```
 1: // $Id: countwords.cpp,v 1.1 2020-06-27 19:59:24-07 - - $
 2:
 3: #include <cerrno>
 4: #include <cstring>
 5: #include <fstream>
 6: #include <iostream>
 7: #include <map>
 8: #include <regex>
 9: #include <string>
10: #include <vector>
11: using namespace std;
12:
13: using wordcount_type = map<string,size_t>;
14:
15: void scan (wordcount_type& words, istream& infile) {
16:    static const regex word_rx {"[[:alpha:]]+"};
17:    for (;;) {
18:       string line;
19:       getline (infile, line);
20:       if (infile.eof()) break;
21:       for (auto& chr: line) chr = tolower (chr);
22:       auto itor = sregex_iterator (line.begin(), line.end(), word_rx);
23:       for (; itor != sregex_iterator(); ++itor) {
24:          ++words[itor->str()];
25:       }
26:    }
27: }
28:
29: int main (int argc, char** argv) {
30:    wordcount_type words;
31:    string exec_name {basename (argv[0])};
32:    int exit_status = EXIT_SUCCESS;
33:    vector<string> filenames (&argv[1], &argv[argc]);
34:    if (filenames.size() == 0) filenames.push_back ("-");
35:    for (const auto& filename: filenames) {
36:       if (filename == "-") scan (words, cin);
37:       else {
38:          ifstream infile (filename);
39:          if (infile) scan (words, infile);
40:          else {
41:             exit_status = EXIT_FAILURE;
42:             cerr << exec_name << ": " << filename << ": "
43:                  << strerror (errno) << endl;
44:          }
45:       }
46:    }
47:    for (const auto& word: words) {
48:       cout << word.first << " " << word.second << endl;
49:    }
50:    return exit_status;
51: }
```

```
 1: # $Id: Makefile,v 1.2 2021-08-11 21:54:47-07 - - $
 2:
 3:
 4: GWARN   = -Wall -Wextra -Wpedantic -Wshadow -Wold-style-cast
 5: GOPTS   = ${GWARN} -fdiagnostics-color=never
 6: GPP     = g++ -std=gnu++2a -g -O0 ${GOPTS}
 7: GRIND   = valgrind --leak-check=full --show-reachable=yes
 8: NODEPS  = ${filter ci clean spotless tar, ${MAKECMDGOALS}}
 9: MKTAR   = gtar --create --verbose --gzip
10:
11: H_FILES =
12: C_FILES = countwords.cpp
13: OBJECTS = ${C_FILES:.cpp=.o}
14: EXECBIN = countwords
15: SOURCES = ${H_FILES} ${C_FILES} Makefile data
16:
17:
18: all : ${EXECBIN}
19:
20: ${EXECBIN} : ${OBJECTS}
21:         ${GPP} -o $@ $^
22:
23: %.o : %.cpp
24:         - cpplint.py.perl $<
25:         - checksource $<
26:         ${GPP} -c $<
27:
28: ci : ${SOURCES}
29:         - checksource $^
30:         cid -is $^
31:
32: clean :
33:         - rm --force ${OBJECTS} test.log test.out test.err
34:
35: lis : ${SOURCES} Makefile.deps
36:         mkpspdf Listing.ps $^
37:
38: spotless : clean
39:         - rm --force ${EXECBIN} Listing.{ps,pdf} Makefile.deps
40:
41: tar : ${SOURCES}
42:         ${MAKE} --no-print-directory spotless
43:         ( DIRNAME=$$(basename $$(pwd)) \
44:         ; cd .. \
45:         ; ${MKTAR} --exclude=RCS --file=countwords.tar.gz $$DIRNAME \
46:         )
47:
48: test : ${EXECBIN}
49:         ${GRIND} --log-file=test.log \
50:                 ${EXECBIN} ${SOURCES} 1>test.out 2>test.err
51:
52: Makefile.deps :
53:         ${GPP} -MM ${C_FILES} >Makefile.deps
54:
55: ifeq (${NODEPS}, )
56: include Makefile.deps
57: endif
58:
```

```
1: This is a test of data.
2: Data of is this a test?
3: Data of is this a test?
4: Data of is this a test?
5: what is this testing?
6: what is this testing?
7: $Id: data,v 1.2 2022-01-05 17:15:41-08 - - $
```

```
1: countwords.o: countwords.cpp
```