

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

SISTEMAS OPERATIVOS I

PRACTICA 7: TUBERIAS (PIPES)



Docente:

Prof. Marcos González Flores

Alumno:

Jesús Huerta Aguilar

Alex Abdiel Ruano Flores

Matricula:

202041509

202075025

NRC: 46152

Sección: 003

QUINTO SEMESTRE

Practica No.7

Nombre: Tuberías (Pipes)

Objetivo: Aprender a crear tuberías con nombre y sin nombre, así como enviar mensajes a través de una tubería.

Desarrollo:

EJERCICIO 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 256

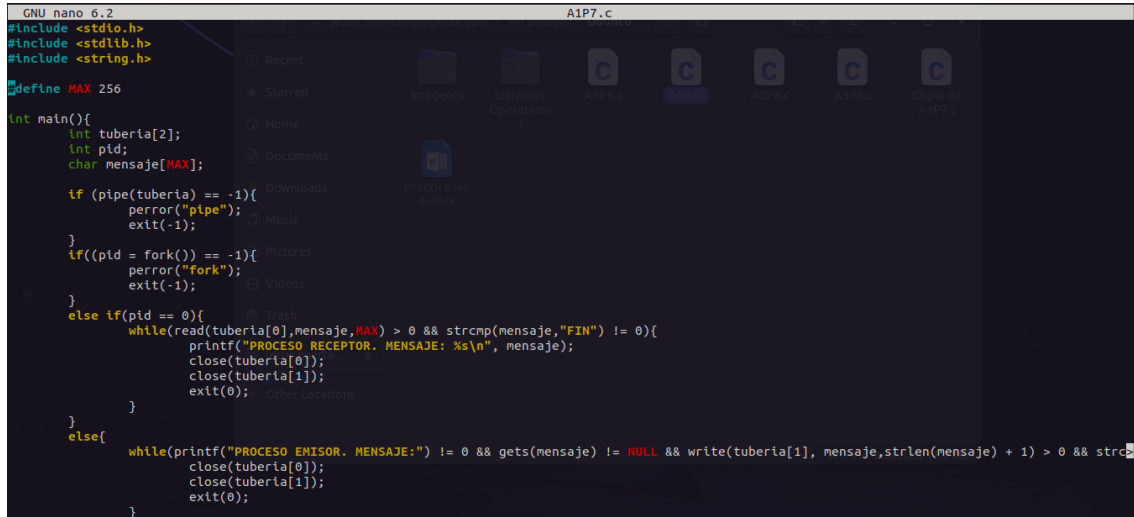
int main(){
    int tuberia[2];
    int pid;
    char mensaje[MAX];

    if (pipe(tuberia) == -1){
        perror("pipe");
        exit(-1);
    }
    if((pid = fork()) == -1){
        perror("fork");
        exit(-1);
    }
    else if(pid == 0){
        while(read(tuberia[0],mensaje,MAX) > 0 &&
strcmp(mensaje,"FIN") != 0){
            printf("PROCESO RECEPTOR. MENSAJE: %s\n",
mensaje);

            close(tuberia[0]);
            close(tuberia[1]);
            exit(0);
        }
    }
    else{
        while(printf("PROCESO EMISOR. MENSAJE:") != 0 &&
gets(mensaje) != NULL && write(tuberia[1], mensaje,strlen(mensaje) +
1) > 0 && strcmp

            close(tuberia[0]);
            close(tuberia[1]);
            exit(0);
        }
    }
}
```

TRANSCRITO + EJECUCIÓN:

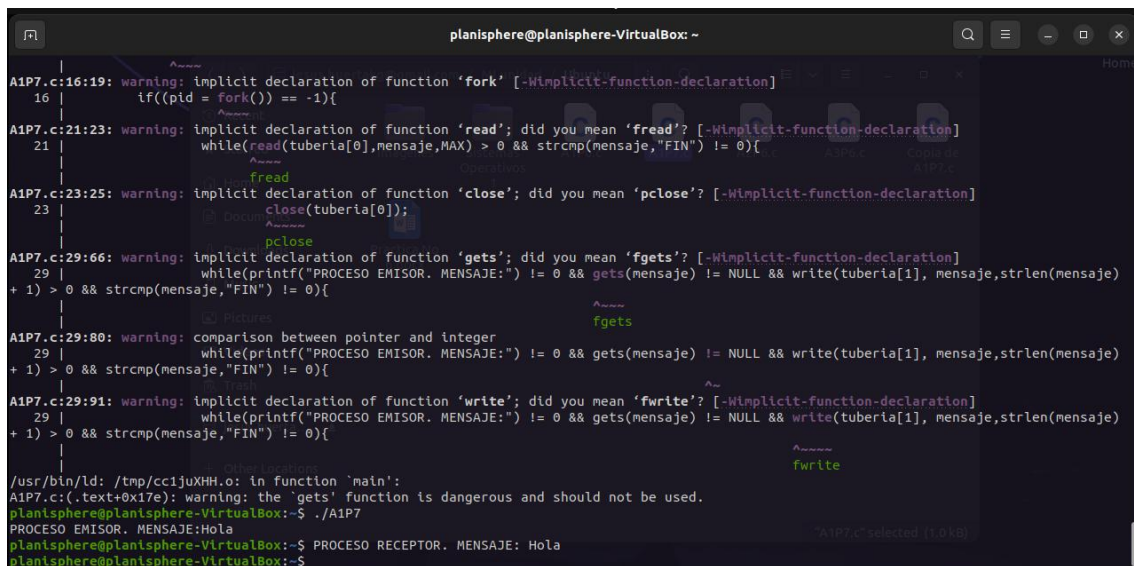


```
GNU nano 6.2 A1P7.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 256

int main(){
    int tuberia[2];
    int pid;
    char mensaje[MAX];

    if (pipe(tuberia) == -1){
        perror("pipe");
        exit(-1);
    }
    if((pid = fork()) == -1){
        perror("fork");
        exit(-1);
    }
    else if(pid == 0){
        while(read(tuberia[0],mensaje,MAX) > 0 && strcmp(mensaje,"FIN") != 0){
            printf("PROCESO RECEPTOR. MENSAJE: %s\n", mensaje);
            close(tuberia[0]);
            close(tuberia[1]);
            exit(0);
        }
    }
    else{
        while(printf("PROCESO EMISOR. MENSAJE:") != 0 && gets(mensaje) != NULL && write(tuberia[1], mensaje,strlen(mensaje) + 1) > 0 && strcmp(mensaje,"FIN") != 0){
            close(tuberia[0]);
            close(tuberia[1]);
            exit(0);
        }
    }
}
```



```
planisphere@planisphere-VirtualBox: ~
A1P7.c:16:19: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
16 |     if((pid = fork()) == -1){
   |                   ^
A1P7.c:21:23: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
21 |     while(read(tuberia[0],mensaje,MAX) > 0 && strcmp(mensaje,"FIN") != 0){
   |             ^
A1P7.c:23:25: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
23 |         close(tuberia[0]);
   |         ^
A1P7.c:29:66: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
29 |         while(printf("PROCESO EMISOR. MENSAJE:") != 0 && gets(mensaje) != NULL && write(tuberia[1], mensaje,strlen(mensaje)
+ 1) > 0 && strcmp(mensaje,"FIN") != 0){
   |                                                              ^
A1P7.c:29:80: warning: comparison between pointer and integer
29 |         while(printf("PROCESO EMISOR. MENSAJE:") != 0 && gets(mensaje) != NULL && write(tuberia[1], mensaje,strlen(mensaje)
+ 1) > 0 && strcmp(mensaje,"FIN") != 0){
   |                                                              ^
A1P7.c:29:91: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
29 |         while(printf("PROCESO EMISOR. MENSAJE:") != 0 && gets(mensaje) != NULL && write(tuberia[1], mensaje,strlen(mensaje)
+ 1) > 0 && strcmp(mensaje,"FIN") != 0){
   |                                                              ^
/usr/bin/ld: /tmp/cc1juxHH.o: in function 'main':
A1P7.c:(.text+0x17e): warning: the 'gets' function is dangerous and should not be used.
planisphere@planisphere-VirtualBox:~$ ./A1P7
PROCESO EMISOR. MENSAJE:Hola
planisphere@planisphere-VirtualBox:~$ PROCESO RECEPTOR. MENSAJE: Hola
planisphere@planisphere-VirtualBox:~$
```

EJERCICIO 2

```
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <wait.h>

#define SIZE 512

int main(int argc, char **argv) {
    pid_t pid;
    int a[2], b[2], readbytes;
    char buffer[SIZE];

    pipe (a);
    pipe (b);

    if((pid = fork()) == 0) {
        close(a[1]); //cierra escritura
        close(b[0]); //cierra lectura
        while((readbytes = read(a[0],buffer,SIZE)) > 0)
            write(1,buffer, readbytes);
        close(a[0]);
        strcpy(buffer, "Soy tu hijo hablandote por la otra
tuberia\n");
        write(b[1],buffer, strlen(buffer));
        close(b[1]);
    }
    else{
        close(a[0]); //cierra lectura
        close(b[1]); //cierra escritura
        strcpy(buffer, "Soy tu padre hablandote por una
tuberia\n");
        write(a[1],buffer,strlen(buffer));
        close(a[1]);
        while((readbytes = read(b[0],buffer,SIZE)) > 0)
            write(1,buffer, readbytes);
        close(b[0]);
    }
    waitpid (pid, NULL,0);
    exit(0);
}
```

TRANSCRITO + EJECUCIÓN:

```
GNU nano 6.2 A2P7.c
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <wait.h>

#define SIZE 512

int main(int argc, char **argv){
    pid_t pid;
    int a[2], b[2], readbytes;
    char buffer[SIZE];

    pipe(a);
    pipe(b);

    if((pid = fork()) == 0){
        close(a[1]); //cierra escritura
        close(b[0]); //cierra lectura
        while((readbytes = read(a[0],buffer,SIZE)) > 0)
            write(1,buffer,readbytes);
        close(a[0]);
        strcpy(buffer,"Soy tu hijo hablandote por la otra tubería\n");
        write(b[1],buffer,strlen(buffer));
        close(b[1]);
    }
    else{
        Read 42 lines
    }
}

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

```
GNU nano 6.2 A2P7.c

    pipe(a);
    pipe(b);

    if((pid = fork()) == 0){
        close(a[1]); //cierra escritura
        close(b[0]); //cierra lectura
        while((readbytes = read(a[0],buffer,SIZE)) > 0)
            write(1,buffer,readbytes);
        close(a[0]);
        strcpy(buffer,"Soy tu hijo hablandote por la otra tubería\n");
        write(b[1],buffer,strlen(buffer));
        close(b[1]);
    }
    else{
        close(a[0]); //cierra lectura
        close(b[1]); //cierra escritura
        strcpy(buffer,"Soy tu padre hablandote por una tubería\n");
        write(a[1],buffer,strlen(buffer));
        close(a[1]);

        while((readbytes = read(b[0],buffer,SIZE)) > 0)
            write(1,buffer,readbytes);
        close(b[0]);
    }
    waitpid(pid,NULL,0);
    exit(0);
}

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```


EJERCICIO 3

A3P7.c

```
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(void) {
    int fd;
    char buf[] = "Hola, pase por la tuberia";
    mkfifo("fifo2",0666);
    FD = open("fifo2",O_WRONLY);
    write(fd,buf,sizeof(buf));
    // printf("Numero de bytes rx: %d \n",n);
    // printf("RX Mensaje: %s\n", buf);
    close(fd);
    return 0;
}
```

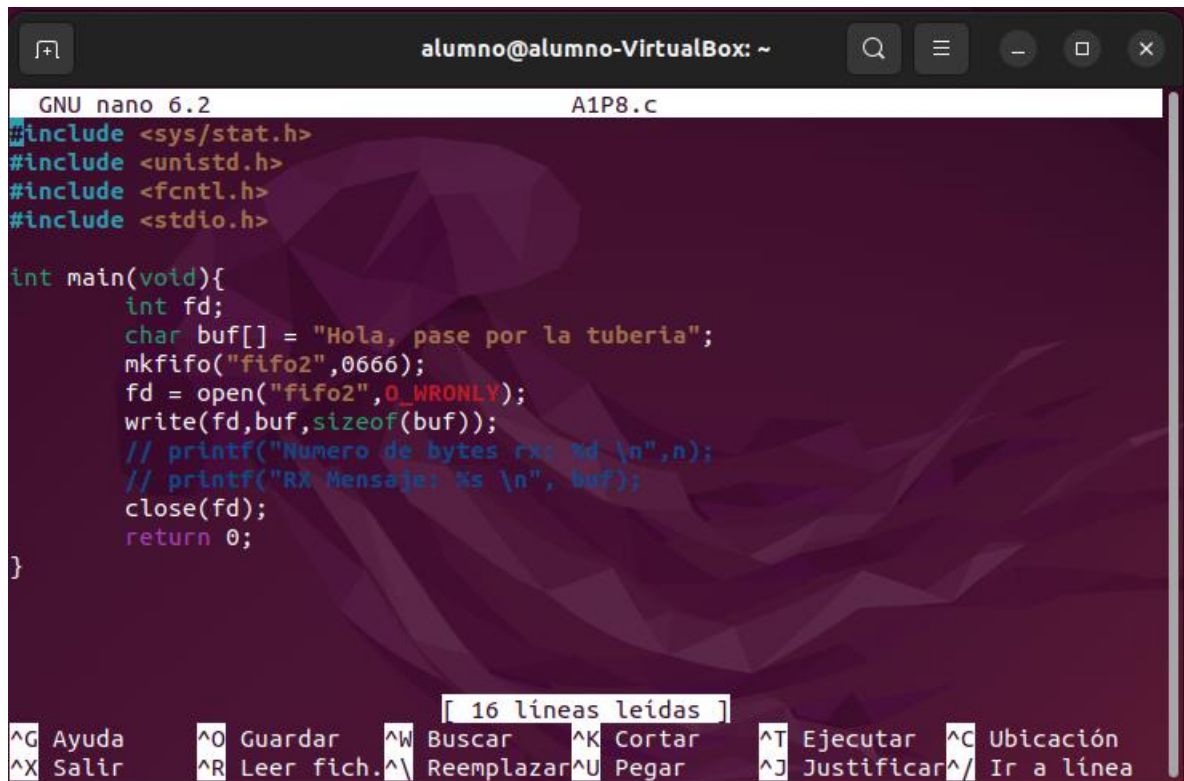
receptor.c

```
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(void) {
    int fd,n;
    char buf[1024];

    fd = open("fifo2",O_RDONLY);
    n = read(fd, buf,sizeof(buf));
    printf("Numero de bytes rx: %d \n",n);
    printf("RX Mensaje: %s\n", buf);
    close(fd);
    return 0;
}
```

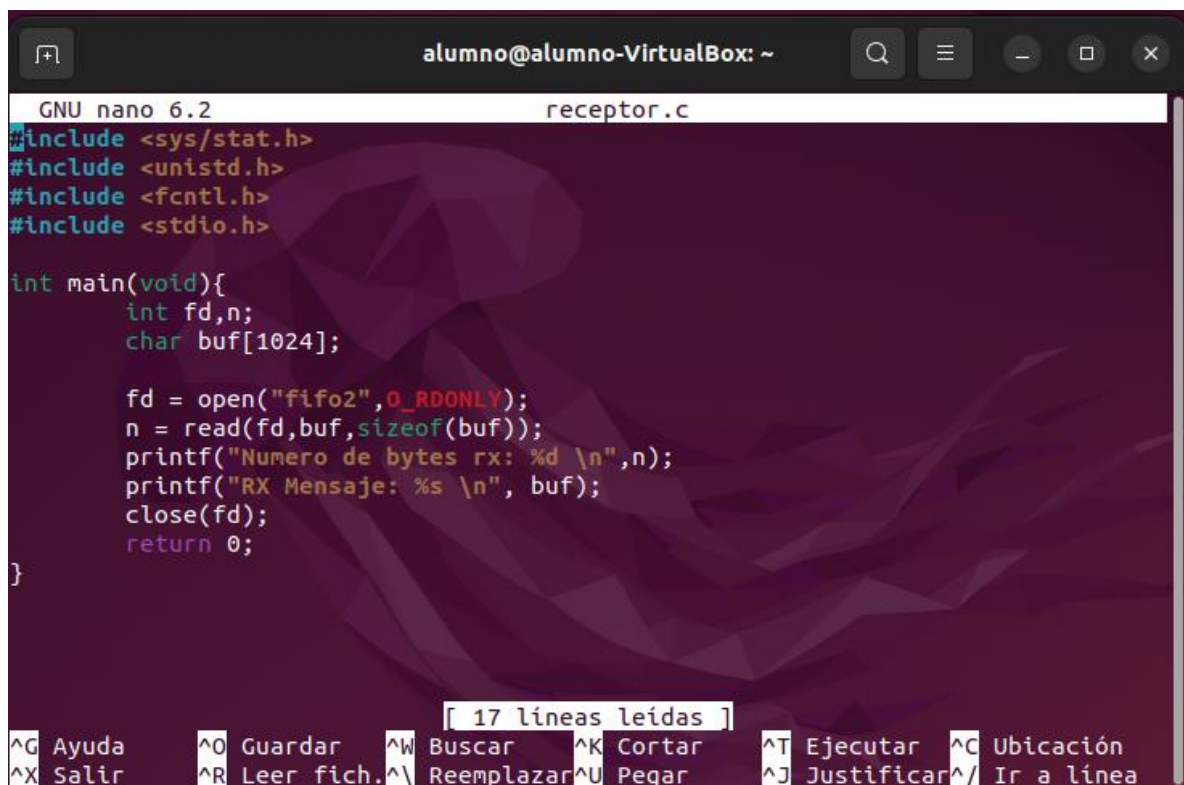
TRANSCRITO + EJECUCIÓN:



```
alumno@alumno-VirtualBox: ~  
GNU nano 6.2 A1P8.c  
#include <sys/stat.h>  
#include <unistd.h>  
#include <fcntl.h>  
#include <stdio.h>  
  
int main(void){  
    int fd;  
    char buf[] = "Hola, pase por la tuberia";  
    mkfifo("fifo2",0666);  
    fd = open("fifo2",O_WRONLY);  
    write(fd,buf,sizeof(buf));  
    // printf("Numero de bytes rx: %d \n",n);  
    // printf("RX Mensaje: %s \n", buf);  
    close(fd);  
    return 0;  
}
```

[16 líneas leídas]

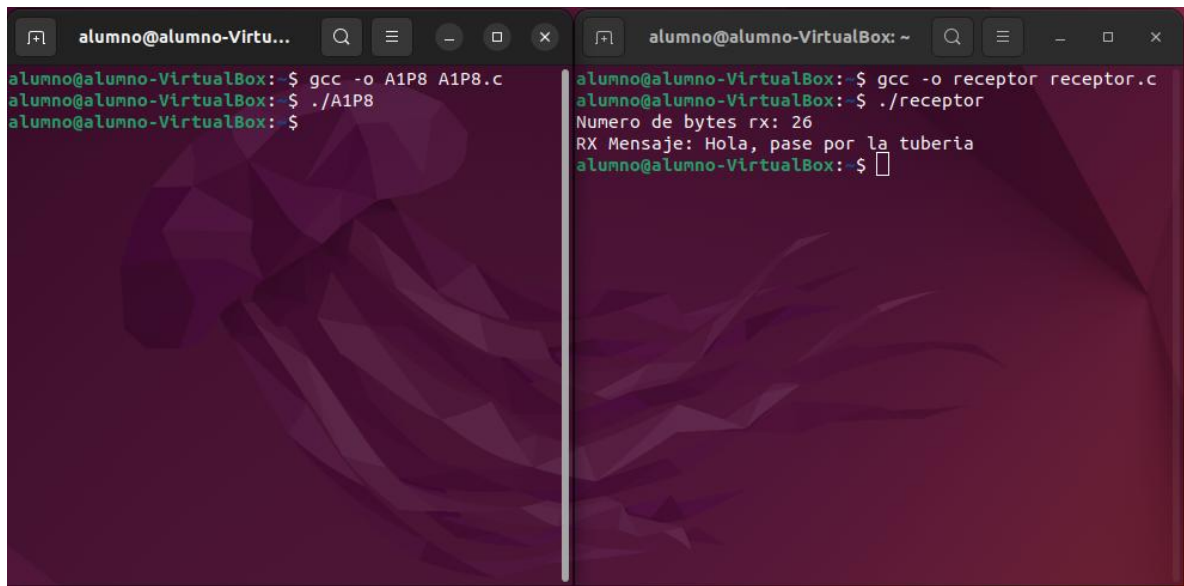
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea



```
alumno@alumno-VirtualBox: ~  
GNU nano 6.2 receptor.c  
#include <sys/stat.h>  
#include <unistd.h>  
#include <fcntl.h>  
#include <stdio.h>  
  
int main(void){  
    int fd,n;  
    char buf[1024];  
  
    fd = open("fifo2",O_RDONLY);  
    n = read(fd,buf,sizeof(buf));  
    printf("Numero de bytes rx: %d \n",n);  
    printf("RX Mensaje: %s \n", buf);  
    close(fd);  
    return 0;  
}
```

[17 líneas leídas]

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea



```
alumno@alumno-Virtu...  
alumno@alumno-VirtualBox:~$ gcc -o A1P8 A1P8.c  
alumno@alumno-VirtualBox:~$ ./A1P8  
alumno@alumno-VirtualBox:~$  
alumno@alumno-VirtualBox:~$ gcc -o receptor receptor.c  
alumno@alumno-VirtualBox:~$ ./receptor  
Numero de bytes rx: 26  
RX Mensaje: Hola, pase por la tuberia  
alumno@alumno-VirtualBox:~$
```