**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

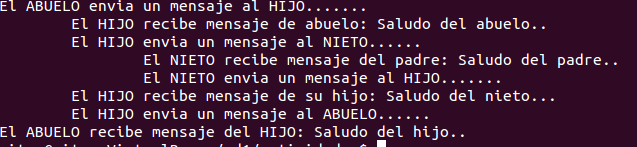
ABUELO –--------> HIJO –---------> NIETO

←------- < ----------

fd2 fd1

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

**Resultado esperado:**



**Solución:**

**#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 lee el 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:**

El problema era que el proceso "hijo" cierra el extremo de escritura de fd2 antes de que el "abuelo" tenga la oportunidad de leer el mensaje final.

**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%).