**OS lab assignment 8**

**Agnim Gupta**

**2028082**

**A-23 CSSE**

**Question 1**

#include<stdio.h>

#include<unistd.h>

int main() {

int pipefds[2]; int returnstatus; int pid;

char writemessages[20][20]={"Hello", "World"};

char readmessage[20];

returnstatus = pipe(pipefds);

if (returnstatus == -1) {

printf("Unable to create pipe\n"); return 1;

}

pid = fork();

// Child process

if (pid == 0) {

read(pipefds[0], readmessage, sizeof(readmessage));

printf("Child Process - Reading from pipe – Message 1 is %s\n", readmessage);

read(pipefds[0], readmessage, sizeof(readmessage));

printf("Child Process - Reading from pipe – Message 2 is %s\n", readmessage);

}

else {

//Parent process

printf("Parent Process - Writing to pipe - Message 1 is %s\n", writemessages[0]);

write(pipefds[1], writemessages[0],sizeof(writemessages[0]));

printf("Parent Process - Writing to pipe - Message 2 is%s\n", writemessages[1]);

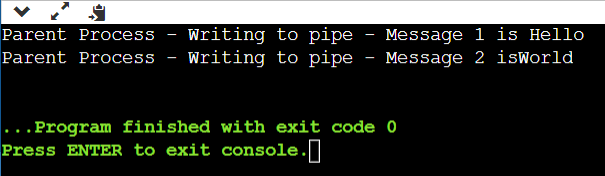
write(pipefds[1], writemessages[1], sizeof(writemessages[1]));

}

return 0;

}

**Output**

****

**Question 2**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(int argc, char \*argv[])

{

    int fd[2];

    int childID = 0;

    pipe(fd);

    for(int i=0;i<5;i++)

    {

    if (fork() != 0)

    {

    close(fd[0]); childID = 1;

    write(fd[1], &childID, sizeof(childID));

    printf("Parent(%d) send childID: %d\n", getpid(), childID);

    close(fd[1]);

    childID++;

    }

    else if (fork()==0)

    {

    close(fd[1]);

    read(fd[0], &childID, sizeof(childID));

    printf("Child(%d) received childID: %d\n", getpid(), childID);

    close(fd[0]);

    childID++;

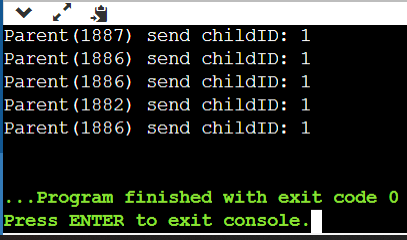
    }

    }

    return 0;

}

**Output**

****

**Question 3**

#include<stdio.h>

#include<unistd.h>

int main() {

int pipefds1[2], pipefds2[2];

int returnstatus1, returnstatus2;

int pid;

char pipe1writemessage[20] = "Hello";

char pipe2writemessage[20] = "Dear Friends";

char readmessage[20];

returnstatus1 = pipe(pipefds1);

if (returnstatus1 == -1) {

    printf("Unable to create pipe 1 \n");

    return 1;

}

returnstatus2 = pipe(pipefds2);

if (returnstatus2 == -1) {

     printf("Unable to create pipe 2 \n");

    return 1;

}

pid = fork();

if (pid != 0){

    close(pipefds1[0]); close(pipefds2[1]);

    printf("In Parent: Writing to pipe 1 – Message is %s\n", pipe1writemessage);

    write(pipefds1[1], pipe1writemessage, sizeof(pipe1writemessage));

    read(pipefds2[0], readmessage, sizeof(readmessage));

    printf("In Parent: Reading from pipe 2 – Message is %s\n", readmessage);

}

else {

    close(pipefds1[1]);

    close(pipefds2[0]);

    read(pipefds1[0], readmessage, sizeof(readmessage)); printf("In Child: Reading from pipe 1 – Message is %s\n",readmessage);

    printf("In Child: Writing to pipe 2 – Message is %s\n", pipe2writemessage);

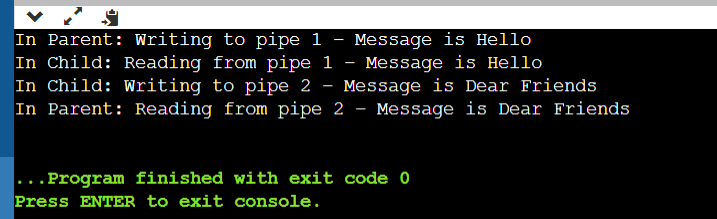
write(pipefds2[1], pipe2writemessage, sizeof(pipe2writemessage));

}

return 0;

}

**Output**

****

**Question 4**

#include<stdio.h>

#include <unistd.h>

#include <fcntl.h>

int main()

{

    int file\_desc = open("test.txt", O\_WRONLY | O\_APPEND);

    if(file\_desc < 0)

        printf("Error opening the file\n");

    int copy\_desc = dup(file\_desc);

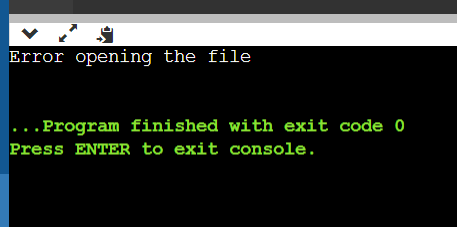
    write(copy\_desc,"This will be output to the file named test.txt\n", 46);

    write(file\_desc,"This will also be output to the file named test.txt\n", 51);

    return 0;

}

**Output**

****

**Question 5**

#include<stdlib.h>

#include<unistd.h>

#include<stdio.h>

#include<fcntl.h>

int main()

{

    int file\_desc = open("test2.txt",O\_WRONLY | O\_APPEND);

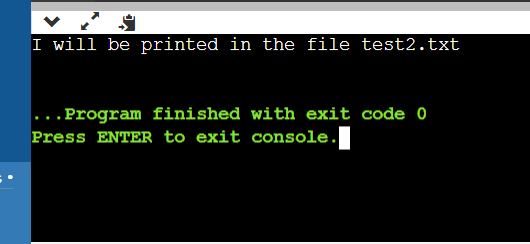
    dup2(file\_desc, 1) ;

    printf("I will be printed in the file test2.txt\n");

    return 0;

}

**Output**

****

**Question 6**

#include<unistd.h>

#include<stdio.h>

#include<fcntl.h>

int main()

{

    int old\_fd, new\_fd;

    old\_fd=open("test.txt",O\_RDWR);

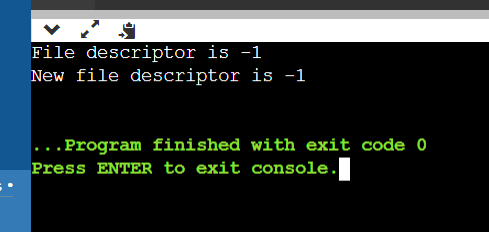
    printf("File descriptor is %d\n",old\_fd);

    new\_fd=dup(old\_fd);

    printf("New file descriptor is %d\n",new\_fd);

}

**Output**

****