

```
1: // $Id: debugf.h,v 1.3 2013-06-07 13:41:07-07 - - $
2:
3: #ifndef __DEBUGF_H__
4: #define __DEBUGF_H__
5:
6: //
7: // DESCRIPTION
8: //     Debugging library containing miscellaneous useful things.
9: //
10:
11: //
12: // Keep track of Exec_Name and Exit_Status.
13: //
14: extern char *Exec_Name;
15: extern int Exit_Status;
16:
17: //
18: // Support for stub messages.
19: //
20: #define STUBPRINTF(...) \
21:     __stubprintf (__FILE__, __LINE__, __func__, __VA_ARGS__)
22: void __stubprintf (char *file, int line, const char *func,
23:                  char *format, ...);
24:
25: //
26: // Debugging utility.
27: //
28:
29: void set_debugflags (char *flags);
30: //
31: // Sets a string of debug flags to be used by DEBUGF statements.
32: // Uses the address of the string, and does not copy it, so it
33: // must not be dangling. If a particular debug flag has been set,
34: // messages are printed. The format is identical to printf format.
35: // The flag "@" turns on all flags.
36: //
37:
38: #ifdef NDEBUG
39: #define DEBUGF(FLAG,...) // DEBUG (FLAG, __VA_ARGS__)
40: #else
41: #define DEBUGF(FLAG,...) \
42:     __debugprintf (FLAG, __FILE__, __LINE__, __func__, __VA_ARGS__)
43: void __debugprintf (char flag, char *file, int line, const char *func,
44:                  char *format, ...);
45: #endif
46:
47: #endif
48:
```

```
1: // $Id: hashset.h,v 1.2 2013-03-04 20:49:14-08 - - $
2:
3: #ifndef __HASHSET_H__
4: #define __HASHSET_H__
5:
6: #include <stdbool.h>
7:
8: typedef struct hashset hashset;
9:
10: //
11: // Create a new hashset with a default number of elements.
12: //
13: hashset *new_hashset (void);
14:
15: //
16: // Frees the hashset, and the words it points at.
17: //
18: void free_hashset (hashset *);
19:
20: //
21: // Inserts a new string into the hashset.
22: //
23: void put_hashset (hashset *, char*);
24:
25: //
26: // Looks up the string in the hashset and returns true if found,
27: // false if not found.
28: //
29: bool has_hashset (hashset *, char*);
30:
31: #endif
32:
```

```
1: // $Id: strhash.h,v 1.1 2012-11-16 18:05:22-08 - - $
2:
3: //
4: // NAME
5: //     strhash - return an unsigned 32-bit hash code for a string
6: //
7: // SYNOPSIS
8: //     hashcode_t strhash (char *string);
9: //
10: // DESCRIPTION
11: //     Uses Horner's method to compute the hash code of a string
12: //     as is done by java.lang.String.hashCode:
13: //     . s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
14: //     Using strength reduction, the multiplication is replaced by
15: //     a shift. However, instead of returning a signed number,
16: //     this function returns an unsigned number.
17: //
18: // REFERENCE
19: //     http://java.sun.com/j2se/1.4.1/docs/api/java/lang/
20: //     String.html#hashCode()
21: //
22: //
23:
24: #ifndef __STRHASH_H__
25: #define __STRHASH_H__
26:
27: #include <inttypes.h>
28:
29: typedef uint32_t hashcode_t;
30:
31: hashcode_t strhash (char *string);
32:
33: #endif
34:
```

```
1: // $Id: yyextern.h,v 1.2 2013-05-21 19:58:24-07 - - $
2:
3: #ifndef __YYEXTERN_H__
4: #define __YYEXTERN_H__
5:
6: //
7: // DESCRIPTION
8: //   Definitions of external names used by flex-generated code.
9: //
10:
11: #include <stdio.h>
12:
13: extern FILE *yyin;           // File currently being read
14:
15: extern char *yytext;        // Pointer to the string that was found
16:
17: extern int yy_flex_debug;    // yylex's verbose tracing flag
18:
19: extern int yylex (void);     // Read next word from opened file yyin
20:
21: extern int yylineno;        // Line number within the current file
22:
23: extern int yylex_destroy (void);
24:                             // Cleans up flex's buffers when done
25:                             // Avoids valgrind memory leak report.
26:
27: #endif
28:
```

```
1: // $Id: debugf.c,v 1.4 2013-06-07 13:41:07-07 - - $
2:
3: #include <errno.h>
4: #include <stdarg.h>
5: #include <stdbool.h>
6: #include <stdio.h>
7: #include <stdlib.h>
8: #include <string.h>
9: #include <unistd.h>
10:
11: #include "debugf.h"
12:
13: char *Exec_Name = NULL;
14: int Exit_Status = EXIT_SUCCESS;
15:
16: static char *debugflags = "";
17: static bool alldebugflags = false;
18:
19: void __stubprintf (char *filename, int line, const char *func,
20:                  char *format, ...) {
21:     va_list args;
22:     fflush (NULL);
23:     fprintf (stdout, "%s: STUB (%s:%d) %s:\n",
24:             Exec_Name, filename, line, func);
25:     va_start (args, format);
26:     vfprintf (stdout, format, args);
27:     va_end (args);
28:     fflush (NULL);
29: }
30:
31: void set_debugflags (char *flags) {
32:     debugflags = flags;
33:     if (strchr (debugflags, '@') != NULL) alldebugflags = true;
34:     DEBUGF ('a', "Debugflags = \"%s\"\n", debugflags);
35: }
36:
37: void __debugprintf (char flag, char *file, int line, const char *func,
38:                   char *format, ...) {
39:     va_list args;
40:     if (alldebugflags || strchr (debugflags, flag) != NULL) {
41:         fflush (NULL);
42:         fprintf (stderr, "%s: DEBUGF(%c) %s (%s:%d):\n",
43:                 Exec_Name, flag, func, file, line);
44:         va_start (args, format);
45:         vfprintf (stderr, format, args);
46:         va_end (args);
47:         fflush (NULL);
48:     }
49: }
50:
```

```
1: // $Id: hashset.c,v 1.7 2013-05-23 15:41:07-07 - - $
2:
3: #include <assert.h>
4: #include <stdio.h>
5: #include <stdlib.h>
6: #include <string.h>
7:
8: #include "debugf.h"
9: #include "hashset.h"
10: #include "strhash.h"
11:
12: #define HASH_NEW_SIZE 15
13:
14: typedef struct hashnode hashnode;
15: struct hashnode {
16:     char *word;
17:     hashnode *link;
18: };
19:
20: struct hashset {
21:     size_t size;
22:     size_t load;
23:     hashnode **chains;
24: };
25:
26: hashset *new_hashset (void) {
27:     hashset *this = malloc (sizeof (struct hashset));
28:     assert (this != NULL);
29:     this->size = HASH_NEW_SIZE;
30:     this->load = 0;
31:     size_t sizeof_chains = this->size * sizeof (hashnode *);
32:     this->chains = malloc (sizeof_chains);
33:     assert (this->chains != NULL);
34:     memset (this->chains, 0, sizeof_chains);
35:     DEBUGF ('h', "%p -> struct hashset {size = %d, chains=%p}\n",
36:             this, this->size, this->chains);
37:     return this;
38: }
39:
40: void free_hashset (hashset *this) {
41:     DEBUGF ('h', "free (%p)\n", this);
42: }
43:
44: void put_hashset (hashset *this, char *item) {
45:     STUBPRINTF ("hashset=%p, item=%s\n", this, item);
46: }
47:
48: bool has_hashset (hashset *this, char *item) {
49:     STUBPRINTF ("hashset=%p, item=%s\n", this, item);
50:     return true;
51: }
52:
```

```
1: // $Id: strhash.c,v 1.3 2013-05-21 19:58:47-07 - - $
2:
3: #include <assert.h>
4: #include <stdio.h>
5: #include <sys/types.h>
6:
7: #include "strhash.h"
8:
9: hashcode_t strhash (char *string) {
10:     assert (string != NULL);
11:     hashcode_t hashcode = 0;
12:     for (; *string != '\0'; ++string) {
13:         hashcode = hashcode * 31 + *string;
14:     }
15:     return hashcode;
16: }
17:
```

```
1: // $Id: spellchk.c,v 1.6 2013-05-21 19:58:24-07 - - $
2:
3: #include <errno.h>
4: #include <libgen.h>
5: #include <stdio.h>
6: #include <stdlib.h>
7: #include <string.h>
8: #include <unistd.h>
9:
10: #include "debugf.h"
11: #include "hashset.h"
12: #include "yyextern.h"
13:
14: #define STDIN_NAME      "-"
15: #define DEFAULT_DICTNAME \
16:     "/afs/cats.ucsc.edu/courses/cmps012b-wm/usr/dict/words"
17: #define DEFAULT_DICT_POS 0
18: #define EXTRA_DICT_POS  1
19: #define NUMBER_DICTS     2
20:
21: void print_error (char *object, char *message) {
22:     fflush (NULL);
23:     fprintf (stderr, "%s: %s: %s\n", Exec_Name, object, message);
24:     fflush (NULL);
25:     Exit_Status = EXIT_FAILURE;
26: }
27:
28: FILE *open_infile (char *filename) {
29:     FILE *file = fopen (filename, "r");
30:     if (file == NULL) print_error (filename, strerror (errno));
31:     DEBUGF ('m', "filename = \"%s\\", file = 0x%p\\n", filename, file);
32:     return file;
33: }
34:
35: void spellcheck (char *filename, hashset *hashset) {
36:     yylineno = 1;
37:     DEBUGF ('m', "filename = \"%s\\", hashset = 0x%p\\n",
38:             filename, hashset);
39:     for (;;) {
40:         int token = yylex ();
41:         if (token == 0) break;
42:         DEBUGF ('m', "line %d, yytext = \"%s\\\"\\n", yylineno, yytext);
43:         STUBPRINTF ("%s: %d: %s\\n", filename, yylineno, yytext);
44:     }
45: }
46:
47: void load_dictionary (char *dictionary_name, hashset *hashset) {
48:     if (dictionary_name == NULL) return;
49:     DEBUGF ('m', "dictionary_name = \"%s\\", hashset = %p\\n",
50:             dictionary_name, hashset);
51:     STUBPRINTF ("Open dictionary, load it, close it\\n");
52: }
53:
```



```
54:
55: int main (int argc, char **argv) {
56:     Exec_Name = basename (argv[0]);
57:     char *default_dictionary = DEFAULT_DICTNAME;
58:     char *user_dictionary = NULL;
59:     hashset *hashset = new_hashset ();
60:     yy_flex_debug = false;
61:
62:     // Scan the arguments and set flags.
63:     opterr = false;
64:     for (;;) {
65:         int option = getopt (argc, argv, "nxyd:@:");
66:         if (option == EOF) break;
67:         switch (option) {
68:             char optopt_string[16]; // used in default:
69:             case 'd': user_dictionary = optarg;
70:                 break;
71:             case 'n': default_dictionary = NULL;
72:                 break;
73:             case 'x': STUBPRINTF ("-x\n");
74:                 break;
75:             case 'y': yy_flex_debug = true;
76:                 break;
77:             case '@': set_debugflags (optarg);
78:                 if (strpbrk (optarg, "@y")) yy_flex_debug = true;
79:                 break;
80:             default : sprintf (optopt_string, "-%c", optopt);
81:                 print_error (optopt_string, "invalid option");
82:                 break;
83:         }
84:     }
85:
86:     // Load the dictionaries into the hash table.
87:     load_dictionary (default_dictionary, hashset);
88:     load_dictionary (user_dictionary, hashset);
89:
90:     // Read and do spell checking on each of the files.
91:     if (optind >= argc) {
92:         yyin = stdin;
93:         spellcheck (STDIN_NAME, hashset);
94:     } else {
95:         for (int fileix = optind; fileix < argc; ++fileix) {
96:             DEBUGF ('m', "argv[%d] = \"%s\"\n", fileix, argv[fileix]);
97:             char *filename = argv[fileix];
98:             if (strcmp (filename, STDIN_NAME) == 0) {
99:                 yyin = stdin;
100:                 spellcheck (STDIN_NAME, hashset);
101:             } else {
102:                 yyin = open_infile (filename);
103:                 if (yyin == NULL) continue;
104:                 spellcheck (filename, hashset);
105:                 fclose (yyin);
106:             }
107:         }
108:     }
109:
110:     yylex_destroy ();
111:     return Exit_Status;
112: }
113:
```

```
1: %{
2: // $Id: scanner.l,v 1.3 2013-05-21 19:58:24-07 - - $
3:
4: #include <stdlib.h>
5:
6: #include "yyextern.h"
7:
8: %}
9:
10: %option 8bit
11: %option debug
12: %option ecs
13: %option interactive
14: %option nodefault
15: %option noyywrap
16: %option yylineno
17:
18: NUMBER  ([[digit:]]+([-:][[:digit:]]+)* )
19: WORD     ([[alnum:]]+([-&' ][[:alnum:]]+)* )
20: OTHER    (.|\\n)
21:
22: %%
23:
24: {NUMBER}      { }
25: {WORD}        { return 1; }
26: {OTHER}       { }
27:
28: %%
29:
```

```
1: # $Id: Makefile,v 1.4 2013-03-04 20:49:28-08 - - $
2:
3: MKFILE      = Makefile
4: DEPSFILE    = ${MKFILE}.deps
5: NOINCLUDE   = ci clean spotless
6: NEEDINCL    = ${filter ${NOINCLUDE}, ${MAKECMDGOALS}}
7: GMAKE       = gmake --no-print-directory
8:
9: GCC         = gcc -g -O0 -Wall -Wextra -std=gnu99
10: MKDEPS      = gcc -MM
11:
12: CSOURCE     = debugf.c hashset.c strhash.c spellchk.c
13: CHEADER     = debugf.h hashset.h strhash.h yyextern.h
14: OBJECTS     = ${CSOURCE:.c=.o} scanner.o
15: EXECBIN     = spellchk
16: SUBMITS     = ${CHEADER} ${CSOURCE} scanner.l ${MKFILE}
17: SOURCES     = ${SUBMITS}
18: LISTING     = Listing.code.ps
19: PROJECT     = cmps012b-wm.w13 asg4
20:
21: all : ${EXECBIN}
22:
23: ${EXECBIN} : ${OBJECTS}
24:      ${GCC} -o $@ ${OBJECTS}
25:
26: scanner.o : scanner.l
27:      flex -oscanner.c scanner.l
28:      gcc -g -O0 -std=gnu99 -c scanner.c
29:
30: %.o : %.c
31:      ${GCC} -c $<
32:
33: ci : ${SOURCES}
34:      cid + ${SOURCES}
35:      checksource ${SUBMITS}
36:
37: lis : ${SOURCES} ${DEPSFILE}
38:      mkpspdf ${LISTING} ${SOURCES} ${DEPSFILE}
39:
40: clean :
41:      - rm ${OBJECTS} ${DEPSFILE} core scanner.c ${EXECBIN}.errs
42:
43: spotless : clean
44:      - rm ${EXECBIN}
45:
46: submit : ${SUBMITS}
47:      submit ${PROJECT} ${SUBMITS}
48:
49: deps : ${CSOURCE} ${CHEADER}
50:      @ echo "# ${DEPSFILE} created `date`" >${DEPSFILE}
51:      ${MKDEPS} ${CSOURCE} >>${DEPSFILE}
52:
53: ${DEPSFILE} :
54:      @ touch ${DEPSFILE}
55:      ${GMAKE} deps
56:
57: again :
58:      ${GMAKE} spotless deps ci all lis
59:
60: ifeq "${NEEDINCL}" ""
61: include ${DEPSFILE}
62: endif
63:
```

```
1: # Makefile.deps created Fri Jun  7 13:41:07 PDT 2013
2: debugf.o: debugf.c debugf.h
3: hashset.o: hashset.c debugf.h hashset.h strhash.h
4: strhash.o: strhash.c strhash.h
5: spellchk.o: spellchk.c debugf.h hashset.h yyextern.h
```