Asking for a Shared Memory Segment - shmget()

The system call that requests a shared memory segment is **shmget()**. It is defined as follows:

In the above definition, \mathbf{k} is of type $\mathbf{key_t}$ or $\mathbf{IPC_PRIVATE}$. It is the numeric key to be assigned to the returned shared memory segment. \mathbf{size} is the size of the requested shared memory. The purpose of \mathbf{flag} is to specify the way that the shared memory will be used. For our purpose, only the following two values are important:

- 1. **IPC_CREAT | 0666** for a server (*i.e.*, creating and granting read and write access to the server)
- 2. **0666** for any client (*i.e.*, granting read and write access to the client)

Note that due to Unix's tradition, **IPC_CREAT** is *correct* and **IPC_CREATE** is *not*!!!

If **shmget()** can successfully get the requested shared memory, its function value is a non-negative integer, the shared memory ID; otherwise, the function value is negative. The following is a server example of requesting a private shared memory of four integers:

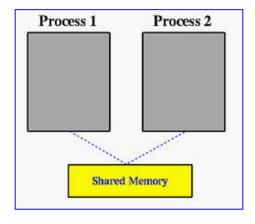
If a client wants to use a shared memory created with <code>IPC_PRIVATE</code>, it must be a child process of the server, created <code>after</code> the parent has obtained the shared memory, so that the private key value can be passed to the child when it is created. For a client, changing <code>IPC_CREAT | 0666</code> to <code>0666</code> works fine. A warning to novice <code>C programmers:</code> don't change <code>0666</code> to <code>666</code>. The leading <code>0</code> of an integer indicates that the integer is an octal number. Thus, <code>0666</code> is <code>110110110</code> in binary. If the leading zero is removed, the integer becomes six hundred sixty six with a binary representation <code>1111011010</code>.

Server and clients can have a parent/client relationship or run as separate and unrelated processes. In the former case, if a shared memory is requested and

1 of 2 11/17/18, 1:10 AM

attached prior to forking the child client process, then the server may want to use IPC_PRIVATE since the child receives an identical copy of the server's address space which includes the attached shared memory. However, if the server and clients are separate processes, using IPC_PRIVATE is unwise since the clients will not be able to request the same shared memory segment with a unique and unknown key.

Suppose process 1, a server, uses **shmget()** to request a shared memory segment successfully. That shared memory segment exists somewhere in the memory, but is not yet part of the address space of process 1 (shown with dashed line below). Similarly, if process 2 requests the same shared memory segment with the same key value, process 2 will be granted the right to use the shared memory segment; but it is not yet part of the address space of process 2. To make a requested shared memory segment part of the address space of a process, use **shmat()**.



2 of 2 11/17/18, 1:10 AM