

# CS 214 Fall 2023

## Project II: Word occurrence count

David Menendez

Due: Friday, November 10, 2023, at 11:59 PM (ET)

For this project, you and your partner will write a program, **words**, which will count the number of occurrences for each word in a file or set of files.

### 1 Program

**words** takes one or more arguments, which may be text files or directories. **words** will open and process each named text file. For directories, **words** will recursively seek and process files in the directory whose names end with “.txt”.

**words** must use the functions `open()`, `close()`, and `read()` to read from files and `write()` to print to standard output. Use of `sprintf()` to format decimal numbers is permitted.

**words** must use the functions `opendir()`, `closedir()`, and `readdir()` to read the contents of directories. It may use `stat()` to determine if a file is a directory.

**words** will process one or more files by keeping maintaining a count of how many times each word was encountered. It maintains a mapping from words to counts, adding new words as they are encountered, and increasing the count of words that have been seen earlier. When processing all files, the count will indicate the number of times a word has been seen in *all* files. (That is, counts are not kept per-file.)

Once all files have been processed, **words** will print the list of words and counts sorted by decreasing appearance count. Words with the same count will be ordered lexicographically. Each word will be printed on a separate line, followed by its number of occurrences.

**words** should not assume a maximum file size or a maximum word length.

**words** should assume that all files and directories in its arguments are distinct. That is, **words** does not need to check whether a file has been listed more than once.

#### 1.1 Example output

When sorting a file containing “spam eggs bacon spam”, **words** should output:

```
spam 2
bacon 1
eggs 1
```

## 1.2 Directory traversal

When an argument to **words** is a directory name, **words** will open the directory and search for (1) regular files whose names end with “.txt” and (2) subdirectories. Regular files are opened and processed as though they had been named on the command line. Subdirectories are opened and scanned recursively.

**Note** Be sure to distinguish between files in the working directory and files in the directory being scanned. For example, if the working directory contains a file “input.txt” and a directory “foo” that contains a file also named “input.txt”, and **words** is given the argument “foo”, it will scan “foo/input.txt”, not the file in the working directory.

## 2 Words

For simplicity, we will assume that a word is a sequence of letters and some punctuation. Specifically, the apostrophe (single quote) may occur anywhere within a word, and a hyphen (dash) may occur in a word, but only if preceded and followed by a letter.

That is, **foo-bar** would be considered one word, while **foo--bar** contains the words “foo” and “bar”.

We consider numbers, whitespace, and other punctuation to separate words.

For simplicity, we will consider capitalization significant. Thus, “foo” and “Foo” are considered different words.

**Note** Accurately determining what words are contained in a text document is actually pretty difficult, and requires knowledge of the document’s language. We have deliberately chosen a relatively simple set of rules for **words**.

### 2.1 Example

Consider a file containing this text:

```
Foo bar? Bar! Foo-bar baz--quux! 'oo? "Bar, bar."
Super23foo.
```

**words** should output:

```
Bar 2
bar 2
'oo 1
Foo 1
Foo-bar 1
Super 1
baz 1
foo 1
quux 1
```

### 3 Submission

Submit a single Tar archive containing your source code, make file, and README. These should be in the top level of the archive, not within a directory.

An example of archive creation:

```
tar -zcf p2.tar words.c Makefile README
```

Use `tar -tf` to confirm the contents of the archive.

We should be able to compile and execute your program “out of the box”. For example,

```
$ tar -xf p2.tar
$ make
$ ./words test_file
```

#### 3.1 README

Your README should be a plain text file (openable in nano, for example) containing the name and net ID of both partners.

Use of Markdown or similar text formatting conventions is permitted.