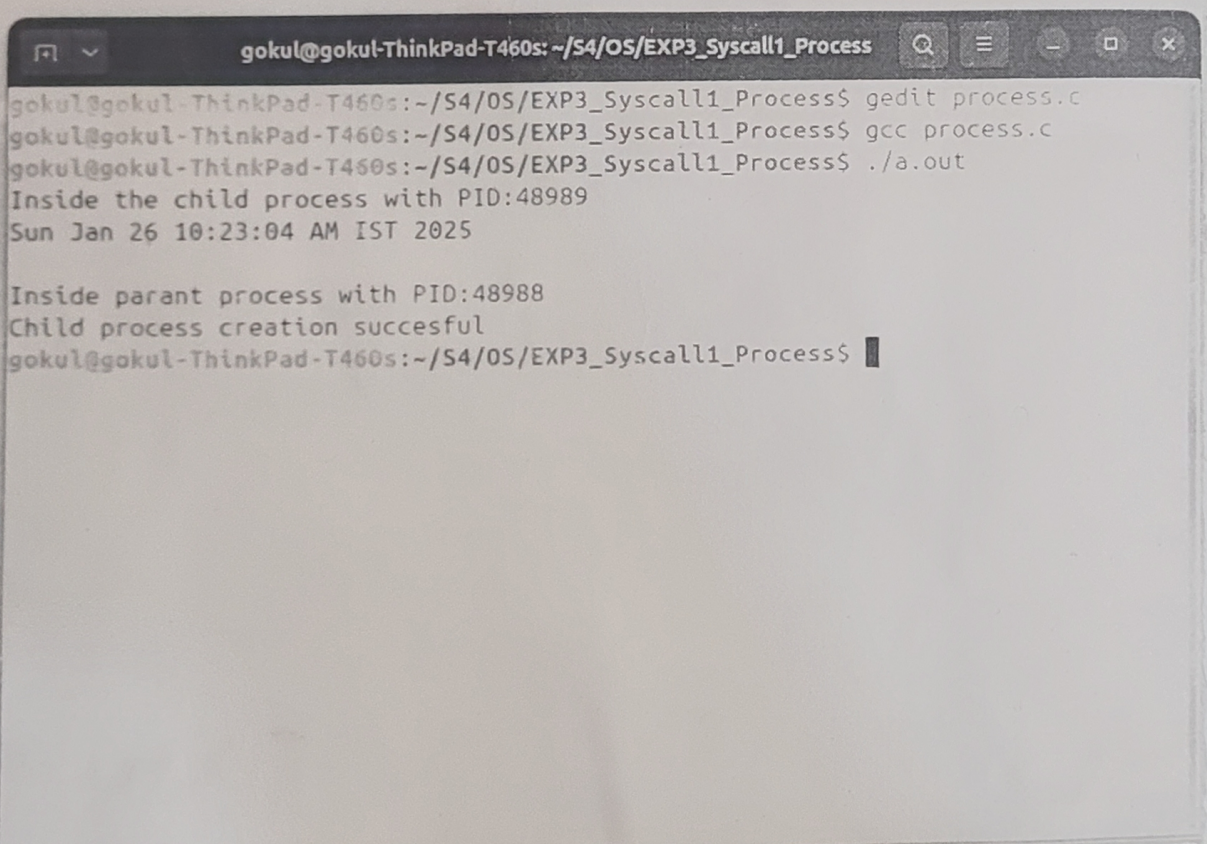
```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>

int main(){
        int status,pid,child_pid;
        pid=fork();
        if(pid==-1)
        {
                printf("Child process creation failed\n");
                return -1;
        }
        else if(pid==0){
                printf("Inside the child process with PID:%d\n",getpid());
                execlp("/bin/date","date",NULL);
                return 0;
        }
        else{
                child_pid=wait(&status);
                printf("\nInside parant process with PID:%d\n",getpid());
                printf("Child process creation succesful\n");
                return 0;
        }
}
```



Experiment-3

Process-System calls

Aim

To write a C program to create a child process in C using system calls.

Algorithm

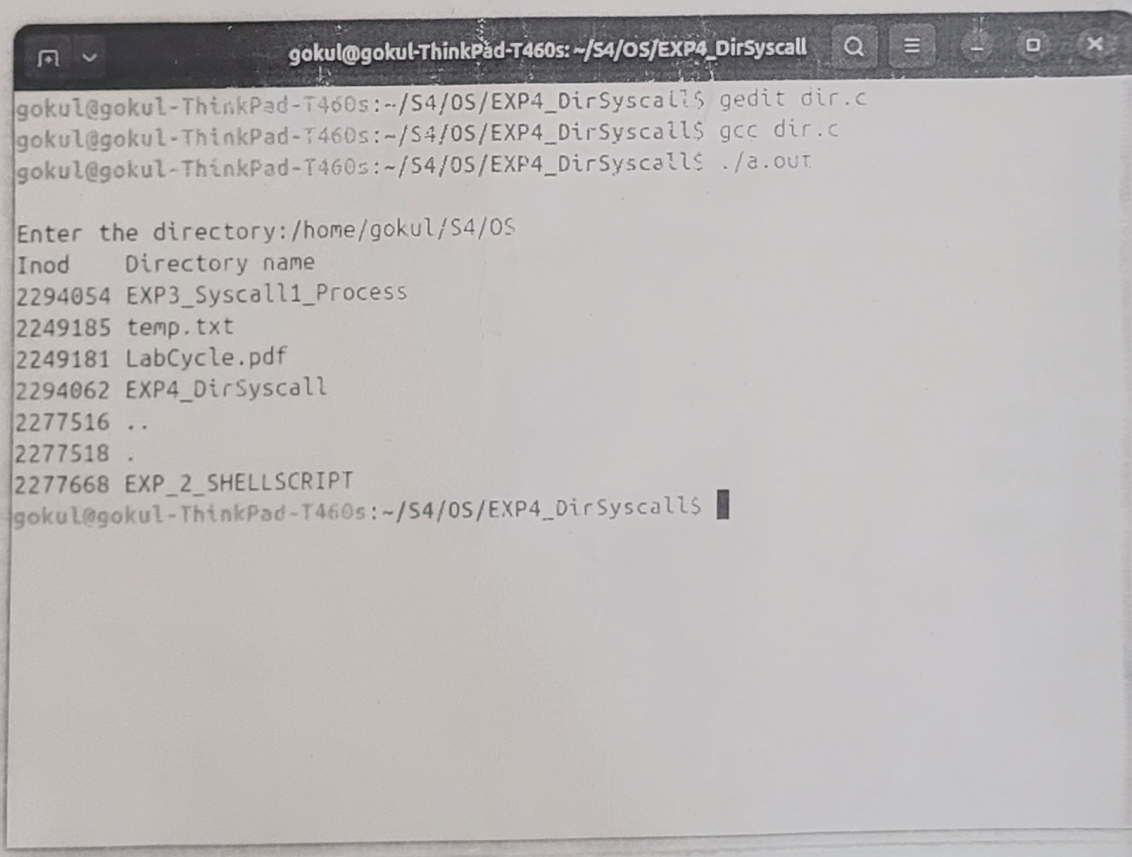
1. Start
2. Declare variables : Status, Pid and ChildPid
3. call fork()
4. If fork returned -1,
   4.1 ; Print error message
5. elseif fork returned 0'
   5.1: Print Pid of child process
   5.2: Print the Current date using execlp
6. else,
   6.1: Wait for Child Process to terminate
   6.2: Print Parent Process ID
7. Print Success message
8. stop

Result

Program executed Succesfully And desired output is recived

```c
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>

void main(){
        DIR *dir;
        struct dirent *ptr;
        char dirname[100];
        printf("\nEnter the directory:");
        scanf("%s",dirname);
        dir=opendir(dirname);
        printf("Inod\tDirectory name\n");
        while((ptr=readdir(dir))!=NULL){
                printf("%ld\t%s\n",ptr->d_ino,ptr->d_name);
        }
        closedir(dir);
        return;
}
```



Experiment-4

Directory management System Calls

Aim

To create a c Program to Print the files and its Inod for a given directory.

Algorithm

1. Start
2. Declare Pointers dir and dirent Pointer Ptr
3. Declare String dirname
4. Print Enter the directory and readit
5. Open directory using opendir()
6. Repeat while (Ptr=read(dir)!=NULL)
   - 6.1: Print the inod directory name of file using Ptr
   - 6.2: End of repeat
7. Close directory using closedir()
8. Stop

Result

Program executed succesfully and desired output is recived.

```c
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>

int main(){
        int fd,fd2;
        char wbuf[128],rbuf[128];
        fd=open("file.txt",O_WRONLY);
        printf("Enter the text to be written\n");
        scanf("%s",wbuf);
        printf("writing to file\n");
        write(fd,wbuf,strlen(wbuf));
        close(fd);
        printf("text written to file\n");
        printf("Reading from file\n");
        fd2=open("file.txt",O_RDONLY);
        printf("File contents are\n");
        read(fd2,rbuf,128);
        printf("%s\n",rbuf);
        close(fd2);
        return 0;
}
```

```
gokul@gokul-ThinkPad-T460s: ~/S4/OS/EXP4_FileSyscall
gokul@gokul-ThinkPad-T460s:~/S4/OS/EXP4_FileSyscall$ gedit file.txt
gokul@gokul-ThinkPad-T460s:~/S4/OS/EXP4_FileSyscall$ gedit file.c
gokul@gokul-ThinkPad-T460s:~/S4/OS/EXP4_FileSyscall$ gcc file.c
gokul@gokul-ThinkPad-T460s:~/S4/OS/EXP4_FileSyscall$ ./a.out
Enter the text to be written
HELLO_WORLD
writing to file
text written to file
Reading from file
File contents are
HELLO_WORLD
gokul@gokul-ThinkPad-T460s:~/S4/OS/EXP4_FileSyscall$
```

Experiment-5

File management system calls

Aim

To write a C program to write to file and read the contents of the file using System calls.

Algorithm

1. Start
2. Declare Two integers fd and fd2
3. Declare Two Strings wbuf, rbuf
4. Open file using open and store the result of Operation in fd.
5. Print 'Enter text to be written'
6. read Buser input to wbuf
7. write wbuf to file using write() System call
8. close the file
9. Open the file to fd2
10. read the data in the file to rbuf using read() System call
11. Print rbuf
12. close the file
13. Stop

Result

Program executed Successfully and desired output is recived.