Larry Bush March 28, 2002
Student ID: 660 220 742 email: bushl2@rpi.edu or Lawrence_Bush@dps.state.ny.us

Final Project
Operating Systems
Grep for Windows

1. **General Description:** Grep (simplified) with replacement for Windows.

This program is something like the grep command in UNIX. It searches through the files in a given directory (including sub-directories) for a word and locates each occurrence of the word. It can also replace each occurrence of that word with another word at the user's request.

This program will also be multi-threaded and pre-emptable.

It is simplified in the following ways:

1. It does not use wildcards (AKA regular expressions). It just searches for the word that the uses enters.

2. It only takes single words, not phrases. In other words, it does not allow spaces in the search string or the replacement string.

How this relates to OS:

The searching of these directories relates to concepts in Chaper 6 ("File Systems") of our text book.

It will involve approximately 10 different system calls and related declarations. I read through chapters 2 and 4 in "Win32 System Services" (by Marchall Brain) and compiled the following list of system calls for moving through directories and manipulate files. This list is not necessarily complete, and some of the calls may not be necessary or useful. Some may be substituted by other similar calls.

File commands to Create the output (results) file:

ReadFile();	
WriteFile();	
CloseHandle(fileHandle);	Closes the fileHandle.

Directory Searching commands:

WIN32_FIND_DATA findData;	Declaration of a structure that describes a file.
FindFirstFile(searchFile,&findData);	Finds the first file "searchFile" and returns a fileHandle and a structure describing the file (via &findData). SearchFile may be a wildcard(*.*).
FindNextFile(fileHandle,& findData);	Finds the next file with the same name as the file pointer "fileHandle" and returns a fileHandle and a structure describing the file (via &findData). SearchFile may be a wildcard(*.*)

FindClose()	This closes the file handle.
findData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY;	Tells me if I found a directory (1) or a file (0).
findData.cFileName	Tells me the file name.
GetCurrentDirectory()	Returns the current directory path.
GetFullPathName()	Equivalent to the concatonation of cFileName with the current directory.

GetFileSize(fileHandle, &sizeHi) or findData.nFileSizeLow
These 2 commands return the file size. There are several ways to do this.

File I/O

The program would also use these file I/O commands, which are not system calls.

File Reading commands:

```
ifstream in;
in.open(name);
in >> ch;
in.close();
```

File Writing commands:

```
ofstream out;
out.open(name);
out << ch;
out.close();
```

2. Non-standard items: None

3. Platform

a. C++
b. Windows 2000

4. Functionality

Primary Functionality

Directory Searching	(25%) Search every file in a given directory (including sub-directories) for a word that is provided by the user.
Word Replacement	(25%) Replace every match found with the replacement word.

Secondary Functionality

Pre-emptable	(25%) The program should allow the user to stop the process while it is running. This can be done using a global variable which is checked by the thread periodically (in between processing files).

Other Important Grading Aspects

Program Design (Organized) and Documentation (Good Commenting) (25%)

Other Expectation Details / Assumptions:

The program is not expected to perform more than 1 grep at a time.
The interface should be straight forward, however, the user semantics/interface explained below is only approximate. The interface is not a significant grading point.
Each line in a file is expected to be '\n' delimited.
The program may not need all the system calls explained above and is therefore not required to use all of them.

Output

For each match found:
File name
The line the word is found on.

5. User semantics

a. To run the program, the user will type in the command grep with arguments at the dos prompt. The program must be in the current directory. (This can also be done through VC++)

The user will input the command line in the following format:

```
grep     word_to_find    [replacement_word]      directory
```

For example, if I want to change the word school to college in all my letters that I have written, I could type:

```
grep school college C:/Windows/Desktop/Letters/
```

The replacement word is optional.
The word to find and the replacement word must each be a single word (no spaces).

b. Tell the user what is about to happen.
Continue y/n.

c. Do grep. User is returned a prompt from which he can cancel the process using a keyword.

d. For each match found, the program will output:
File name
The line the word is found on.

This program is oriented towards UNIX users who wish windows had grep with replacement. Therefore, they will be familiar with the command line interface style and file structures.

6. Individual Project – No Partners.