Example program

``**shmget** system call example" is a menu-driven program. It allows all possible combinations of using the **shmget** system call 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 4-7) by including the required header files as specified by the **shmget**(S) manual page. Note that the **sys/errno.h** header file is included as opposed to declaring **errno** as an external variable; either method will work.

Variable names have been chosen to be as close as possible to those in the synopsis for the system call. 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:

key
: used to pass the value for the desired **key**

opperm
: used to store the desired operation permissions

flags
: used to store the desired control commands (flags)

shmid
: used for returning the message queue identification number for a successful system call or the error code (**-1**) for an unsuccessful one

size
: used to specify the shared memory segment size

opperm_flags
: used to store the combination from the logical ORing of the **opperm** and **flags** variables; it is then used in the system call to pass the **shmflg** argument

The program begins by prompting for a hexadecimal **key**, an octal operation permissions code, and finally for the control command combinations (flags) which are selected from a menu (lines 14-31). All possible combinations are allowed even though they might not be viable. This allows observing the errors for invalid combinations.

Next, the menu selection for the flags is combined with the operation permissions; the result is stored in the **opperm_flags** variable (lines 35-50).

A display then prompts for the size of the shared memory segment; it is stored in the **size** variable (lines 51-54).

The system call is made next; the result is stored in the **shmid** variable (line 56).

Since the **shmid** variable now contains a valid message queue identifier or the error code (**-1**), it is tested to see if an error occurred (line 58). If **shmid** equals **-1**, a message indicates that an error resulted and the

external **errno** variable is displayed (line 60).

If no error occurred, the returned shared memory segment identifier is displayed (line 64).

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

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


 4    #include    <sys/types.h>
 5    #include    <sys/ipc.h>
 6    #include    <sys/shm.h>
 7    #include    <errno.h>


 8    /*Start of main C language program*/
 9    main()
10    {
11        key_t key;             /*declare as long integer*/
12        int opperm, flags;
13        int shmid, size, opperm_flags;
14        /*Enter the desired key*/
15        printf("Enter the desired key in hex = ");
16        scanf("%x", &key);


17        /*Enter the desired octal operation
18          permissions.*/
19        printf("\nEnter the operation\n");
20        printf("permissions in octal = ");
21        scanf("%o", &opperm);


22        /*Set the desired flags.*/
23        printf("\nEnter corresponding number to\n");
24        printf("set the desired flags:\n");
25        printf("No flags                  = 0\n");
26        printf("IPC_CREAT                 = 1\n");
27        printf("IPC_EXCL                  = 2\n");
28        printf("IPC_CREAT and IPC_EXCL    = 3\n");
29        printf("            Flags         = ");
30        /*Get the flag(s) to be set.*/
31        scanf("%d", &flags);


32        /*Check the values.*/
33        printf ("\nkey =0x%x, opperm = 0%o, flags = %d\n",
34            key, opperm, flags);


35        /*Incorporate the control fields (flags) with
36          the operation permissions*/
37        switch (flags)
38        {
39        case 0:    /*No flags are to be set.*/
40            opperm_flags = (opperm | 0);
41            break;
42        case 1:    /*Set the IPC_CREAT flag.*/
43            opperm_flags = (opperm | IPC_CREAT);
44            break;
45        case 2:    /*Set the IPC_EXCL flag.*/
46            opperm_flags = (opperm | IPC_EXCL);
47            break;
48        case 3:    /*Set the IPC_CREAT and IPC_EXCL flags.*/
49            opperm_flags = (opperm | IPC_CREAT | IPC_EXCL);
50        }
```

```
51        /*Get the size of the segment in bytes.*/
52        printf ("\nEnter the segment");
53        printf ("\nsize in bytes = ");
54        scanf ("%d", &size);


55        /*Call the shmget system call.*/
56        shmid = shmget (key, size, opperm_flags);


57        /*Perform the following if the call is unsuccessful.*/
58        if(shmid == -1)
59        {
60            printf ("\nThe shmget call failed, error number = %d\n", errno);
61        }
62        /*Return the shmid upon successful completion.*/
63        else
64            printf ("\nThe shmid = %d\n", shmid);
65        exit(0);
66    }
```

shmget system call example

Next topic: Controlling shared memory
Previous topic: Operation permissions codes

SCO OpenServer Release 6.0.0 -- 02 June 2005