U18CO018

Shubham Shekhaliya

Assignment – 1 (SS)

**Fork()**

**Code:-**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

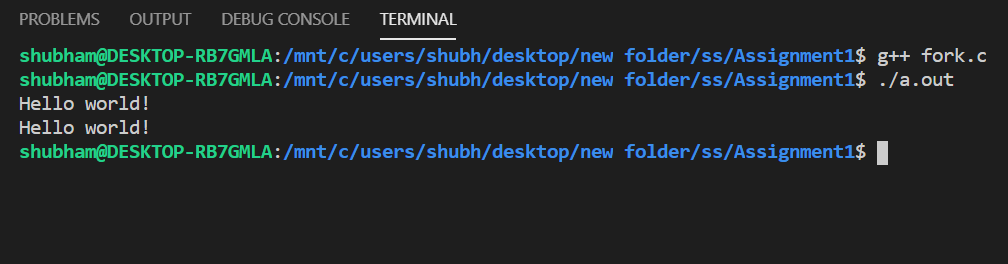
    fork();

    printf("Hello world!\n");

    return 0;

}

**Output:-**

****

**Exec()**

**Code:-**

**Execmain.c**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main() {

        char \*args[]={"./EXEC",NULL};

        execvp(args[0],args);

        printf("Ending-----");

    return 0;

}

**Exec.c**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main() {

        char \*args[]={"./EXEC",NULL};

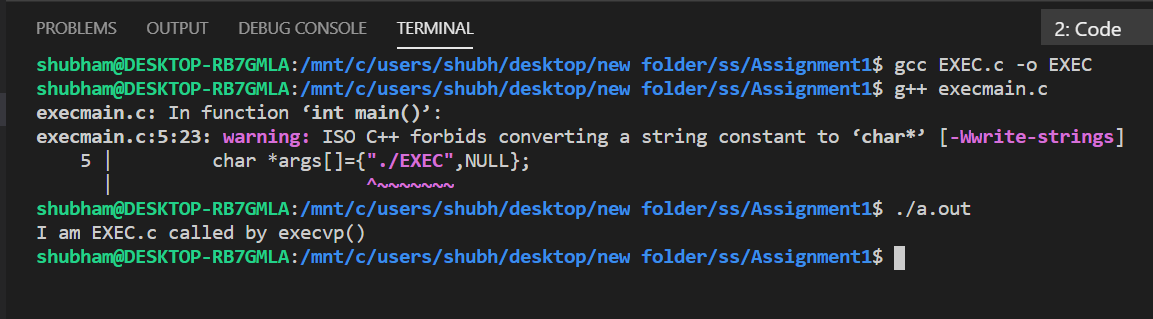
        execvp(args[0],args);

        printf("Ending-----");

    return 0;

}

**Output:-**

****

**Getpid()**

**Code:-**

#include<stdio.h>

#include<unistd.h>

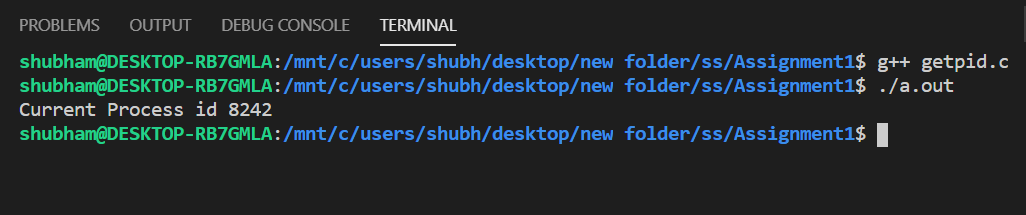
int main() {

    printf("Current Process id %d\n" , getpid());

    return 0;

}

**Output:-**

****

**exit()**

**Code:-**

#include<stdio.h>

#include<stdlib.h>

int main() {

    for (int i = 1;i<=10;i++) {

        printf("%d\n",i);

        if(i==5) {

            exit(0);

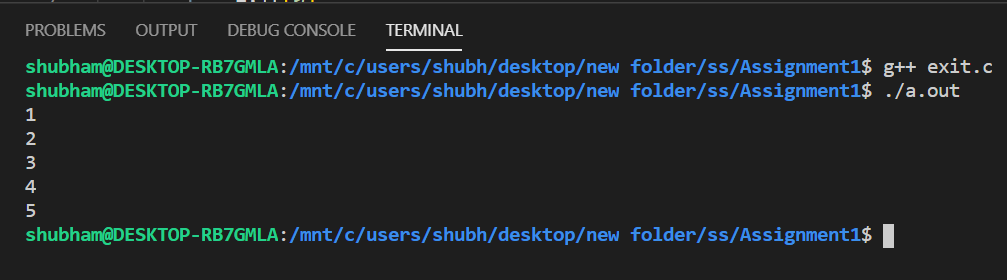
        }

    }

    return 0;

}

**Output:-**

****

**Open(), read(), write(), close(), stat()**

**Code:-**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <unistd.h>

int main() {

    int fd1 = open("./input.txt", O\_RDONLY);

    int fd2 = open("./output.txt", O\_WRONLY);

    if (fd1 < 0 || fd2 < 0) {

        printf("Cannot open files\n");

        exit(1);

    }

    struct stat st;

    fstat(fd1, &st);

    int size = st.st\_size;

    char \*c = (char \*)calloc(size, sizeof(char));

    read(fd1, c, size);

    printf("%s", c);

    write(fd2, c, size);

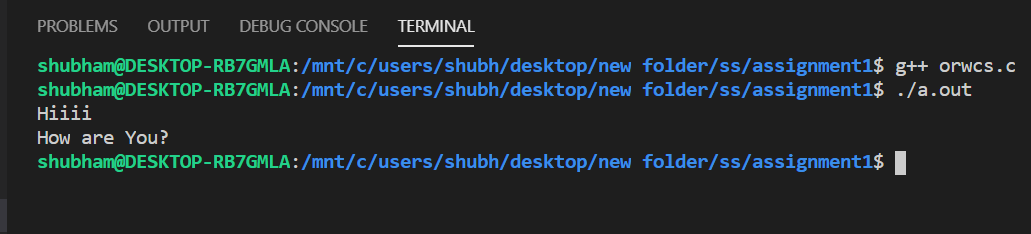
    close(fd1);

    close(fd2);

    return 0;

}

**Output:-**

****

**Opendir() closedir() readdir()**

**Code:-**

#include <dirent.h>

#include <stdio.h>

#include <unistd.h>

int main(void) {

    struct dirent \*de;

    DIR \*dr = opendir(".");

    if (dr == NULL) {

        printf("Could not open current directory");

        \_exit(1);

    }

    while ((de = readdir(dr)) != NULL) {

        printf("%s\n", de->d\_name);

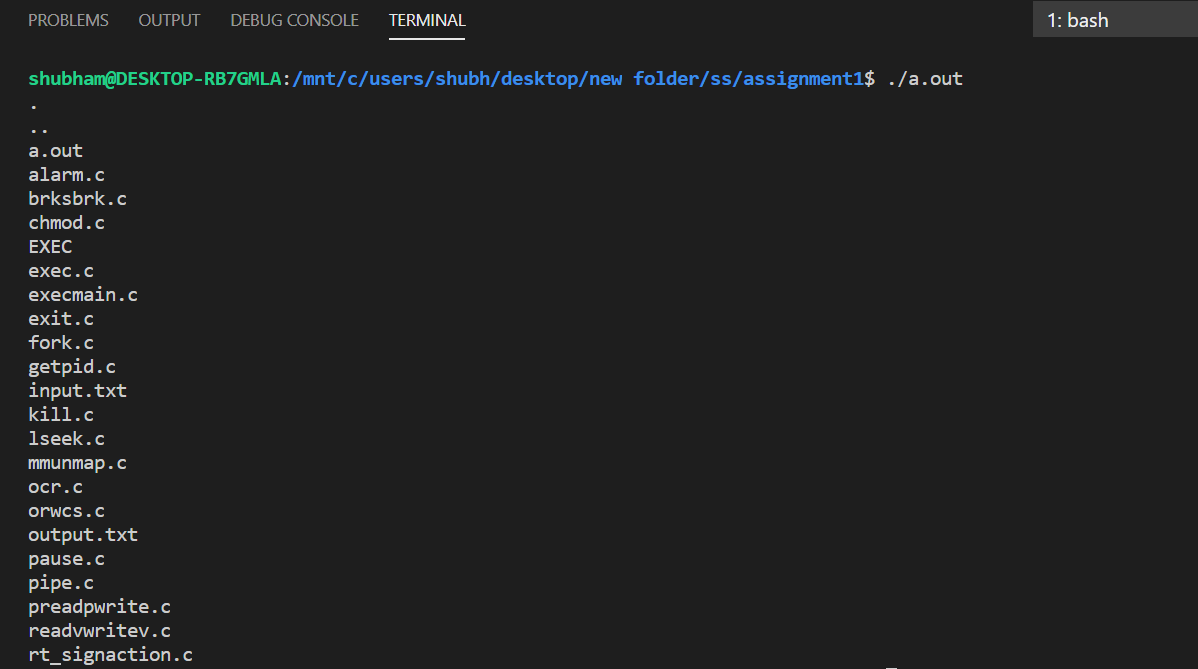
    }

    closedir(dr);

    return 0;

}

**Output:-**

****

**Chmod()**

**Code:-**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/stat.h>

int main() {

    mode\_t mode = 0755;

    uid\_t owner = 01000;

    uid\_t group = 01000;

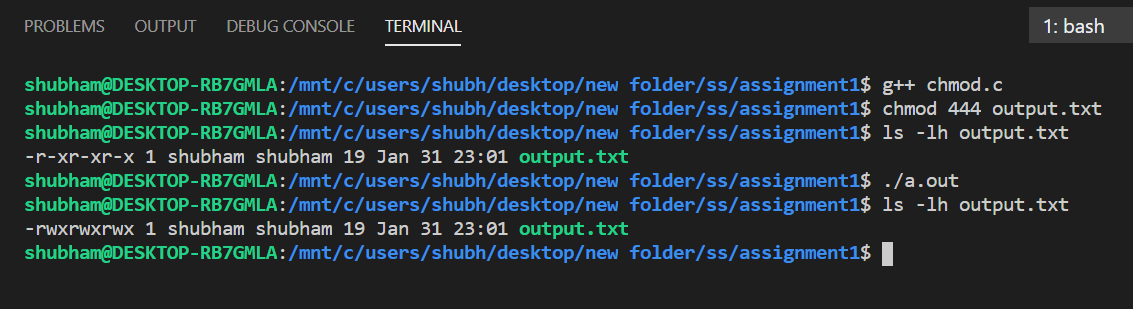
    chmod("./output.txt", mode);

    chown("./output.txt", owner, group);

    return 0;

}

**Output:-**

****

**lseek()**

**Code:-**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

    int fd = open("input.txt", O\_RDONLY);

    struct stat st;

    fstat(fd, &st);

    int size = st.st\_size;

    char \*c1 = (char \*)calloc(size, sizeof(char));

    char \*c2 = (char \*)calloc(size, sizeof(char));

    read(fd, c1, size);

    printf("%s\n", c1);

    // Move back to beginning of file

    lseek(fd, 0, SEEK\_SET);

    read(fd, c2, size);

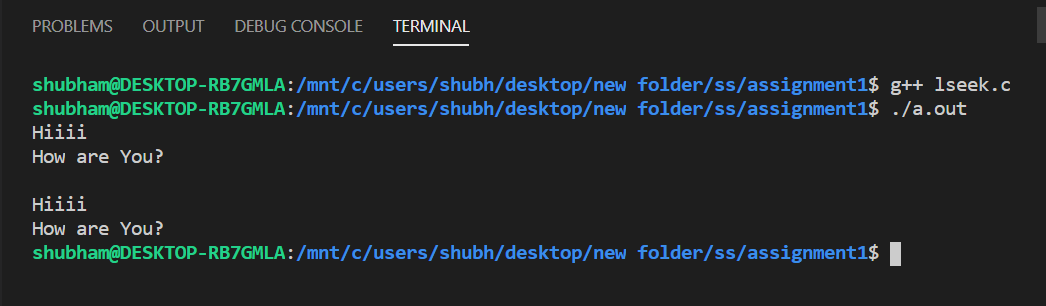
    printf("%s", c2);

    close(fd);

    return 0;

}

**Output:-**

****

**mmap(), munmap()**

**Code:-**

#include <fcntl.h>

#include <stdio.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

    int fd = open("./input.txt", O\_RDONLY);

    struct stat st;

    fstat(fd, &st);

    int size = st.st\_size;

    char \*data = (char \*)mmap(NULL, size, PROT\_READ, MAP\_PRIVATE, fd, 0);

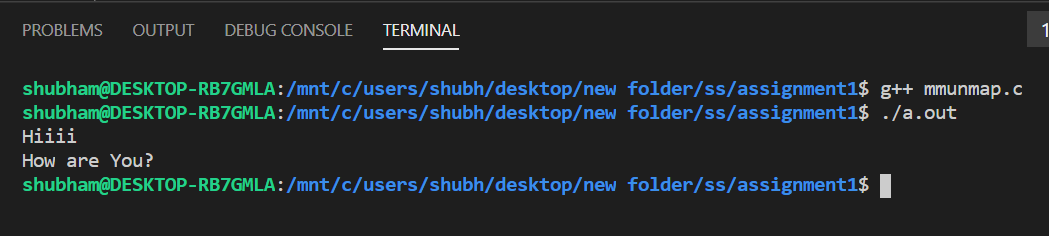
    printf("%s", data);

    munmap(data, size);

    return 0;

}

**Output:-**

****

**brk(), sbrk()**

**Code:-**

#include <unistd.h>

int main() {

    int \*p = (int \*)sbrk(0);

    //brk (p + 4);

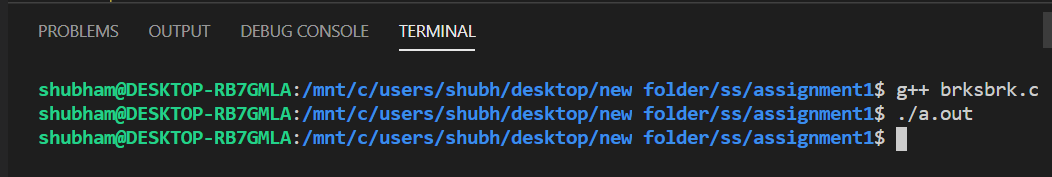
    \*p = 1;

    return 0;

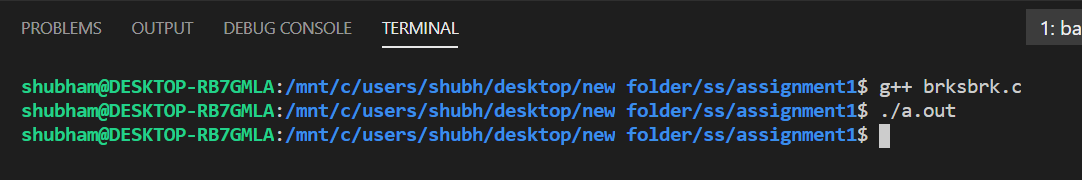
}

**Output:-**

**Without brk()**

****

**With brk()**

****

**rt\_sigaction()**

**Code:-**

#include <signal.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

void handler(int num) {

    char \*buf = "Ctrl C Called! \n";

    int len = strlen(buf);

    write(STDOUT\_FILENO, buf, len);

}

int main() {

    struct sigaction sa;

    sa.sa\_handler = handler;

    sigaction(SIGINT, &sa, NULL);

    printf("PID: %d", getpid());

    while (1)

    {

        printf("...");

        fflush(stdout);

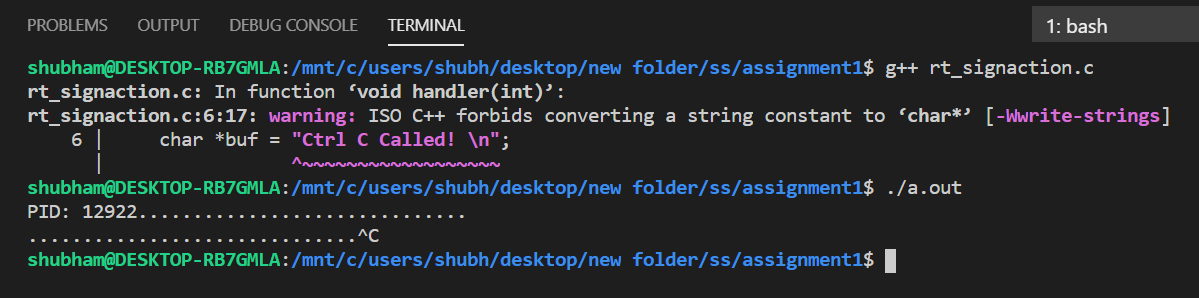
        sleep(1);

    }

    return 0;

}

**Output:-**

****

**I**t should not stop after prees ctrl+c but on my pc not working.

**pread(), pwrite()**

**Code:-**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <unistd.h>

#define OFFSET 4

int main() {

    int fd1 = open("./input.txt", O\_RDONLY);  //input

    int fd2 = open("./output.txt", O\_WRONLY); //output

    if (fd1 < 0 || fd2 < 0)

    {

        printf("Cannot open files\n");

        exit(1);

    }

    struct stat st;

    fstat(fd1, &st);

    int size = st.st\_size;

    char \*c = (char \*)calloc(size, sizeof(char));

    pread(fd1, c, size, OFFSET);

    printf("%s", c);

    pwrite(fd2, c, size - OFFSET, 0);

    close(fd1);

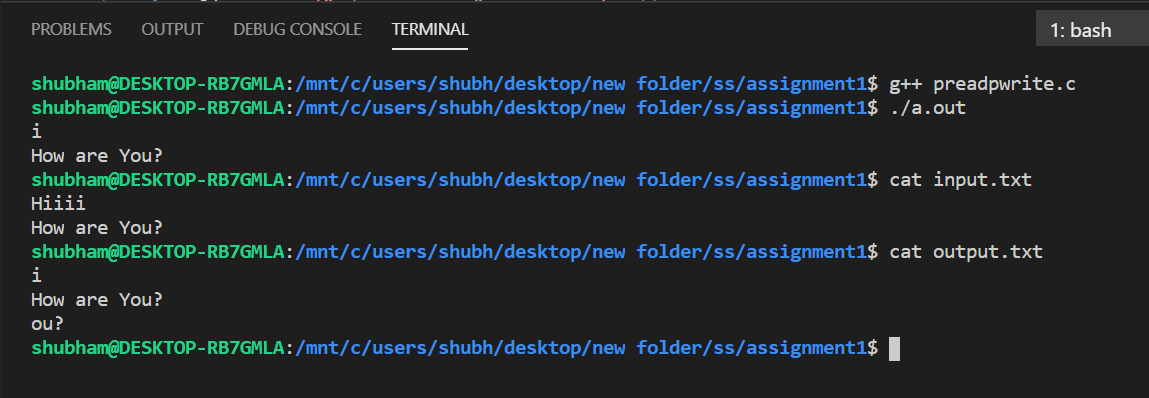
    close(fd2);

    free(c);

    return 0;

}

**Output:-**

****

**readv(), writev()**

**Code:-**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <sys/uio.h>

#include <unistd.h>

#define VECTOR\_COUNT 3

int main() {

    int fd1 = open("input.txt", O\_RDONLY); //input

    int fd2 = open("output.txt", O\_WRONLY); //output

    struct iovec vec[VECTOR\_COUNT];

    struct stat st;

    fstat(fd1, &st);

    int size = st.st\_size;

    for (int i = 0; i < VECTOR\_COUNT - 1; ++i)

    {

        vec[i].iov\_base = (char \*)calloc(size / VECTOR\_COUNT, sizeof(char));

        vec[i].iov\_len = size / VECTOR\_COUNT;

    }

    int rem = (size - (size / VECTOR\_COUNT) \* (VECTOR\_COUNT - 1));

    vec[VECTOR\_COUNT - 1].iov\_base = (char \*)calloc(rem, sizeof(char));

    vec[VECTOR\_COUNT - 1].iov\_len = rem;

    readv(fd1, vec, VECTOR\_COUNT);

    writev(fd2, vec, VECTOR\_COUNT);

    for (int i = 0; i < VECTOR\_COUNT; ++i)

    {

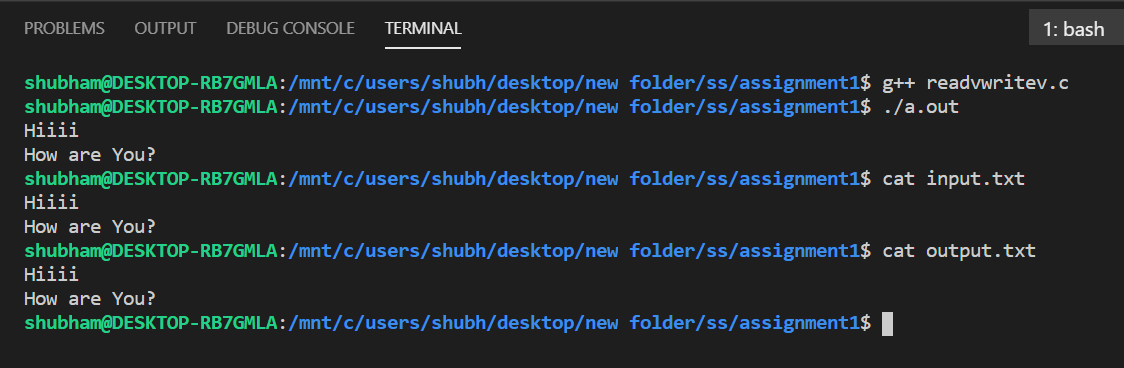
        printf("%s", (char \*)vec[i].iov\_base);

    }

    return 0;

}

**Output:-**

****

**alarm()**

**Code:-**

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <unistd.h>

void sig\_handler(int signum) {

    printf("Alarm Called\n");

    exit(0);

}

int main() {

    signal(SIGALRM, sig\_handler);

    alarm(2);

    while (1)

    {

        printf("...");

        fflush(stdout);

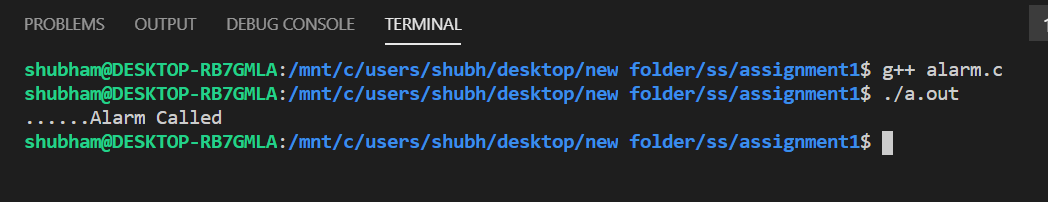
        sleep(1);

    }

    return 0;

}

**Output:-**

****

**kill()**

**Code:-**

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include <sys/types.h>

int main(){

    int pid = getpid(), x = 1;

    printf("PID: %d\n", pid);

    while (x++)

    {

        printf("Running...\n");

        sleep(1);

        if (x == 5)

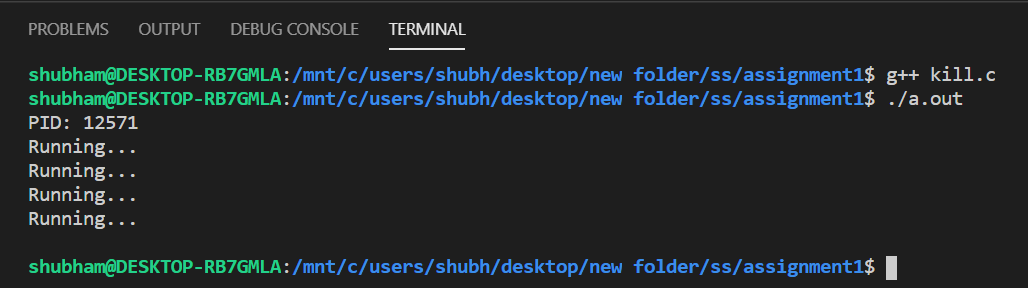
            kill(pid, SIGINT);

    }

    return 0;

}

**Output:-**

****

**pipe()**

**Code:-**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define OPSIZE 16

char \*op1 = "output 1";

char \*op2 = "output 2";

char \*op3 = "output 3";

int main() {

    char inbuf[OPSIZE];

    int p[2], i;

    if (pipe(p) < 0)

        exit(EXIT\_FAILURE);

    write(p[1], op1, OPSIZE);

    write(p[1], op2, OPSIZE);

    write(p[1], op3, OPSIZE);

    for (i = 0; i < 3; i++)

    {

        read(p[0], inbuf, OPSIZE);

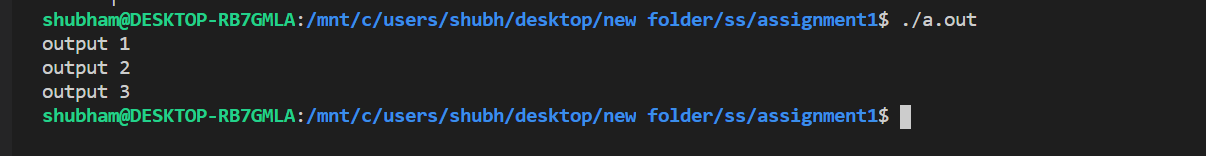
        printf("%s\n", inbuf);

    }

    return 0;

}

**Output:-**

****

**pause()**

**Code:-**

#include <stdio.h>

#include <unistd.h>

int main(void) {

    for (int i = 0; i < 10; i++) {

        printf("%d\n",i);

        if (i == 5)

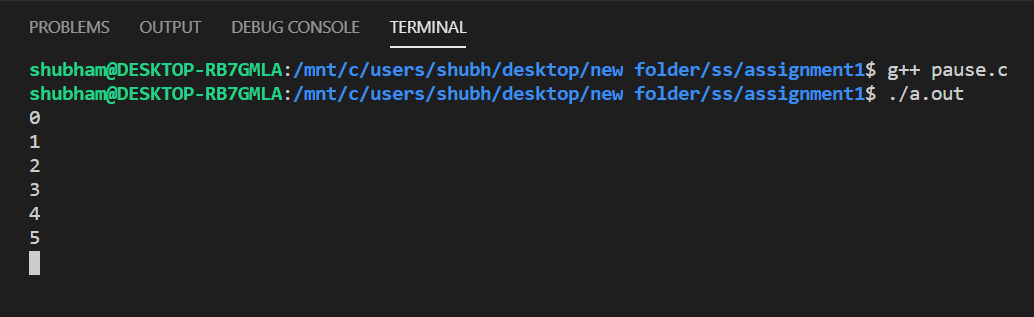
            pause();

    }

    return 0;

}

**Output:-**

****