# TASK 1:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

int fd[2],n;

char buffer[100];

pid\_t p;

pipe(fd);

p=fork();

if(p>0)

{

printf("Parent Passing value to child\n");

write(fd[1],"hello\n",6);

wait(50);

}

else

{

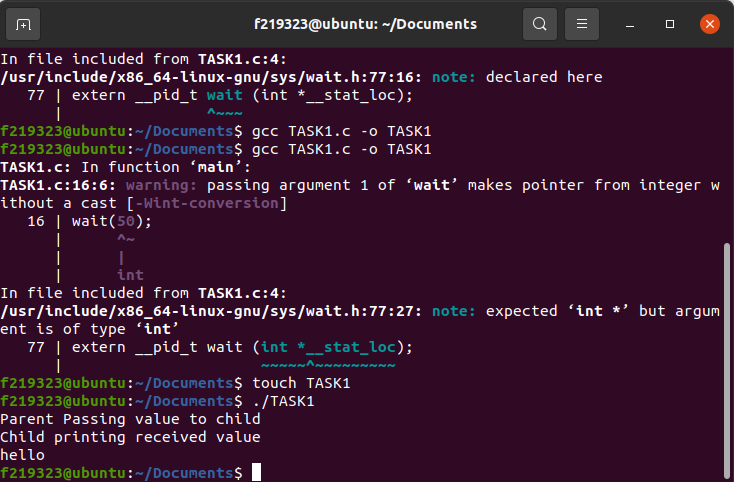
printf("Child printing received value\n");

n=read(fd[0],buffer,100);

write(1,buffer,n);

}

}



# TASK2:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#define BUFFER\_SIZE 100

int main() {

int fd[2];

pid\_t pid;

char buffer[BUFFER\_SIZE];

if (pipe(fd) == -1) {

fprintf(stderr, "Pipe failed");

return 1;

}

pid = fork();

if (pid < 0) {

fprintf(stderr, "Fork failed");

return 1;

}

if (pid > 0) { // parent process

close(fd[1]); // close the write end of the pipe

while (1) {

printf("Waiting for message from P2...\n");

read(fd[0], buffer, BUFFER\_SIZE); // read message from P2

printf("P2:

# TASK 3:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#define READ\_END 0

#define WRITE\_END 1

int main() {

int pipe\_add\_sub[2];

int pipe\_mul\_div[2];

int pid1, pid2, pid3;

int a, b, c, d, result\_add\_sub, result\_mul\_div;

if (pipe(pipe\_add\_sub) == -1) {

perror("Error creating pipe for addition and subtraction");

exit(EXIT\_FAILURE);

}

if (pipe(pipe\_mul\_div) == -1) {

perror("Error creating pipe for multiplication and division");

exit(EXIT\_FAILURE);

}

pid1 = fork();

if (pid1 == -1) {

perror("Error forking first child");

exit(EXIT\_FAILURE);

} else if (pid1 == 0) {

close(pipe\_add\_sub[WRITE\_END]);

read(pipe\_add\_sub[READ\_END], &a, sizeof(int));

read(pipe\_add\_sub[READ\_END], &b, sizeof(int));

result\_add\_sub = a + b;

write(pipe\_add\_sub[WRITE\_END], &result\_add\_sub, sizeof(int));

close(pipe\_add\_sub[READ\_END]);

exit(EXIT\_SUCCESS);

} else {

close(pipe\_add\_sub[READ\_END]);

printf("Enter two numbers for addition and subtraction: ");

scanf("%d %d", &a, &b);

write(pipe\_add\_sub[WRITE\_END], &a, sizeof(int));

write(pipe\_add\_sub[WRITE\_END], &b, sizeof(int));

read(pipe\_add\_sub[READ\_END], &result\_add\_sub, sizeof(int));

printf("Result of addition and subtraction: %d\n", result\_add\_sub);

close(pipe\_add\_sub[WRITE\_END]);

pid2 = fork();

if (pid2 == -1) {

perror("Error forking second child");

exit(EXIT\_FAILURE);

} else if (pid2 == 0) {

close(pipe\_mul\_div[WRITE\_END]);

read(pipe\_mul\_div[READ\_END], &c, sizeof(int));

read(pipe\_mul\_div[READ\_END], &d, sizeof(int));

result\_mul\_div = c \* d;

write(pipe\_mul\_div[WRITE\_END], &result\_mul\_div, sizeof(int));

close(pipe\_mul\_div[READ\_END]);

exit(EXIT\_SUCCESS);

} else {

close(pipe\_mul\_div[READ\_END]);

printf("Enter two numbers for multiplication and division: ");

scanf("%d %d", &c, &d);

write(pipe\_mul\_div[WRITE\_END], &c, sizeof(int));

write(pipe\_mul\_div[WRITE\_END], &d, sizeof(int));

read(pipe\_mul\_div[READ\_END], &result\_mul\_div, sizeof(int));

# TASK 4:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#define BUFFER\_SIZE 1024

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

int pipefd[2];

pid\_t pid;

char buffer[BUFFER\_SIZE];

ssize\_t num\_bytes;

int total\_bytes = 0;

int fd;

// Ensure the user provided a shell command

if (argc < 2) {

printf("Usage: %s command\n", argv[0]);

return 1;

}

// Create the pipe

if (pipe(pipefd) == -1) {

perror("pipe");

return 1;

}

// Fork the process

pid = fork();

if (pid == -1) {

perror("fork");

return 1;

}

if (pid == 0) {

// Child process

close(pipefd[0]); // Close the unused read end of the pipe

dup2(pipefd[1], STDOUT\_FILENO); // Redirect stdout to the write end of the pipe

close(pipefd[1]); // Close the write end of the pipe

// Execute the shell command provided by the user

execlp(argv[1], argv[1], NULL);

perror("execlp"); // This line should not be reached if execlp is successful

exit(1);

} else {

// Parent process

close(pipefd[1]); // Close the unused write end of the pipe

// Open the file for writing

fd = open("result.txt", O\_WRONLY |

# TASK 5:

My npiper reader.c:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#define FIFO\_NAME "/tmp/mypipe"

int main()

{

int fd;

char buffer[256];

mkfifo(FIFO\_NAME, 0666);

fd = open(FIFO\_NAME, O\_RDONLY);

printf("Waiting for input...\n");

while (1) {

ssize\_t bytes\_read = read(fd, buffer, sizeof(buffer));

if (bytes\_read <= 0) {

break;

}

buffer[bytes\_read] = '\0';

printf("Got it: '%s'\n", buffer);

if (strcmp(buffer, "exit\n") == 0) {

break;

}

}

printf("Exiting\n");

close(fd);

unlink(FIFO\_NAME);

return 0;

}

My npiper writer.c:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#define FIFO\_NAME "/tmp/mypipe"

int main()

{

int fd;

char buffer[256];

fd = open(FIFO\_NAME, O\_WRONLY);

printf("Opening named pipe: %s\n", FIFO\_NAME);

while (1) {

printf("Enter Input: ");

fgets(buffer, sizeof(buffer), stdin);

write(fd, buffer, strlen(buffer));

printf("Writing buffer to pipe...done\n");

if (strcmp(buffer, "exit\n") == 0) {

break;

}

}

printf("Exiting\n");

close(fd);

return 0;

}