1 Objectives

The objectives of this project are to get you more familiar with:

1. The programming environment
2. Working with the basics of the C language (e.g., syntax, types, loops, displaying, etc.)
3. Working with a make file
4. Working with command-line arguments via `argc` and `argv`
5. Reading and understanding `man` pages
6. Converting strings to numbers using the `strtol` library function

2 Requirements

Big picture: write a program called `days` that takes the date given on the command line (month day) and calculates the number of days in the year up to that point (excluding leap years).

2.1 Interface Requirements

1. The program shall be written in a file called `days.c`.
2. The compiled version of the program shall be called `days`.
3. `days` shall take two strings as input, indicating the month and day of the month (e.g., "`./days Jan 8`"). The month shall be given as the three-character abbreviation of a month, either as all lower-case characters (e.g., "feb") or with only the first character capitalized (e.g., "Feb"). The day shall be given as a decimal number.
4. `days` shall detect when the input is invalid.
   When `days` detects invalid input, it shall display a meaningful error message, and then exit with a value of -1. There are many ways that input can be invalid, such as:
   a. Too many or too few arguments were provided on the command-line.
   b. The input month is not recognized.
   c. The input day is less than 1 or greater than the number of possible days in the input month (e.g., "Feb 31").
   d. The input day is not a valid number (e.g., abc and 1ab).
5. After verifying that the input is valid, the program shall then calculate the day of the year. For example, if someone enters "`./days Jan 1`", then the calculated answer is 1; if someone enters "`./days Dec 31`", then the calculated answer is 365.
6. An example of the required input and output is given below (where the '$' is the command-line prompt):

```
$ ./days Jan 12
The number of days = 12
$
```

7. When `days` does not detect an input error and is able to calculate the required result, it shall display the answer and then exit with a value of 0.

2.2 Makefile Requirements

1. Create a file called “Makefile”
2. Create the following targets in the file:
    a. all
    This target shall cause the program `days` to be created.
    b. days
    If `days` does not exist, or if `days.c` is newer than `days`, then this target shall try to compile `days`. Otherwise, when `days` is newer than `days.c`, then make shall not compile `days.c`.

3 Submission

Before the deadline, post to Sakai a file called `dist1.tar`, which you can create by doing the following on the command-line:

```
tar -cvf dist1.tar Makefile days.c
```

Late submissions will not be accepted.

4 Grading

Severe deductions shall be given in the following situations:

1. Compilation errors (i.e., your code will not compile).
2. Segmentation faults or other crashes of your program occur under some circumstances.
3. The code appears to be hard-coded to give the correct answers to the test input below.
4. You have 365 `if` statements to determine what the output should be (and other such obvious inefficient programming).
5. No Makefile was turned in.
6. I reserve the right to include other reasons I have not yet encountered.

Other deductions of various degrees shall be given in the following situations:

1. Warning statements are seen when your code is compiled.
2. The following **invalid** input does not produce an error message or does not return value of -1:
    a. ./days
    b. ./days hello
    c. ./days hello world
    d. ./days Jan 31 abc
    e. ./days Jan 3rd

   f. ./days Jan abc
   g. ./days Feb 0
   h. ./days Feb -1
   i. ./days Sep 31
3. The following **valid** input does not produce the correct answers and a return value of 0:
   a. ./days Jan 1 (answer is 1)
   b. ./days Dec 31 (answer is 365)
   c. ./days May 31 (answer is 151)
   d. ./days may 31 (answer is 151)
   e. ./days Jul 4 (answer is 185)
   f. ./days Nov 11 (answer is 315)
4. Style guide violations in your code.
5. The Makefile does not function as required.
6. I reserve the right to include other reasons I have not yet encountered.

5 Advice

1. **See the "Tips" link at the top of the main Wiki page**.
2. You may find the following library functions useful: `isupper`, `islower`, `isdigit`, `toupper`. See their man pages.
3. Remember that the `strlen` function can be used to find the length of a string, and the `strcmp` function is used to compare two strings for equality. See their man pages.