Chapter 27: Error handling

Section 27.1: errno

When a standard library function fails, it often sets errno to the appropriate error code. The C standard requires at least 3 values for errno be set:

Value	Meaning
EDOM	Domain error
ERANGE	Range error
EILSEQ	Illegal multi-byte character sequence

Section 27.2: strerror

If `perror` is not flexible enough, you may obtain a user-readable error description by calling `strerror` from **`<string.h>`**.

```
int main(int argc, char *argv[])
{
    FILE *fout;
    int last_error = 0;

    if ((fout = fopen(argv[1], "w")) == NULL) {
        last_error = errno;
         /* reset errno and continue */
         errno = 0;
    }

    /* do some processing and try opening the file differently, then */


    if (last_error) {
        fprintf(stderr, "fopen: Could not open %s for writing: %s",
                argv[1], strerror(last_error));
        fputs("Cross fingers and continue", stderr);
    }

    /* do some other processing */

    return EXIT_SUCCESS;
}
```

Section 27.3: perror

To print a user-readable error message to `stderr`, call `perror` from <stdio.h>.

```
int main(int argc, char *argv[])
{
   FILE *fout;

   if ((fout = fopen(argv[1], "w")) == NULL) {
      perror("fopen: Could not open file for writing");
      return EXIT_FAILURE;
   }
return EXIT_SUCCESS;
}
```

This will print an error message concerning the current value of `errno`.