```c
/*
 * Example of using hugepage memory in a user application using the mmap
 * system call with MAP_HUGETLB flag.  Before running this program make
 * sure the administrator has allocated enough default sized huge pages
 * to cover the 256 MB allocation.
 *
 * For ia64 architecture, Linux kernel reserves Region number 4 for hugepages.
 * That means the addresses starting with 0x800000... will need to be
 * specified.  Specifying a fixed address is not required on ppc64, i386
 * or x86_64.
 */
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define LENGTH (256UL*1024*1024)
#define PROTECTION (PROT_READ | PROT_WRITE)

#ifndef MAP_HUGETLB
#define MAP_HUGETLB 0x40000 /* arch specific */
#endif

/* Only ia64 requires this */
#ifdef __ia64__
#define ADDR (void *)(0x8000000000000000UL)
#define FLAGS (MAP_PRIVATE | MAP_ANONYMOUS | MAP_HUGETLB | MAP_FIXED)
#else
#define ADDR (void *)(0x0UL)
#define FLAGS (MAP_PRIVATE | MAP_ANONYMOUS | MAP_HUGETLB)
#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;

    addr = mmap(ADDR, LENGTH, PROTECTION, FLAGS, 0, 0);
    if (addr == MAP_FAILED) {
        perror("mmap");
        exit(1);
```

```
    }

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

    munmap(addr, LENGTH);

    return 0;
}
```