10** – 10th Day
* **06** – 6th Month (June)
* **\*** – Every day of the week

# Schedule a Job for More Than One Instance (e.g. Twice a Day)

The following script takes a incremental backup twice a day every day. This example executes the specified incremental backup shell script (incremental-backup) at 11:00 and 16:00 on every day. The comma separated value in a field specifies that the command needs to be executed in all the mentioned time.

00 11,16 \* \* \* /home/username/bin/incremental-backup

* **00** – 0th Minute (Top of the hour)
* **11,16** – 11 AM and 4 PM
* **\*** – Every day
* **\*** – Every month
* **\*** – Every day of the week

# Schedule a Job for Specific Range of Time (e.g. Only on Weekdays)

* + To schedule the job for every hour with in a specific range of time then use the following.

Cron Job everyday during working hours

This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

00 09-18 \* \* \* /home/username/bin/check-db-status

* **00** – 0th Minute (Top of the hour)
* **09-18** – 9 am, 10 am,11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
* **\*** – Every day
* **\*** – Every month
* **\*** – Every day of the week

# Schedule a Job for Every Minute Using Cron.

Ideally user may not have a requirement to schedule a job every minute. But understanding this example will help user understand the other examples mentioned below in this article.

\* \* \* \* \* CMD

The \* means all the possible unit — i.e every minute of every hour throughout the year. More than using this \* directly, user will find it very useful in the following cases.

* When user specify \*/5 in minute field means every 5 minutes.
* When user specify 0-10/2 in minute field mean every 2 minutes in the first 10 minute.
* Thus the above convention can be used for all the other 4 fields.

# Schedule a Background Cron Job For Every 10 Minutes.

Use the following, to check the disk space every 10 minutes.

\*/10 \* \* \* \* /home/username/check-disk-space

It executes the specified command check-disk-space every 10 minutes throughout the year.

There are special cases in which instead of the above 5 fields you can use @ followed by a keyword — such as reboot, midnight, yearly, hourly.

Table: Cron special keywords and its meaning

|  |  |
| --- | --- |
| **Keyword** | **Equivalent** |
| @yearly | 0 0 1 1 \* |
| @daily | 0 0 \* \* \* |
| @hourly | 0 \* \* \* \* |
| @reboot | Run at startup. |

# Schedule a Job for First Minute of Every Year using @yearly

User can specify a job to be executed on the first minute of every year, then user can use the **@yearly** cron keyword as shown below.

This will execute the system annual maintenance using annual-maintenance shell script at 00:00 on Jan 1st for every year.

@yearly /home/username/red-hat/bin/annual-maintenance

# Schedule a Cron Job Beginning of Every Month using @monthly

Executes the command monthly once using **@monthly** cron keyword.

This will execute the shell script tape-backup at 00:00 on 1st of every month. @monthly /home/username/suse/bin/tape-backup

# Schedule a Background Job Every Day using @daily

Using the @daily cron keyword, this will do a daily log file cleanup using cleanup-logs shell scriptat 00:00 on every day.

@daily /home/username/arch-linux/bin/cleanup-logs "day started"

# To Execute a Linux Command After Every Reboot using @reboot

Using the **@reboot** cron keyword, this will execute the specified command once after the machine got booted every time.

@reboot CMD

# To Disable/Redirect the Crontab Mail Output using MAIL keyword

By default crontab sends the job output to the user who scheduled the job. To redirect the output to a specific user, add or update the MAIL variable in the crontab as shown below.

username@dev-db$ crontab -l MAIL="username"

@yearly /home/username/annual-maintenance

\*/10 \* \* \* \* /home/username/check-disk-space

[Note: Crontab of the current logged in user with MAIL variable]

To stop the crontab output to be emailed, add or update the MAIL variable in the crontab as shown below.

MAIL=""

# Specify PATH Variable in the Crontab

To set absolute path of the Linux command or the shell-script :

Instead of specifying /home/username/tape-backup, user can specify tape-backup, then add the path

/home/username to the PATH variable in the crontab as shown below.

username@dev-db$ crontab -l PATH=/bin:/sbin:/usr/bin:/usr/sbin:/home/username @yearly annual-maintenance

\*/10 \* \* \* \* check-disk-space

[Note: Crontab of the current logged in user with PATH variable]

# To Install Crontab from a Cron File

Instead of directly editing the crontab file, user can also add all the entries to a cron-file first.

Once userhave all thoese entries in the file, user can upload or install them to the cron as shown below. username@dev-db$ crontab -l

no crontab for username

$ cat cron-file.txt

@yearly /home/username/annual-maintenance

\*/10 \* \* \* \* /home/username/check-disk-space username@dev-db$ crontab cron-file.txt username@dev-db$ crontab -l

@yearly /home/username/annual-maintenance

\*/10 \* \* \* \* /home/

username/check-disk-space

**To View Crontab Entries:**

View Current Logged-In User‘s Crontab entries To view crontab entries type

crontab -l

Username@dev-db$ crontab -l

@yearly /home/username/annual-maintenance

\*/10 \* \* \* \* /home/username/check-disk-space

[Note: This displays crontab of the current logged in user]

# To View Root Crontab entries

Login as root user (su – root) and do crontab -l as shown below. root@dev-db# crontab -l

no crontab for root

# To View Other Linux User’s Crontabs entries:

To view crontab entries of other Linux users, login to root and use **-u {username} -l**

root@dev-db# crontab -u username -l @monthly /home/username/monthly-backup

00 09-18 \* \* \* /home/username/check-db-status

# To Edit Crontab Entries:

Edit Current Logged-In User‘s Crontab entries To edit a crontab entries,

use crontab –e

By default this will edit the current logged-in users crontab. username@dev-db$ crontab -e

@yearly /home/username/centos/bin/annual-maintenance

\*/10 \* \* \* \* /home/username/debian/bin/check-disk-space "/tmp/crontab.XXXXyjWkHw" 2L, 83C

[Note: This will open the crontab file in Vim editor for editing.

Please note cron created a temporary /tmp/crontab.XX... ]

When user save the above temporary file with :wq, it will save the crontab and display the following message indicating the crontab is successfully modified.

QUESTIONS FOR PRACTICE:

Q1. Schedule a task to display the following message on the monitor for every 2 minutes.

Q2. Schedule a task to take backup of your important file (say file f1) for every 30 minutes

Q3. Schedule a task to take backup of login information everyday 9:30am

# Ex. No. 7 SHELL PROGRAMS

**Date:**

# Aim:

**How to run a Shell Script**

* Edit and save your program using editor
* Add execute permission by *chmod* command
* Run your program using the name of your program

./program-name

# Important Hints

* No space before and after the assignment operator Ex. sum=0
* *Single quote* ignores all special characters. Dollar sign, Back quote and Back slash are not ignored inside *Double quote*. *Back quote* is used as command substitution. *Back slash* is used to remove the special meaning of a character.
* Arithmetic expression can be written as follows : i=$((i+1) or i=$(expr

$i + 1)

* Command line arguments are referred inside the program as $1, $2, ..and so on
* $\* represents all arguments, $# specifies the number of arguments
* read statement is used to get input from input device. Ex. read a b

**Syntax for if statement**

if [ condition ] then

...

elif [ condition ] then

else

...

fi ...

**Syntax for case structure**

case value in pat1) ...

pat2) ...

\*) ...

esac

**Syntax for for-loop**

statement;; Statement;; Statement;;

**for** var **in** list-of-values

**do**

**done**

...

...

**Syntax for printf statement**

printf “string and format” arg1 arg2 … …

* Break and continue statements functions similar to C programming
* Relational operators are –lt, -le, -gt, -ge, -eq,-ne
* Ex. (i>= 10) is written as [ $i -ge 10 ]
* Logical operators (and, or, not) are -o, -a, !
* Ex. (a>b) && (a>c) is written as [ $a –gt $b –a $a –gt $c ]
* Two strings can be compared using = operator

**Q1.** Given the following values

num=10, x=\*, y=`date` a="Hello, 'he said'"

**Execute and write the output of the following commands**

|  |  |
| --- | --- |
| **Command** | **Output** |
| ***echo num*** |  |
| ***echo $num*** |  |
| ***echo $x*** |  |
| ***echo ‘$x’*** |  |
| ***echo “$x”*** |  |
| ***echo $y*** |  |
| ***echo $(date)*** |  |
| ***echo $a*** |  |
| ***echo \$num*** |  |
| ***echo \$$num*** |  |

**Q2. Find the output of the following shell scripts**

**$** vi ex51

echo Enter value for n read n

sum=0 i=1

while [ $i –le $n ] do

done

sum=$((sum+i)) i=$((i+2))

echo Sum is $sum

# Output :

**Q3. Write a program to check whether the file has execute permission or not. If not, add thepermission**.

$ vi ex52

echo Enter name of the file

read name

if [ -x $name ]

then

echo Yes $name has Execute Permission

else

echo No $name has NO Execute Permission

fi

**Output:**

# Q4. Write a shell script to print a greeting as specified below.

If hour is greater than or equal to 0 (midnight) and less than or equal to 11 (up to 11:59:59), "Good morning" is displayed.

If hour is greater than or equal to 12 (noon) and less than or equal to 17 (up to 5:59:59 p.m.), "Good afternoon" is displayed.

If neither of the preceding two conditions is satisfied, "Good evening" is displayed.

**$** vi ex53

hour=$(date | cut -c12-13)

if [ "$hour" -ge 0 -a "$hour" -le 11 ]

then

echo Good Morning

elif [ $hour -le 17 ]

then

echo Good Afternoon

else

echo Good Evening

fi

**Output:**

**Q5. Write a shell script to list only the name of sub directories in the present working directory**

$ vi ex54

for i in \*

do

if [ -d $i ]

then

echo $i

fi

done

**Output:**

# Q6. Write a program to check all the files in the present working directory for a pattern (passed through command line) and display the name of the file followed by a message stating that the pattern is available or not available.

$ vi ex55

for i in \*

do

if [ -f $i ]

then

grep $1 $i > /dev/null

if [ $? -eq 0 ]

then

echo $i found

else

echo $i not found

fi

fi

done

**Output:**

**Result**:

**Ex. No. 8 PROCESS CREATION**

**Date:**

**Aim:**

**Syntax for process creation**

int fork();

Returns 0 in child process and child process ID in parent process.

**Other Related Functions**

int getpid() → returns the current process ID int

getppid() → returns the parent process ID

wait() → makes a process wait for other process to complete

**Virtual fork**

vfork() function is similar to fork but both processes shares the same address space.

**Q1. Find the output of the following program**

#include <stdio.h>

#include<unistd.h>

int main()

{

int a=5,b=10,pid;

printf("Before fork a=%d b=%d \n",a,b); pid=fork();

if(pid==0)

{

a=a+1;

b=b+1;

printf("In child a=%d b=%d \n",a,b);

}

else

{

sleep(1);

a=a-1;

b=b-1;

printf("In Parent a=%d b=%d \n",a,b);

}

return 0;

}

**Output :-**

**Q2. Rewrite the program in Q1 using vfork() and write the output**

#include <stdio.h>

#include<unistd.h>

int main()

{

int a=5,b=10,pid;

printf("Before fork a=%d b=%d \n",a,b);

pid=vfork();

if(pid==0)

{

a=a+1;

b=b+1;

printf("In child a=%d b=%d \n",a,b);

}

else

{

sleep(1);

a=a-1;

b=b-1;

printf("In Parent a=%d b=%d \n",a,b);

}

return 0;

}

**Output :**

**Q3. Calculate the number of times the text “SRMIST” is printed.**

#include <stdio.h>

#include<unistd.h>

int main()

{

fork();

fork();

fork();

printf(“SRMIST\n”);

return 0;

}

**Output :**

**Q4. Complete the following program as described below :**

The child process calculates the sum of odd numbers and the parent process calculate the sum of even numbers up to the number ‘n’. Ensure the Parent process waits for the child process to finish.

#include <stdio.h>

#include<unistd.h>

int main()

{

int pid,n,oddsum=0,evensum=0;

printf("Enter the value of n : ",a); scanf(“%d”,&n);

pid=fork();

// Complete the program

if (pid==0)

{

for(i=1;i<=n;i=i+2)

oddsum+=i;

printf("Sum of odd number is %d\n",oddsum);

}

else

{

sleep(1);

for(i=0;i<=n;i=i+2)

evensum+=i;

printf("Sum of even number is %d\n",evensum);

}

return 0;

}

**Output :**

**Q5. How many child processes are created for the following code?**

Hint : Check with small values of ‘n’.

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

fork();

**Output :**

**Q6. Write a program to print the Child process ID and Parent process ID in both Child and Parent processes**

#include <stdio.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==0)

{

printf("In child......\n child ID is %d Parents ID is %d \n",getpid(),getppid());

}

else

{

sleep(1);

printf("In parents ......\n child Id is %d parent ID is %d \n",pid,getpid());

}

return 0;

}

**Output:**

**Q7. How many child processes are created for the following code?**

#include <stdio.h>

#include<unistd.h>

int main()

{

fork();

fork()&&fork()||fork();fork();

printf(“Yes ”);

return 0;

}

**Output :**

**Result:**