Name- Manish kumar

Roll no- 2020ITB007

Assignment: OS\_IPC 2

1.Write a C program where the parent and a child process will exchange their information through PIPE IPC.

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <string.h>

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

{

    int pipefd[2];

    pid\_t x;

    char str[1];

    pipe(pipefd);

    x = fork();

    if (x == 0)

    {

        printf("From child: \n");

        close(pipefd[1]);

        while (read(pipefd[0], &str, 1))

        {

            printf("%s", str);

        }

        printf("\n");

        close(pipefd[0]);

    }

    else

    {

        close(pipefd[0]);

        write(pipefd[1], argv[1], strlen(argv[1]));

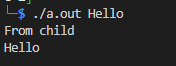
        close(pipefd[1]);

        wait(NULL);

    }

}

OUTPUT



2. Write a C program where the main process and the threading processes will exchange their data through pipe IPC.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <string.h>

void \*display(void \*arg)

{

    pthread\_detach(pthread\_self());

    printf("Displaying from inside the thread:\n");

    int \*pipefd = arg;

    char buf[1];

    while (read(pipefd[0], &buf, 1))

    {

        // printf("%s", buf);   // Gives garbage values write(STDOUT\_FILENO, &buf, 1);

    }

    close(pipefd[0]);

    pthread\_exit(NULL);

}

int main()

{

    int pipefd[2];

    // char str[100];

    char \*str = (char \*)malloc(sizeof(char));

    pipe(pipefd);

    printf("Enter a string: ");

    scanf("%[^\n]s", str);

    write(pipefd[1], str, strlen(str));

    close(pipefd[1]);

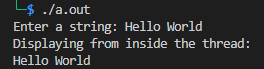
    pthread\_t thread;

    pthread\_create(&thread, NULL, &display, &pipefd);

    pthread\_join(thread, NULL);

    pthread\_exit(NULL);

}

OUTPUT

3. Write another C program where the parent process will create a child and then replace it with an existing process from the hard disk storage by calling the EXEC family functions. The main process and the new process will now communicate with the PIPE IPC. The main process will open a pipe and will pass the descriptors through the suitable EXEC family functions to the new process for IPC.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <unistd.h>

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

{

    int pipefd[2];

    pipe(pipefd);

    char \*msg = (char \*)malloc(sizeof(char));

    printf("Enter the message: ");

    scanf("%[^\n]s", msg);

    pid\_t x;

    x = fork();

    if (x == 0)

    {

        close(pipefd[1]);

        // int size = snprintf(NULL, 0, "%d", pipefd[0]); int size = sizeof(pipefd[0]) / sizeof(int);

        char \*str = (char \*)malloc(size + 1);

        // char str[size + 1];

        printf(str, size + 1, "%d", pipefd[0]);

        execlp("./file", "monke", str, NULL);

    }

    else

    {

        close(pipefd[0]);

        write(pipefd[1], msg, strlen(msg));

        close(pipefd[1]);

        wait(NULL);

    }

    return 0;

}

FILE.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

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

{

    int pipefd;

    char buf[1];

    printf("From file.c\n");

    sscanf(argv[1], "%d", &pipefd);

    while (read(pipefd, &buf, 1))

    {

        printf("%s", buf);

    }

    printf("\n");

    close(pipefd);

    return 0;

}

# Compile file.c & execute ipc2\_3.c

