Week 5 Lab

File I/O and Make

Task 1. Working with text files

Write a program called `letterfreq` that does the following things:

- takes a filename as a command line parameter and attempts to open it for reading
- if that attempt fails, append ".txt" to the filename and try to open that
- displays a useful error message if the file could not be opened
- reads the file a character at a time
- counts how many of each letter of the alphabet appears in the file (ignoring case and disregarding nonalphabetic characters)
- outputs letter counts for each letter at the end of the run

Write a `Makefile` that has `letterfreq` as a target.

Task 2. Working with binary files

Write a program called `hexdump` that takes a filename as a command line parameter and outputs its contents to `stdout` as a sequence of space-separated two-digit hexadecimal numbers.

Write a second program, `dumphex`, that does the reverse: it takes a filename as a command line parameter, reads a sequence of space-separated hexadecimal numbers on `stdin`, and stores the byte stream represented by those numbers into the file.

Note that one byte is always represented as two hexadecimal digits, so you can assume that you'll always be reading one character at a time.

Add targets for `hexdump` and `dumphex` to your `Makefile`. Also add a target for `all`, which should build all of these programs, and one for `clean`, which deletes all object files.

Task 3. More strings and program structure

As you're aware, strings in C do not store their lengths. Design a datatype that can store a string alongside its length. Define this datatype to be called `Newstring`.

Support this datatype by reimplementing versions of some standard C string functions to work with it. You should write (at least):

- `int newstrlen(Newstring *str);`
- `Newstring *newstrcat(Newstring *s1, Newstring *s2);`
- `Newstring *newstrcpy(Newstring *s1, Newstring *s2);`

Look at the manual pages for the functions `strlen()`, `strcat()`, and `strcpy()` if you are not familiar with them. You may put all these functions in the same C source file.

In a *separate* C source file, write a `main()` function that creates several `Newstring`s and uses them to test your functions. Run this under `valgrind` to ensure that your program does not leak memory.

Create appropriate `Makefile` targets for the new files, and keep your old `clean` and `all` targets up to date too.