```
 1: // $Id: wordct.c,v 1.2 2014-04-22 19:38:08-07 - - $
 2:
 3: //
 4: // NAME
 5: //    wordct - count lines, words, and characters in files
 6: //
 7: // SYNOPSIS
 8: //    wordct [-lwc] [file...]
 9: //
10: // DESCRIPTION
11: //    Print the character (byte), word, and newline counts for
12: //    each file, and a total line if more than one file is
13: //    specified.  If no file is specified, or if file is just
14: //    a -, read stdin.  A word is any white-space delimited
15: //    sequence of characters.
16: //
17: // OPTIONS
18: //    If no options are specified, print all three counts.
19: //    If any options are specified, print only those requested.
20: //    -l  print the line counts
21: //    -w  print the word counts
22: //    -c  print the byte counts
23: //
24:
25: #include <assert.h>
26: #include <ctype.h>
27: #include <errno.h>
28: #include <libgen.h>
29: #include <stdarg.h>
30: #include <stdbool.h>
31: #include <stdio.h>
32: #include <stdlib.h>
33: #include <string.h>
34: #include <sys/stat.h>
35: #include <unistd.h>
36:
37: char *program_name = NULL;
38: int exit_status = EXIT_SUCCESS;
39: const char stdin_name[] = "-";
40:
41: struct options {
42:    bool lines;
43:    bool words;
44:    bool chars;
45:    int file_count;
46:    char **file_names;
47: };
48:
49: struct counts {
50:    size_t lines;
51:    size_t words;
52:    size_t chars;
53: };
54:
```

```
 55:
 56: void error (const char *format, ...) {
 57:    va_list fmt_args;
 58:    fflush (NULL);
 59:    assert (program_name != NULL);
 60:    fprintf (stderr, "%s: ", program_name);
 61:    va_start (fmt_args, format);
 62:    vfprintf (stderr, format, fmt_args);
 63:    va_end (fmt_args);
 64:    fflush (NULL);
 65:    exit_status = EXIT_FAILURE;
 66: }
 67:
 68: void scan_options (int argc, char **argv, struct options *opts) {
 69:    bool all_flags = true;
 70:    opts->chars = opts->words = opts->lines = false;
 71:    opterr = false;
 72:    for (;;) {
 73:       int flag = getopt (argc, argv, "cwl");
 74:       if (flag == EOF) break;
 75:       switch (flag) {
 76:          case 'c': opts->chars = true; all_flags = false; break;
 77:          case 'w': opts->words = true; all_flags = false; break;
 78:          case 'l': opts->lines = true; all_flags = false; break;
 79:          default : error ("-%c: invalid option", optopt); break;
 80:       }
 81:    }
 82:    if (all_flags) opts->chars = opts->words = opts->lines = true;
 83:    opts->file_count = argc - optind;
 84:    opts->file_names = &argv[optind];
 85: }
 86:
 87: bool is_plain_file (FILE *file, const char *filename) {
 88:    struct stat stat;
 89:    int rc = fstat (fileno (file), &stat);
 90:    if (rc != 0) {
 91:       error ("%s: %s\n", filename, strerror (errno));
 92:       return false;
 93:    }
 94:    if (S_ISREG (stat.st_mode)) return true;
 95:    const char *reason = S_ISDIR (stat.st_mode)
 96:                       ? "is a directory"
 97:                       : "is not a plain file";
 98:    error ("%s: %s\n", filename, reason);
 99:    return false;
100: }
101:
```

```
102:
103: void print_count (struct options *opts, struct counts *count,
104:                   const char *name) {
105:    if (opts->lines) printf ("%8zd", count->lines);
106:    if (opts->words) printf ("%8zd", count->words);
107:    if (opts->chars) printf ("%8zd", count->chars);
108:    if (name != NULL) printf (" %s", name);
109:    printf ("\n");
110: }
111:
112: void count_file (FILE *file, const char *filename,
113:                  struct options *opts, struct counts *totals) {
114:    if (! is_plain_file (file, filename)) return;
115:    struct counts file_counts = {0, 0, 0};
116:    bool spaces = true;
117:    for (;;) {
118:       int byte = fgetc (file);
119:       if (byte == EOF) break;
120:       ++file_counts.chars;
121:       if (byte == '\n') ++file_counts.lines;
122:       if (isspace (byte)) {
123:          spaces = true;
124:       }else if (spaces) {
125:          ++file_counts.words;
126:          spaces = false;
127:       }
128:    }
129:    print_count (opts, &file_counts, filename);
130:    totals->lines += file_counts.lines;
131:    totals->words += file_counts.words;
132:    totals->chars += file_counts.chars;
133: }
134:
```

```
135:
136: int main (int argc, char **argv) {
137:    program_name = basename (argv[0]);
138:    struct options opts = {false, false, false, 0, NULL};
139:    struct counts totals = {0, 0, 0};
140:    scan_options (argc, argv, &opts);
141:    if (opts.file_count == 0) {
142:       count_file (stdin, NULL, &opts, &totals);
143:    }else {
144:       for (int filenr = 0; filenr < opts.file_count; ++filenr) {
145:          char *filename = opts.file_names[filenr];
146:          if (strcmp (filename, stdin_name) == 0) {
147:             count_file (stdin, filename, &opts, &totals);
148:          }else {
149:             FILE *file = fopen (filename, "r");
150:             if (file == NULL) {
151:                error ("%s: %s", filename, strerror (errno));
152:             }else {
153:                count_file (file, filename, &opts, &totals);
154:                fclose (file);
155:             }
156:          }
157:       }
158:       if (opts.file_count > 1) print_count (&opts, &totals, "total");
159:    }
160:    return exit_status;
161: }
162:
163: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
164: //TEST// grind wordct *.c >wordct.out 2>&1
165: //TEST// mkpspdf wordct.ps wordct.c* wordct.out*
166:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting wordct.c
2: wordct.c:
3:      $Id: wordct.c,v 1.2 2014-04-22 19:38:08-07 - - $
4: gcc -g -O0 -Wall -Wextra -rdynamic -std=gnu11 wordct.c
5:         -o wordct -lglut -lGLU -lGL -lX11 -lrt -lm
6: rm -f wordct.o
7: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished wordct.c
```

```
    1: ==10747== Memcheck, a memory error detector
    2: ==10747== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
    3: ==10747== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
    4: ==10747== Command: wordct catbychar.c catbyline.c getoptex.c undefvar.c
wordct.c
    5: ==10747==
    6:       64     196    1561 catbychar.c
    7:       67     230    1795 catbyline.c
    8:       78     232    1936 getoptex.c
    9:       24      82     570 undefvar.c
   10:      166     623    4732 wordct.c
   11:      399    1363   10594 total
   12: ==10747==
   13: ==10747== HEAP SUMMARY:
   14: ==10747==     in use at exit: 0 bytes in 0 blocks
   15: ==10747==   total heap usage: 6 allocs, 6 frees, 2,856 bytes allocated
   16: ==10747==
   17: ==10747== All heap blocks were freed -- no leaks are possible
   18: ==10747==
   19: ==10747== For counts of detected and suppressed errors, rerun with: -v
   20: ==10747== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```