Problem 1 -- mmap vs read for grep

A common UNIX utility is the `grep` command which searches for patterns in files. We are going to write a variant of this program. Unlike the traditional grep command which was meant for line-delimited text files, we are going to be searching for binary patterns. Here is the syntax of this command:

```
SYNOPSIS:
        bgrep {OPTIONS} -p pattern_file {file1 {file 2} ...}
        bgrep {OPTIONS} pattern {file1 {file2} ...}
DESCRIPTION:
        Searches for a binary pattern in one or more input files.
        The pattern may be specified in one of two ways:
         1)A file may be specified with the -p option.  The contents
           of that file are the exact binary pattern
         2)The pattern can be specified as the first argument after the
           option flags.  For binary patterns that contain non-printable
           characters, this may be awkward and method (1) may be easier
         These two methods are mutually exclusive.  If there is no -p
         option flag, then the first argument after the option flags is
         the pattern.

        If no files are named then bgrep's sole input file is standard input

        bgrep reports, for each instance of a match, the name of the file
        which matches (standard input is reported as <standard input>)
        along with the byte offset of the start of the match, and surrounding
        context if requested with the -c option
OPTIONS:
        -p pattern_file
                 Read the pattern from pattern_file

        -c context_bytes
                 When a match is found, also output the binary context
                 surrounding the match for context_bytes before and after.
                 If there are fewer than context_bytes either before or after,
                 that part of the output is truncated.
RETURNS:
        If any system call errors occur, bgrep returns -1.  Otherwise,
        if at least one instance of the pattern is found, bgrep returns 0,
        otherwise returns 1.  For each instance of the pattern matching,
        it prints to standard output the file name and byte offset within
        the file of the start of the match.  If the -c option is given,
        it also prints a human-readable and a hex output of the binary context
        on either side of the match, including the matched bytes too.
```

This is a tremendous pain in numerous areas to implement with conventional I/O using the read system call, requiring

some tricky buffer manipulations. The pattern finding itself is not that bad, but capturing the leading and trailing context for all cases, and doing so with efficient use of system calls and memory, is a difficult problem. However, we are going to take the easy route and use the `mmap` system call to search each file as a block of memory.

However, there are limitations with this approach. You will not be able to pipe into this command, because pipes can not be mmap'd. You will not be able to use this program from standard input if it is the terminal, again because terminals are character device files that can't be mmap'd. If you can't use mmap on a given input source, report an error and skip it, but continue searching the other inputs (if applicable). Read the definition of the program's return value carefully.

The size of a file being examined could change while you are scanning it. If the file grows, you aren't required to detect that and scan the new part (you could get stuck in an endless loop that way too). But, if the file shrinks, that could cause `SIGBUS`. You must handle that, close the file in question and move on to the other files. Since this is an error, the return code will be -1 (255) regardless if any other files matched. Here are some example runs:

```
$ ./bgrep -c 6 JFIF f1.jpg  f2.gif f3.gif f4.tif f5.tif
f1.jpg:6 ? ? ? ? ? ? J F I F ? ? ? ? ? , ?    FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C
f4.tif:7206 ? ? ? ? ? ? J F I F ? ? ? ? ? H ?    FF D8 FF E0 00 10 4A 46 49 46 00 01 02 01 00
f4.tif:2947840 F I G F G I J F I F E C D E D B A    46 49 47 46 47 49 4A 46 49 46 45 43 44 45
f5.tif:493928 ? ? ? ? ? ? J F I F ? ? ? ? ? ? ?    FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00
$ echo $?
0
$ ./bgrep FOOBAR badfile.txt
Can't open bf.txt for reading:No such file or directory
$ echo $?
255
$ ./bgrep NOSUCHPATTERN f1.jpg
$ echo $?
1
$ ./bgrep pattern largefile1 largefile2  largefile3
largefile1:1024
largefile1:16384
SIGBUS received while processing file largefile2
largefile3:6
$ echo $?
255
$ perl -e 'print "\177ELF\001";' >pattern.bin
$ hexdump -c pattern.bin
0000000 177   E   L   F 001
0000005
$ ./bgrep -c 8 -p pattern.bin ./bgrep
 ./bgrep:0 ? E L F ? ? ? ? ? ? ? ? ?    7F 45 4C 46 01 01 01 00 00 00 00 00 00 00
```