```
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <signal.h>

#define _GNU_SOURCE

const char *smallfile = "small.file";
const char *largefile = "large.file";

const char *data = "some test data";

void sig_handle(int s) {
  fflush(stdout); // Just so we can se the order of things
  psignal(s, "Caught signal");
  exit(1);
}

void quit(const char *msg) {
  perror(msg);
  exit(1);
}

size_t file_size(int fd) {
  struct stat s;
  if (fstat(fd, &s) == -1)
    quit("fstat() failed");
  return s.st_size;
}

void* _mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset
) {
  void *m;
  if ((m = mmap(addr, length, prot, flags, fd, offset)) == MAP_FAILED)
    quit("mmap() failed");
  return m;
}

int _open(const char *pathname, int flags) {
  int fd;
  if ((fd = open(pathname, flags)) == -1)
    quit("open() failed");
  return fd;
}

off_t _lseek(int fd, off_t offset, int whence) {
  off_t o;
  if ((o = lseek(fd, offset, whence)) == -1)
    quit("lseek() failed");
  return o;
}

ssize_t _read(int fd, void *buf, size_t count) {
```

```
  ssize_t s;
  if ((s = read(fd, buf, count)) == -1)
    quit("read() failed");
  return s;
}

ssize_t _write(int fd, const void *buf, size_t count) {
  ssize_t s;
  if ((s = write(fd, buf, count)) == -1)
    quit("write() failed");
  return s;
}

void test_A() {
  printf("Opening small file\n");
  int fd = _open(smallfile, O_RDONLY);
  size_t length = file_size(fd);
  printf("File has length %zuB\n", length);
  void *m = _mmap(NULL, length, PROT_READ, MAP_SHARED, fd, 0);
  printf("Writing to map...\n");
  sprintf(m, "a");
}

void test_BC(int shared) {
  size_t offset = 0;
  printf("Opening test file\n");
  int fd = _open(largefile, O_RDWR);
  size_t length = file_size(fd);
  printf("File has length %zuB\n", length);
  void *m = _mmap(NULL, length, PROT_READ|PROT_WRITE, (shared ? MAP_SHARED : MAP
_PRIVATE), fd, offset);
  printf("Writing the following to mapped file: %s\n", data);
  sprintf(m, "%s", data);
  _lseek(fd, offset, SEEK_SET);
  char buf [32];
  _read(fd, buf, strlen(data));
  printf("Read the following from file: %s\n", buf);
  printf("Data matched: %s\n", strcmp(data, buf) == 0 ? "yes" : "no");
}

void test_DE(int e) {
  size_t offset = 0;
  printf("Opening test file\n");
  int fd = _open(largefile, O_RDWR);
  size_t length = file_size(fd);
  printf("File has length %zuB\n", length);
  void *m = _mmap(NULL, length, PROT_READ|PROT_WRITE, MAP_SHARED, fd, offset);
  printf("Writing one byte past end of file\n");
  sprintf(m + length, "%s", data);
  size_t newlength = file_size(fd);
  printf("File now has length %zuB\n", newlength);
  printf("File expanded: %s\n", newlength == length ? "no" : "yes");
  // This doesn't work because you're writing to memory that hasn't even been ma
pped to a file

  if (e) {
    char buf[32];
    printf("Expanding file\n");
    _lseek(fd, strlen(data), SEEK_END);
```

```
    _write(fd, "\0", 1);
    _lseek(fd, length, SEEK_SET);
    _read(fd, buf, 32);
    newlength = file_size(fd);
    printf("File now has length %zuB\n", newlength);
    printf("Data read from file: %s\n", buf);
    printf("Data was written to file: %s\n", strcmp(buf, data) == 0 ? "yes" : "no");
  }
}

void test_F() {
  printf("Opening small test file\n");
  int fd = _open(smallfile, O_RDWR);
  size_t length = file_size(fd);
  printf("File has length %zuB\n", length);
  char *m = (char*)_mmap(NULL, 8196, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
  printf("Accessing first page\n");
  char test = m[300];
  printf("Accessing first page successful\nNow accessing second page\n");
  test = m[6000];
  printf("Accessing second page successful\n");
  // Writing to the first page works because that page exists (since file was re
ad into it)
  // Writing to the second page doesn't work because it doesn't exists yet
}

int main( int argc, const char* argv[] ) {
  if (signal(SIGSEGV,sig_handle) == SIG_ERR)
    quit("signal() failed");
  if (signal(SIGBUS,sig_handle) == SIG_ERR)
    quit("signal() failed");
  switch (argv[1][0]) {
    case 'A':
      test_A();
      break;
    case 'B':
      test_BC(1);
      break;
    case 'C':
      test_BC(0);
      break;
    case 'D':
      test_DE(0);
      break;
    case 'E':
      test_DE(1);
      break;
    case 'F':
      test_F();
      break;
  }
}
```

```
#!/bin/sh
smallfile='dd bs=1 count=10 if=/dev/urandom of=small.file'
largefile='dd bs=1 count=10000 if=/dev/urandom of=large.file'

if [ -e ./small.file ]
  then
  rm small.file
fi

if [ -e ./large.file ]
  then
  rm large.file
fi

echo "Running test A\n"
eval $smallfile
./mtests A
rm small.file
echo "\n\n"

echo "Running test B\n"
eval $largefile
./mtests B
rm large.file
echo "\n\n"

echo "Running test C\n"
eval $largefile
./mtests C
rm large.file
echo "\n\n"

echo "Running test D\n"
eval $largefile
./mtests D
rm large.file
echo "\n\n"

echo "Running test E\n"
eval $largefile
./mtests E
rm large.file
echo "\n\n"

echo "Running test F\n"
eval $smallfile
./mtests F
rm small.file
```

```
Running test A

10+0 records in
10+0 records out
10 bytes (10 B) copied, 0.0003063 s, 32.6 kB/s
Opening small file
File has length 10B
Writing to map...
Caught signal: Segmentation fault



Running test B

10000+0 records in
10000+0 records out
10000 bytes (10 kB) copied, 0.157535 s, 63.5 kB/s
Opening test file
File has length 10000B
Writing the following to mapped file: some test data
Read the following from file: some test data
Data matched: yes



Running test C

10000+0 records in
10000+0 records out
10000 bytes (10 kB) copied, 0.129229 s, 77.4 kB/s
Opening test file
File has length 10000B
Writing the following to mapped file: some test data
Read the following from file: ^]ýmñjýð
½^?aFG+
Data matched: no



Running test D

10000+0 records in
10000+0 records out
10000 bytes (10 kB) copied, 0.132988 s, 75.2 kB/s
Opening test file
File has length 10000B
Writing one byte past end of file
File now has length 10000B
File expanded: no



Running test E

10000+0 records in
10000+0 records out
10000 bytes (10 kB) copied, 0.130892 s, 76.4 kB/s
Opening test file
File has length 10000B
```

```
Writing one byte past end of file
File now has length 10000B
File expanded: no
Expanding file
File now has length 10015B
Data read from file: some test data
Data was written to file: yes



Running test F

10+0 records in
10+0 records out
10 bytes (10 B) copied, 0.000228325 s, 43.8 kB/s
Opening small test file
File has length 10B
Accessing first page
Accessing first page successful
Now accessing second page
Caught signal: Bus error
```