**ENUNCIADO**

**Actividad 8**

Realiza un programa en C donde los procesos abuelo, hijo y nieto se envíen mensajes de acuerdo al siguiente gráfico:

fd1 fd2

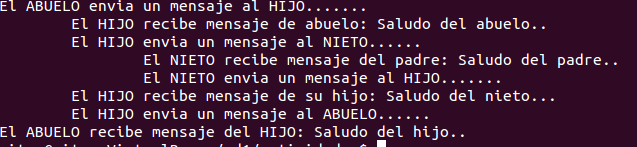
ABUELO –--------> HIJO –---------> NIETO

←------- < ----------

fd2 fd1

Nota: como se puede observar en el gráfico, se utilizan 2 pipes.

**Resultado esperado:**



**Solución:**

Incluir aquí el código del programa Actividad8.c

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

void main ()

{

int fd1[2];

char buffer[80];

char saludoAbuelo[]="Saludo del abuelo..";

pid\_t pid;

pipe(fd1); // Se crea el pipe o tubería

pid = fork();

switch (pid) {

case -1: // error

printf("No se ha podido crear el proceso hijo...\n");

exit (-1);

break;

case 0: // Hijo recibe

close(fd1[1]); // Cierra el descriptor de escritura

read (fd1[0], buffer, sizeof(buffer)); // leo el pipe

printf ("\tEl Hijo recibe mensaje de abuelo: %s\n", buffer);

// crear el mensaje para el nieto

int fd2[2];

char saludoPadre[]="Saludo del padre..";

char saludoHijo[]="Saludo del hijo..";

pipe(fd2); // Se crea el pipe o tubería

pid = fork();

switch (pid) {

case -1: // error

printf("No se ha podido crear el proceso nieto...\n");

exit (-1);

break;

case 0: // Nieto recibe

close(fd2[1]); // Cierra el descriptor de escritura

read (fd2[0], buffer, sizeof(buffer)); // leo el pipe

printf ("\t\tEl Nieto recibe mensaje del padre: %s\n", buffer);

// crear el mensaje para el hijo

char saludoNieto[]="Saludo del nieto..";

pipe(fd1); // Se crea el pipe o tubería

pid = fork();

switch (pid) {

case -1: // error

printf("No se ha podido crear el proceso hijo...\n");

exit (-1);

break;

case 0: // Hijo recibe

close(fd1[1]); // Cierra el descriptor de escritura

read (fd1[0], buffer, sizeof(buffer)); // leo el pipe

printf ("\tEl Hijo recibe mensaje de su hijo: %s\n", buffer);

// crear el mensaje para el abuelo

pipe(fd2); // Se crea el pipe o tubería

pid = fork();

switch (pid) {

case -1: // error

printf("No se ha podido crear el proceso hijo...\n");

exit (-1);

break;

case 0: // Hijo recibe

close(fd2[1]); // Cierra el descriptor de escritura

read (fd2[0], buffer, sizeof(buffer)); // leo el pipe

printf ("El Abuelo recibe mensaje del Hijo: %s\n", buffer);

break;

default: // Hijo envia a Abuelo

close(fd2[0]); // Cierra el descriptor de lectura

write (fd2[1], saludoHijo, sizeof(saludoHijo));

printf ("\tEl Hijo envía un mensaje al abuelo...\n");

wait (NULL); // Espera al proceso hijo

break;

}

break;

default: // Nieto envia a hijo

close(fd1[0]); // Cierra el descriptor de lectura

write (fd1[1], saludoNieto, sizeof(saludoNieto));

printf ("\t\tEl Nieto envía un mensaje al Hijo...\n");

wait (NULL); // Espera al proceso hijo

break;

}

break;

default: // Padre Envía a nieto

close(fd2[0]); // Cierra el descriptor de lectura

write (fd2[1], saludoPadre, sizeof(saludoPadre));

printf ("\tEl Hijo envía un mensaje al Nieto...\n");

wait (NULL); // Espera al proceso hijo

break;

}

break;

default: // Abuelo Envía

close(fd1[0]); // Cierra el descriptor de lectura

write (fd1[1], saludoAbuelo, sizeof(saludoAbuelo));

printf ("El Abuelo envía un mensaje al hijo...\n");

wait (NULL); // Espera al proceso hijo

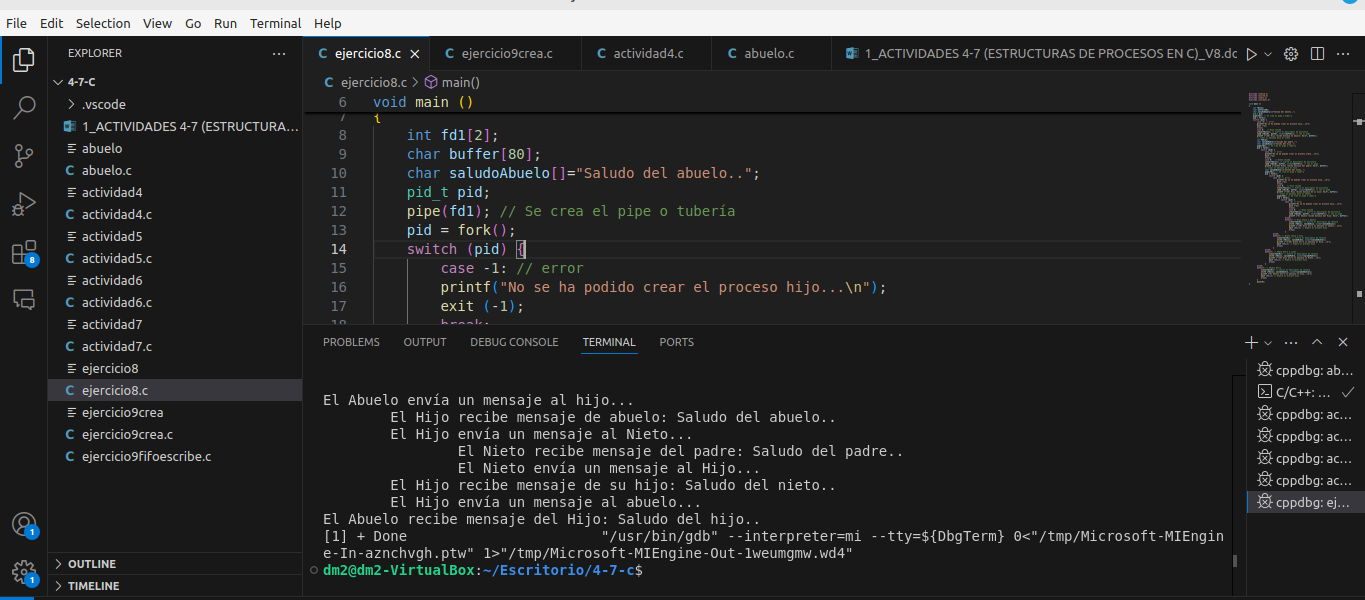
break;

}

exit(0);

}

**Resolución de problemas:**

****

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Correcto funcionamiento de los programas desarrollados en C (35%).
3. Ajuste a los resultados esperados (35%).
4. Claridad del código C (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida).
6. Avisos o errores de compilación (-15%).
7. Nivel de comentarios en el código fuente (-15%).