Ejercicio para comenzar a entender el funcionamiento de los pipes

El siguiente es un ejercicio para la práctica del uso de pipes dentro de la programación en lenguaje C, solamente tiene un objetivo didáctico y no un objetivo aplicativo dentro de los sistemas operativos.

Enunciado del problema:

Tenemos un proceso padre y dos procesos hijos, se entiende que los procesos hijos son creados por el proceso padre usando la llamada al sistema pid_t fork(void), esto determina que los hijos hereden los recursos creados previamente por el padre, de esta manera los hijos heredarán el recurso pipe del padre para realizar la comunicación entre ellos.

Queremos que el padre y uno de los hijos (Hijo 2) trabajen cooperativamente para la creación del abecedario en mayúsculas, el padre escribe la letra ‘A’ en un pipe, el hijo debe poner detrás de la letra ‘A’, la letra ‘B’, el padre debe poner la letra ‘C’ detrás de la ‘B’, hasta el momento en un pipe tenemos “CBA”, suponiendo que la escritura se realice en el extremo izquierdo y la lectura por el extremo derecho, un punto final indica que se acabaron las letras en el pipe entonces quedaría en el pipe lo siguiente ‘.CBA’

Cuando se completó el abecedario en un pipe el otro hijo (Hijo 1) que no participo de la creación del abecedario lee del pipe el abecedario y lo muestra por pantalla.

Observaciones

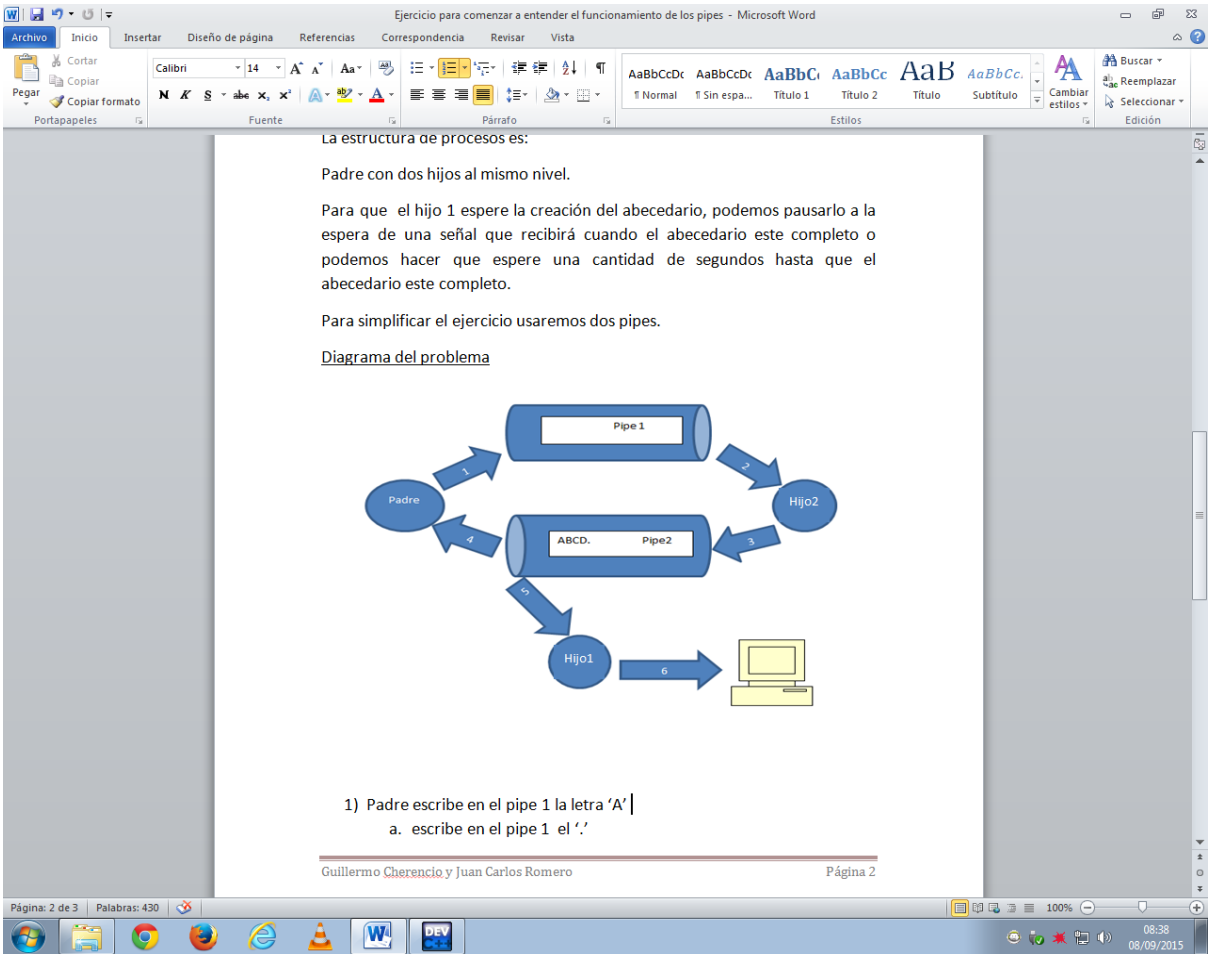
La estructura de procesos es:

Padre con dos hijos al mismo nivel.

Para que el hijo 1 espere la creación del abecedario, podemos pausarlo a la espera de una señal que recibirá cuando el abecedario este completo o podemos hacer que espere una cantidad de segundos hasta que el abecedario este completo.

Para simplificar el ejercicio usaremos dos pipes.

Diagrama del problema



1) Padre escribe en el pipe 1 la letra ‘A’
    a. escribe en el pipe 1 el ‘.’
2) Hijo 2 lee del pipe 1 la letra ‘A’
3) Hijo 2 escribe en el pipe 2 la letra ‘A’
    a. escribe en el pipe 2 la letra ‘B’
    b. escribe en el pipe 2 el ‘.’
4) Padre lee del pipe 2 la letra ‘A’
    a. escribe en pipe 1 la letra ‘A’
    b. lee del pipe 2 la letra ‘B’
    c. escribe en el pipe 1 la letra ‘B’
    d. escribe en el pipe 1 la letra ‘C’
    e. escribe en el pipe 1 el ‘.’
5) Cuando el abecedario está completo y el hijo 1 recibe la señal o simplemente ya pasaron los segundos de espera, lee del pipe y escribe en pantalla.

Puedes consultar a juancarlosjromer@gmail.com

Solución 1: PipeAbcPunto01.c

Lo más importante a destacar de esta solución es que el proceso padre y el proceso hijo2, escriben y leen en sus pipes correspondientes en forma concurrente, esto quiere decir que mientras el proceso Padre escribe en el Pipe 1 el proceso Hijo 2 puede estar leyendo del Pipe 1 o puede estar escribiendo en el Pipe 2, de forma no simultánea, solo en forma concurrente porque estamos en un entorno mono procesamiento.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

void Padre(void);
void Hijo1(void);
void Hijo2(void);

int p1[2],p2[2] ;
char letra , letraant;
pid_t pid1 , pid2 ;

int main(void)
{
   pipe(p1);
   pipe(p2);
   pid1 = fork();
   if ( pid1 == 0 ) //hijo 1
         Hijo1();
```

```
   else  // padre crea hijo 2
     {
        pid2 = fork() ;
        if (pid2 ==  0)
         {
           close(p1[1]); //el hijo2 no escribe en el pipe 1
           close(p2[0]); //el hijo2 no lee del pipe 2
           while(1)  Hijo2();
         }
        else
         {
            close(p1[0]); //el padre no lee del pipe 1
            close(p2[1]); //el padre no escribe en el pipe 2
            write(p1[1],"A.",2);
            while(1)  Padre();
         }
     }
   exit(0) ;
}

void Padre(void)
{
   while(read(p2[0],&letra,1)>0)
     {
//      printf("Padre dice: Lee de p2 %c\n",letra);
        if ( letra == 'Z')
        {
  //       printf("Padre dice: Lei Z Escribi Z \n");
```

```
          write(p1[1],&letra,1);
          close(p2[0]);
          close(p1[1]);
          close(p1[0]);
          exit(0) ; //si el proceso termina no es necesario hacer los close anteriores, solo es a modo
//didáctico
        }
       if ( letra != '.' )
        {
          letraant = letra ;
          write(p1[1],&letra,1);
        }
       else
         if ( letra == '.' )
           {
             letra = letraant + 1 ;
             write(p1[1],&letra,1);
             write(p1[1],".",1);
             if (letra == 'Z')
               {
            //    printf("Padre dice: Genere la Z Escribi la Z\n");
                  close(p2[0]);
                  close(p1[1]);
                  close(p1[0]);
                  kill(pid2,SIGINT);
                  exit(0); //si el proceso termina no es necesario hacer los close anteriores, solo es a
//modo didáctico
               }
             break;
```

```
            }
      }
 }


void Hijo2(void)
{
  while(read(p1[0],&letra,1)>0)
    {
   // printf("Hijo2 dice: Lee de p1 %c\n",letra);
      if ( letra == 'Z')
        {
      //  printf("Hijo 2 dice: Lei Z Escribi Z \n");
          write(p2[1],&letra,1);
          close(p1[0]);
          close(p2[1]);
          close(p2[0]);
          exit(0); //si el proceso termina no es necesario hacer los close anteriores, solo es a modo
//didáctico
        }
      if ( letra != '.' )
        {
          letraant = letra ;
          write(p2[1],&letra,1);
        }
      else
        if ( letra == '.' )
          {
            letra = letraant + 1 ;
```

```
            write(p2[1],&letra,1);
            write(p2[1],".",1);
            if (letra == 'Z')
              {
//              printf("Hijo 2 dice: Genere la Z Escribi la Z\n");
                close(p1[0]);
                close(p2[1]);
                close(p2[0]);
                kill(getppid(),SIGINT);
                exit(0) ; //si el proceso termina no es necesario hacer los close anteriores, solo es a
//modo didáctico
              }
          break;
        }
    }
}

void Hijo1(void)
{
  close(p1[1]);
  close(p2[1]);
  printf("Hijo 1 dice: Espere tres segundos\n");
  sleep(3);
  printf("Hijo 1 dice: leo de p1\n");
  while(read(p1[0],&letra,1)>0)
    {
      printf("%c " , letra);
      if ( letra == 'Z') break ;
```

```
    }
  printf("\n");
  printf("Hijo 1 dice: p1 Vacio\n");


  printf("Hijo 1 dice: leo de p2\n");
  while(read(p2[0],&letra,1)>0)
    {
      printf("%c " , letra);
      if ( letra == 'Z') break;
     }
   printf("\n");
   printf("Hijo 1 dice: p2 Vacio\n");
   exit(0);
}
```

Solución 2: PipeAbcPunto02.c

Los aspectos más importantes a destacar de esta solución son:

1) El proceso Padre y el proceso Hijo2, muestran el contenido del pipe que escribieron, el padre luego de escribir en el pipe1 muestra su contenido y el hijo 2 luego de escribir en el pipe2 muestra su contenido.
2) Se utilizan señales para lograr el punto anterior, por ejemplo el hijo 2 escribe en el pipe2 “.DCBA” luego muestra lo que escribió en pantalla luego señaliza al padre para que comience a leer del pipe2 y escriba en el pipe1.
3) El punto anterior implica que mientras el hijo2 está escribiendo en el pipe2, el padre no puede leer del pipe2 y no puede escribir en el pipe1, esta es una gran diferencia con la solución anterior.
4) Otra diferencia con la solución 1, es que en este caso el hijo 1 se pone en pausa esperando que el padre o el hijo 2 le envíen una señal

cuando el abecedario este completo en algún pipe, para pueda comenzar a mostrar por pantalla.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

void  MuestraContenidoPipe1(void);
void  MuestraContenidoPipe2(void);

void leerp2escribirp1(void);
void leerp1escribirp2(void);

void Hijo1(void);

char letra = '?' , letraant;
pid_t  pid1 , pid2 ;
int p1[2],p2[2] ;

int main(void)
{

  pipe(p1);
  pipe(p2);

  system("clear");
```

```
pid1 = fork() ;
if (pid1 > 0 ) // Padre
{
  pid2 = fork() ;
  if ( pid2 > 0 ) // padre
   {
    signal(SIGUSR1,(void *)leerp2escribirp1);
    close(p2[1]);
    write(p1[1],"A.",2);
    MuestraContenidoPipe1();
    while(1)
     {
       sleep(1);
       kill(pid2,SIGUSR2);
       pause();
       MuestraContenidoPipe1();
     }
 }
 else // hijo 2
     {
        signal(SIGUSR2,(void *)leerp1escribirp2);
        close(p1[1]);
        while(1)
        {
           pause();
           MuestraContenidoPipe2();
           sleep(1);
```

```
            kill(getppid(),SIGUSR1);
          }
        }
    }
    else // Hijo 1
     {
       signal(SIGINT,(void*)Hijo1) ;
       pause();
     }
     return 0 ;
}

//Padre
void leerp2escribirp1(void)
{
    close(p2[1]);
    while(read(p2[0],&letra,1)>0)
      {
        if ( letra == 'Z')
          {
            printf("Padre dice: Lei Z Escribi Z \n");
            write(p1[1],&letra,1);
            MuestraContenidoPipe1();
            exit(0) ;
          }

        if ( letra != '.' )
          {
```

```
            letraant = letra ;
            write(p1[1],&letra,1);
          }
        else
          if ( letra == '.' )
            {
              letra = letraant + 1 ;
              write(p1[1],&letra,1);
              write(p1[1],".",1);
              if (letra == 'Z')
                {
                  printf("Padre dice: Genere la Z Escribi la Z\n");
                  MuestraContenidoPipe1();
                  close(p2[0]);
                  close(p1[1]);
                  close(p1[0]);
                  kill(pid2,SIGINT);
                  kill(pid1,SIGINT);
                  exit(0);
                 }
              break;
             }
    }
}



//Hijo 2
void leerp1escribirp2(void)
```

```
{
    close(p1[1]);
    while(read(p1[0],&letra,1)>0)
      {
        if ( letra == 'Z')
          {
            printf("Hijo 2 dice: Lei Z Escribi Z \n");
            write(p2[1],&letra,1);
            MuestraContenidoPipe2();
            exit(0);
          }

        if ( letra != '.' )
         {
            letraant = letra ;
            write(p2[1],&letra,1);
         }
        else
          if ( letra == '.' )
            {
              letra = letraant + 1 ;
              write(p2[1],&letra,1);
              write(p2[1],".",1);
              if (letra == 'Z')
                {
                  printf("Hijo 2 dice: Genere la Z Escribi la Z\n");
                  MuestraContenidoPipe2();
                  close(p1[0]);
```

```
                    close(p2[1]);
                    close(p2[0]);
                    kill(getppid(),SIGINT);
                    kill(pid1,SIGINT);
                    exit(0) ;
                  }
                break;
              }
    }
}


void MuestraContenidoPipe1(void)
{
   char letra ;
   printf("Padre muestra pipe 1 \n") ;
   while(read(p1[0],&letra,1))
       {
          write(p1[1],&letra,1);
          printf("%c",letra) ;
          if (letra == '.') break;
       }
   printf("\n") ;
}


void MuestraContenidoPipe2(void)
{
```

```
   char letra ;
   printf("Hijo 2 muestra pipe 2 \n") ;
   while(read(p2[0],&letra,1))
       {
          write(p2[1],&letra,1);
          printf("%c",letra) ;
          if (letra == '.') break;
       }
   printf("\n") ;
}



void Hijo1(void)
{
  close(p1[1]);
  close(p2[1]);
  printf("Hijo 1 dice: leo de p1\n");
  while(read(p1[0],&letra,1)>0)
    {
       printf("%c " , letra);
       if ( letra == 'Z') break ;
    }
  printf("\n");
  printf("Hijo 1 dice: p1 Vacio\n");

  printf("Hijo 1 dice: leo de p2\n");
  while(read(p2[0],&letra,1)>0)
    {
```

```
      printf("%c " , letra);
      if ( letra == 'Z') break;
     }
   printf("\n");
   printf("Hijo 1 dice: p2 Vacio\n");
   exit(0);
}
```