Program 4a: getopt()

CSCI 4547 / 6647 Systems Programming
Fall 2017

1 Goals

1. To develop a command language for DiskSweeper.
2. To use `getopt()` to process single-letter options.
3. To build the first phase of the DiskSweeper application.
4. To learn or review some parts of C++ that will be needed in this course.

2 The Project

The Disk Sweeper application will search your hard disk for files that are copies of each other and report the full pathnames of all copies of the file. The command name will be `sweep`. The arguments will be:

- -o filename (required): open and use the named file for output.
- -v (optional): provide verbose debugging feedback.
- -d (optional): delete all duplicates except the first.
- A small integer (optional): a minimum file size to work on, in kilobytes. (That is, 5 represents 5K bytes.)
- A pathname (a string, required): the directory at which to start sweeping.

In stage 4b of this program, we will add some long switches.

3 Instructions

Write a program to parse the command line for DiskSweeper. Due Sept. 22 (Friday).

Define a class named Params. Members of the class should include:

- An ofstream.
- The pathname of the starting directory, a C-style string.
- An integer: the minimum size file to process. The default should be 0 for this optional parameter.
- A boolean variable for each switch. (All are optional, the defaults are all false.)
- The Params constructor should have two parameters: argc and argv. Process the command-line arguments using `getopt()` (instructions follow) and initialize the data members to the settings that you find on the command line. Open the ofstream for output to the file named on the command line.
- print( ): print all the members except the stream in a nice format to the open ofstream.

In your main function, declare an instance of Params and pass argc and argv to its constructor. When construction is finished, call Params::print() to display the params.

4 Using `getopt()` to Decode Command-line Arguments

The function `getopt()` automates and vastly simplifies the process of decoding a command line. The example shown here is NOT what you need for your project; it is provided only so that you can understand the syntax of the getopt() function.
Syntax: `int = getopt( int argc, char* argv[], const char* opts );`
Usage:

```
   for (;;) {
      ch = getopt(argc, argv, "i:avRou");
      if( ch == -1 ) break;
      switch( ch ){
      ...
```

- argc and argv are the parameters received from the shell by your program.
- opts is a string containing all the short switch options that your program supports: "i:avRou" This string of letters means that the application will accept the switches `i, a, v, R, o, u`. A command line could give any combination of switches, in any order.
- An option letter followed by a colon requires a parameter (usually a file name). In this example, the `-i` switch must be followed by the name of an input file.
- An option letter followed by a double colon has an optional parameter (no example here).

Processing the switches. Your program must use getopt() in a loop, as shown, to read and validate the switches that are present and store that information in Param's variables. Print a usage comment after any error you discover. `Getopt` returns -1 when there are no more switches to process.

Processing the required arguments. After returning from `getopt()`, set a local variable, `required = optind`. Optind is a global variable, so this should be done immediately. The number is the subscript of the first non-switch argument on the command line, and the first thing you must process after returning from `getopt()`.

- Since there is only one required argument in DiskSweeper, `required` should be `argc-1`. If this is not true, print a usage error comment.
- Set your `path`, equal to the required pathname.

Finishing up. After processing the command-line arguments, print a report like the one shown below and end. Test all of the command-line options and capture the results from all tests in one file, using append mode. Here is some sample output:

```
Command: sweep -v -o dupfiles.txt   10 ~/A_UNH/Teaching
    Verbose? Yes
    Delete? No
    Output file name:  dupfiles.txt
    Size: 10 K or greater
    Directory: ~/A_UNH/Teaching
```