Example program

``**shmop** system call example" is a menu-driven program. It allows all possible combinations of using the **shmat** and **shmdt** system calls to be exercised.

From studying this program, you can observe the method of passing arguments and receiving return values. The user-written program requirements are pointed out.

This program begins (lines 5-9) by including the required header files as specified by the **shmop**(S) manual page. Note that in this program **errno** is declared as an external variable; therefore, the **sys/errno.h** header file does not have to be included.

Variable and structure names have been chosen to be as close as possible to those in the synopsis. Their declarations are self explanatory. These names make the program more readable and are perfectly valid since they are local to the program.

The variables declared for this program and what they are used for are as follows:

addr
: used to store the address of the shared memory segment for the **shmat** and **shmdt** system calls and to receive the return value from the **shmat** system call

laddr
: used to store the desired attach/detach address entered by the user

flags
: used to store the codes of the **SHM_RND** or **SHM_RDONLY** flags for the **shmat** system call

i
: used as a loop counter for attaching and detaching

attach
: used to store the desired number of attach operations

shmid
: used to store and pass the desired shared memory segment identifier

shmflg
: used to pass the value of flags to the **shmat** system call

retrn
: used to store the return values from the **shmdt** system call

detach
: used to store the desired number of detach operations

This example program combines both the **shmat** and **shmdt** system calls. The program prompts for the number of attachments and enters a loop until they are done for the specified shared memory identifiers. Then, the program prompts for the number of detachments to be performed and enters a loop until they are done for the specified shared memory segment addresses.

shmat

The program prompts for the number of attachments to be performed, and the value is stored at the address of the attach variable (lines 19-23).

A loop is entered using the attach variable and the **i** counter (lines 23-72) to perform the specified number of attachments.

In this loop, the program prompts for a shared memory segment identifier (lines 26-29); it is stored in the **shmid** variable (line 30). Next, the program prompts for the address where the segment is to be attached (lines 32-36); it is stored in the **laddr** variable (line 37) and converted to a pointer (line 39). Then, the program prompts for the desired flags to be used for the attachment (lines 40-47), and the code representing the flags is stored in the flags variable (line 48). The flags variable is tested to determine the code to be stored for the **shmflg** variable used to pass them to the **shmat** system call (lines 49-60). The system call is executed (line 63). If successful, a message stating so is displayed along with the attach address (lines 68-70). If unsuccessful, a message stating so is displayed and the error code is displayed (line 65). The loop then continues until it finishes.

shmdt

After the attach loop completes, the program prompts for the number of detach operations to be performed (lines 73-77) and the value is stored in the detach variable (line 76).

A loop is entered using the detach variable and the **i** counter (lines 80-98) to perform the specified number of detachments.

In this loop, the program prompts for the address of the shared memory segment to be detached (lines 81-85); it is stored in the **laddr** variable (line 86) and converted to a pointer (line 88). Then, the **shmdt** system call is performed (line 89). If successful, a message stating so is displayed along with the address that the segment was detached from (lines 95, 96). If unsuccessful, the error number is displayed (line 92). The loop continues until it finishes.

The example program for the **shmop** system calls follows. We suggest that you name the source program file *shmop.c* and the executable file **shmop**.

```
 1    /*This is a program to illustrate
 2     *the shared memory operations, shmop(),
 3     *system call capabilities.
 4     */


 5    /*Include necessary header files.*/
 6    #include    <stdio.h>
 7    #include    <sys/types.h>
 8    #include    <sys/ipc.h>
 9    #include    <sys/shm.h>
10    /*Start of main C language program*/
11    main()
12    {
13        extern int errno;
14        void *addr;
15        long laddr;
16        int flags, i, attach;
17        int shmid, shmflg, retrn, detach;


18        /*Loop for attachments by this process.*/
19        printf("Enter the number of\n");
20        printf("attachments for this\n");
21        printf("process (1-4).\n");
22        printf("       Attachments = ");


23        scanf("%d", &attach);
24        printf("Number of attaches = %d\n", attach);
```

```
25        for(i = 1; i <= attach; i++) {
26            /*Enter the shared memory ID.*/
27            printf("\nEnter the shmid of\n");
28            printf("the shared memory segment to\n");
29            printf("be operated on = ");
30            scanf("%d", &shmid);
31            printf("\nshmid = %d\n", shmid);


32            /*Enter the value for shmaddr.*/
33            printf("\nEnter the value for\n");
34            printf("the shared memory address\n");
35            printf("in hexadecimal:\n");
36            printf("            Shmaddr = ");
37            scanf("%lx", &laddr);
38            addr = (void*) laddr;
39            printf("The desired address = 0x%lx\n", (long)addr);


40            /*Specify the desired flags.*/
41            printf("\nEnter the corresponding\n");
42            printf("number for the desired\n");
43            printf("flags:\n");
44            printf("SHM_RND               = 1\n");
45            printf("SHM_RDONLY            = 2\n");
46            printf("SHM_RND and SHM_RDONLY = 3\n");
47            printf("            Flags     = ");
48            scanf("%d", &flags);


49            switch(flags)
50            {
51            case 1:
52                shmflg = SHM_RND;
53                break;
54            case 2:
55                shmflg = SHM_RDONLY;
56                break;
57            case 3:
58                shmflg = SHM_RND | SHM_RDONLY;
59                break;
60            }
61            printf("\nFlags = 0%o\n", shmflg);


62            /*Do the shmat system call.*/
63            addr = shmat(shmid, addr, shmflg);
64            if(addr == (char*) -1) {
65                printf("\nShmat failed, error = %d\n", errno);
66            }
67            else {
68                printf ("\nShmat was successful\n");
69                printf("for shmid = %d\n", shmid);
70                printf("The address = 0x%lx\n", (long)addr);
71            }
72        }


73        /*Loop for detachments by this process.*/
74        printf("Enter the number of\n");
75        printf("detachments for this\n");
76        printf("process (1-4).\n");
77        printf("       Detachments = ");


78        scanf("%d", &detach);
79        printf("Number of attaches = %d\n", detach);
80        for(i = 1; i <= detach; i++) {
```

```
81            /*Enter the value for shmaddr.*/
82            printf("\nEnter the value for\n");
83            printf("the shared memory address\n");
84            printf("in hexadecimal:\n");
85            printf("            Shmaddr = ");
86            scanf("%lx", &laddr);
87            addr = (void*) laddr;
88            printf("The desired address = 0x%lx\n", (long)addr);


89            /*Do the shmdt system call.*/
90            retrn = shmdt(addr);
91            if(retrn == -1)  {
92                printf("Error = %d\n", errno);
93            }
94            else {
95                printf ("\nShmdt was successful\n");
96                printf("for address  = 0x%lx\n", (long)addr);


97            }
98        }
99    }
```

shmop system call example

Next topic: IPC programming example for liber
Previous topic: Detaching shared memory segments

© 2005 The SCO Group, Inc. All rights reserved.
SCO OpenServer Release 6.0.0 -- 02 June 2005