Memory/mmap Test Programs

A sophisticated programming technique is to create "test programs" which probe for the existence of certain features on a target platform, or which "prove" that something works a certain way. For example, many open-source programs use the GNU autoconf utility to automatically test for things such as the size of a long, and create header files so that code can be compiled automatically on a variety of platforms.

In this assignment, you will be creating test programs to discover the answers to a variety of questions having to do with the virtual memory system. Of course, you (should) already know the answers....they are probably in this unit's lecture notes, but your task is to create a program or system of programs to learn each answer **without user intervention. To be a true test program, it must be capable of determing the answer through conditional test, not simply printing out a foregone conclusion!** For example, the following is a valid test:

```
if (2+2==4) printf("ints are at least 3 bits long\n");
```

Write a program or series of programs to answer these questions about memory-mapped files:

A: "When one has mapped a file for read-only access, but attempts to write to that mapped area, what signal is generated?"

B: "If one maps a file with MAP_SHARED and then writes to the mapped memory, is that update visible when accessing the file through the traditional lseek(2)/read(2) system calls?"

C: Same question as above, except for MAP_PRIVATE.

D: "Say a pre-existing file of a certain size which is not an exact multiple of the page size is mapped with MAP_SHARED and read/write access, and one writes to the mapped memory just beyond the byte corresponding to the last byte of the existing file. Does the size of the file through the traditional interface (e.g. `stat(2)`) change? (*In your narrative, explain your reasoning why this is or is not the case*)

E: "Let us say that after performing the aforementioned memory write, one then increased the size of the file beyond the written area, without over-writing the intervening contents (e.g. by using `lseek(2)` and then `write(2)` with a size of 1), thus creating a 'hole' in the file. What happens to the data that had been written to this hole previously via the memory-map? Are they visible in the file"

F: Let's say there is an existing small file (say 10 bytes). Can you establish a valid mmap region two pages (8192 bytes) long? If so, what signal is delivered when attempting to access memory in the second page? What about the first page? Explain any differences in these outcomes (*your explanation need not be printed out by the test program ... it can be in your accompanying assignment write-up or in the source code comments*)

Sample output. Your output does not have to match this exactly:

```
$ ./mm2 D
Creating 8195 byte random file testfile
1+0 records in
1+0 records out
The size of the file as reported by fstat is 8195
about to MAP_SHARED
About to write 4 bytes to offset 8195
The new size of the file as reported by fstat is 8195
File dump starting at offset 8195, read req 4, ret 0
```

```
<EOF>
Memory dump starting at offset 8195
 <58>  <59>  <5A>  <00>
In response to question D,  NO, the file size does not change
About to expand file by 13 bytes and write(2) to new end
Memory dump starting at offset 8195 for 16 bytes
 <58>  <59>  <5A>  <00>  <00>  <00>  <00>  <00>  <00>  <00>  <41>  <42>  <43>
  <00>  <00>  <00>
File dump starting at offset 8195, read req 16, ret 13
 <58>  <59>  <5A>  <00>  <00>  <00>  <00>  <00>  <00>  <00>  <41>  <42>  <43>
<EOF>
In response to question E:  YES, the data previously written to the hole remain
```

To properly code these test programs, you will need to establish signal handlers as discussed in Unit 4. It is permissible to write one program which contains all of the tests and either select them with a command-line flag, or run them sequentially (this is more cumbersome as it requires clean-up from the last test), or to write a series of individual programs which each test one thing.