```c
#include <stdio.h>   // FILE, size_t, perror, f* I/O functions
#include <stdlib.h>  // size_t, EXIT_FAILURE, EXIT_SUCCESS

#define READBUFSIZE 128

/* Copy a file byte-by-byte one user buffer size at a time */
int main(int argc, char** argv) {
  FILE* fin;
  FILE* fout;
  char readbuf[READBUFSIZE];
  size_t readlen;

  // Take the filenames from command line arguments
  if (argc != 3) {
    fprintf(stderr, "usage: ./cp_example infile outfile\n");
    return EXIT_FAILURE;
  }

  // Open the input file
  fin = fopen(argv[1], "rb");  // "rb" --> read, binary mode
  if (fin == NULL) {
    perror("fopen for read failed");
    return EXIT_FAILURE;
  }

  // Open the output file
  fout = fopen(argv[2], "wb");  // "wb" --> truncate & write, binary mode
  if (fout == NULL) {
    perror("fopen for write failed");
    fclose(fin);
    return EXIT_FAILURE;
  }


  // Read from the file, write to fout.
  while ((readlen = fread(readbuf, 1, READBUFSIZE, fin)) > 0) {
    // Test to see if we encountered an error while reading.
    if (ferror(fin)) {
      perror("fread failed");
      fclose(fin);
      fclose(fout);
      return EXIT_FAILURE;
    }
    if (fwrite(readbuf, 1, readlen, fout) < readlen) {
      perror("fwrite failed");
      fclose(fin);
      fclose(fout);
      return EXIT_FAILURE;
    }
  }

  // No need to error-check fclose - usually called just before exiting.
  // From man page:  "In either case [success or failure], any futher access
  // (including another call to fclose()) to the stream results in undefined
  // behavior."
  fclose(fin);
  fclose(fout);

  return EXIT_SUCCESS;
}
```