Instituto Politécnico Nacional

Escuela Superior de

Cómputo

Sistemas Operativos

Laboratorio de Sistemas Operativos

Práctica No.7

“Memoria Compartida”

Ayona López Eugenio Milton

Morales Blas David Israel

Grupo: 2CM8

Profesor: Juárez Méndez Ana Belem

PRACTICA NO.7

"MEMORIA COMPARTIDA"

OBJETIVO

Utilizar los conocimientos de las herramientas de memoria compartida en aplicaciones de sistemas operativos.

INTRODUCCIÓN

Memoria compartida. La memoria compartida es uno de los mecanismos de Comunicación Entre Procesos (IPC) que hay en Linux. Disponemos de 4 llamadas al sistema:

- ✓ shmget(): Obtención de la zona de memoria Obtención de la zona de memoria
- ✓ shmctl(): Proporciona operaciones para el control de la memoria compartida la memoria compartida
- ✓ shmat(): Se adjunta la zona de memoria compartida al espacio de direcciones del proceso al espacio de direcciones del proceso.
- ✓ shmdt(): Desvincula una zona de memoria compartida del proceso.

Para usar memoria compartida en Linux es necesario seguir una serie de pasos que luego se traducen a las llamadas al sistema.

1. Necesitamos obtener un identificador de IPC. Para ello convertimos una ruta (ruta) del sistema en un identificador IPC. Este identificador es necesario para crear la zona de memoria virtual. Esto es muy sencillo de hacer con la llamada al sistema ftok..
2. Crear el segmento de memoria compartida con la llamada al sistema shmget.
3. Operar con la memoria compartida. Indica lo que queremos compartir con la llamada al sistema shmat.
4. Destruimos el segmento de memoria compartida con la llamada al sistema shmdt y shmctl.

   key_t ftok (const char * pathname, int proj_id);

   El primer parámetro es una ruta. Lo que se puede hacer es tomar una ruta que siempre está en el sistema, por ejemplo "/bin/cp". El segundo parámetro es un número aleatorio, pero que conoce todos los procesos que deben unirse al mismo segmento de memoria compartida (al igual que la ruta).

   int shmget (clave_t clave, tamaño_t tamaño, int shmflg);

El primer parámetro es la clave obtenida en la llamada al sistema anterior. El segundo parámetro es el tamaño de la memoria a compartir datos en bytes. El tercer

parámetro son flags o banderas para el modo de uso y quién accede al segmento de memoria. Las banderas:
IPC_CREAT: crea un nuevo segmento.
IPC_EXCL: se combina con IPC_CREAT y si el segmento ya existe, entonces la creación del segmento falla.
banderas de modo : son los permisos del segmento sobre usuario, grupo y otros, además de los permisos especiales.

DESARROLLO

I. EJERCICIO 1

Realiza un programa que mediante el uso de memoria compartida visto en clase, comparta con otros programas una pila de 10 elementos, donde cada elemento de la pila sea un carácter. Cada programa mostrará el siguiente menú de opciones al usuario.

1. Ingresar elemento
2. Retirar elemento
3. Visualizar elemento
4. Visualizar pila
5. Finalizar

Las opciones deben realizar lo siguiente.

1. Ingresar elemento: operación push de la pila. Se le pedirá al usuario el carácter a ingresar en la pila y lo ingresará a la pila.
2. Retirar elemento: operación pop de la pila. Leer y quitar el elemento superior de la pila. Antes de eliminarlo de la pila, se debe mostrar en pantalla.
3. Visualizar elemento: operación top de la pila. Leer y mostrar el elemento superior de la pila sin retirarlo.
4. Visualizar pila: recorrer toda la pila y mostrar todos los elementos de la pila.
5. Finalizar: Liberar la memoria y finalizar

el programa. SOLUCIÓN

Durante el desarrollo de la solución a este problema fue necesario primeramente implementar aquel modulo que maneja la estructura que se desea compartir, el uso de la estructura “pila”. Para ello en un archivo por aparte se llevó cabo la implementación de la definición de la estructura pila, así como las operaciones necesarias para esta misma, todas estas contenidas en los archivos TADPilaEst.h y TADPilaEst.c

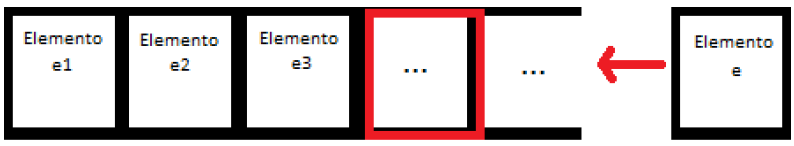
De esta forma se hacen uso de la definición de las funciones u operaciones de la pila a través del archivo de librería .h y su implementación, contenidas en el archivo con extensión .c, estos son utilizados para compilar la solución original para operar. Además, son colocados en la cabecera del archivo Memoria_comp_pila.c para poder hacer uso de su contenido.

De esta forma, se procedió a construir un método nombrado seleccionMenu(), el cual no retorna ningún valor, solo muestra en pantalla, con fines prácticos, el menú de las opciones solicitadas anteriormente solo es un método de apoyo para la impresión de información en pantalla de la cinta de opciones.

Una vez realizado todo lo anterior, ahora es importante establecer en orden la seria de pasos para llevar a cabo la creación de la memoria compartida, es por ello que se crea primeramente la key_t nombrada "llave" e inmediatamente la inicialización de la misma a través del uso del método ftok("/bin/ls",40), asignando el valor entero 40 para operar sobre la pila compartida y creándola sobre el directorio "/bin/ls", para asegurar la existencia del directorio y evitar problemas, así finalmente se procede a hacer la validación de la operación exitosa de esta función.

Cabe destacar que posteriormente a lo anterior se crean algunas variables auxiliares para llevar a cabo la creación de la memoria compartida como shmid, el cual es ocupado para obtener la identificación de la zona de memoria.

Así mismo, como la creación de una variable de tipo apuntador a la estructura pila (pila
*pilac), el cual es nombrada "pilac", así como la creación de los elementos auxiliares para introducir a la pila. Ya que, de acuerdo con el modelo de esta estructura, la estructura pila es un arreglo o conjunto de elementos nombrados "elementos" y cada elemento puede contener distintos tipos de datos, en este caso un carácter (char c), de esta forma es una descripción de la ilustración siguiente.



PILA

Además de otras variables auxiliares de tipo entero para almacenar la opción del usuario a operar sobre la pila y otros de tipo carácter para imprimir en pantalla o recibir carácter desde el teclado para introducir en la pila.

Se procede inmediatamente después a realizar el segundo paso, haciendo uso del método shmget(llave,sizeof(pila),IPC_CREAT|0777), el cual de estos indica la creación de la memoria compartida con identificación o apuntador con el entero 40 definido en la llave anteriormente, con una reservación de memoria del tamaño de nuestra estructura definida

como pila y las banderas de inicialización para llevar a cabo la creación con los permisos definidos a través de los bits significativos en 0777. Posteriormente se realiza la validación de la correcta operación del método.

Después de esto se realiza la obtención de un apuntador que direcciona a la zona de memoria compartida para lograr leer sobre ella o escribir, de esta forma, se hace uso del método para ello escrito como shmat(shmid,0,0), el cual este hace uso del valor obtenido anteriormente llamado shmid, el cual es el identificador a la zona de memoria y los demás atributos son necesarios por el momento colocarlos en cero, así para finalmente realizar una vez más la correcta validación de la correcta operación del método.

Cabe destacar que el método shmat retorna un valor de tipo char*, el cual debe ser transformado al tipo que es necesario operar, en este caso se realiza un casteo de tipo apuntador a char a un apuntador a una estructura nombrada pila, resultando de esta forma.

$$pilac = (struct\ pila *)\ shmat(shmid, 0,0)$$

De esta forma se realiza una asignación al apuntador a la estructura pila nombrada “pilac”,
para después realizar una vez más la validación del correcto funcionamiento del método.

Una vez completado todos los pasos anteriores, la operación principales de la operaciones sobre la pila en memoria compartida fue implementada dentro de un ciclo iterativo “while” infinito (while(1)), para continuar realizando diferentes operaciones en la ejecución del programa, así dentro de este ciclo se procede a imprimir en pantalla la cinta de opciones haciendo uso del método auxiliar seleccionMenu(), posteriormente haciendo uso de las librerías estándar, ser recibe por teclado la opción del usuario a utilizar.

Tomando este valor como condición de operación para una estructura “switch” y de esta forma cada una de la opciones esta conformada de cada uno de los pintos que se enlistan a continuación.

1. Case 1: Ingresar un elemento a la pila.
2. Case 2: Retirar el elemento superior de la pila.
3. Case 3: Visualizar el elemento superior de la pila.
4. Case 4: Visualizar todos los elementos de la pila.
5. Case 5: Salir o finalizar del programa.

1.Ingresar un elemento a la pila

Para el desarrollo de esta sección, se procede a solicitarle al usuario a través de los recursos de la librería estándar el carácter a ingresar, el

cual se almacena en un valor carácter auxiliar nombrado “c” (char c), el cual se almacena en uno de los elementos de la estructura nombrada “elemento” con una instancia e1, para después utilizar la operación para introducir un elemento a la pila, es decir, la operación Push(pilac,e1).

La cual recibe los siguientes parámetros; la pila en la cual se va a operar como un apuntador a ella y el elemento que se quiere ingresar, en este caso el que se almaceno con anterioridad en e1.

2.Retirar el elemento superior de la pila

Para el desarrollo de esta opción, se hace uso de la operación eaux=Top(pilac), el cual esta asignada a un elemento "eaux" auxiliar de tipo elemento, del cual se trata de una estructura, por la cual para hacer acceso al carácter que contiene se realizar la operación vaux=eaux.c; todo con motivo de imprimirlo e indicarlo al usuario cual es el elemento superior que se retira con el uso de la operación Pop(pilac), el cual solo recibe el apuntador a la pila sobre la cual se retirara el elemento.

3.Visualizar el elemento superior de la pila

Como se mencionó anteriormente se hace uso de la operación e2=Top(pilac), el cual esta asignada a un elemento "e2" auxiliar de tipo elemento, del cual se trata de una estructura, por la cual para hacer acceso al carácter que contiene se realizar la operación valortop=e2.c; para finalmente mostrarlo en pantalla.

4.Visualizar todos los elementos de la pila

Para visualizar todos los elementos contenidos en la pila se hace uso de la función Shows(pilac), el cual recibe como parámetro el apuntador a la pila sobre la cual esta va a operar, cabe destacar que el funcionamiento de este método se basa en un recorrido simple de un arreglo lineal, es por ello que, en la implementación de la solución, solo se llama a la función ya implementada, además ya contiene formatos de impresión.

5.Salir o finalizar del programa

Finalmente, al terminar el programa o Salir de el a través de esta opción, es necesario implementar primeramente la liberación de la memoria compartida con el uso del método shmdt((char *)pilac) el cual libera el apuntador a la memoria reservada, sin embargo, es necesario realizar una conversión al tipo de dato char*, así como del uso de la función shmctl(shmid,IPC_RMID,0), la cual recibe como parámetros al identificador de la zona de memoria compartida creada así como las banderas correspondientes para su eliminación.

Finalmente, con el caso default, se realiza la validación de opciones inexistentes dentro de la estructura switch. Así, a continuación, se muestra la implementación desarrollada y explicada anteriormente escrita en lenguaje c.

CODIGO LENGUAJE C

*memoria.c

File Edit Search Options Help

```
#include <stdio.h>
#include <stdlib.h>
#include"TADPilaEst.h"
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHM_SIZE 1024  /* make it a 1K shared memory segment */

void seleccionMenu();
int main(int argc, char *argv[])
{

   char* data;
   char* cadena;//error: sucede debido a que como la variable, no era parte del segmento, no tenia acceso

  key_t key;
  int shmid;
  pila* mi_pila;
  elemento mi_elemento,mi_elemento2,mi_elementoaux;
  int opcion;
char c,vaux,top;

```

Menu [Inb... *m... Pra... Lin... Lin... Lin... 7:57:18

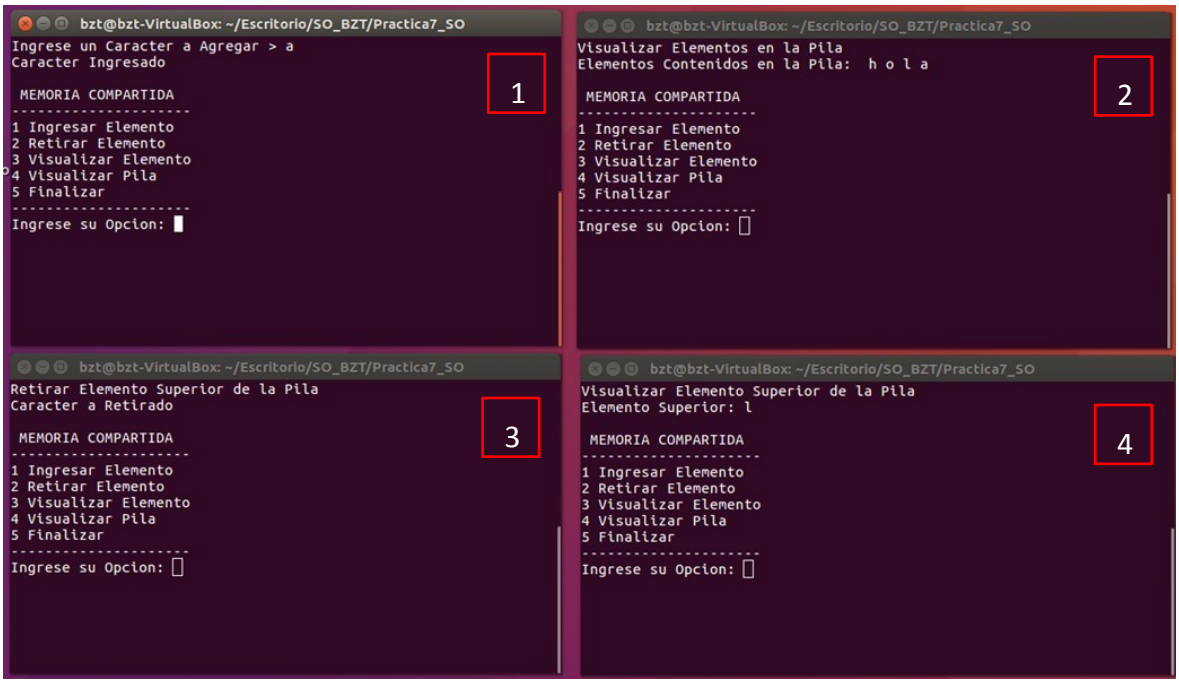
```
   /* Crea la llave para compartir la memoria*/
   if ((key = ftok("hello1.txt", 40)) == -1) /*Here the file must exist */
{
      perror("ftok");
      exit(1);
   }

   /* Creacion de la memoria compartida */
   if ((shmid = shmget(key, sizeof(pila),  IPC_CREAT|0644)) == -1) {
      perror("shmget");
      exit(1);
   }

//se crea acceso a las variables dentro del segmento

   mi_pila=(struct pila*) shmat(shmid, NULL, 0);
   Initialize(mi_pila);

   if (mi_pila==NULL ) {
      perror("shmat");
      exit(1);
   }
while(1){
seleccionMenu();
scanf("%d",&opcion);
switch(opcion)
{
    case 1:
         system("clear");
         printf("1. Ingresa elemento a la pila\n");
         fflush(stdin);
         scanf("%c",&c);
        //strncpy(data,  cadena, SHM_SIZE);
         mi_elemento.c=c;
         Push(mi_pila,mi_elemento);

   break;

    case 2:
              system("clear");
              mi_elementoaux=Top(mi_pila);
              vaux=mi_elementoaux.c;
              Pop(mi_pila);
              printf("Se retiro el elemento %c" ,vaux);
```

```
 case 2:
            system("clear");
            mi_elementoaux=Top(mi_pila);
            vaux=mi_elementoaux.c;
            Pop(mi_pila);
            printf("Se retiro el elemento %c" ,vaux);
 break;

 case 3:
             system("clear");
             mi_elemento2=Top(mi_pila);
             top= mi_elemento2.c;
             printf("3. Visualiza elemento superior de la pila %c\n",top);
 break;

 case 4:
                      system("clear");
                      printf("Visualizar Elementos en la Pila\n");
                      Shows(mi_pila);
                      break;
 break;

 case 5:
          Destroy(mi_pila);
         shmdt((char *)mi_pila);
          shmctl(shmid,IPC_RMID,(struct shmid_ds *) NULL);

printf("Finalizo\n");
 break;

default:
system("clear");
 printf("Ingrese la opcion correcta\n");


}

    }
 /* detach from the segment: */
 if (shmdt(data) == -1) {
    perror("shmdt");
    exit(1);
 }

 return 0;
}
```

```
void seleccionMenu(){
	printf("\n MEMORIA COMPARTIDA\n");
	for(int i=0;i<21;i++)
		printf(":v");
	printf("\n1 Ingresar Elemento\n2 Retirar Elemento\n3 Visualizar Elemento");
	printf("\n4 Visualizar Pila\n5 Finalizar\n");
	for(int i=0;i<21;i++)
		printf(":v");
	printf("\nIngrese su Opcion: ");
}
```

PRUEBAS Y RESULTADOS

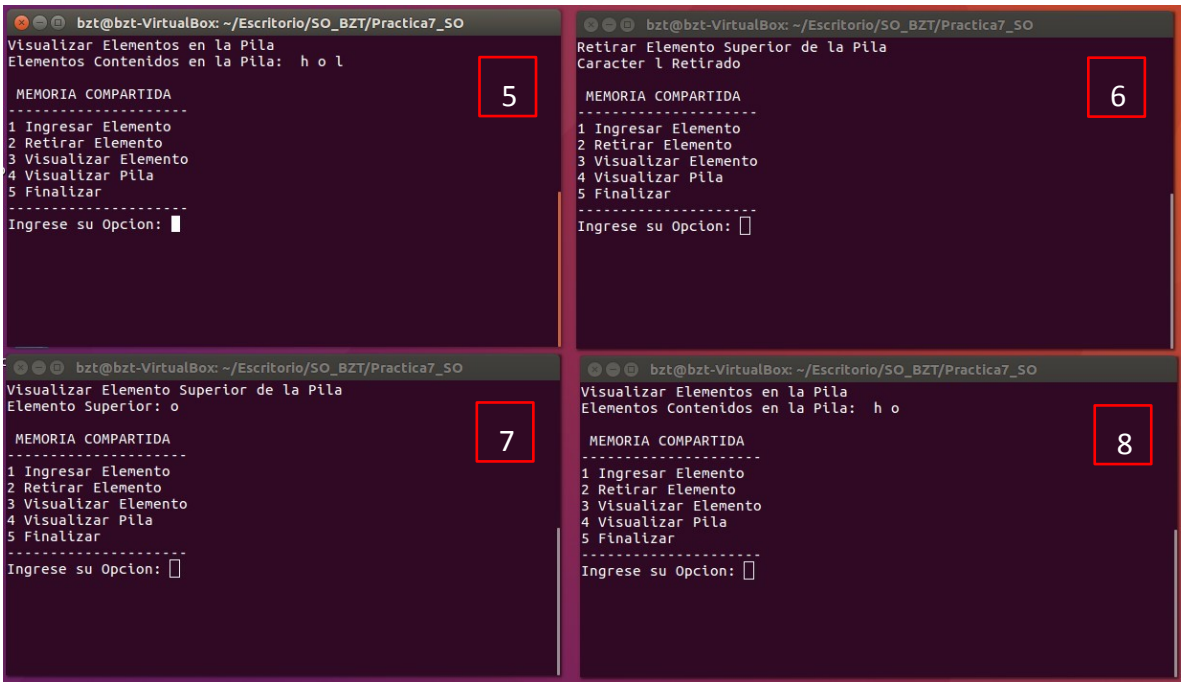
A continuación, se realiza la prueba del desarrollo ejecutando en 4 terminales diferentes, el uso de todas las operaciones implementadas, de esta forma a continuación, se enumera la secuencia de operación para la prueba en las terminales que se muestra en la Ilustración



Como se puede observar en la ilustración, se muestra una serie de secuencias de prueba, las cuales describen el siguiente funcionamiento.

1. Se procede a agregar elemento a elemento con la primer opción el conjunto de caracteres “h o l a”. Esto se puede observar, ya que el último elemento añadido es el carácter “a”.
2. Se procede a visualizar todos los elementos de la pila con la cuarta opción, el cual se puede observar en pantalla la cadena anterior que se ingresó carácter a carácter.
3. Se procede a retirar el elemento superior de la pila con la segunda opción, el cual al ser el último carácter “a”, es el que se muestra en pantalla que es el elemento tope eliminado como se observa en el punto 3.
4. Una vez más se consulta el elemento superior con la tercera opción, al eliminar el carácter “a” de la cadena “h o l a”, el elemento superior pasa a resultar el carácter “l”, de esta forma se muestra de forma correcta en pantalla.

A continuación, para complementar estas operaciones se muestra el siguiente conjunto de cuatro capturas de terminales, complementando la secuencia desarrollada anteriormente, con el uso de las cadenas ya introducidas anteriormente en la pila, de esta forma se desarrolla el procedimiento siguiente.

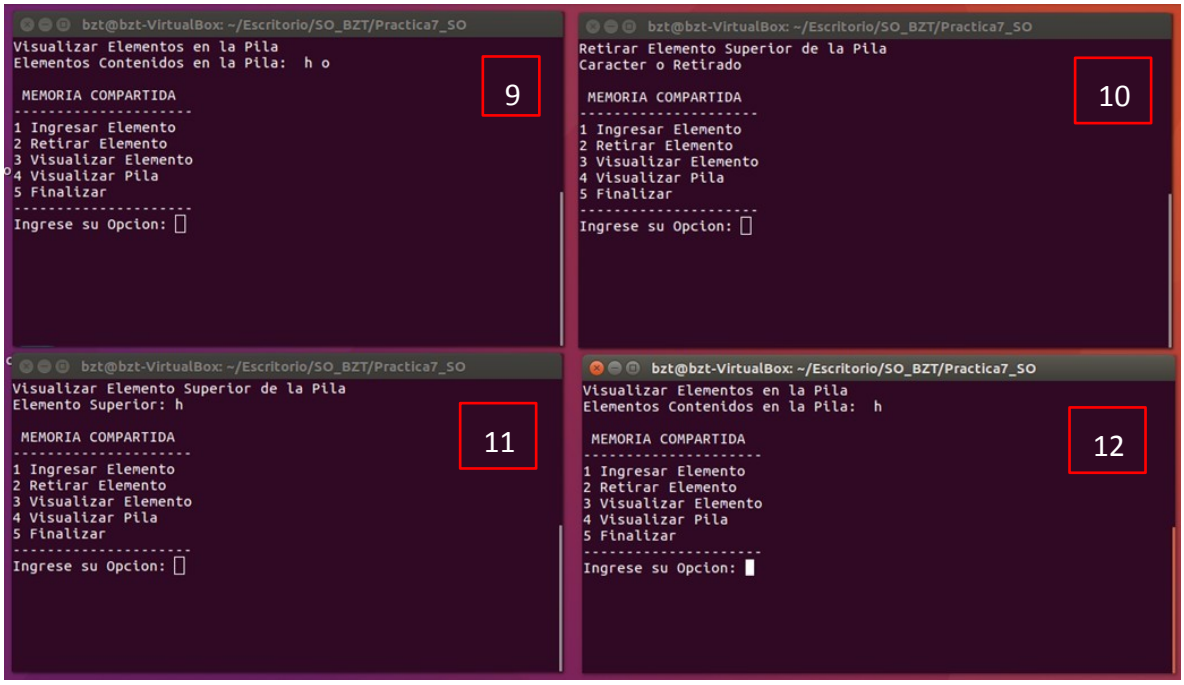


Como se puede observar en la ilustración, se muestra una serie de secuencias de prueba para complementar el funcionamiento de la implementación.

5. Continuando con la secuencia, al observar que el elemento superior es el carácter “l”, en esta operación se observa el contenido de la pila la cual debe ser “h o l”.
6. Una vez mas haciendo uso de la segunda opción, se retira el elemento superior, e
   cual es “l”, resultando así la cadena “h o”.
7. Se hace la consulta del elemento superior, con base a lo anterior, se muestra que
   este es elemento “o”, como se observa en la Ilustración.
8. Una vez mas reafirmando, se realiza la consulta de los elementos de la pila, al ser el
   elemento superior el carácter “o”, el contenido deberá ser “h o”.

Consiguientemente, para complementar y finalizar estas operaciones se muestra el siguiente conjunto de cuatro capturas de terminales.

Complementando las secuencias desarrolladas anteriormente, con el uso de las cadenas ya introducidas anteriormente en la pila, de esta forma se desarrolla el procedimiento siguiente.



Como se puede observar en la última ilustración, se muestra una serie de secuencias de prueba para complementar el funcionamiento de la implementación.

9. Se consulta una vez mas el contenido de la pila, el cual es el conjunto de caracteres que se muestra en la Ilustración “h o”.
10. Al ser la cadena “h o” el contenido de la pila, el elemento superior deberá ser “o”, así mismo, se procede a usar la segunda opción para eliminar dicho elemento como se puede observar en la Ilustración.
11. Al eliminar el elemento superior “o”, de la cadena de la pila “h o”, el contenido restante de la pila deberá ser el conjunto “h”, como se observa con el uso de la tercera opción para visualizar el elemento superior de la pila, así mismo como ser elemento superior de la pila.
12. Finalmente, como anteriormente se mostró, el elemento superior deberá ser “h”, así
finalmente se muestra el correcto funcionamiento.

Al finalizar todo el desarrollo de esta secuencia, se logro observar que la implementación descrita y desarrollada en el apartado de desarrollo, es correcta para todos los puntos solicitados en el inicio. Finalmente se puede deducir que la solución es correcta, cabe destacar que la implementación de la pila es orientada a un modelo estructurado, sin embargo, es funcional para este propósito descrito en este documento.

CONCLUSIONES

Gracias al desarrollo de esta experimentación realizada, se logró observar que la memoria compartida o el recurso de memoria compartida es una parte fundamental de los métodos de comunicación entre procesos en cualquier Sistema Operativo, especialmente en UNIX principalmente se componen de 3 tipos: a través del uso de tuberías, semáforos y el uso de memoria compartida, el cual se logro observar las características y los recursos necesarios para lograr utilizarlas entre distintos procesos.

Cabe destacar que, la implementación realizada en este desarrollo de la estructura pila, no es la única, ya que como se mencionó se puede implementar de diferentes formas, sin embargo, con fines prácticos se tomó dicha implementación para hacer uso del correcto modelado teórico de la estructura pila. Sin embargo, durante las pruebas se pudo observar que en cualquier instante un proceso puede acceder a dicha memoria y leer en ese momento su contenido.

A diferencia de métodos como la sincronización de procesos a través del uso de semáforos, además este tipo de sincronización no detiene o pasa a estados dormidos los procesos que hacen uso de ella a diferencia de los semáforos.

REFERENCIAS

Deitel, H.M.; “Sistemas Operativos”, Ed. Addison Wesley, 2a ed., México

Juárez, A. (2020). Sistemas Operativos: Comunicación de Procesos [Notas de Clase].

Snatverk. (2010). Memoria compartida en Linux. Recuperado de https://snatverk.blogspot.com/2010/09/memoria-compartida-en-linux.html

Alemán, O. (s. f.). Memoria Compartida. Recuperado de http://sopa.dis.ulpgc.es/ii-dso/leclinux/ipc/mem_compartida/memcomp.pdf

ANEXOS

Implementación de la librería de la estructura pila, escrita en lenguaje c.

```
//DEFINICIONES DE CONSTANTES
#define MAX_ELEMENT 10
#define TRUE 1
#define FALSE     0

//DEFINICIONES DE TIPOS DE DATO
//Definir un boolean (Se modela con un "char")
typedef unsigned char boolean;

//Definir un elemento (Se modela con una estructura "elemento")
typedef struct elemento
{
      //Variables de la estructura "elemento" (El usuario puede modificar)
      char c;
      //***
      //***
}elemento;

//Definir una pila (Se modela con una estructura que unicamente incluye un
puntero a "elemento")
typedef struct pila
{
      elemento arreglo[MAX_ELEMENT];       //La pila es un arreglo
estático de MAX_ELEMET
      int tope;                                          //El tope es un
entero (Indice del arreglo de elementos)
}pila;

//DECLARACIÓN DE FUNCIONES
void Initialize(pila *s);                  //Inicializar pila (Iniciar una
pila para su uso)
void Push(pila *s, elemento e);            //Empilar (Introducir un elemento
a la pila)
elemento Pop (pila *s);                    //Desempilar (Extraer un
elemento de la pila)
boolean Empty(pila *s);                    //Vacia (Preguntar si la pila
esta vacia)
elemento Top(pila *s);                     //Tope (Obtener el "elemento" del
tope de la pila si extraerlo de la pila)
int Size(pila *s);                         //Tamaño de la pila (Obtener
el número de elementos en la pila)
void Shows(pila *s);                       //Muestra en pantalla todos los
elementos en la pila
void Destroy(pila *s);                     //Elimina pila (Borra a todos
los elementos y a la pila de memoria)
```