```
/*
 * hugepage-mmap:
 *
 * Example of using huge page memory in a user application using the mmap
 * system call.  Before running this application, make sure that the
 * administrator has mounted the hugetlbfs filesystem (on some directory
 * like /mnt) using the command mount -t hugetlbfs nodev /mnt. In this
 * example, the app is requesting memory of size 256MB that is backed by
 * 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.
 */

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define FILE_NAME "/mnt/hugepagefile"
#define LENGTH (256UL*1024*1024)
#define PROTECTION (PROT_READ | PROT_WRITE)

/* Only ia64 requires this */
#ifdef __ia64__
#define ADDR (void *)(0x8000000000000000UL)
#define FLAGS (MAP_SHARED | MAP_FIXED)
#else
#define ADDR (void *)(0x0UL)
#define FLAGS (MAP_SHARED)
#endif

static void check_bytes(char *addr)
{
    printf("First hex is %x\n", *((unsigned int *)addr));
}

static void write_bytes(char *addr)
{
    unsigned long i;

    for (i = 0; i < LENGTH; i++)
        *(addr + i) = (char)i;
}

static void read_bytes(char *addr)
{
    unsigned long i;

    check_bytes(addr);
    for (i = 0; i < LENGTH; i++)
        if (*(addr + i) != (char)i) {
            printf("Mismatch at %lu\n", i);
            break;
        }
}

int main(void)
{
    void *addr;
    int fd;
```

```
    fd = open(FILE_NAME, O_CREAT | O_RDWR, 0755);
    if (fd < 0) {
        perror("Open failed");
        exit(1);
    }

    addr = mmap(ADDR, LENGTH, PROTECTION, FLAGS, fd, 0);
    if (addr == MAP_FAILED) {
        perror("mmap");
        unlink(FILE_NAME);
        exit(1);
    }

    printf("Returned address is %p\n", addr);
    check_bytes(addr);
    write_bytes(addr);
    read_bytes(addr);

    munmap(addr, LENGTH);
    close(fd);
    unlink(FILE_NAME);

    return 0;
}
```