Lab 9: Functions & files in image management

Goals

Demonstrate proficiency in creating functions that perform basic operations that are useful in image management and in using the fopen(), fread(), fwrite() and fclose() to read and write disk resident .ppm files.

Background

Use of files other than *stdin, stdout,* and *stderr.*

- Heretofore, all of our input and output operations have been directed to the predefined FILEs *stdin, stdout,* and *stderr.* However, it is easy to associate logical FILE structures directly with specific disk resident files.

- To read directly from a disk file the following steps are necessary:

- Declare a FILE structure pointer as follows:

```
FILE *myfile;
```

- Use the *fopen()* function to bind your file name to the physical disk file. The two parameters passed to *fopen()* are the name of the disk file and the operation type desired "r", "w", or "rw". The disk file name can contain a full path specification. If the full path is omitted the file being opened must reside in the current working directory.

```
myfile = fopen("union-jack.ppm", "r");
```

- Make sure that *fopen()* was successful.

```
if (myfile == 0)
{
   fprintf(stderr, "Open failed\n");
   perror("bad open:");
   return(-1);
}
```

- Now the fscanf(), fread(), fprintf(), fwrite() functions may be used as usual, but with the *myfile* instead of stdin or stdout used as the argument:

```
howmany = fscanf(myfile, "%s", image->id);
count = fread(pixmap, sizeof(pix_t),
              pixcount, myfile);

fprintf(myout, "P6\n");
```

- When reading is complete the FILE * should be closed using the *fclose* function.

```
fclose(myfile);
```

Assignment:

Create a lab9 directory and copy the contents of the labs07/lab9 directory to your lab9 directory. The main() function is found in main.c and is already complete. Your mission is to *complete four functions* in the image.c file. Two of the functions: *get_pixel()* and *set_pixel()* are the same functions required for this week's codelab exercise.

The *int load_image(img_t *image)* function should perform the following operations:

(1) Create a *FILE** file structure pointer and set it to the value returned by *fopen()* when fopen() is passed the file name contained in the img_t structure and an operation code of "r".

(2) Verify that fopen() was successful and return(-1) if not.

(3) Invoke the *read_ppmhdr()* function passing it the img_t pointer *and your FILE * pointer.* If read_ppmhdr() returns a value other than 0, immediately return the code returned by read_ppmhdr().

(4) Use malloc() to allocate space for the pixmap, storing the location of the pixmap in *image->pixmap*.

(5) Use *count = fread()* to read the image data into the malloc'ed area. Verify that count == image->width * image->depth and return(-2) if not.

(6) return(0);

The *int save_image(img_t *image)* function should perform the following operations:

(1) Create a FILE * file structure pointer and set it to the value returned by fopen() when fopen() is passed the file name contained in the img_t structure and an operation code of "w".

(2) Verify that fopen() was successful and return(-1) if not.

(3) Use fprintf() with *your file structure pointer* as the first parameter to print a valid .ppm header.

(4) Use count = fwrite() to write the image data from the pixmap. Verify that count == image->width * image->depth and return(-2) if not.

(5) return(0);

Turn In Work

Show your TA that you completed the assignment. Then turn in your image.c program using the command:

sendlab.101.*section_number* 9 image.c

DO NOT TURN IN main.c, imp.h, imphdrs.h or any image files.