Note: This example, taken from pages 174–176 of the text by Haviland et al., shows how one can set up a named pipe between a sender process and a receiver process.

It should be noted that the sender and receiver processes don’t have a parent-child relationship. They are created independently, and they communicate using the named pipe.

To run this program, the following sequence of steps must be carried out: (1) Create the executables for both the sender and the receiver. (2) Run the receiver in the background. (3) Run the sender.

/* Sender program. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MSGSIZE 63
#define FIFO_MODE 0666
char *fifo = "FIFO"; /* Name of FIFO. */

int main(int argc, char *argv[]) {
    int fd, j, nwrite;
    char msgbuf[MSGSIZE+1]; /* Messages to fifo will be at most 63 bytes long. */
    if (argc < 2) {
        fprintf(stderr, "Usage: send m1 m2 ... \n"); exit(1);
    }
    /* The sender assumes that the fifo will be created by the receiver. */
    /* Open the fifo for non-blocking writes. */
    if ((fd = open(fifo, O_WRONLY | O_NONBLOCK)) == -1) {
        fprintf(stderr, "Sender: FIFO open failed.\n"); exit(1);
    }
    /* Send messages. */
    for (j = 1; j < argc; j++) {
        if (strlen(argv[j]) > MSGSIZE) {
            fprintf(stderr, "Sender: Message too long -- %s\n", argv[j]); exit(1);
        }
        strcpy(msgbuf, argv[j]);
        if ((nwrite = write(fd, msgbuf, MSGSIZE+1)) == -1) {
            fprintf(stderr, "Sender: Write to fifo failed.\n"); exit(1);
        }
    } /* End of for loop. */
    return 0;
} /* End of main for sender. */
/* Receiver program. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>

#define MSGSIZE 63
#define FIFO_MODE 0666

char *fifo = "FIFO"; /* Name of FIFO. */

int main(void) {
    int fd;
    char msgbuf[MSGSIZE+1]; /* Messages to fifo will be at most 63 bytes long. */

    /* Create fifo if it doesn't already exist. */
    if (mkfifo(fifo, FIFO_MODE) == -1) {
        if (errno != EEXIST) {
            fprintf(stderr, "Receiver: Couldn't create fifo.\n"); exit(1);
        }
    }

    /* Open the fifo for read and write. */
    if ((fd = open(fifo, O_RDWR)) == -1) {
        fprintf(stderr, "Receiver: FIFO open failed.\n"); exit(1);
    }

    /* Receive messages. */
    for (;;) {
        if (read(fd, msgbuf, MSGSIZE+1) == -1) {
            fprintf(stderr, "Receiver: Reading from fifo failed.\n"); exit(1);
        }
        /* Print the message. */
        printf("Received message: \%s\n", msgbuf);
    }
    /* End of for loop. */
    return 0;
} /* End of main for receiver. */