Apple Business Systems
Technical Notes

Product Technical Support

AWS23: SHMAT limitations

Written by: Sandhya Vora April, 1993
Modified by: Ben Beasley May, 1994

This technical note discusses a memory limitation that may be encountered by UNIX® COFF (Common Object File Format) applications using shared memory on A/UX releases 3.1 and earlier versions and a programming technique that can be used to work-around this limitation.

The memory limitation occurs if an application attempts to expand the size of its data segment (or heap space) by at least 256K bytes (cumulative) after it has allocated one or more shared memory segments which were attached at an address chosen by the system (i.e., in the invocation of the *shmat*(2) system call, the value of the `shmaddr` argument is zero).

In Figure 1, the address space of the application is depicted as it might appear immediately following the invocation of *shmat*(2). The arrow indicates the direction of future growth of the application's data segment.

This memory limitation is perceived by the application whether it attempts to expand its data segment directly using *sbrk(*2) or indirectly using *malloc*(3) or some interface supported by its run-time environment, e.g., an interpreted language. *Malloc*(3) and *sbrk*(2) will fail and set errno to ENOMEM (decimal 12).

® UNIX is a registered trademark of UNIX System Laboratories, a subsidiary of Novell.

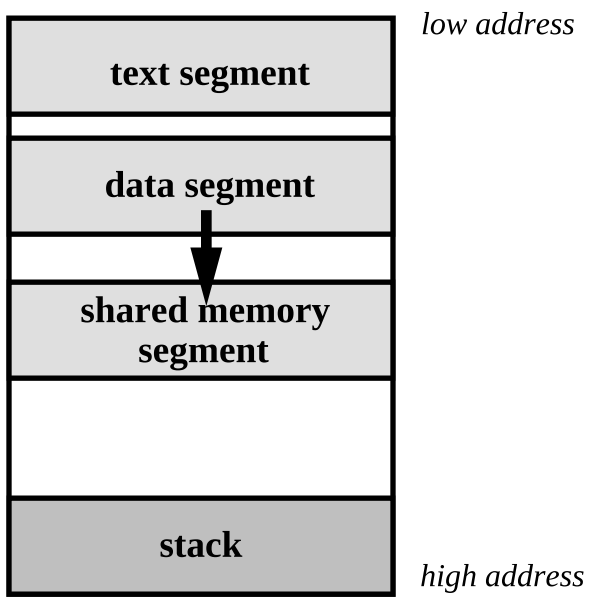

Figure 1. Application Address Space after Invocation of *shmat*(2)

The ENOMEM error indicates that the available address space is not large enough to fulfill the requested growth of the data segment. In Figure 2, the address space of the application is depicted as it might appear when the ENOMEM error occurs. In the scenario described above, the error occurs because the expanded data segment would overlap the shared memory segment(s), which would be a violation of the virtual memory protection scheme. Even though there is still additional “free space” in the applications address space, the system cannot fragment the data segment, since some applications expect it to be contiguous.

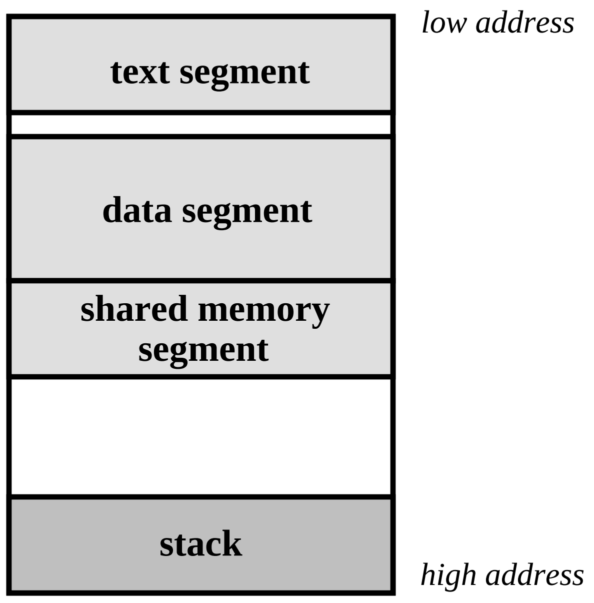

Figure 2. Application Address Space When ENOMEM Occurs

The work-around for this problem is for the application to request a specific address as the second argument to *shmat*(2). Given a prudent choice for the value of this address, the application can arrange for the shared memory segment to be placed high enough in memory to avoid conflict with the growing heap. The value of this address may be determined by obtaining the current size of the data segment and then adding the maximum future data requirements of the process.

In the following code fragment, the programmer has determined that the maximum future data requirements for this application is `0x100000` (5 MB); when *sbrk(2)* is called with an argument of zero, it returns the current end of the data segment. The shared memory segment will be attached at an address which is at least 5 MB beyond the end of the data segment.

```
if (shmat(shmid, sbrk(0) + 0x100000, SHM_RND) == -1)
{
   perror("shmat");
}
```