**EX NO:3 UNIX SYSTEM CALLS**

**DATE:11.03.2021**

**Aim:**To write and execute system calls in Unix environment

**Programs:**

**1.Write a program to execute the following system calls: fork(), getpid(), getppid()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3.** Using header files,call the needed headers for the program

**4**. Using ‘if’ command, check whether the program id is lesser than or equal to zero.

**5.**If program id is equal to zero display child and parent process’s id,orelse display error.

**6.**Using cc filename.c and ./a.out display the output.

**7.** Stop.

**Program:**

# include <stdio.h>

# include <sys/types.h>

# include <stdlib.h>

# include <unistd.h>

void main()

{

pid\_tprgmid;

prgmid=fork();

if(prgmid<0)

{

printf("error");

exit(0);

}

else if(prgmid==0)

{

printf("child process");

printf("%d",getpid);

printf("%d",getppid);

}

else

{

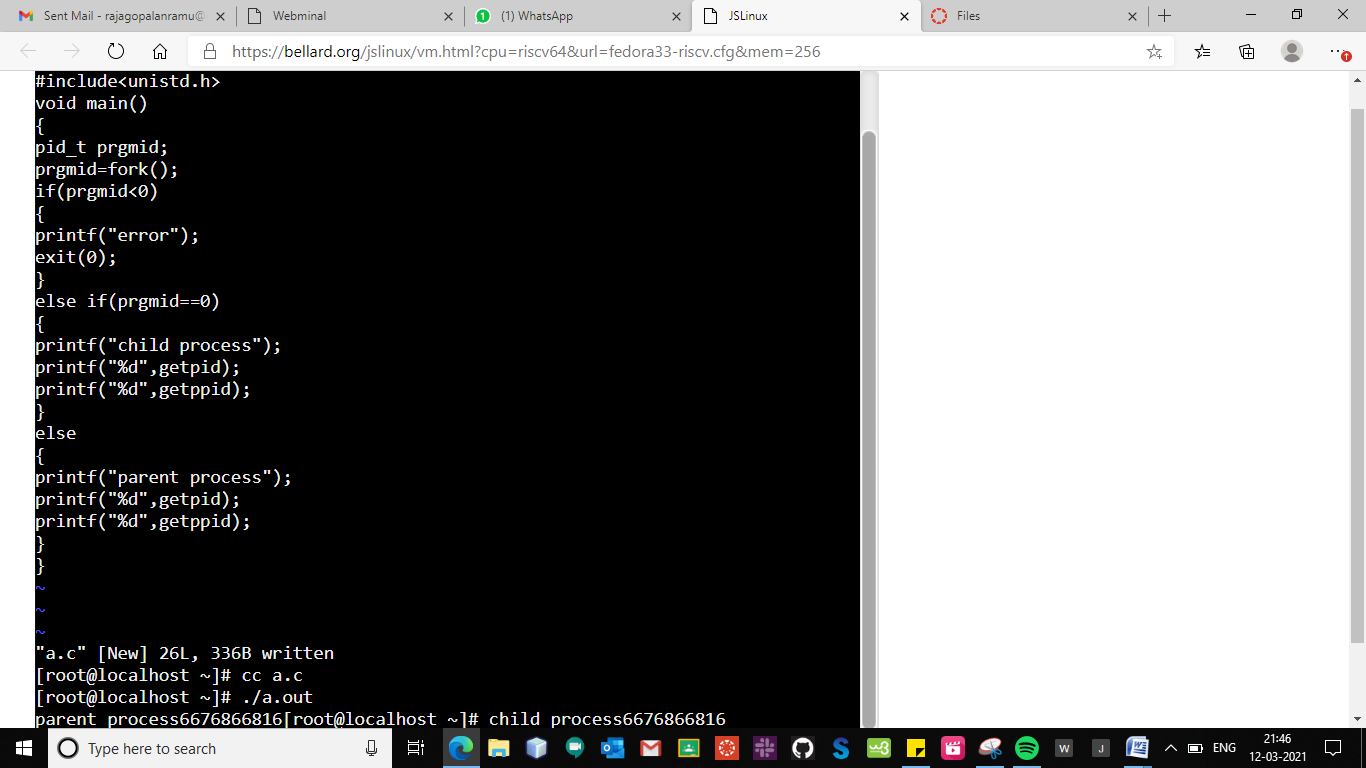
printf("parent process");

printf("%d",getpid);

printf("%d",getppid);

}

**Output:**



**2.Write a program to execute the following system calls:wait();**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Using header files,call the needed headers for the program

**4.**If program id is lesser than zero,display creation failed, else if program id is greater than zero,display content of parent process and end it,else display content of child process and end it.

**5.**Using cc filename.c and ./a.out display the output.

**6.**Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

void main()

{

inti;

pid\_tpid;

pid=fork();

if(pid<0)

{

printf("Creation failed");

exit(-1);

}

else if(pid>0)

{

wait(NULL);

printf("\n Parent starts thus printing even numbers");

for(i=2;i<=10;i+=2)

{

printf("\n %d ",i);

}

printf("\n End of parent process");

}

else

{

printf("\n Child starts thus printing odd numbers");

for(i=1;i<=9;i++)

{

if(i%2!=0)

{

printf("\n %d ",i);

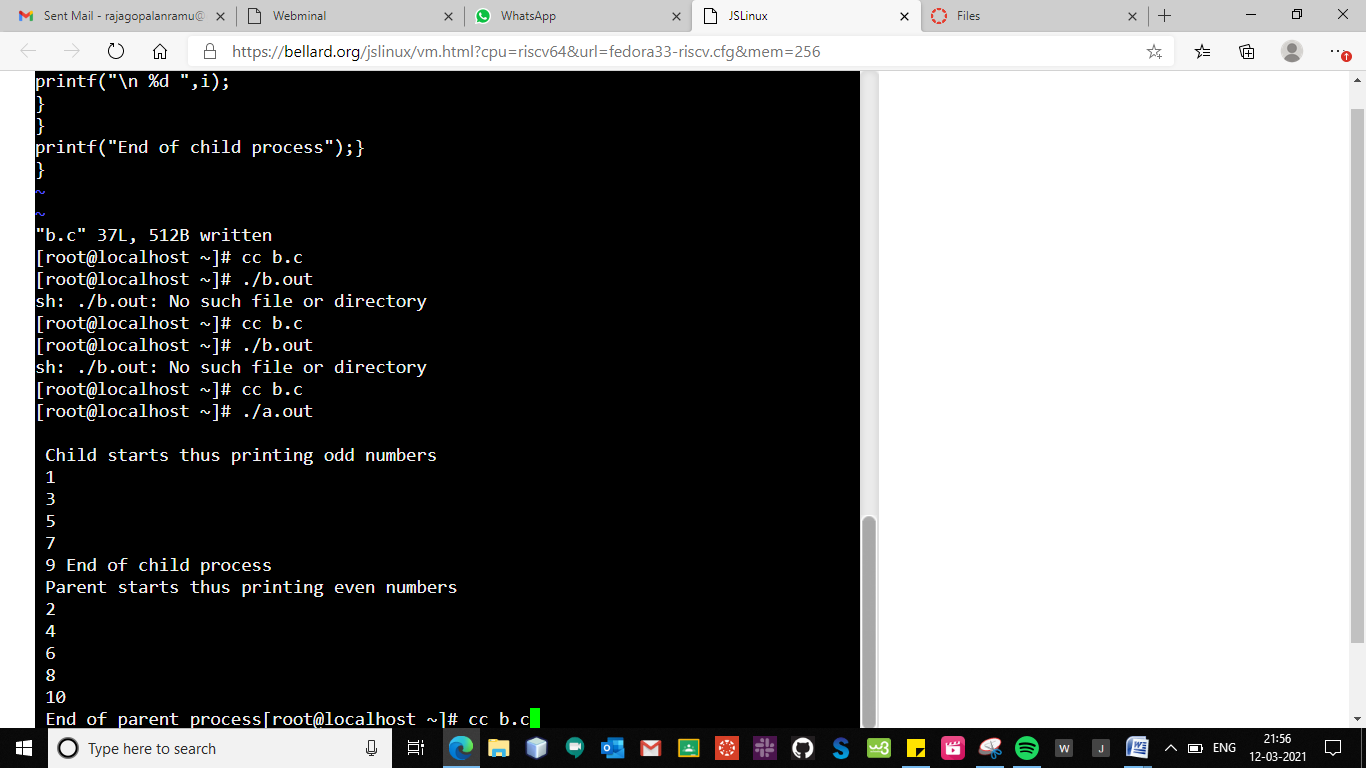
}

}

printf("End of child process");}

}

**Output:**



**3.Write a program to execute the following system calls:exec();**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Using ‘exec’ command, displays the commands inside the brackets.

**4.** Using cc filename.c and ./a.out display the output.

**5.**Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

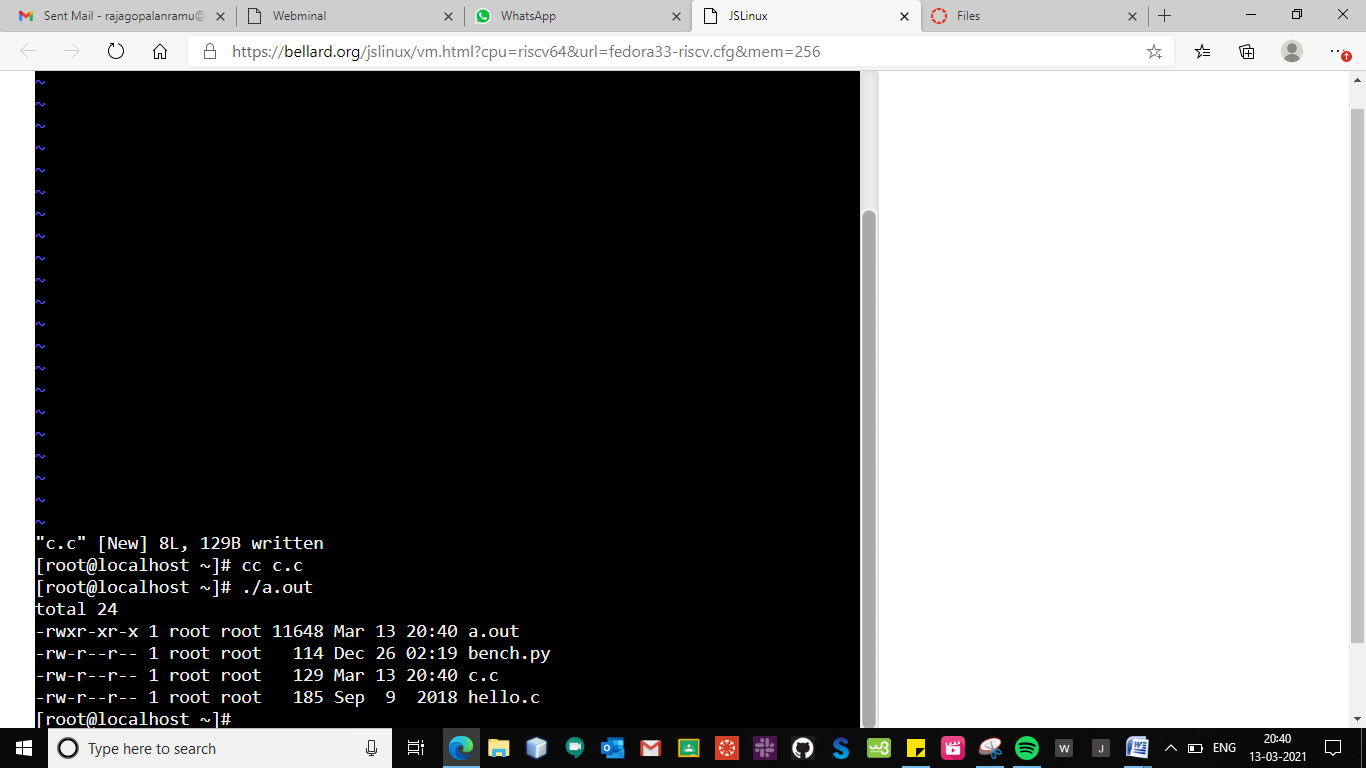
void main()

{

execl("/bin/ls","ls","-l",0,NULL);

}

**Output:**

****

**4.Write a program to execute the following system calls:stat();**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Using ‘stat’ command, ,if file and address of file is equal to one it shows that the file does not exist,else displays the information of the file given.

**4.** Using cc filename.c and ./a.out display the output.

**5.**Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<sys/stat.h>

#include<sys/statvfs.h>

#include<sys/types.h>

void main()

{

struct stat file;

int n;

if((n=stat("c.c",&file))==-1)

{

perror("File doesn't exist");

exit(-1);

}

printf("User id: %u \n",file.st\_uid);

printf("Group id: %u \n",file.st\_gid);

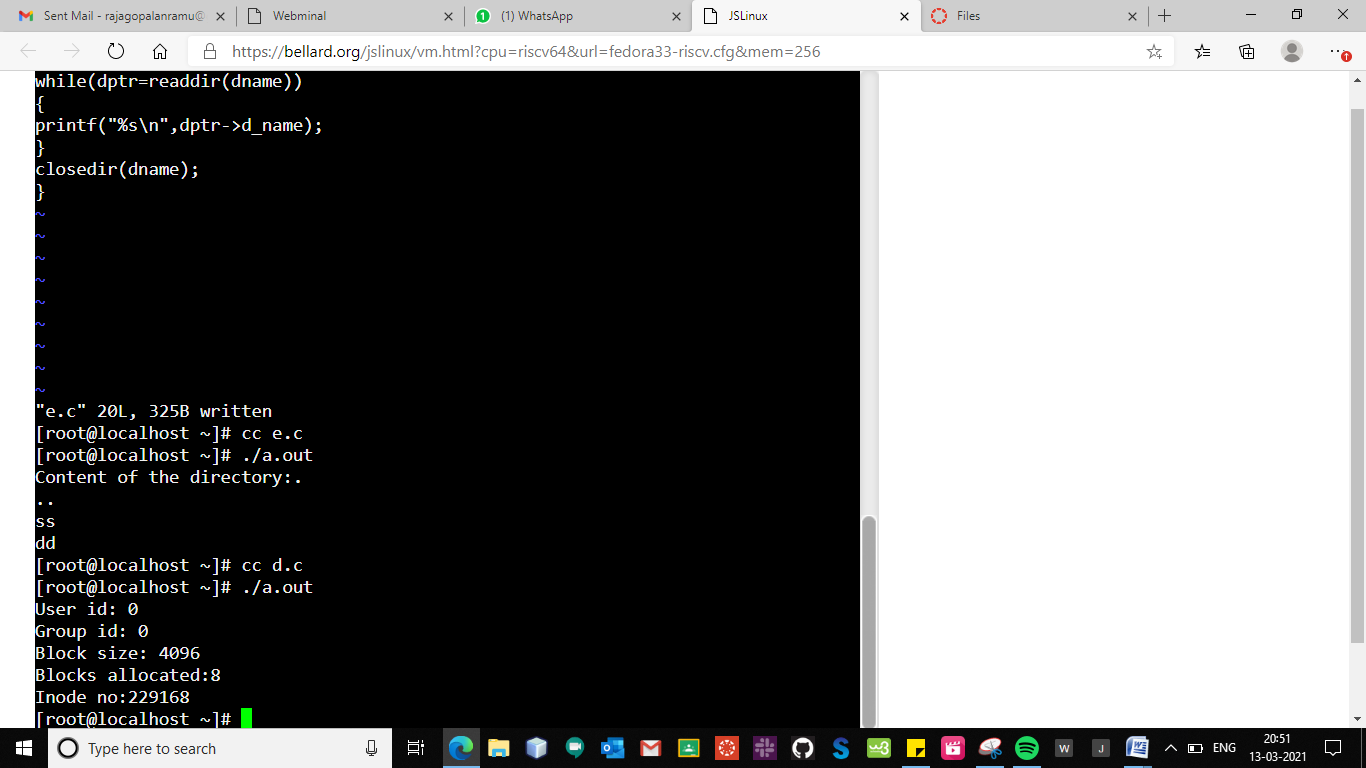
printf("Block size: %ld \n",file.st\_blksize);

printf("Blocks allocated:%ld \n",file.st\_blocks);

printf("Inode no:%ld \n",file.st\_ino);

}

**Output:**

****

**5.Write a program to execute the following system calls:readdir(),opendir();**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Using ‘opendir and readdir’ commands,we can open a file and read the contents in the file.

**4.** Using cc filename.c and ./a.out display the output.

**5.**Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<dirent.h>

#include<sys/types.h>

void main()

{

struct dirent \*dptr;

dir \*dname;

if((dname=opendir("dir1"))==NULL)

{

perror("Directory does not exist");

exit(-1);

}

while(dptr=readdir(dname))

{

printf("Contents of the opened directory: ");

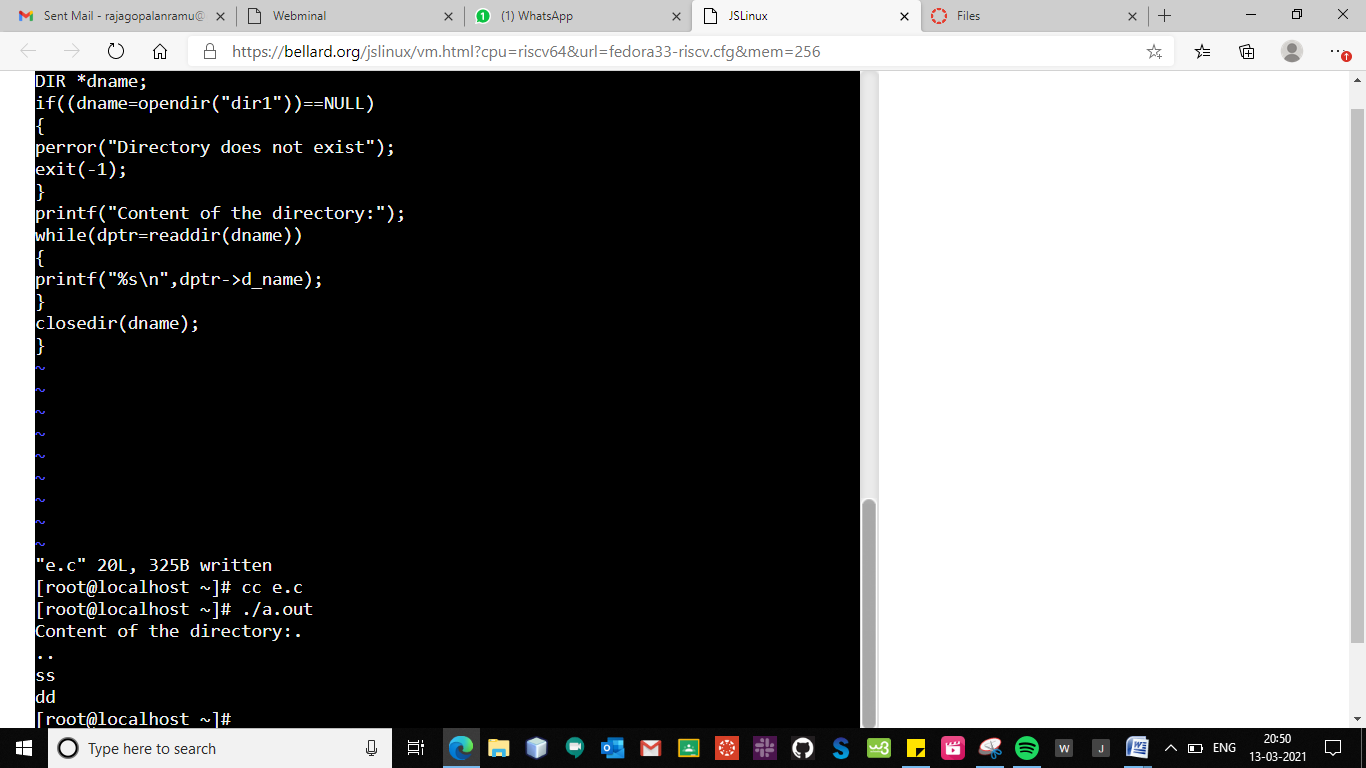
printf("%s\n",dptr->d\_name);

}

closedir(dname);

}

**Output:**



|  |  |
| --- | --- |
| **Observation(20)** |  |
| **Record(5)** |  |
| **Total(25)** |  |
| **Initial** |  |

**Result:**

Thus the basic Unix system calls were executed and outputs were noted.