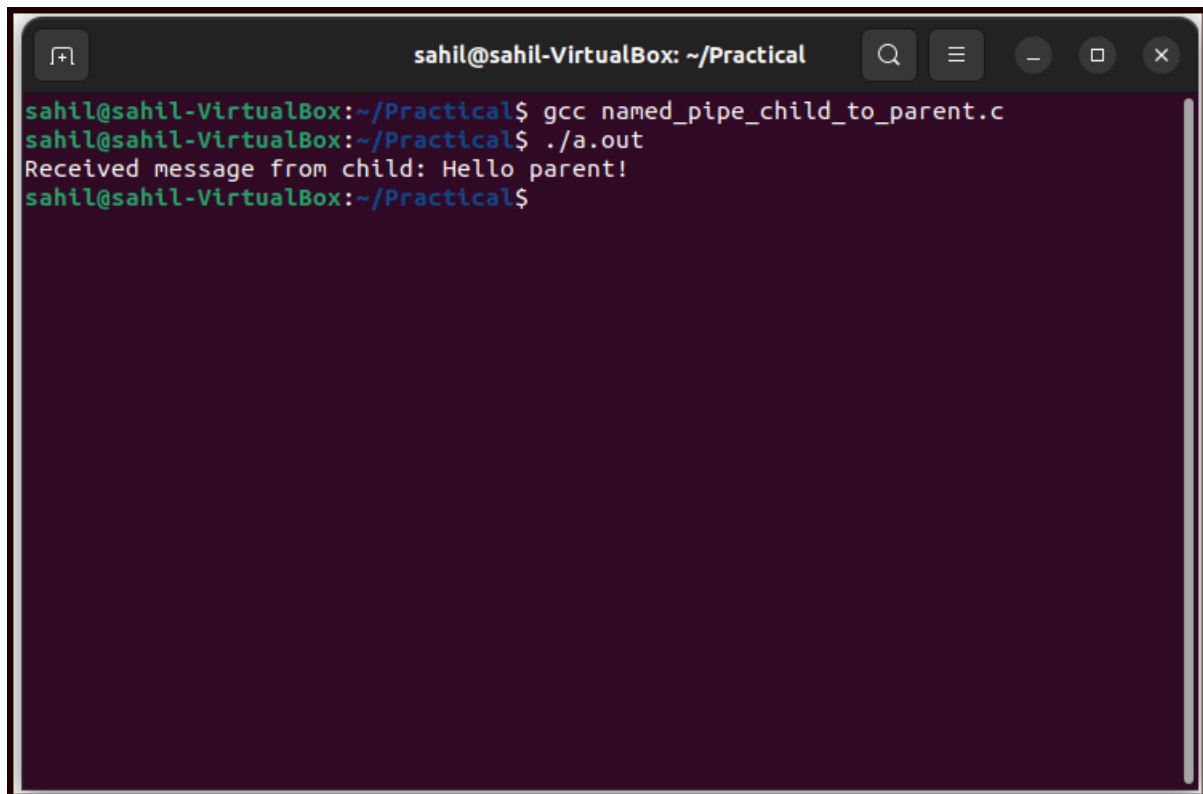


1. Write a program to create a named pipe where child sends message to parent

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
int main() {
    int fd,pid;
    char * myfifo = "namedpipe";
    char buf[256];
    /* create the named pipe */
    mkfifo(myfifo, 0666);
    pid = fork();
    if (pid > 0)
    {
        fd = open(myfifo, O_RDONLY);
        read(fd, buf, sizeof(buf));
        printf("Received message from child: %s\n", buf);
        close(fd);
        unlink(myfifo);
        _exit(0);
    }
    else
    {
        fd = open(myfifo, O_WRONLY);
        write(fd, "Hello parent!", sizeof("Hello parent!"));
        close(fd);
        _exit(0);
    }
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc named_pipe_child_to_parent.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
Received message from child: Hello parent!
sahil@sahil-VirtualBox:~/Practical$
```

2. Write a program in C to print the file names of a specified directory.

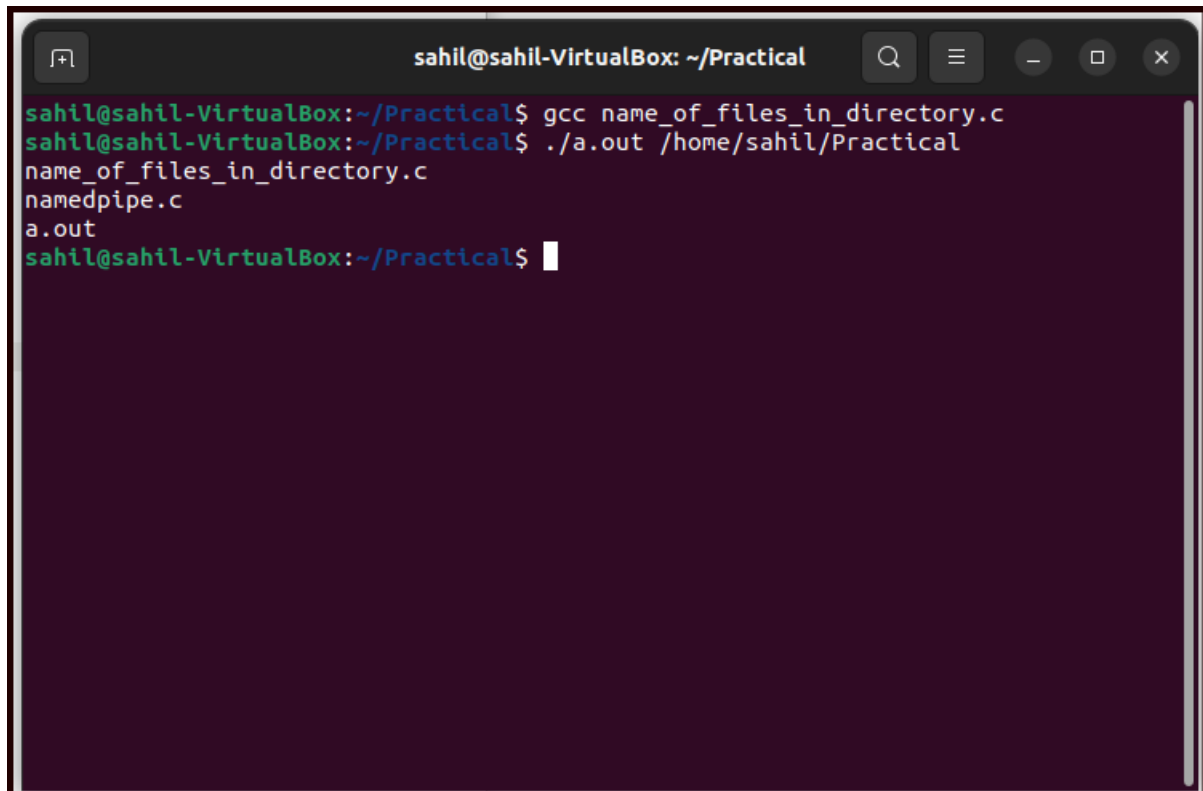
Code :

```
#include<stdio.h>
#include<dirent.h>
#include<unistd.h>
void main(int argc ,char *argv[])

{
    DIR *dp;
    struct dirent *entry;
    dp = opendir(argv[1]);
    if (dp == NULL) {
        printf("Directory does not exists.\n");
        _exit(0);
    }
    while (entry = readdir(dp))
        if (entry->d_type == DT_REG)
            printf("%s\n",entry->d_name);
    closedir(dp);
}
```

}

Output :

A terminal window titled 'sahil@sahil-VirtualBox: ~/Practical' with standard window controls. The terminal shows the following commands and output:

```
sahil@sahil-VirtualBox:~/Practical$ gcc name_of_files_in_directory.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out /home/sahil/Practical
name_of_files_in_directory.c
namedpipe.c
a.out
sahil@sahil-VirtualBox:~/Practical$
```

3. Write a program to catch SIGINT signal five times and print message 'SIGINT signal occurred ' every timed and exit at sixth occurrence. Also ignore every occurrence of SIGTSTP signal

Code :

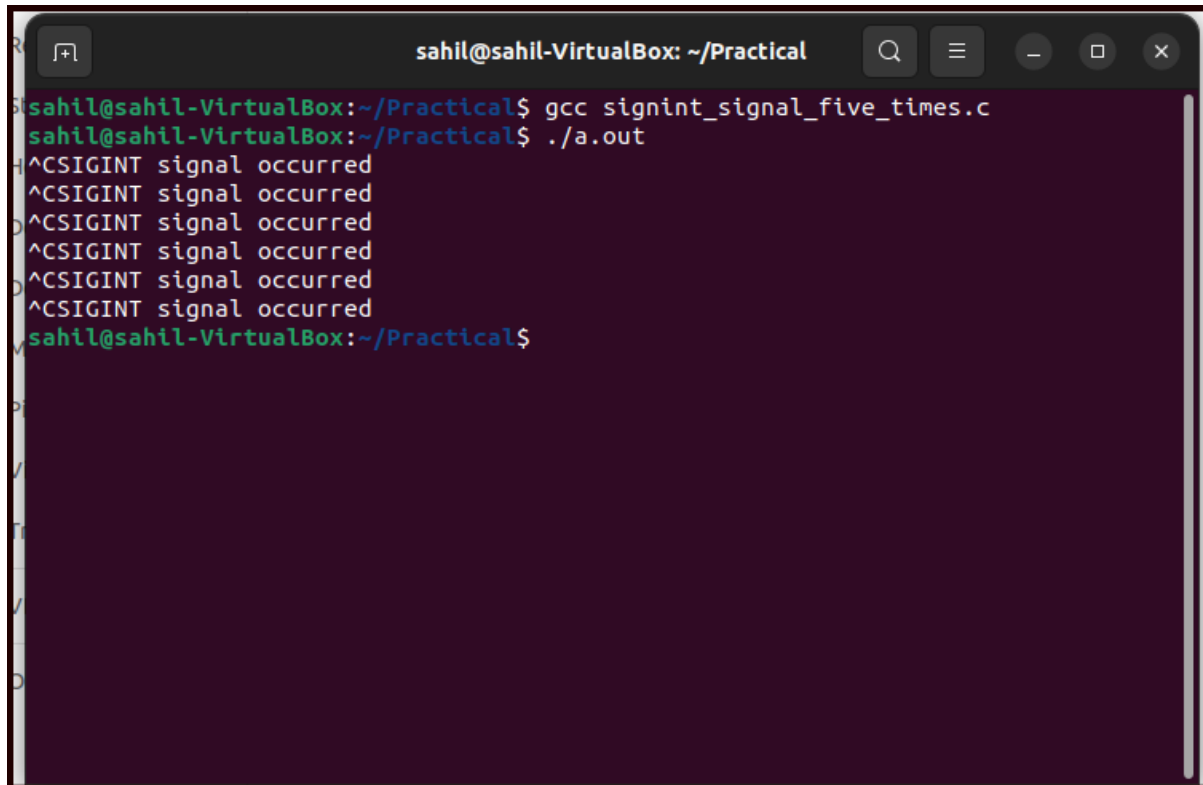
```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

int count = 0;
void sigint_handler(int sig) {
    count++;
    printf("SIGINT signal occurred\n");
    if (count == 6)
        exit(0);
}

int main() {
    signal(SIGINT, sigint_handler);
```

```
while (1) {  
    // Wait for signals  
}  
return 0;  
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical  
sahil@sahil-VirtualBox:~/Practical$ gcc signint_signal_five_times.c  
sahil@sahil-VirtualBox:~/Practical$ ./a.out  
^CSIGINT signal occurred  
^CSIGINT signal occurred  
^CSIGINT signal occurred  
^CSIGINT signal occurred  
^CSIGINT signal occurred  
^CSIGINT signal occurred  
sahil@sahil-VirtualBox:~/Practical$
```

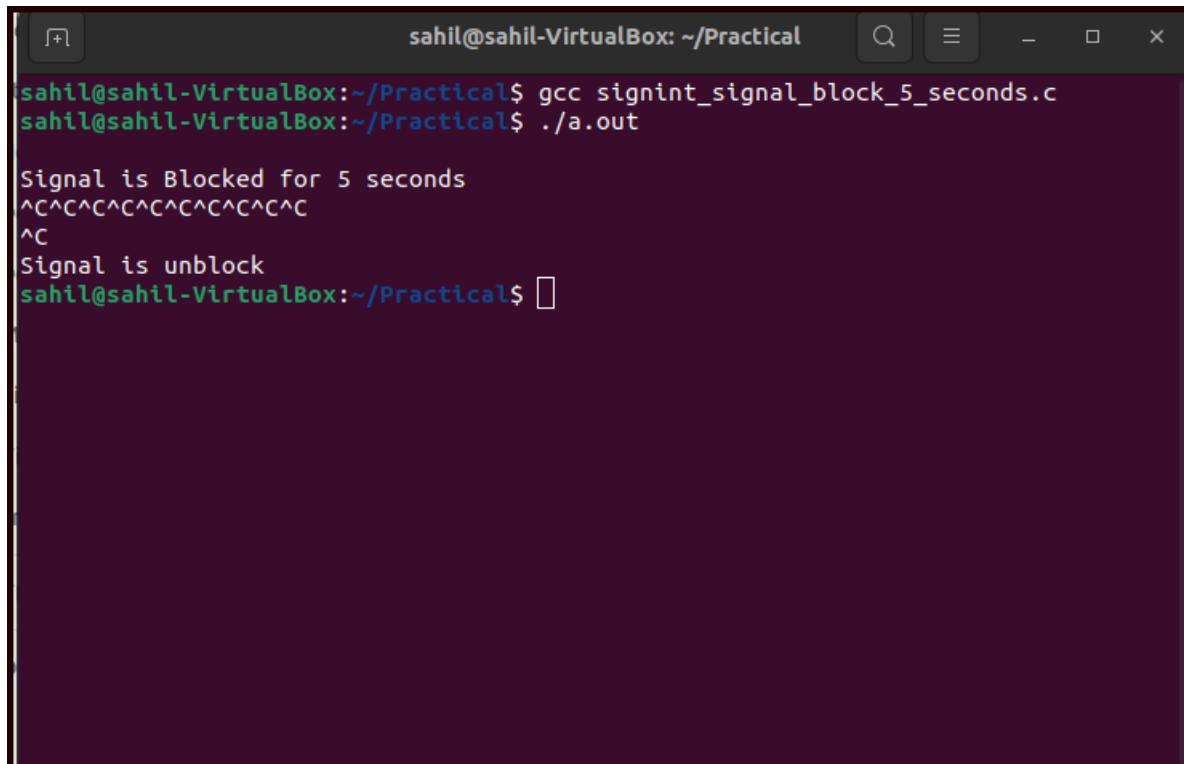
4. Write a program to block SIGINT signal for 5 seconds

Code :

```
#include <stdio.h>  
#include <signal.h>  
#include <stdlib.h>  
#include <unistd.h>  
  
void sigint_handler(int sig) {  
    sleep(5);  
    printf("\nSignal is unblock\n");  
    _exit(0);  
}  
  
int main() {
```

```
signal(SIGINT, sigint_handler);
printf("\nSignal is Blocked for 5 seconds\n");
while (1) {
    // Wait for signals
}
return 0;
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc signint_signal_block_5_seconds.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out

Signal is Blocked for 5 seconds
^C^C^C^C^C^C^C^C^C^C
^C
Signal is unblock
sahil@sahil-VirtualBox:~/Practical$
```

5. Write a program in LINUX to simulate extended shell. Show the prompt and accept standard shell command which will be executed by child process using one of the exec family system calls. Parent process waits until child finished execution the command may consist of at the most 5 parameters. The process should be repeated till user types "exit".

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    char line[80];
    char* args[5];
    int status;
    while (1) {
        printf("extended-shell> ");
        if (fgets(line,80, stdin) == NULL)
            break;
        char* token = strtok(line, " \\t\\n");
        int i = 0;
        while (token != NULL && i < 5) {
            args[i] = token;
            token = strtok(NULL, " \\t\\n");
            i++;
        }
        args[i] = NULL;
        int pid = fork();
        if (pid == -1) {
            perror("fork");
            exit(1);
        } else if (pid == 0) {
            if (execvp(args[0], args) == -1) {
                perror("execvp");
                exit(1);
            }
        } else
            wait(&status);

        if (strcmp(args[0], "exit") == 0)
            break;
    }
    return 0;
}
```

Output :

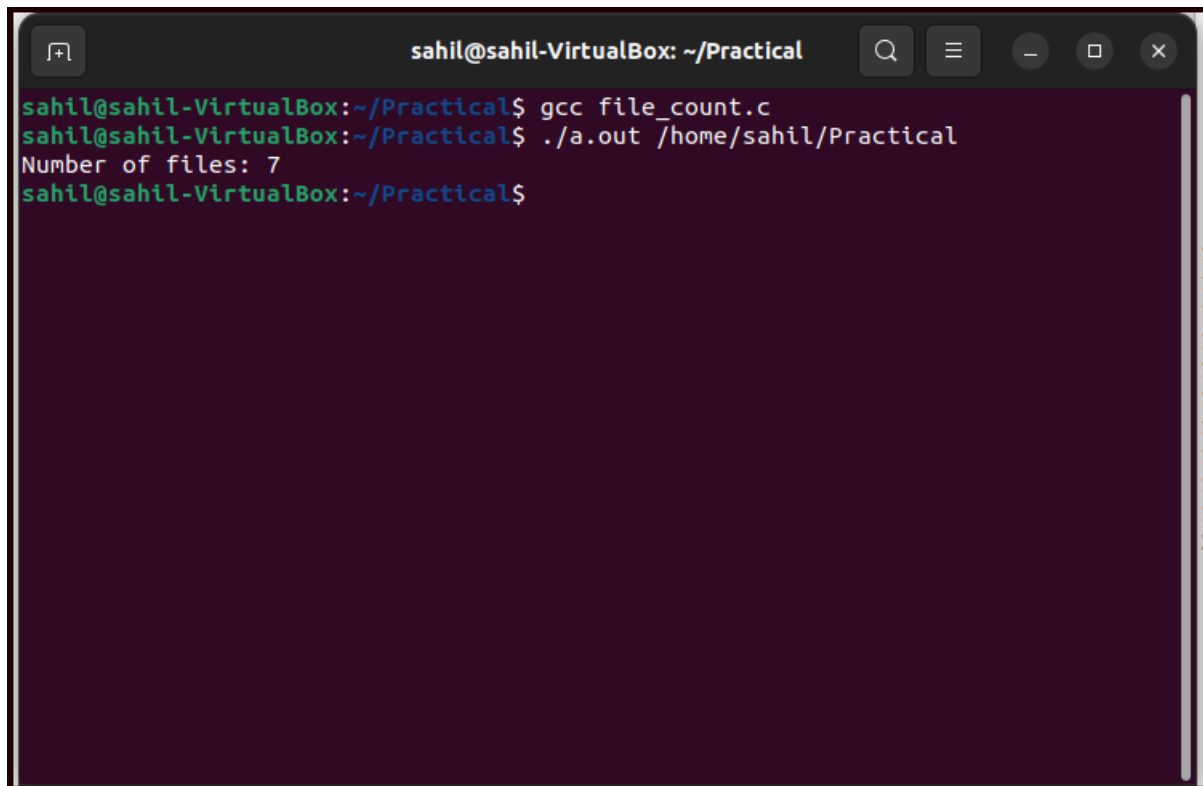
```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc simulate_shell_atmost_5_command.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
extended-shell> ls
a.out                                signint_signal_block_5_seconds.c
namedpipe.c                         signint_signal_five_times.c
name_of_files_in_directory.c        simulate_shell_atmost_5_command.c
extended-shell> cd
execvp: No such file or directory
extended-shell> cmd
execvp: No such file or directory
extended-shell> md
execvp: No such file or directory
extended-shell> exit
execvp: No such file or directory
sahil@sahil-VirtualBox:~/Practical$
```

6. Write a program to count the no of files of a specified directory.

Code :

```
#include<stdio.h>
#include<dirent.h>
#include<unistd.h>
void main(int argc , char *argv[])
{
    int cnt = 0;
    DIR *dp;
    struct dirent *entry;
    dp = opendir(argv[1]);
    if (dp == NULL) {
        printf("Directory does not exists.\n");
        _exit(0);
    }
    while (entry = readdir(dp))
        if (entry->d_type == DT_REG)
            cnt++;
    printf("Number of files: %d\n", cnt);
    closedir(dp);
}
```

Output :

A terminal window titled 'sahil@sahil-VirtualBox: ~/Practical' with standard window controls. The terminal shows the following commands and output:

```
sahil@sahil-VirtualBox:~/Practical$ gcc file_count.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out /home/sahil/Practical
Number of files: 7
sahil@sahil-VirtualBox:~/Practical$
```

7. Use of dup and dup2 system call. Create one file named temp.txt. Write some contents in the file. Make use of dup and dup2 system calls to write following content in the file. Makes use of four descriptors and channel zero.

Code :

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<string.h>

void main() {
    char text1[] = "M.Sc. Computer Science\n";
    char text2[] = "Semesterâ€™II Class\n";
    char text3[] = "Advanced Operating Systems\n";
    char text4[] = "Practical Examination\n";
```

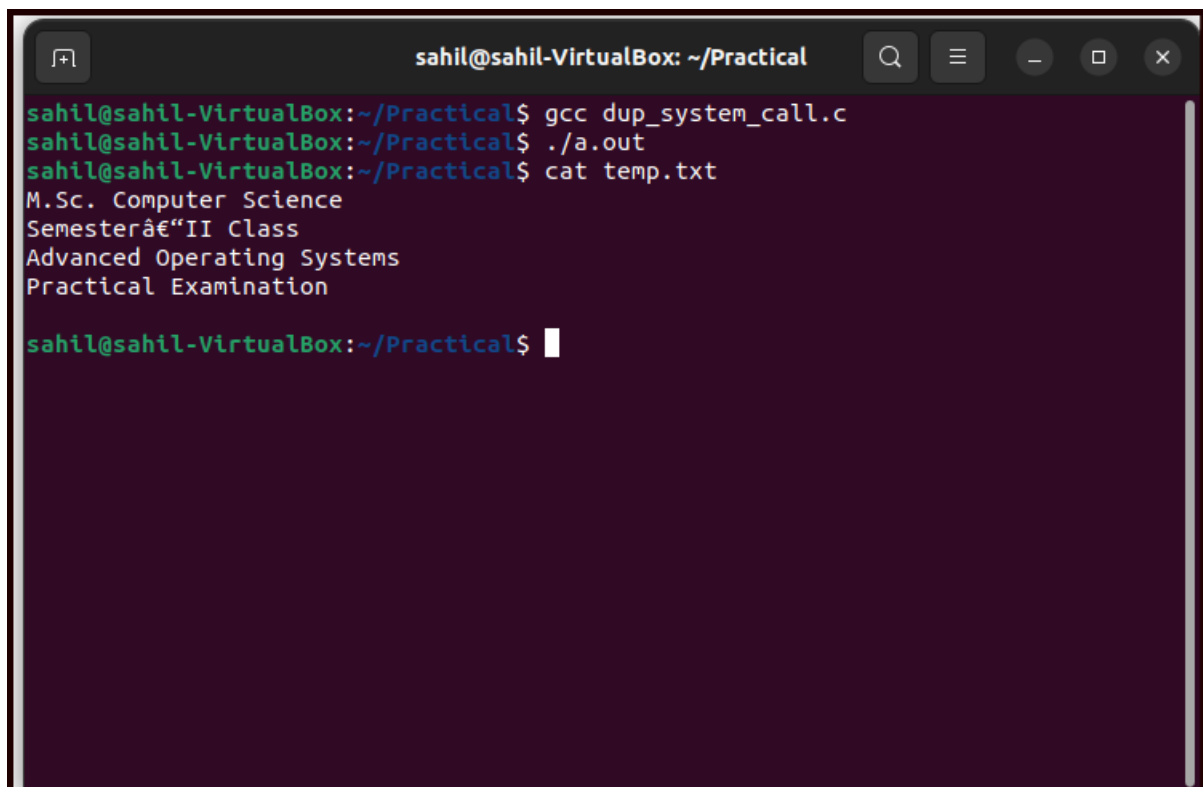


```
int fd1 = open("temp.txt", O_WRONLY);
int fd2 = dup(fd1);
int fd3 = dup2(fd2, 0);
int fd4 = dup2(fd3, 1);

write(fd1, text1, strlen(text1));
write(fd2, text2, strlen(text2));
write(fd3, text3, strlen(text3));
printf("%s", text4);

}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc dup_system_call.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
sahil@sahil-VirtualBox:~/Practical$ cat temp.txt
M.Sc. Computer Science
Semesterâ€™II Class
Advanced Operating Systems
Practical Examination
sahil@sahil-VirtualBox:~/Practical$
```

8. Check whether the specified file is regular file or directory and print user permissions of the file.

Code :

```
#include <stdio.h>
#include <dirent.h>
```

```

#include <unistd.h>
#include <sys/stat.h>
void main(int argc , char *argv[])
{
    char *fileName = argv[1];
    struct stat pathStat;
    stat(fileName, &pathStat);

    if (S_ISREG(pathStat.st_mode))
        printf("Its file!\n");
    else if (S_ISDIR(pathStat.st_mode))
        printf("its Directory!\n");
    printf("File permissions are : ");
    printf((pathStat.st_mode & S_IRUSR) ? "r" : "-");
    printf((pathStat.st_mode & S_IWUSR) ? "w" : "-");
    printf((pathStat.st_mode & S_IXUSR) ? "x\n" : "-\n");
}

```

Output :

```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc check_file_or_directory_user_permissions.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out temp.txt
Its file!
File permissions are : rw-
sahil@sahil-VirtualBox:~/Practical$

```

9. Write a program in Linux to block SIGQUIT signal for 5 seconds

Code :

```

#include <stdio.h>
#include <signal.h>

```

```
#include <unistd.h>

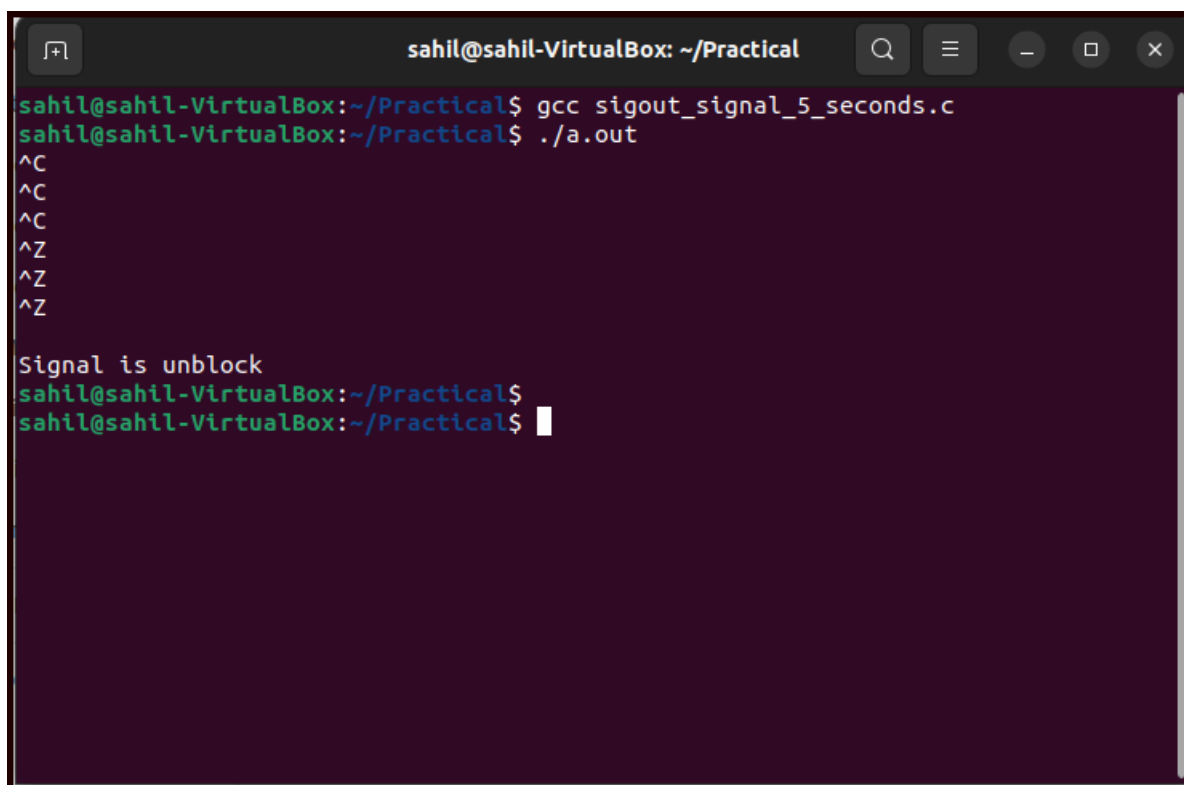
void sigint_handler(int sig) {
}

void handle_sigtstp(int sig) {
    sleep(5);
    printf("\nSignal is unblock\n");
    _exit(0);
}

int main() {
    signal(SIGINT, sigint_handler);
    signal(SIGTSTP, handle_sigtstp);

    while (1) {
        // Loop indefinitely
    }
    return 0;
}
```

Output :



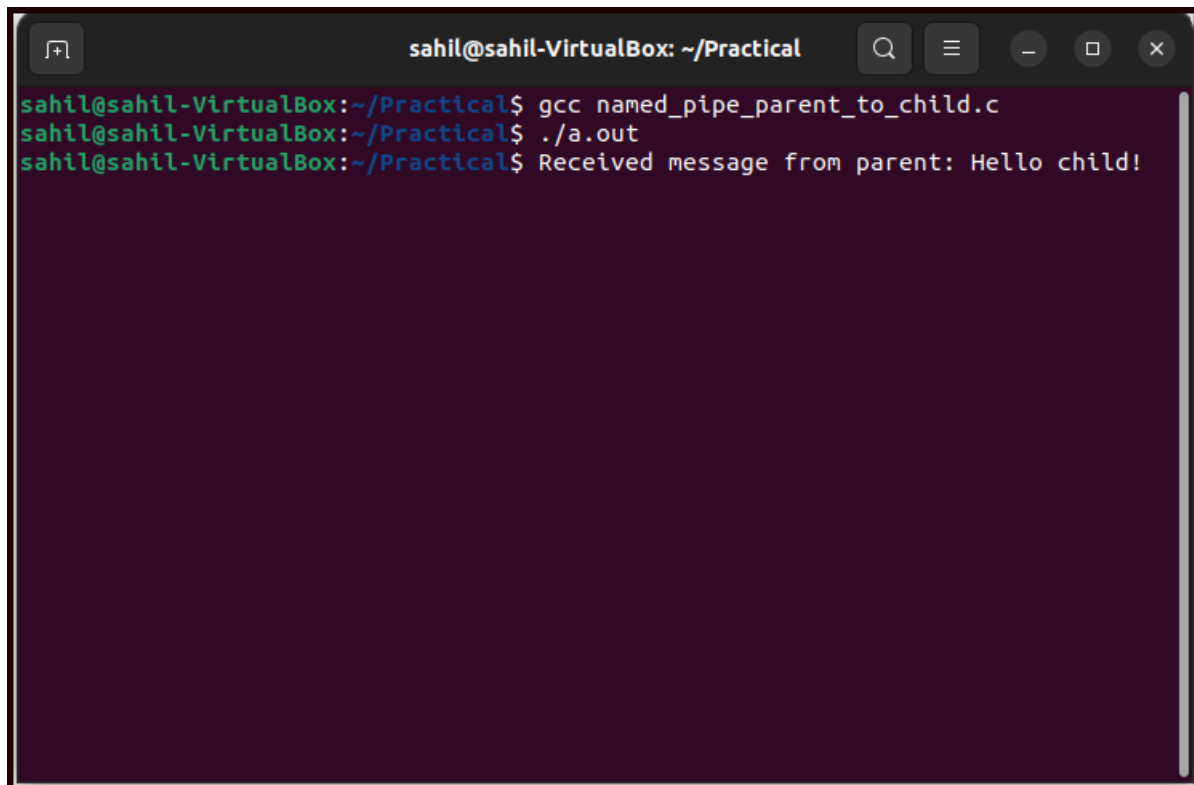
```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc sigout_signal_5_seconds.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
^C
^C
^C
^Z
^Z
^Z
Signal is unblock
sahil@sahil-VirtualBox:~/Practical$
sahil@sahil-VirtualBox:~/Practical$
```

10. Write a program to create a named pipe where parent sends message to child.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
int main() {
    int fd,pid;
    char * myfifo = "namedpipe";
    char buf[256];
    /* create the named pipe */
    mkfifo(myfifo, 0666);
    pid = fork();
    if (pid > 0)
    {
        fd = open(myfifo, O_WRONLY);
        write(fd, "Hello child!", sizeof("Hello child!"));
        close(fd);
        _exit(0);
    }
    else
    {
        fd = open(myfifo, O_RDONLY);
        read(fd, buf, sizeof(buf));
        printf("Received message from parent: %s\n", buf);
        close(fd);
        unlink(myfifo);
        _exit(0);
    }
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc named_pipe_parent_to_child.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
sahil@sahil-VirtualBox:~/Practical$ Received message from parent: Hello child!
```

11. Write program to handle SIGINT, SIGALRM and SIFTSTP signals.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void handle_sigint(int sig) {
    printf("Received SIGINT signal\n");
}

void handle_sigalrm(int sig) {
    printf("Received SIGALRM signal\n");
}

void handle_sigtstp(int sig) {
    printf("Received SIGTSTP signal\n");
}

int main() {
    signal(SIGINT, handle_sigint);
    signal(SIGALRM, handle_sigalrm);
    signal(SIGTSTP, handle_sigtstp);
```

```

while (1) {
    alarm(1);
    sleep(1);
}
return 0;
}

```

Output :

```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc sigint_sigalrm_siftstp.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
Received SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal
Received SIGINT signal
^CReceived SIGINT signal
^CReceived SIGINT signal
^ZReceived SIGTSTP signal
^ZReceived SIGTSTP signal
^ZReceived SIGTSTP signal
Received SIGTSTP signal
^ZReceived SIGALRM signal
^CReceived SIGINT signal
^CReceived SIGINT signal
^ZReceived SIGTSTP signal
Received SIGTSTP signal
^ZReceived SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal
Received SIGALRM signal

```

12. Write program to create hole in it.(Use Lseek system call). Write “Hello world !” five times after every 100 blank character.

Code :

```

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>
int main(argc,argv)
int argc;
char *argv[];
{

```

```
int fd,skval;

char c;

if(argc !=2)
    _exit(0);

fd = open(argv[1],O_WRONLY|O_CREAT);

if(fd == -1)
    _exit(0);

for(int j=0; j<3; j++)
{
    skval = lseek(fd,100L,1);

    for(int i=0; i<5;i++)
        write(fd,"Hello world !",strlen("Hello world !"));
}

close(fd);

return 0;

}
```

Output :

```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc create_hole_using_lseek.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out temp.txt
sahil@sahil-VirtualBox:~/Practical$ cat temp.txt
Hello world !Hello world !Hello world !Hello world !Hello world !Hello world !He
llo world !Hello world !Hello world !Hello world !Hello world !Hello world !Hell
o world !Hello world !Hello world !sahil@sahil-VirtualBox:~/Practical$
```

13. Write program that illustrate suspending and resuming processes using signals

Code :

```

#include <stdio.h>
#include <signal.h>
#include <unistd.h>

int main() {
    int pid = fork();
    if (pid == 0)
    {
        printf("Child process %d is running...\n", getpid());
        while (1)
        {
            printf("Child process %d is running...\n", getpid());
            sleep(1);
        }
    }
    else
    {
        printf("Parent process %d is running...\n", getpid());

        sleep(2);
        printf("Sending SIGSTOP signal to child process %d...\n", pid);
        kill(pid, SIGSTOP);
        printf("Child process %d has been suspended.\n", pid);

        sleep(2);
        printf("Sending SIGCONT signal to child process %d...\n", pid);
        kill(pid, SIGCONT);
        printf("Child process %d has been resumed.\n", pid);

        sleep(2);
        printf("Sending SIGTERM signal to child process %d...\n", pid);
        kill(pid, SIGTERM);
        printf("Child process %d has been terminated.\n", pid);
    }
    return 0;
}

```

Output :


```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc suspending_resuming_process.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
Parent process 3154 is running...
Child process 3155 is running...
Child process 3155 is running...
Child process 3155 is running...
Sending SIGSTOP signal to child process 3155...
Child process 3155 has been suspended.
Sending SIGCONT signal to child process 3155...
Child process 3155 has been resumed.
Child process 3155 is running...
Child process 3155 is running...
Sending SIGTERM signal to child process 3155...
Child process 3155 has been terminated.
sahil@sahil-VirtualBox:~/Practical$
```

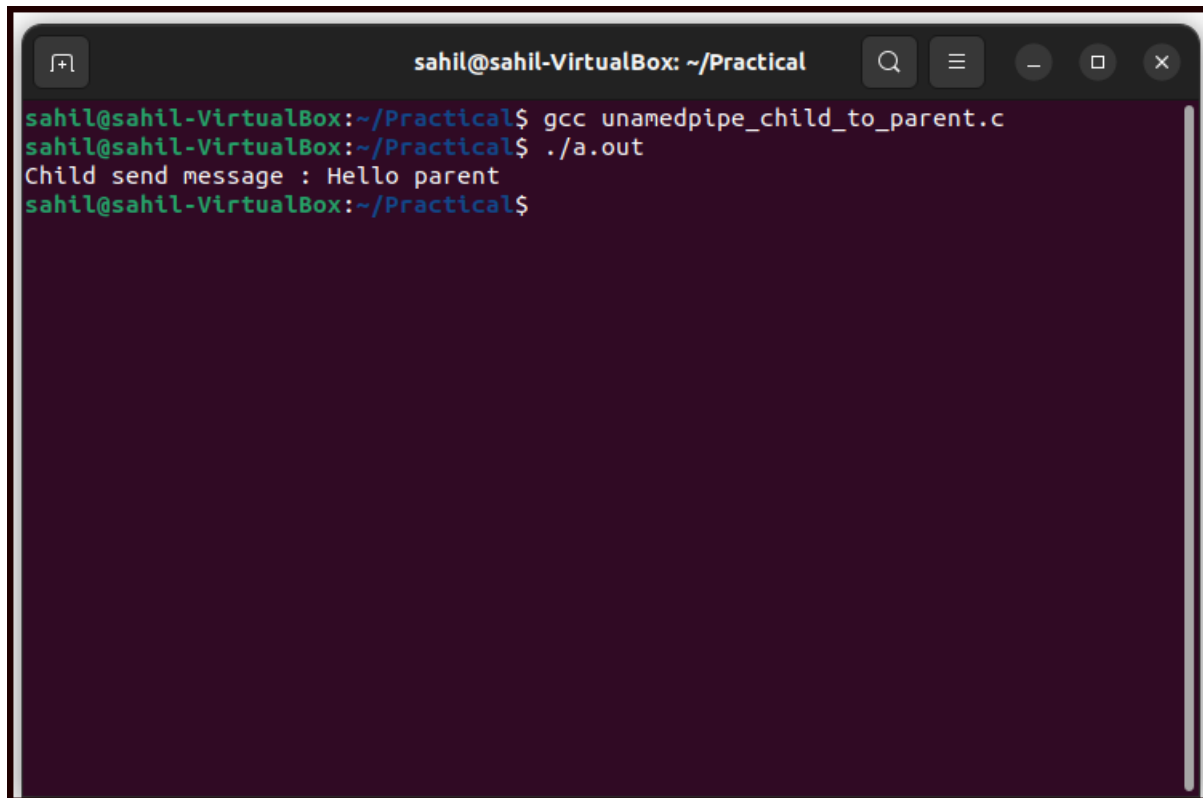
14. Write a program to create an unmanned pipe where child sends messages to parent.

Code :

```
#include<stdio.h>
#include<string.h>
#include <sys/types.h>
#include <unistd.h>
#include<fcntl.h>
int main()
{
    int fd[2], pid;
    char buff[50];
    char msg[]="Hello parent";
    pipe(fd);
    if((pid=fork()) != 0)
    {
        close(fd[1]);
        read(fd[0],buff,sizeof(msg));
        printf("Child send message : %s\n",buff);
        close(fd[0]);
    }
    else
    {
```

```
    close(fd[0]);
    write(fd[1],msg,strlen(msg));
    close(fd[1]);
}
return 0;
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc unnamedpipe_child_to_parent.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
Child send message : Hello parent
sahil@sahil-VirtualBox:~/Practical$
```

15. Print the type of file and group permission of it where file name is accepted thru command line arguments

Code :

```
#include <stdio.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>

void main(int argc , char *argv[])
{
    char *fileName = argv[1];
    struct stat pathStat;
```

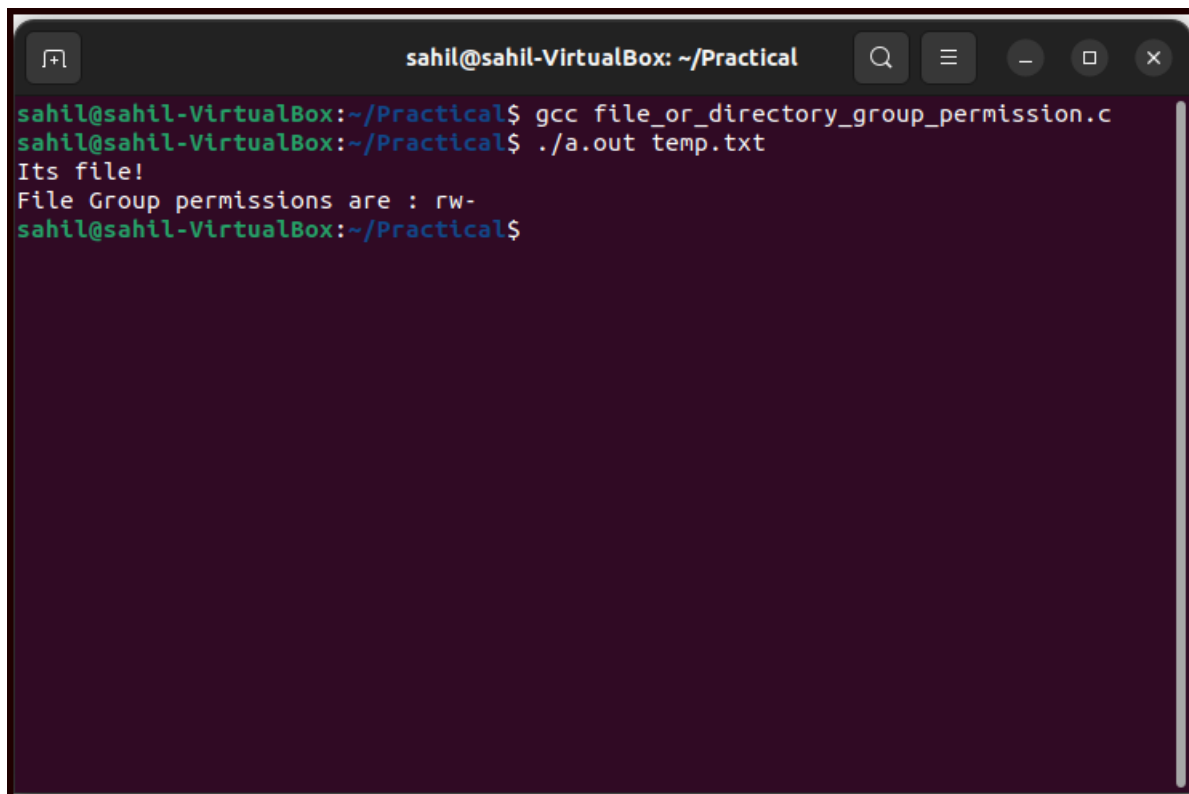
```

stat(fileName, &pathStat);

if (S_ISREG(pathStat.st_mode))
    printf("Its file!\n");
else if (S_ISDIR(pathStat.st_mode))
    printf("its Directory!\n");
printf("File Group permissions are : ");
printf((pathStat.st_mode & S_IRGRP) ? "r" : "-");
printf((pathStat.st_mode & S_IWGRP) ? "w" : "-");
printf((pathStat.st_mode & S_IXGRP) ? "x\n" : "-\n");
}

```

Output :



```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc file_or_directory_group_permission.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out temp.txt
Its file!
File Group permissions are : rw-
sahil@sahil-VirtualBox:~/Practical$

```

16. Write a program to catch death of child signal by parent process after 5 seconds. Use alarm system call

Code :

```

#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

```

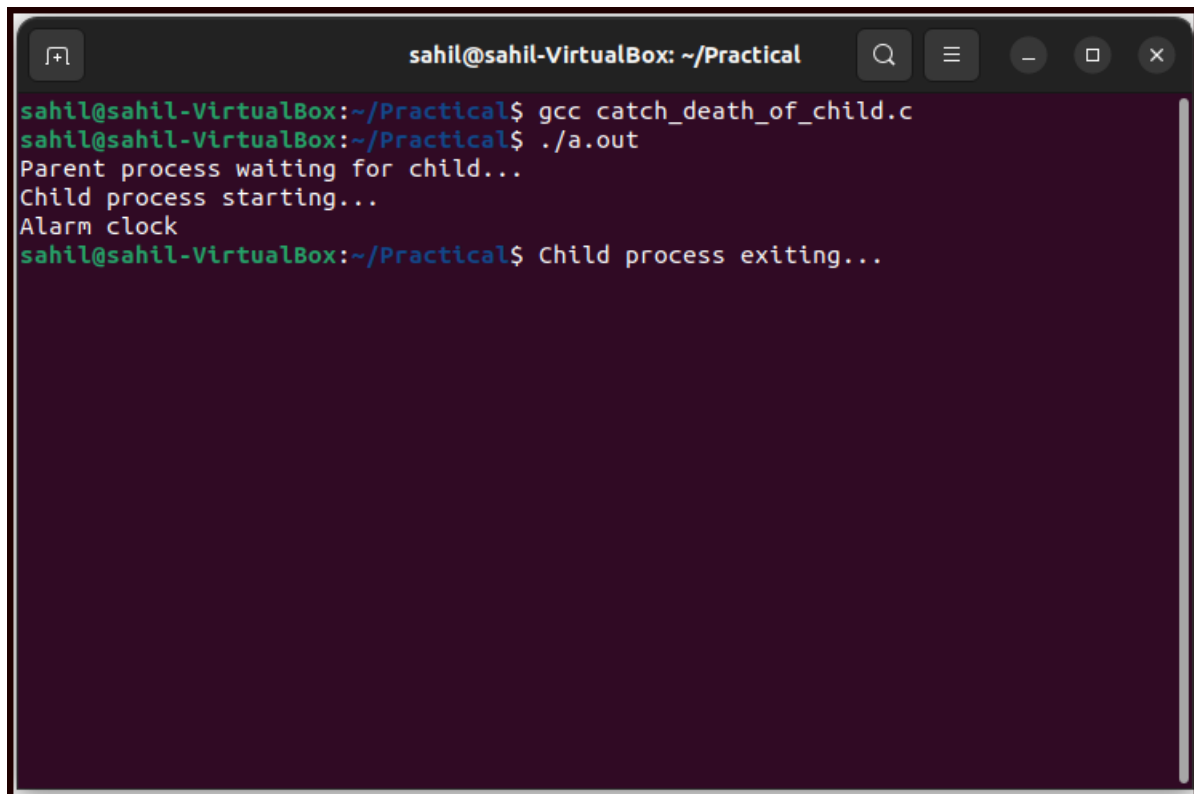
```
#include <sys/types.h>
#include <sys/wait.h>

void sigchld_handler(int sig) {
    printf("Child process terminated.\n");
}

int main() {
    pid_t pid;
    int status;

    signal(SIGCHLD, sigchld_handler);
    pid = fork();
    if (pid == 0) {
        printf("Child process starting...\n");
        sleep(10);
        printf("Child process exiting...\n");
        exit(0);
    }
    else if (pid > 0)
    {
        printf("Parent process waiting for child...\n");
        alarm(5);
        wait(&status);
    }
    else
    {
        printf("Fork failed.\n");
        exit(0);
    }
    printf("Parent process exiting.\n");
    return 0;
}
```

Output :



```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc catch_death_of_child.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
Parent process waiting for child...
Child process starting...
Alarm clock
sahil@sahil-VirtualBox:~/Practical$ Child process exiting...
```

17. Write a program where parent and child share file access

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <string.h>

int main() {
    int fd;
    pid_t pid;
    char buf[256];

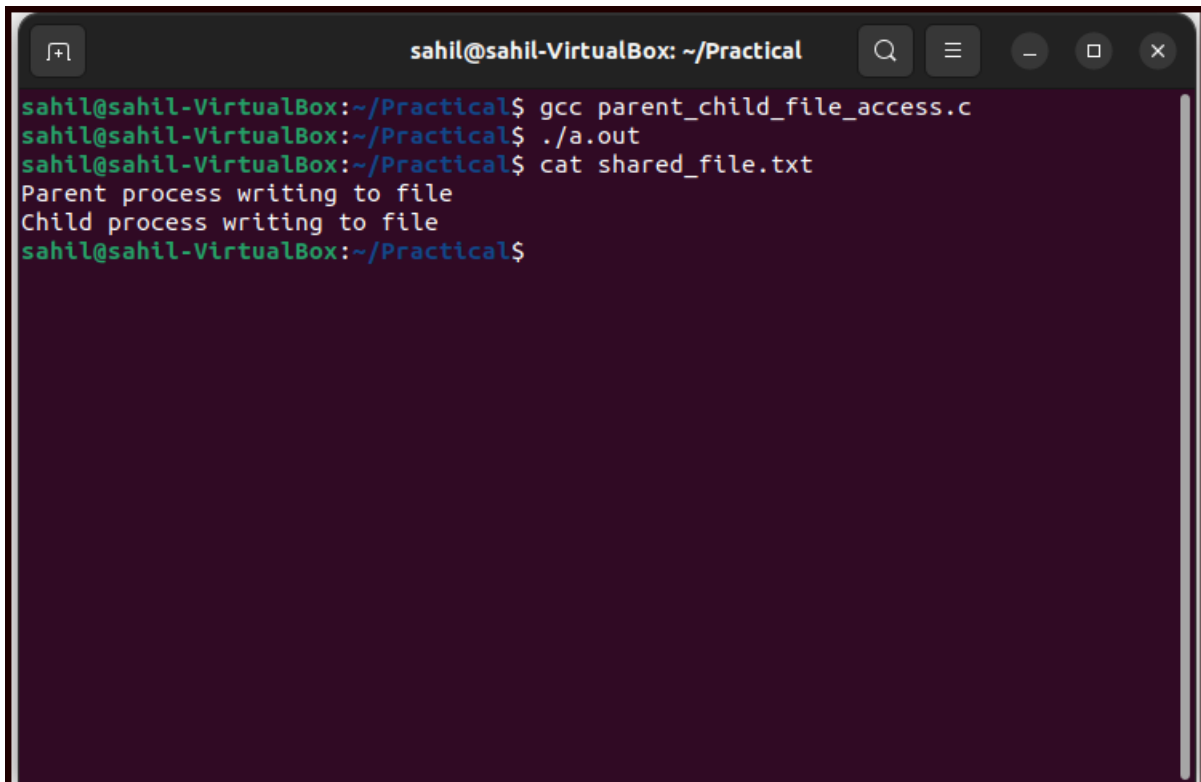
    fd = open("shared_file.txt", O_RDWR | O_CREAT, 0644);
    pid = fork();
    if (pid < 0) {
        perror("fork");
        exit(0);
    }
```

```

    }
    else if (pid == 0)
    {
        sprintf(buf, "Child process writing to file\n");
        write(fd, buf, strlen(buf));
        exit(0);
    }
    else
    {
        sprintf(buf, "Parent process writing to file\n");
        write(fd, buf, strlen(buf));
        exit(0);
    }
    return 0;
}

```

Output :



```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc parent_child_file_access.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
sahil@sahil-VirtualBox:~/Practical$ cat shared_file.txt
Parent process writing to file
Child process writing to file
sahil@sahil-VirtualBox:~/Practical$

```

18. Write the program that accepts directory name as input and print its content

Code :

```
#include<stdio.h>
```

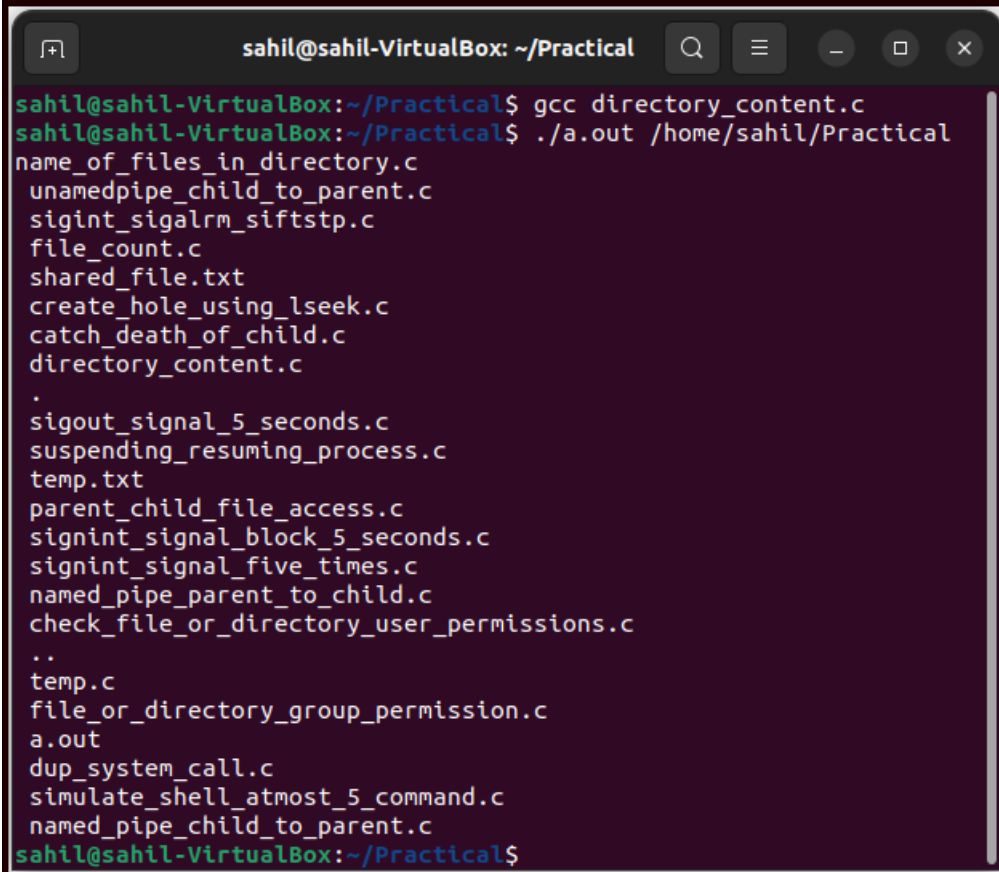
```

#include<dirent.h>
#include<unistd.h>

void main(int argc , char *argv[])
{
    DIR *dp;
    struct dirent *entry;
    dp = opendir(argv[1]);
    if (dp == NULL) {
        printf("\n Directory does not exists .\n");
        _exit(0);
    }
    while( entry = readdir(dp))
        printf("%s\n ",entry-> d_name);
    closedir(dp);
}

```

Output :



```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc directory_content.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out /home/sahil/Practical
name_of_files_in_directory.c
unamedpipe_child_to_parent.c
sigint_sigalrm_siftstp.c
file_count.c
shared_file.txt
create_hole_using_lseek.c
catch_death_of_child.c
directory_content.c
.
sigout_signal_5_seconds.c
suspending_resuming_process.c
temp.txt
parent_child_file_access.c
signint_signal_block_5_seconds.c
signint_signal_five_times.c
named_pipe_parent_to_child.c
check_file_or_directory_user_permissions.c
..
temp.c
file_or_directory_group_permission.c
a.out
dup_system_call.c
simulate_shell_atmost_5_command.c
named_pipe_child_to_parent.c
sahil@sahil-VirtualBox:~/Practical$

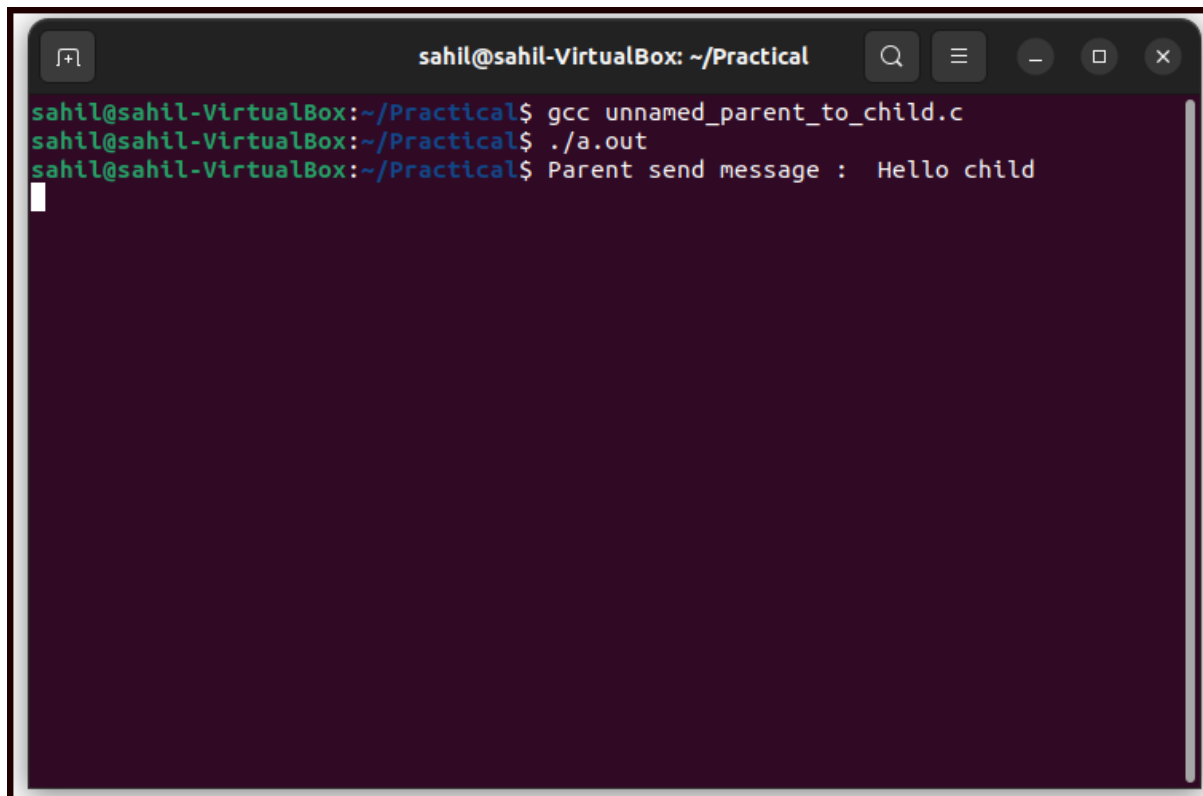
```

19. Write a program to create an unnamed pipe where parent sends message to child

Code :

```
#include<stdio.h>
#include<string.h>
#include <sys/types.h>
#include <unistd.h>
#include<fcntl.h>
int main()
{
    int fd[2], pid;
    char buff[50];
    char msg[]="Hello child";
    pipe(fd);
    if((pid=fork()) != 0)
    {
        close(fd[0]);
        write(fd[1],msg,strlen(msg));
        close(fd[1]);
    }
    else
    {
        close(fd[1]);
        read(fd[0],buff,sizeof(msg));
        printf("Parent send message : %s\n",buff);
        close(fd[0]);
    }
    return 0;
}
```

Output :



The screenshot shows a terminal window titled "sahil@sahil-VirtualBox: ~/Practical". The user has entered three commands: `gcc unnamed_parent_to_child.c`, `./a.out`, and the output is "Parent send message : Hello child".

```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc unnamed_parent_to_child.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
sahil@sahil-VirtualBox:~/Practical$ Parent send message : Hello child
```

20. Write a program to implement following commands as linux.
- a) Typelines +5 <filename> print first 5 lines of a file
 - b) Typelines -8 <filename> print last 20 lines of a file

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<string.h>
void typeline(char *op, char *fn)
{
```

```
int fh,i,j,n;

char c;

fh = open(fn,O_RDONLY);

if(fh == -1)

{

    printf("File %s not found.\n", fn);

    return;

}

if(atoi(op)==5)

{

    i = 0;

    while(read(fh, &c, 1) > 0)

    {

        printf("%c", c);

        if(c == '\n')

            i++;

        if(i == 5)

            break;

    }

}

if(strcmp(op,"-8")==0)

{

    i = 0;

    while(read(fh, &c, 1) > 0)

    {

        if(c == '\n')

            i++;

    }

    lseek(fh, 0, SEEK_SET);

    j = 0;

    while(read(fh, &c, 1) > 0)
```

```

    {
        if(c == '\n')
            j++;
        if(j == i-20)
            break;
    }
    while(read(fh, &c, 1) > 0)
        printf("%c", c);
    }
    close(fh);
}

```

```

int main()
{
    char command[60],t1[20],t2[20],t3[20];
    int n;
    while(1)
    {
        printf("myshell$ ");
        fflush(stdin);
        fgets(command,60,stdin);
        n = sscanf(command,"%s %s %s",t1,t2,t3);

        switch(n)
        {
            case 1:
                char a=t1[0];
                if(a=='q')
                    return 0;
                if(!fork())
                {

```

```
        execlp(t1,t1,NULL);
        perror(t1);
    }
    break;
case 2:
    if(!fork())
    {
        execlp(t1,t1,t2,NULL);
        perror(t1);
    }
    break;
case 3:
    if(strcmp(t1,"Typelines")==0)
        typeline(t2,t3);
    else
    {
        if(!fork())
        {
            execlp(t1,t1,t2,t3,NULL);
            perror(t1);
        }
    }
    break;
}
}
```

Output :

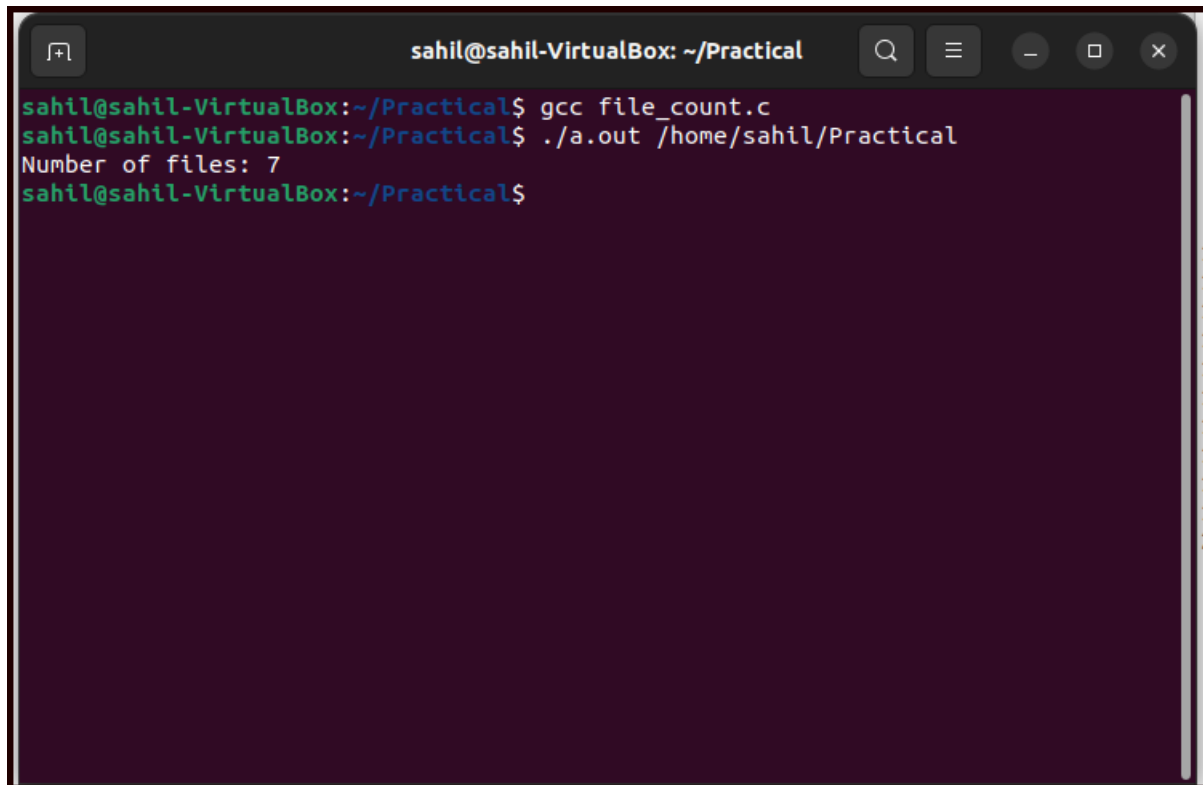
```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc typeline_shell.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
mysHELL$ Typelines 5 temp.txt
Veneshwar Mahadev temple is dedicated to Lord Shiva, situated in the centre of Somnath and
opposite to Somnath Trust Dharmshala. As we look into the history of Somnath, Prabhas Patan
center and pillar of Indian Culture was attacked, ruined and looted multiple times by the Muslim
Rulers. In 1025 AD, the Rajputa King Vaja was ruler of Somnath during the attacked by Mahmud
Ghazni.
mysHELL$ Typelines -8 temp.txt
center and pillar of Indian Culture was attacked, ruined and looted multiple times by the Muslim
Rulers. In 1025 AD, the Rajputa King Vaja was ruler of Somnath during the attacked by Mahmud
Ghazni.
As per the legends, Rajputa king gave Gazni tough fight and not surrendered himself
against the Ghazni. Finally Ghazni had decided to kidnap his daughter named Veni, the great
devotee of Lord Shiva, who visit temple daily to worship Lord Shiva. Soldiers of Gazani
attempted to abduction Veni, the Shivaling unexpectedly got divided and the princess got
embryonic into Shivlinga. Thus Lord Shiva temple here since then is known as "Veneshwar
Mahadev". The hair of the Veni and marks of sword on the Shiva Lingam being split open can still
be seen on Shivlinga. Creative Gujarati novelist K M Munshi had cover this extraordinary
incident in his novel.Nishkalank Mahadev Temple is a Hindu temple located at Koliyak near
Bhavnagar, Gujarat. Situated in Koliyak Beach, it is one of the rarest sea temples in India you
must include in your Gujarat trip and among the most visited places of pilgrimage in
BhavnagarLocated about a kilometer into the sea, the temple is dedicated to Lord Shiva. The
temple has 5 distinct swayambhu Shiva lingams on a square platform and each is having a Nandi
statue facing it. This temple is submerged during high tides in the sea and emerges during low
tides to reveal itself majestically, promising its devotees to wash away all sins. During the
high tide, the idol of the lord is submerged and all that can be seen are the flag and a pillar.
The temple was built with special care to withstand high tides and it truly remains an unsolved
mystery to modern engineers and technology experts
mysHELL$
```

21. Write a program to count the no of files of a specified directory.

Code :

```
#include<stdio.h>
#include<dirent.h>
#include<unistd.h>
void main(int argc , char *argv[])
{
    int cnt = 0;
    DIR *dp;
    struct dirent *entry;
    dp = opendir(argv[1]);
    if (dp == NULL) {
        printf("Directory does not exists.\n");
        _exit(0);
    }
    while (entry = readdir(dp))
        if (entry->d_type == DT_REG)
            cnt++;
    printf("Number of files: %d\n", cnt);
    closedir(dp);
}
```

Output :

A terminal window titled 'sahil@sahil-VirtualBox: ~/Practical' with search, menu, and window control icons. It shows the compilation of 'file_count.c' with 'gcc' and the execution of the resulting binary './a.out' on the directory '/home/sahil/Practical'. The program outputs 'Number of files: 7' before returning to the prompt.

```
sahil@sahil-VirtualBox: ~/Practical$ gcc file_count.c
sahil@sahil-VirtualBox: ~/Practical$ ./a.out /home/sahil/Practical
Number of files: 7
sahil@sahil-VirtualBox: ~/Practical$
```

22. Write a program to catch SIGQUIT signal five times and print message 'SIGQUIT signal occurred' every time and exit at sixth occurrence. Also ignore every occurrence of SIGTSTP signal.

Code :

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>

int count = 0;
void sig_handler(int sig) {
    printf("SIGQUIT signal occurred\n");
    count++;
    if (count == 5)
    {
```

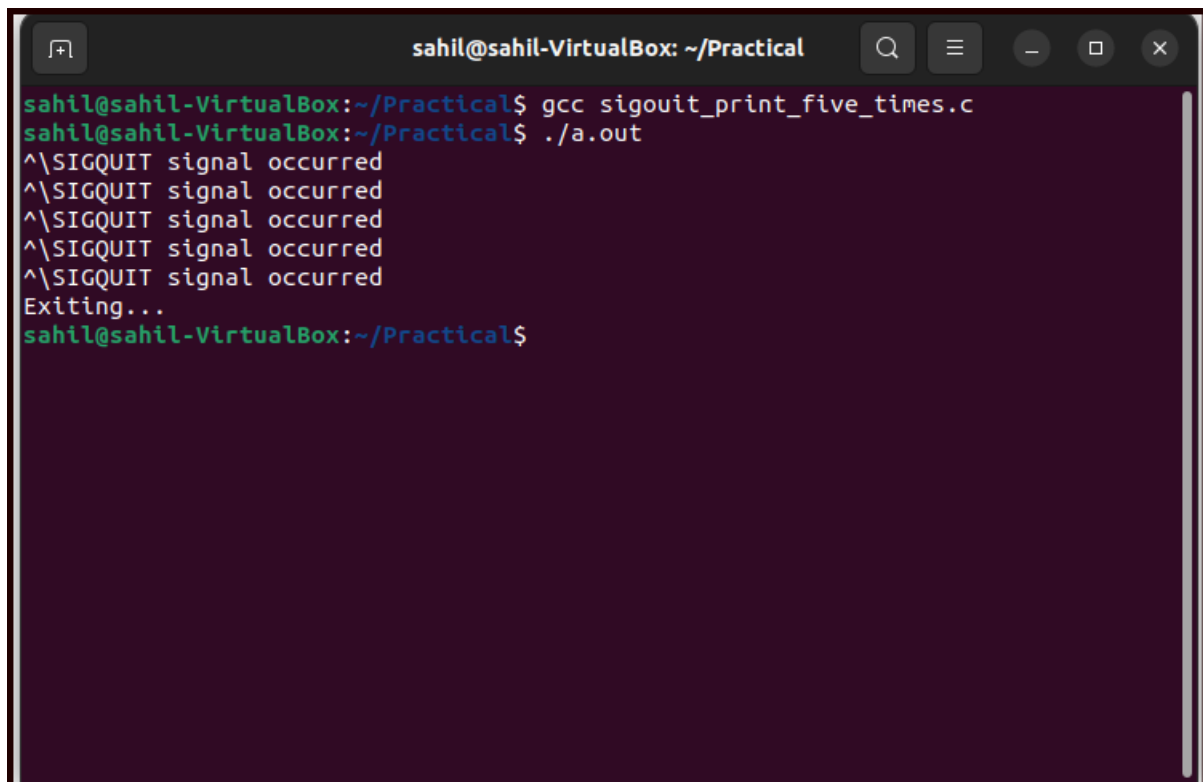
```

        printf("Exiting...\n");
        _exit(0);
    }
}

int main() {
    signal(SIGQUIT, sig_handler);
    while (1) {
        // Wait for signals
    }
    return 0;
}

```

Output :



```

sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc sigouit_print_five_times.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
^\\SIGQUIT signal occurred
^\\SIGQUIT signal occurred
^\\SIGQUIT signal occurred
^\\SIGQUIT signal occurred
^\\SIGQUIT signal occurred
Exiting...
sahil@sahil-VirtualBox:~/Practical$

```

23. Write a program in LINUX to simulate extended shell. Show the prompt and accept standard shell command which will be executed by child process using one of the exec family system calls. Parent process waits until child finished execution the command may consist of

at the most 5 parameters. The process should be repeated till user types “exit”.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    char line[80];
    char* args[5];
    int status;
    while (1) {
        printf("extended-shell> ");
        if (fgets(line, 80, stdin) == NULL)
            break;
        char* token = strtok(line, " \\t\\n");
        int i = 0;
        while (token != NULL && i < 5) {
            args[i] = token;
            token = strtok(NULL, " \\t\\n");
            i++;
        }
        args[i] = NULL;
        int pid = fork();
        if (pid == -1) {
            perror("fork");
            exit(1);
        } else if (pid == 0) {
            if (execvp(args[0], args) == -1) {
                perror("execvp");
                exit(1);
            }
        } else
            wait(&status);

        if (strcmp(args[0], "exit") == 0)
            break;
    }
    return 0;
}
```

Output :


```
sahil@sahil-VirtualBox: ~/Practical
sahil@sahil-VirtualBox:~/Practical$ gcc simulate_shell_atmost_5_command.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out
extended-shell> ls
a.out                                signint_signal_block_5_seconds.c
namedpipe.c                          signint_signal_five_times.c
name_of_files_in_directory.c         simulate_shell_atmost_5_command.c
extended-shell> cd
execvp: No such file or directory
extended-shell> cmd
execvp: No such file or directory
extended-shell> md
execvp: No such file or directory
extended-shell> exit
execvp: No such file or directory
sahil@sahil-VirtualBox:~/Practical$
```

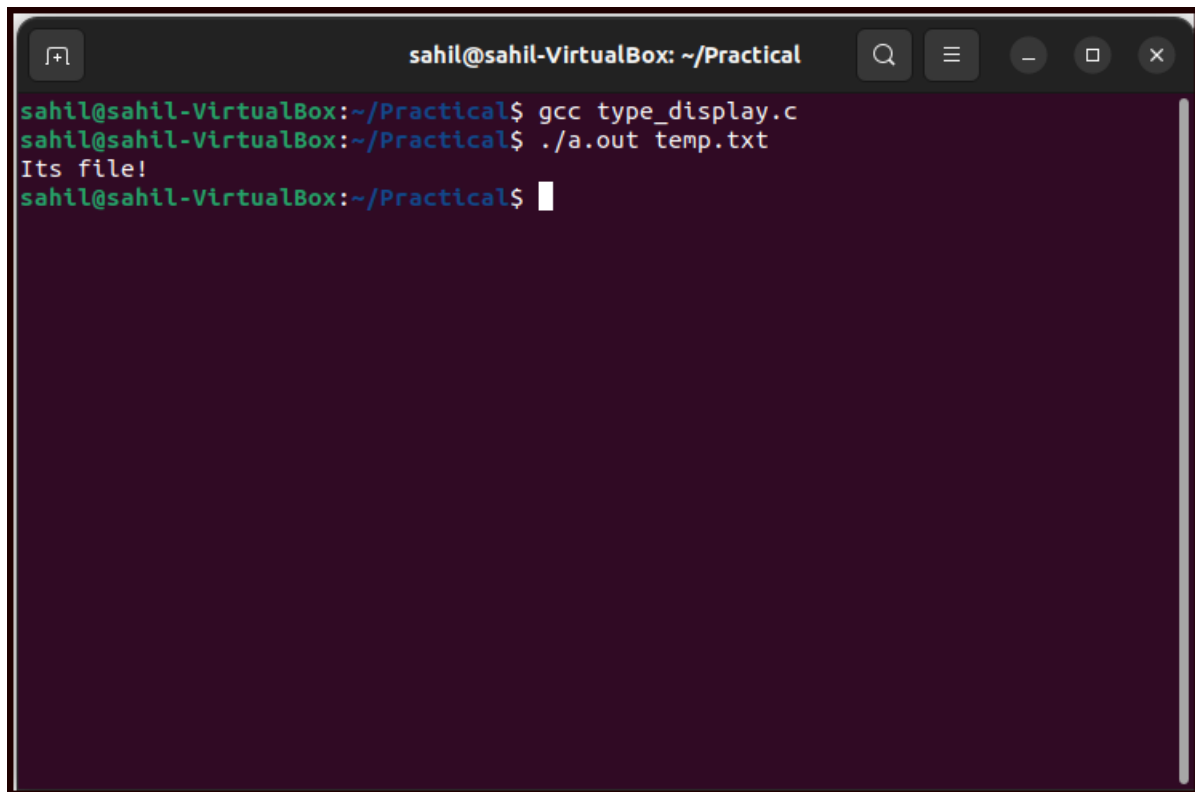
24. Print the type of file where file name is accepted thru command line argument

Code :

```
#include <stdio.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
void main(int argc , char *argv[])
{
    char *fileName = argv[1];
    struct stat pathStat;
    stat(fileName, &pathStat);

    if (S_ISREG(pathStat.st_mode))
        printf("Its file!\n");
    else if (S_ISDIR(pathStat.st_mode))
        printf("its Directory!\n");
}
```

Output :



A terminal window titled "sahil@sahil-VirtualBox: ~/Practical" with standard window controls. The terminal shows the following commands and output:

```
sahil@sahil-VirtualBox:~/Practical$ gcc type_display.c
sahil@sahil-VirtualBox:~/Practical$ ./a.out temp.txt
Its file!
sahil@sahil-VirtualBox:~/Practical$
```