```
 1: // $Id: cppstrtok.cpp,v 1.6 2016-08-18 13:00:16-07 - - $
 2:
 3: // Use cpp to scan a file and print line numbers.
 4: // Print out each input line read in, then strtok it for
 5: // tokens.
 6:
 7: #include <string>
 8: using namespace std;
 9:
10: #include <errno.h>
11: #include <libgen.h>
12: #include <stdio.h>
13: #include <stdlib.h>
14: #include <string.h>
15: #include <wait.h>
16:
17: const string CPP = "/usr/bin/cpp";
18: constexpr size_t LINESIZE = 1024;
19:
20: // Chomp the last character from a buffer if it is delim.
21: void chomp (char* string, char delim) {
22:    size_t len = strlen (string);
23:    if (len == 0) return;
24:    char* nlpos = string + len - 1;
25:    if (*nlpos == delim) *nlpos = '\0';
26: }
27:
28: // Print the meaning of a signal.
29: static void eprint_signal (const char* kind, int signal) {
30:    fprintf (stderr, ", %s %d", kind, signal);
31:    const char* sigstr = strsignal (signal);
32:    if (sigstr != NULL) fprintf (stderr, " %s", sigstr);
33: }
34:
35: // Print the status returned from a subprocess.
36: void eprint_status (const char* command, int status) {
37:    if (status == 0) return;
38:    fprintf (stderr, "%s: status 0x%04X", command, status);
39:    if (WIFEXITED (status)) {
40:       fprintf (stderr, ", exit %d", WEXITSTATUS (status));
41:    }
42:    if (WIFSIGNALED (status)) {
43:       eprint_signal ("Terminated", WTERMSIG (status));
44:       #ifdef WCOREDUMP
45:       if (WCOREDUMP (status)) fprintf (stderr, ", core dumped");
46:       #endif
47:    }
48:    if (WIFSTOPPED (status)) {
49:       eprint_signal ("Stopped", WSTOPSIG (status));
50:    }
51:    if (WIFCONTINUED (status)) {
52:       fprintf (stderr, ", Continued");
53:    }
54:    fprintf (stderr, "\n");
55: }
56:
```

```
 57:
 58: // Run cpp against the lines of the file.
 59: void cpplines (FILE* pipe, char* filename) {
 60:    int linenr = 1;
 61:    char inputname[LINESIZE];
 62:    strcpy (inputname, filename);
 63:    for (;;) {
 64:       char buffer[LINESIZE];
 65:       char* fgets_rc = fgets (buffer, LINESIZE, pipe);
 66:       if (fgets_rc == NULL) break;
 67:       chomp (buffer, '\n');
 68:       printf ("%s:line %d: [%s]\n", filename, linenr, buffer);
 69:       // http://gcc.gnu.org/onlinedocs/cpp/Preprocessor-Output.html
 70:       int sscanf_rc = sscanf (buffer, "# %d \"%[^\"]\"",
 71:                               &linenr, filename);
 72:       if (sscanf_rc == 2) {
 73:          printf ("DIRECTIVE: line %d file \"%s\"\n", linenr, filename);
 74:          continue;
 75:       }
 76:       char* savepos = NULL;
 77:       char* bufptr = buffer;
 78:       for (int tokenct = 1;; ++tokenct) {
 79:          char* token = strtok_r (bufptr, " \t\n", &savepos);
 80:          bufptr = NULL;
 81:          if (token == NULL) break;
 82:          printf ("token %d.%d: [%s]\n",
 83:                  linenr, tokenct, token);
 84:       }
 85:       ++linenr;
 86:    }
 87: }
 88:
 89: int main (int argc, char** argv) {
 90:    const char* execname = basename (argv[0]);
 91:    int exit_status = EXIT_SUCCESS;
 92:    for (int argi = 1; argi < argc; ++argi) {
 93:       char* filename = argv[argi];
 94:       string command = CPP + " " + filename;
 95:       printf ("command=\"%s\"\n", command.c_str());
 96:       FILE* pipe = popen (command.c_str(), "r");
 97:       if (pipe == NULL) {
 98:          exit_status = EXIT_FAILURE;
 99:          fprintf (stderr, "%s: %s: %s\n",
100:                   execname, command.c_str(), strerror (errno));
101:       }else {
102:          cpplines (pipe, filename);
103:          int pclose_rc = pclose (pipe);
104:          eprint_status (command.c_str(), pclose_rc);
105:          if (pclose_rc != 0) exit_status = EXIT_FAILURE;
106:       }
107:    }
108:    return exit_status;
109: }
110:
```

```
 1: # $Id: Makefile,v 1.13 2016-08-18 13:01:32-07 - - $
 2:
 3: GCC        = g++ -g -O0 -Wall -Wextra -std=gnu++14
 4: MKDEP      = g++ -MM -std=gnu++14
 5: VALGRIND   = valgrind --leak-check=full --show-reachable=yes
 6:
 7: MKFILE     = Makefile
 8: DEPFILE    = Makefile.dep
 9: SOURCES    = cppstrtok.cpp
10: OBJECTS    = ${SOURCES:.cpp=.o}
11: EXECBIN    = cppstrtok
12: SRCFILES   = ${SOURCES} ${MKFILE}
13: SMALLFILES = ${DEPFILE} foo.oc foo1.oh foo2.oh
14: CHECKINS   = ${SRCFILES} ${SMALLFILES}
15: LISTING    = Listing.ps
16:
17: all : ${EXECBIN}
18:
19: ${EXECBIN} : ${OBJECTS}
20:         ${GCC} -o${EXECBIN} ${OBJECTS}
21:
22: %.o : %.cpp
23:         ${GCC} -c $<
24:
25: ci :
26:         cid + ${CHECKINS}
27:         checksource ${CHECKINS}
28:
29: clean :
30:         - rm ${OBJECTS}
31:
32: spotless : clean
33:         - rm ${EXECBIN} ${LISTING} ${LISTING:.ps=.pdf} ${DEPFILE} \
34:              test.out misc.lis
35:
36: ${DEPFILE} :
37:         ${MKDEP} ${SOURCES} >${DEPFILE}
38:
39: dep :
40:         - rm ${DEPFILE}
41:         ${MAKE} --no-print-directory ${DEPFILE}
42:
43: include Makefile.dep
44:
45: test : ${EXECBIN}
46:         ${VALGRIND} ${EXECBIN} foo.oc 1>test.out 2>&1
47:
48: misc.lis : ${DEPFILE} foo.oc foo1.oh foo2.oh
49:         catnv ${DEPFILE} foo.oc foo1.oh foo2.oh >misc.lis
50:
51: lis : misc.lis test
52:         mkpspdf ${LISTING} ${SRCFILES} misc.lis test.out
53:
54: again :
55:         ${MAKE} spotless dep all test lis
56:
```

```
  1: ::::::::::::::::::::::::::::::::
  2: Makefile.dep
  3: ::::::::::::::::::::::::::::::::
  4:      1  cppstrtok.o: cppstrtok.cpp
  5: ::::::::::::::::::::::::::::::::
  6: foo.oc
  7: ::::::::::::::::::::::::::::::::
  8:      1  line 1// $Id: foo.oc,v 1.3 2013-09-19 18:03:21-07 - - $
  9:      2  __FILE__ __LINE__ __DATE__ __TIME__
 10:      3  foo.oc, line 3.
 11:      4  #include "foo1.oh"
 12:      5  foo.oc, line 5.
 13:      6  #include "foo2.oh"
 14:      7  /* Comment */ on line 7
 15:      8  FOO1 + FOO2;
 16:      9  foo.oc, line 9, last line.
 17: ::::::::::::::::::::::::::::::::
 18: foo1.oh
 19: ::::::::::::::::::::::::::::::::
 20:      1  // $Id: foo1.oh,v 1.2 2011-09-29 19:06:34-07 - - $
 21:      2  __FILE__ __LINE__ __DATE__ __TIME__
 22:      3  foo1.h, line 3.
 23:      4  foo1.h, line 4.
 24:      5  // Comment.
 25:      6  foo1.h, line 6. /* Comment */ last line
 26:      7  #define FOO1 "foo1"
 27: ::::::::::::::::::::::::::::::::
 28: foo2.oh
 29: ::::::::::::::::::::::::::::::::
 30:      1  // $Id: foo2.oh,v 1.2 2011-09-29 19:06:34-07 - - $
 31:      2  __FILE__ __LINE__ __DATE__ __TIME__
 32:      3  foo2.h, line 3.
 33:      4  foo2.h, line 4.
 34:      5  // Comment.
 35:      6  foo2.h, line 6. /* Comment */ last line
 36:      7  #define FOO2 "foo2"
```

```
    1: ==23048== Memcheck, a memory error detector
    2: ==23048== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
    3: ==23048== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
    4: ==23048== Command: cppstrtok foo.oc
    5: ==23048==
    6: command="/usr/bin/cpp foo.oc"
    7: foo.oc:line 1: [# 1 "foo.oc"]
    8: DIRECTIVE: line 1 file "foo.oc"
    9: foo.oc:line 1: [# 1 "<built-in>"]
   10: DIRECTIVE: line 1 file "<built-in>"
   11: <built-in>:line 1: [# 1 "<command-line>"]
   12: DIRECTIVE: line 1 file "<command-line>"
   13: <command-line>:line 1: [# 1 "/usr/include/stdc-predef.h" 1 3 4]
   14: DIRECTIVE: line 1 file "/usr/include/stdc-predef.h"
   15: /usr/include/stdc-predef.h:line 1: [# 1 "<command-line>" 2]
   16: DIRECTIVE: line 1 file "<command-line>"
   17: <command-line>:line 1: [# 1 "foo.oc"]
   18: DIRECTIVE: line 1 file "foo.oc"
   19: foo.oc:line 1: [line 1]
   20: token 1.1: [line]
   21: token 1.2: [1]
   22: foo.oc:line 2: ["foo.oc" 2 "Sep 21 2016" "16:50:56"]
   23: token 2.1: ["foo.oc"]
   24: token 2.2: [2]
   25: token 2.3: ["Sep]
   26: token 2.4: [21]
   27: token 2.5: [2016"]
   28: token 2.6: ["16:50:56"]
   29: foo.oc:line 3: [foo.oc, line 3.]
   30: token 3.1: [foo.oc,]
   31: token 3.2: [line]
   32: token 3.3: [3.]
   33: foo.oc:line 4: [# 1 "foo1.oh" 1]
   34: DIRECTIVE: line 1 file "foo1.oh"
   35: foo1.oh:line 1: []
   36: foo1.oh:line 2: ["foo1.oh" 2 "Sep 21 2016" "16:50:56"]
   37: token 2.1: ["foo1.oh"]
   38: token 2.2: [2]
   39: token 2.3: ["Sep]
   40: token 2.4: [21]
   41: token 2.5: [2016"]
   42: token 2.6: ["16:50:56"]
   43: foo1.oh:line 3: [foo1.h, line 3.]
   44: token 3.1: [foo1.h,]
   45: token 3.2: [line]
   46: token 3.3: [3.]
   47: foo1.oh:line 4: [foo1.h, line 4.]
   48: token 4.1: [foo1.h,]
   49: token 4.2: [line]
   50: token 4.3: [4.]
   51: foo1.oh:line 5: []
   52: foo1.oh:line 6: [foo1.h, line 6. last line]
   53: token 6.1: [foo1.h,]
   54: token 6.2: [line]
   55: token 6.3: [6.]
   56: token 6.4: [last]
```

```
  57: token 6.5: [line]
  58: foo1.oh:line 7: [# 5 "foo.oc" 2]
  59: DIRECTIVE: line 5 file "foo.oc"
  60: foo.oc:line 5: [foo.oc, line 5.]
  61: token 5.1: [foo.oc,]
  62: token 5.2: [line]
  63: token 5.3: [5.]
  64: foo.oc:line 6: [# 1 "foo2.oh" 1]
  65: DIRECTIVE: line 1 file "foo2.oh"
  66: foo2.oh:line 1: []
  67: foo2.oh:line 2: ["foo2.oh" 2 "Sep 21 2016" "16:50:56"]
  68: token 2.1: ["foo2.oh"]
  69: token 2.2: [2]
  70: token 2.3: ["Sep]
  71: token 2.4: [21]
  72: token 2.5: [2016"]
  73: token 2.6: ["16:50:56"]
  74: foo2.oh:line 3: [foo2.h, line 3.]
  75: token 3.1: [foo2.h,]
  76: token 3.2: [line]
  77: token 3.3: [3.]
  78: foo2.oh:line 4: [foo2.h, line 4.]
  79: token 4.1: [foo2.h,]
  80: token 4.2: [line]
  81: token 4.3: [4.]
  82: foo2.oh:line 5: []
  83: foo2.oh:line 6: [foo2.h, line 6. last line]
  84: token 6.1: [foo2.h,]
  85: token 6.2: [line]
  86: token 6.3: [6.]
  87: token 6.4: [last]
  88: token 6.5: [line]
  89: foo2.oh:line 7: [# 7 "foo.oc" 2]
  90: DIRECTIVE: line 7 file "foo.oc"
  91: foo.oc:line 7: [              on line 7]
  92: token 7.1: [on]
  93: token 7.2: [line]
  94: token 7.3: [7]
  95: foo.oc:line 8: ["foo1" + "foo2";]
  96: token 8.1: ["foo1"]
  97: token 8.2: [+]
  98: token 8.3: ["foo2";]
  99: foo.oc:line 9: [foo.oc, line 9, last line.]
 100: token 9.1: [foo.oc,]
 101: token 9.2: [line]
 102: token 9.3: [9,]
 103: token 9.4: [last]
 104: token 9.5: [line.]
 105: ==23048==
 106: ==23048== HEAP SUMMARY:
 107: ==23048==     in use at exit: 0 bytes in 0 blocks
 108: ==23048==   total heap usage: 3 allocs, 3 frees, 342 bytes allocated
 109: ==23048==
 110: ==23048== All heap blocks were freed -- no leaks are possible
 111: ==23048==
 112: ==23048== For counts of detected and suppressed errors, rerun with: -v
 113: ==23048== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```