$Id: asg1j-jgrep-files.mm,v 1.2 2014-03-24 18:33:58-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps012b-wm/Assignments/asg1j-jgrep-files
URL: http://www2.ucsc.edu/courses/cmps012b-wm/:/Assignments/asg1j-jgrep-files/

1. Introduction

In this assignemnt you will perform options analysis and read sequences of files, printing out lines that match a regular expression. Your Java program will be placed in a jar file. Your program should behave in a manner similar to **grep**(1).

2. Program specification

The program is presented in the form of a Unix **man**(1) page.

NAME

jgrep — search files for a pattern

SYNOPSIS

jgrep [**-ilnv**] *pattern* [*filename* . . .]

DESCRIPTION

The **jgrep** utility searches text files for a pattern and prints out all lines that match that pattern. Patterns are specified as regular expressions and make use of **java.util.regex.Pattern**.

OPTIONS

The options word must be the first word and begin with a minus sign (**-**) if present. Option letters in the options word may occur in any order. Operands appear thereafter.

-i Ignore upper- and lower-case distinctions in the specified pattern.

-l Prints only the names of the files with matching lines, one filename per line. A filename is printed only once.

-n Precede each line by its line number within the file, with the first line being line 1.

-v Prints all lines except for those that match the pattern. The sense of the match is complemented.

OPERANDS

The first operand is required and is a regular expression according to the syntax recognized by the **java.util.regex.Pattern** class. All other operands are filenames. Each file is opened for input in turn. if no filenames are specified, then **stdin** is read instead.

EXIT STATUS

The following exit status values are returned when the program exits :

0 One or more matches were found.

1 No matches were found.

2 Syntax errors or inaccessible files were found.

SEE ALSO

grep(1), **egrep**(1), **fgrep**(1).

3. Starter code

In the `code/` subdirectory, you have been provided with some starter code which you should read and understand. Also, read the man page for `grep`(1), to see how it works.

(1) The main function begins by compiling the regular expression specified on the command line into a `Pattern`. If there is a syntax error, that compilation will generate a `PatternSyntaxException`, which is then reported, causing the program to terminate.

(2) If that succeeds, a `Scanner` is created either for reading `stdin`, or a loop is performed to open each file in sequence. Opening a file can cause an `IOException`, which is then reported.

(3) Note that a pattern error is fatal, while a file access error allows the program to continue.

(4) The function `scanfile` iterates over all lines in the file using `hasNextLine` and `nextLine`. The line

```
boolean matches = regex.matcher (line).find();
```

sets the boolean variable to true or false, depending on whether the line matches the regex.

(5) Error messages are always printed to `System.err`, not to `System.out`. An error message always comes in three parts: the name of the program issuing the error, what it is complaining about, and the reason for the complaint. For example:

```
jgrep: somefile.foo (No Such File)
```

4. Development sequence

You will need to develop the program a little at a time, in some stages. At each point in your development, compare the behaviour of your program with that of `grep`(1).

(1) Begin by studying the man page for `grep`(1). Also study Java's regular expressions `java.util.regex.Pattern`. Following are some examples of regexes.

`abcd` matches any line containing the string "`abcd`".

`^abcd$` matches any line containing exactly the string "`abcd`". The "`^`" matches the beginning of the line and "`$`" matches the end of the line.

`ab.*cd` matches any line containing the string "`ab`" followed somewhere later by "`cd`". The dot (`.`) matches any character and the asterisk (`*`) matches zero or more occurrences of whatever preceds it.

`ab|cd` matches any line containing an "`ab`" or a "`cd`".

`[a-z]` matches any line containing any lower case letter, i.e., the letter "`a`" or "`z`" or any character lexicographically in between. The brackets indicate that a set of characters is specified.

(2) Modify the program to recognize the four flags specified in the assignment. Unlike normal Unix commands, this java version requires the options to be clustered, but in any order. So the following mean the same :

```
jgrep -inv patt file
jgrep -nvi patt file
```

Flags may occur in any order but always must precede the pattern and filenames.

(3) Use a small convenience class **flags** which you can pass around to the necessary parts of the program :

```
class options {
   boolean insensitive;
   boolean filename_only;
   boolean number_lines;
   boolean reverse_match;
   String regex;
   String[] filenames;
}
```

Put this in a file called **options.java**. Its constructor should have **args** passed into it.

(4) If the **-i** flag is specified, the pattern is compiled with

```
Pattern pattern = Pattern.compile (regex,
        Pattern.CASE_INSENSITIVE);
```

(5) If the **-l** option is specified, only the names of the files that match are printed. And as soon as a match is found, the rest of the file is ignored. What name is printed by **grep**(1) for **stdin**?

(6) Then modify your code to handle the **-n** option.

(7) If **-v** is specified, the boolean variable **matches** is flipped.

(8) Test your program extensively against **grep**.

(9) Print error messages if invalid options are specified. The usage message is already done for you. And for bad filenames, print the results of **getMessage**, which is more complete than what **grep** shows.

(10) Modify **messages** to make sure that the exit status in the program specification is handled correctly.

(11) Modify **Makefile** to add these extra classes.

(12) Note that normal output goes to **stdout**, while error messages go to **stderr**. To verify the exit status of your program, the command **echo $?** (bash) or **echo $status** (csh) can be used.

(13) The **README** should have a ***very brief*** note in it if there is something you want the grader to know before starting the grading. Every file you submit should have your name and username in a comment at the top of the file.

(14) Run `checksource` on all files you submit. It should be silent.

5. What to submit

`README`, `Makefile`, `jgrep.java`, `messages.java`, `options.java`.

Make sure that the directory `/afs/cats.ucsc.edu/courses/cmps012b-wm/bin/` is in your `$PATH` or `$path` (depending on which shell you use). You will find the scripts `cid`, `checksource`, and `testsubmit` useful.

If you are doing pair programming, follow the Syllabus instructions detailing the pair programming requirements and submit the `PARTNER` file .