Program 2: getopt_long()

CSCI 4547 / 6647 Systems Programming
Fall 2020

1 Goals

1. To develop a command language for a utility that combines features from find and grep.
2. To use `getopt_long()` to process options.
3. To learn or review some parts of C++ that will be needed in this course.

2 The Project

The command that you develop will be named `findit`. It will search your disk for files that contain one or more of a set of words that will be given on the command line. The possible arguments will be:

- –dir or -d followed by a pathname. The app will start its search with this directory. This switch is required.
- -i Do a case-insentitive search if this switch is present, case-sensitive is the default.
- -R Do a recursive search if this switch is present. The default is to search only one directory.
- -o a pathname (optional): Open the named file and use it for output; the default is screen output.
- –verbose (optional): print the name of every file that is opened. If one of the search words is not found in that file, print a comment. This ability is crucial during development and debugging.
- A string of search words, separated by spaces.

3 Instructions

Write a program that uses `getopt_long()` to parse a command line for findit. Due September 15.

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

- An ofstream.
- A C++ string to store the string of search words. (This will be unpacked into a vector of words in Program 3.)
- The pathname of the starting directory, a C-style string, or null.
- A boolean variable for each switch. (The defaults are all false.)
- The Params constructor should have two parameters: argc and argv. Process the command-line arguments using `getopt_long()` and initialize the data members to the settings that you find on the command line.
- For the search words, argv gives you a c-string containing the words. You need to convert that to a C++ string and store it. The easiest way is to just call the C++ string constructor with the argv input; the prototype is `string (const char* s)` . Refer to the stringstr.cpp demo, attached.
- 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.

Testing. Make a special test directory on your disk and use it consistently.

- Each week, as you develop this application, you will add some files or directories to it.
- For now, select a theme and two to five search words related to that theme. For example, theme: election. search words: Trump, Biden, president, candidate, election.
- In your test directory, create several short files that contain different subsets of your search words, including files with zero, one, and more of these words.

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: findit --verbose -o found.txt  --dir ~/A_UNH/Teaching  "CSCI"
    Verbose? Yes
    Case insensitive? No
    Recursive? No
    Output file name:  found.txt
    Directory: ~/A_UNH/Teaching
```