Assignment 12

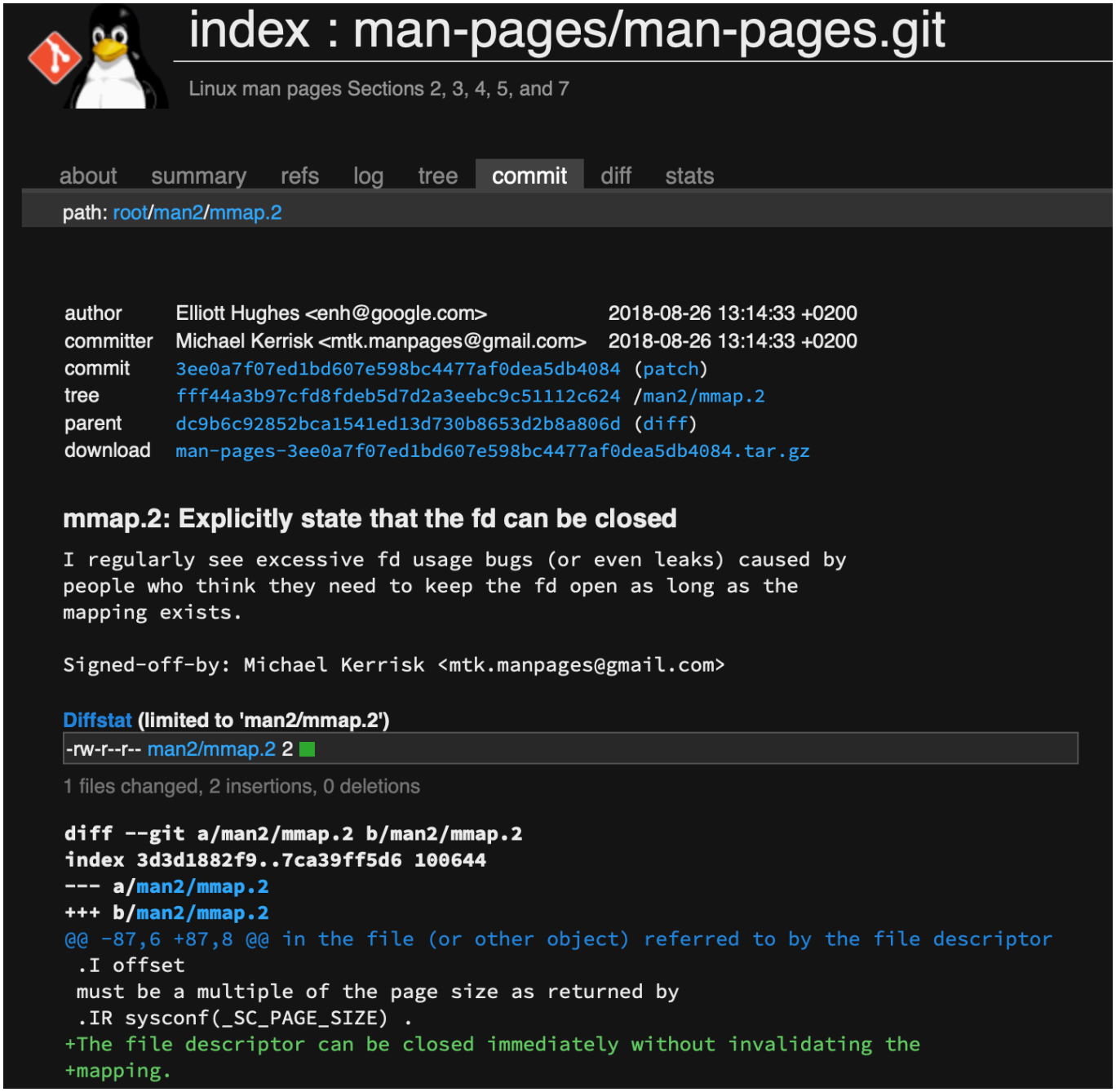
- Code

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

#define FILE_MODE 0644
#define COPYINCR (1024 * 1024l * 1024)

#define err_sys(fmt, ...) fprintf(stderr, (fmt "\n"), ##__VA_ARGS__)
#define err_quit(fmt, ...) do { err_sys(fmt, ##__VA_ARGS__); exit(1);

int main(int argc, char *argv[])
{
        int fdin, fdout;
        void *src, *dst;
        size_t copysz;
        struct stat sbuf;
        off_t fsz = 0;

        if (argc != 3)
                err_quit("usage: %s <fromfile> <tofile>", argv[0]);

        if ((fdin = open(argv[1], O_RDONLY)) < 0)
                err_sys("can't open %s for reading", argv[1]);

        if ((fdout = open(argv[2], O_RDWR | O_CREAT | O_TRUNC, FILE_M(
                err_sys("can't creat %s for writing", argv[2]);

        if (fstat(fdin, &sbuf) < 0)         /* need size of input file
                err_sys("fstat error");

        if (ftruncate(fdout, sbuf.st_size) < 0) /* set output file si
                err_sys("ftruncate error");

        while (fsz < sbuf.st_size) {
            if ((sbuf.st_size - fsz) > COPYINCR)
                copysz = COPYINCR;
```

```
42              else
43                  copysz = sbuf.st_size - fsz;
44
45              /* TODO: Copy the file using mmap here */
46              if ((src = mmap(0, copysz, PROT_READ, MAP_SHARED, fdin, f
47                  err_sys("mmap input error");
48              if ((dst = mmap(0, copysz, PROT_READ | PROT_WRITE, MAP_SH
49                  err_sys("mmap output error");
50              close(fdin);
51              memcpy(dst, src, copysz);
52              munmap(src, copysz);
53              munmap(dst, copysz);
54              fsz += copysz;
55          }
56          exit(0);
57  }
58
```

(2) Will closing the file descriptor invalidate the memory-mapped I/O?

- No. By our experiment and the reference (https://git.kernel.org/pub/scm/docs/man-pages/man-pages.git/commit/man2/mmap.2?id=3ee0a7f07ed1bd607e598bc4477af0dea5db4084) for the manual of mmap.



(3) Describe your implementation in the report.

- Line 46 to Line 49
  Set our source and destination by `mmap`.
  - `0` means the starting address, normally set to zero for portability.
  - `copysz` represents the length of the characters we want to copy.
  - We only need to open the `PROT_READ` mode for the source while both `PROT_READ` and `PROT_WRITE` for the destination.
  - `fdin` and `fdout` are the file descriptors of our input file and output file.
  - `fsz` is the starting offset of the file.
- Line 50
  Close the input file. (The spec requires)

- Line 51
  Invoke `memcpy` to copy the characters from `src` to `dst`.

- Line 52 and Line 53
  Unmap the region of `src` and `dst`. Note that the region is released here, not `close(fin)` or `close(fout)`.

- Line 54
  Add `copysz` to the offset.