```
/*
 * hugepage-shm:
 *
 * Example of using huge page memory in a user application using Sys V shared
 * memory system calls.  In this example the app is requesting 256MB of
 * memory that is backed by huge pages.  The application uses the flag
 * SHM_HUGETLB in the shmget system call to inform the kernel that it is
 * requesting huge pages.
 *
 * For the ia64 architecture, the Linux kernel reserves Region number 4 for
 * huge pages.  That means that if one requires a fixed address, a huge page
 * aligned address starting with 0x800000... will be required.  If a fixed
 * address is not required, the kernel will select an address in the proper
 * range.
 * Other architectures, such as ppc64, i386 or x86_64 are not so constrained.
 *
 * Note: The default shared memory limit is quite low on many kernels,
 * you may need to increase it via:
 *
 * echo 268435456 > /proc/sys/kernel/shmmax
 *
 * This will increase the maximum size per shared memory segment to 256MB.
 * The other limit that you will hit eventually is shmall which is the
 * total amount of shared memory in pages. To set it to 16GB on a system
 * with a 4kB pagesize do:
 *
 * echo 4194304 > /proc/sys/kernel/shmall
 */

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/mman.h>

#ifndef SHM_HUGETLB
#define SHM_HUGETLB 04000
#endif

#define LENGTH (256UL*1024*1024)

#define dprintf(x)  printf(x)

/* Only ia64 requires this */
#ifdef __ia64__
#define ADDR (void *)(0x8000000000000000UL)
#define SHMAT_FLAGS (SHM_RND)
#else
#define ADDR (void *)(0x0UL)
#define SHMAT_FLAGS (0)
#endif

int main(void)
{
    int shmid;
    unsigned long i;
    char *shmaddr;

    if ((shmid = shmget(2, LENGTH,
                SHM_HUGETLB | IPC_CREAT | SHM_R | SHM_W)) < 0) {
        perror("shmget");
        exit(1);
    }
    printf("shmid: 0x%x\n", shmid);
```

```
    shmaddr = shmat(shmid, ADDR, SHMAT_FLAGS);
    if (shmaddr == (char *)-1) {
        perror("Shared memory attach failure");
        shmctl(shmid, IPC_RMID, NULL);
        exit(2);
    }
    printf("shmaddr: %p\n", shmaddr);

    dprintf("Starting the writes:\n");
    for (i = 0; i < LENGTH; i++) {
        shmaddr[i] = (char)(i);
        if (!(i % (1024 * 1024)))
            dprintf(".");
    }
    dprintf("\n");

    dprintf("Starting the Check...");
    for (i = 0; i < LENGTH; i++)
        if (shmaddr[i] != (char)i)
            printf("\nIndex %lu mismatched\n", i);
    dprintf("Done.\n");

    if (shmdt((const void *)shmaddr) != 0) {
        perror("Detach failure");
        shmctl(shmid, IPC_RMID, NULL);
        exit(3);
    }

    shmctl(shmid, IPC_RMID, NULL);

    return 0;
}
```