Práctica 3

Aguilar Zúñiga,Gibran 308071087
Alexis Hernández castro 313006636
Jesus Martin Ortega Martinez 310183534
Daniel Lopez Hernández 309167282
Jaime Alberto Martínez López 309256753

10 de diciembre de 2018

Resumen

Manejo de mmap(), read(), Child's Play, fork(), pipe(), write() y kill().

1. Preguntas

1. mmap()

   a) Leer un archivo a través de la llamada al sistema read(), imprimir su contenido y el valor de su descriptor de archivo.

Figura 1: Código lectura archivo con Read()

b) Describir como se mapea este archivo en memoria

read: Lee el archivo indicado por el descriptor de archivo fd, la función read () lee bytes de entrada cnt en el área de memoria indicada por buf. Una lectura () exitosa actualiza el tiempo de acceso para el archivo.

c) Abrir un archivo mediante la llamada al sistema mmap() e imprimir su contenido

Figura 2: Código lectura archivo con MMMPA()

d) Describir como se mapea este archivo en memoria.

e) Realizar una comparación entre ambos métodos y las ventajas de uno sobre el otro

mmap utiliza tabla de páginas de su proceso, una estructura de datos que utiliza su CPU para asignar espacios de direcciones. La CPU traducirá las direcciones virtuales a las físicas, y lo hará de acuerdo con la tabla de páginas configurada por su núcleo.

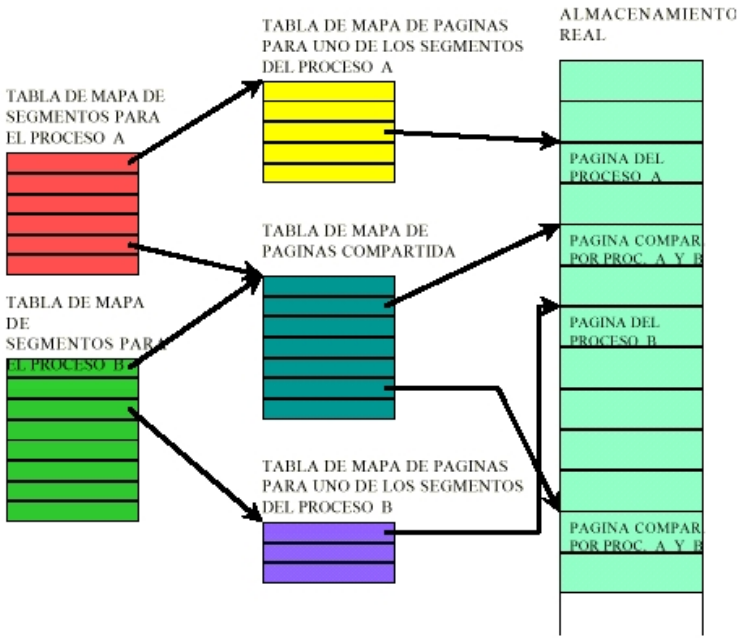
Cuando accede a la memoria asignada por primera vez, su CPU genera un error de página. El núcleo del sistema operativo puede saltar, .ªrreglar.el acceso a la memoria no válida asignando memoria y haciendo E S de archivos en el búfer recién asignado, luego continuar la ejecución del programa como si nada hubiera pasado.

f) Para ambos casos realizar un diagrama de la estructura del proceso en memoria

Si estás utilizando archivos de manera aleatoria, mmap es más sencillo e implica menos llamadas al sistema.

Además, si cuenta la cantidad de veces que se copian los datos a medida que se mueven desde el disco a su programa, el uso de lectura generalmente implica 1 copia adicional.

En el lado de desventaja, el manejo de errores con mmap () se realiza con señales, lo que puede agregar complejidad.



Figura~3.33: Dos procesos compartiendo un sistema de paginación y segmentación.

Figura 3: Diagrama de estructura de proceso en memoria.

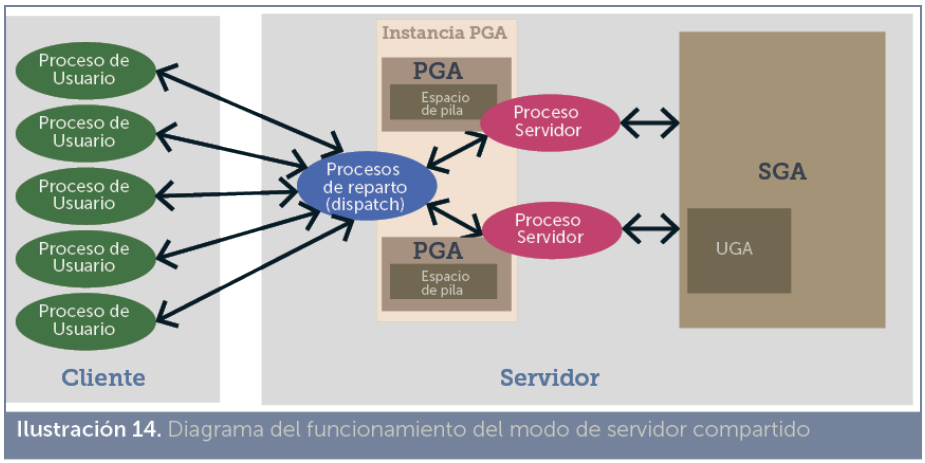


Ilustración 14. Diagrama del funcionamiento del modo de servidor compartido

Figura 4: Diagrama de estructura de proceso en memoria.

2. Child's Play.

   a) Crear un programa que cree un proceso hijo a través de la llamada al sistema fork(), el proceso original deberá imprimir su Process ID, y su Parent Process ID.

```
-bash: cd: Programa2: No such file or directory
[23]   Done                    cd ..
MacBook-Air-de-Gibran:Programa1 gibranaguilar$ cd ..
MacBook-Air-de-Gibran:Practica3 gibranaguilar$ cd Programa2
MacBook-Air-de-Gibran:Programa2 gibranaguilar$ gcc -o PadreHijo PadreHijo.c
PadreHijo.c:12:1: warning: return type of 'main' is not 'int' [-Wmain-return-type]
void  main(void)
^
PadreHijo.c:12:1: note: change return type to 'int'
void  main(void)
^~~~
int
PadreHijo.c:18:12: warning: implicit declaration of function 'fork' is invalid in C99
      [-Wimplicit-function-declaration]
    pid = fork();
          ^
2 warnings generated.
MacBook-Air-de-Gibran:Programa2 gibranaguilar$ ./PadreHijo
 Esta linea es para los hijos, valor = 1
 Esta linea es para los hijos, valor = 2
 Esta linea es para los hijos, valor = 3
 Esta linea es para los hijos, valor = 4
 Esta linea es para los hijos, valor = 5
 Esta linea es para los hijos, valor = 6
 Esta linea es para los hijos, valor = 7
 Esta linea es para los hijos, valor = 8
 Esta linea es para los hijos, valor = 9
 Esta linea es para los hijos, valor = 10
 Esta linea es para los hijos, valor = 11
 Esta linea es para los hijos, valor = 12
 Esta linea es para los hijos, valor = 13
 Esta linea es para los hijos, valor = 14
 Esta linea es para los hijos, valor = 15
 Esta linea es para los hijos, valor = 16
 Esta linea es para los hijos, valor = 17
 Esta linea es para los hijos, valor = 18
 Esta linea es para los hijos, valor = 19
 Esta linea es para los hijos, valor = 20
 Esta linea es para los hijos, valor = 21
 Esta linea es para los hijos, valor = 22
 Esta linea es para los hijos, valor = 23
 Esta linea es para los hijos, valor = 24
 Esta linea es para los hijos, valor = 25
 Esta linea es para los hijos, valor = 26
 Esta linea es para los hijos, valor = 27
 Esta linea es para los hijos, valor = 28
 Esta linea es para los hijos, valor = 29
 Esta linea es para los hijos, valor = 30
 Esta linea es para los hijos, valor = 31
 Esta linea es para los hijos, valor = 32
 Esta linea es para los hijos, valor = 33
 Esta linea es para los hijos, valor = 34
 Esta linea es para los hijos, valor = 35
 Esta linea es para los hijos, valor = 36
 Esta linea es para los hijos, valor = 37
 Esta linea es para los hijos, valor = 38
 Esta linea es para los hijos, valor = 39
 Esta linea es para los hijos, valor = 40
 Esta linea es para los hijos, valor = 41
 Esta linea es para los hijos, valor = 42
 Esta linea es para los hijos, valor = 43
 Esta linea es para los hijos, valor = 44
 Esta linea es para los hijos, valor = 45
```

Figura 5: Código lcreacion padre e hijo

   b) Incluir una condición para la creación del proceso hijo, es decir, ejecutar el fork() unicamente cuando ocurra un evento en el sistema

operativo que lo desencadene.

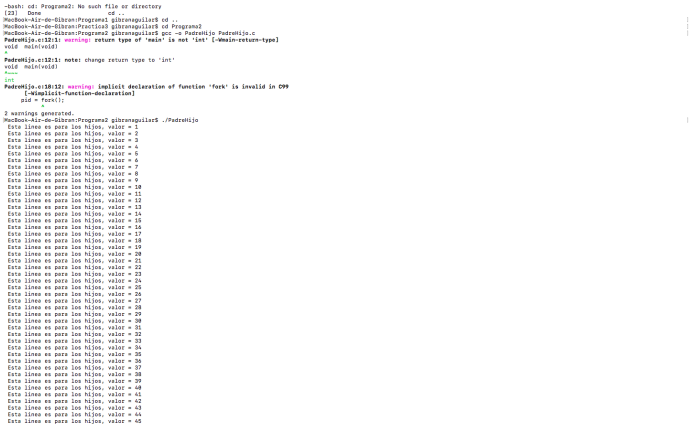


Figura 6: Código la creacion padre e hijo

c) El proceso hijo debe imprimir de igual manera su PID y su PPID, y deberá remarcarse la asociación entre ambos procesos.

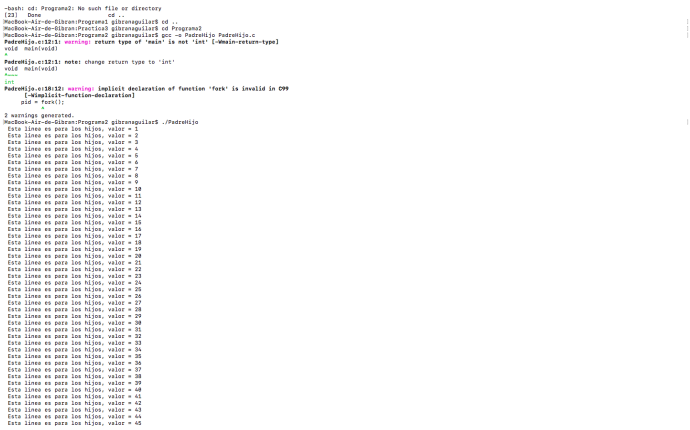


Figura 7: Código la creacion padre e hijo

d) Generar código diferente para cada proceso teniendo en cuenta lo siguiente

```
p = f o r k ( ) ;
i f (p>0)
```

Codigo para e l XXXXX

e l s e

Codigo para e l XXXX

```
MacBook-Air-de-Gibran:Programa2 gibranaguilar$ gcc -o CodigoProceso CodigoProceso.c
clang: error: no such file or directory: 'CodigoProceso.c'
clang: error: no input files
MacBook-Air-de-Gibran:Programa2 gibranaguilar$ gcc -o CodigoProceso CodigoporProceso.c
CodigoporProceso.c:9:1: warning: return type of 'main' is not 'int' [-Wmain-return-type]
void  main(void)
^~~~
CodigoporProceso.c:9:1: note: change return type to 'int'
void  main(void)
~~~~
int
CodigoporProceso.c:13:12: warning: implicit declaration of function 'fork' is invalid in C99
      [-Wimplicit-function-declaration]
    pid = fork();
           ^
2 warnings generated.
MacBook-Air-de-Gibran:Programa2 gibranaguilar$ ./CodigoProceso
This line is from parent, value = 1
This line is from parent, value = 2
This line is from parent, value = 3
This line is from parent, value = 4
This line is from parent, value = 5
This line is from parent, value = 6
This line is from parent, value = 7
This line is from parent, value = 8
This line is from parent, value = 9
This line is from parent, value = 10
This line is from parent, value = 11
This line is from parent, value = 12
This line is from parent, value = 13
This line is from parent, value = 14
This line is from parent, value = 15
This line is from parent, value = 16
This line is from parent, value = 17
This line is from parent, value = 18
This line is from parent, value = 19
This line is from parent, value = 20
This line is from parent, value = 21
This line is from parent, value = 22
This line is from parent, value = 23
This line is from parent, value = 24
This line is from parent, value = 25
This line is from parent, value = 26
This line is from parent, value = 27
This line is from parent, value = 28
This line is from parent, value = 29
This line is from parent, value = 30
This line is from parent, value = 31
This line is from parent, value = 32
This line is from parent, value = 33
This line is from parent, value = 34
This line is from parent, value = 35
This line is from parent, value = 36
This line is from parent, value = 37
This line is from parent, value = 38
This line is from parent, value = 39
This line is from parent, value = 40
This line is from parent, value = 41
This line is from parent, value = 42
This line is from parent, value = 43
This line is from parent, value = 44
This line is from parent, value = 45
This line is from parent, value = 46
```

Figura 8: Código la creacion padre e hijo

3. Tuberias

   a) Escribir un programa el cual genere dos procesos, P1 y P2. P1 recibirá una cadena de caracteres y se la enviará a P2. P2 concatenará la cadena recibida con otra cadena definida en P2 sin emplear funciones de manejo de cadenas contenidas en string.h. La cadena generada será devuelta a P1 para ser impresa en la salida estandar.

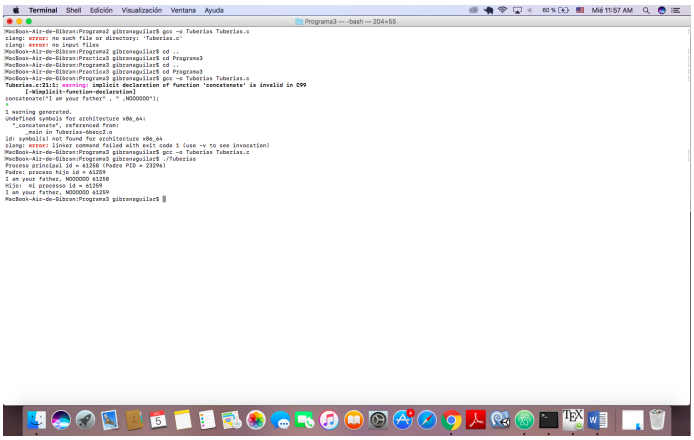


Figura 9: Código comunicación de proceso padre e hijo

b) En este problema deberan utilizarse las llamadas al sistema read(), write(), close(), fork() y pipe() teniendo en cuenta lo siguiente

```
i n t fd [ 2 ] ; //Ar r eglo de d e s c r i p t o r e s de a r chi v o s
pipe ( fd ) ; //Creacion de l pipe , pasando como argumento e l a r
r e g l o de d e s c r i p t o r e s
fd [ 0 ] ; // De s c r ipt o r de a r chivo de l pipe para l e c t u r a
fd [ 1 ] ; // De s c r ipt o r de a r chivo de l pipe para e s c r i t u r a
```

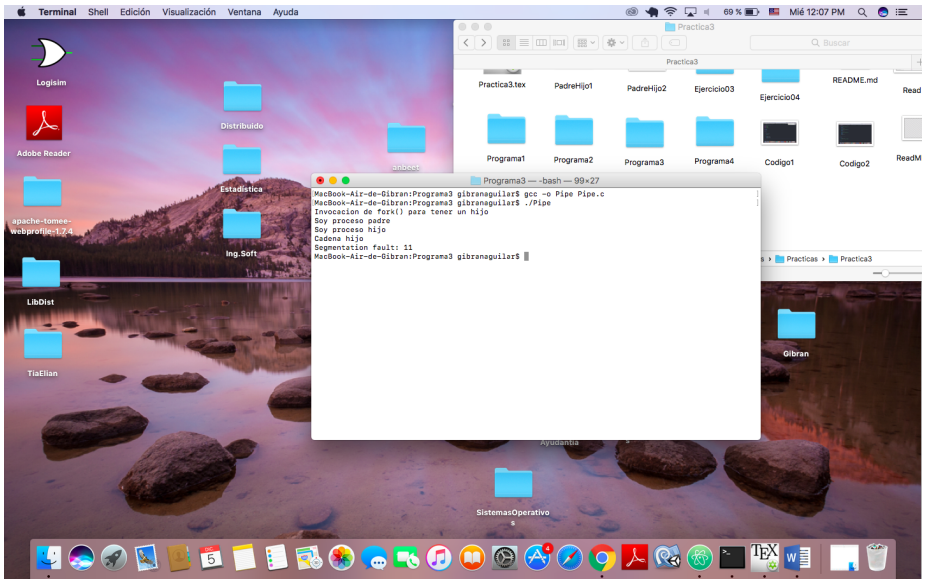


Figura 10: Código comunicación de proceso padre e hijo

c) Considerar que se deben crear dos pipes, y por lo tanto dos arreglos de descriptores de archivos, y que el código debe ser diferente para el proceso padre y el hijo.

4. Signal

a) A través de la implementación de "signal.hçomunicar dos procesos (un proceso padre y un hijo), permitiendo la interacción entre ambos, y ánalizar el programa "matando.ªl proceso hijo.

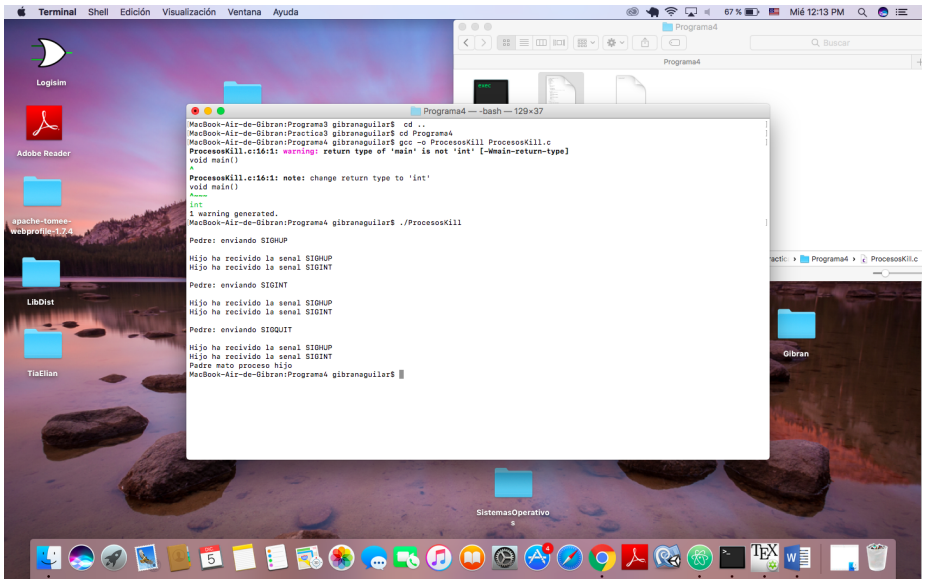


Figura 11: Código comunicación de proceso padre e hijo

b) Cada senal a implementar deberá desencadenar una acción, por ejemplo la impresión de un mensaje.

c) Tener en cuenta que debe implementarse una función que será llamada con la ejecución de cada senal.

d) La llamada al sistema kill() recibe como parametro el PID y la senal a enviar.

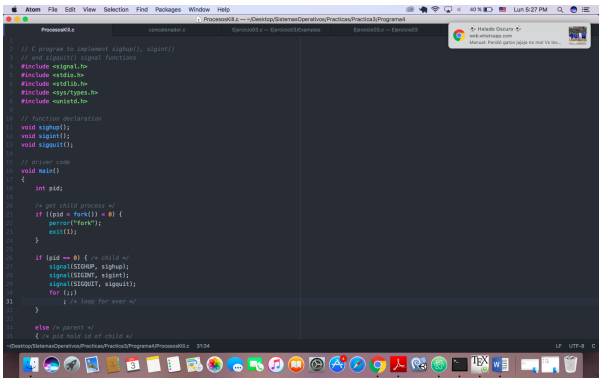


Figura 12: Código Signal

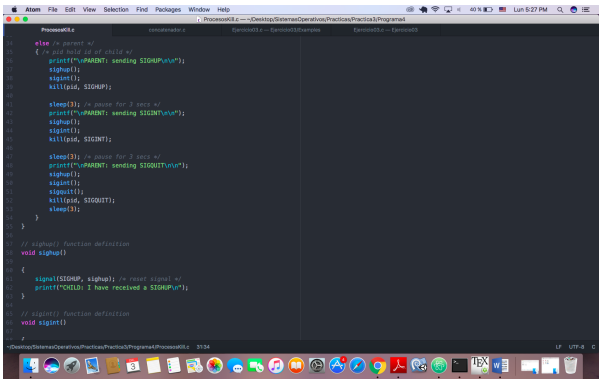


Figura 13: Código Signal

Referencias

[1] https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxbd00/rtrea.htm. *Fundamentos de sistemas operativos.*

[2] https://www.geeksforgeeks.org/signals-c-set-2/. *Fundamentos de sistemas operativos.*