```
 1: // $Id: oclib.oh,v 1.24 2011-11-18 17:34:29-08 - - $
 2:
 3: #ifndef __OCLIB_OH__
 4: #define __OCLIB_OH__
 5:
 6: #ifdef __OCLIB_C__
 7: #   define __(ID)          __##ID
 8: #   define NONE__          void
 9: #   define VOID__(ID)      void __##ID
10: #   define BOOL__(ID)      ubyte __##ID
11: #   define CHAR__(ID)      ubyte __##ID
12: #   define INT__(ID)       int __##ID
13: #   define STRING__(ID)    ubyte *__##ID
14: #   define STRINGS__(ID)   ubyte **__##ID
15: #   define null            0
16: #   define false           0
17: #   define true            1
18: typedef unsigned char ubyte;
19: void *xcalloc (int nelem, int size);
20: #else
21: #   define EOF             (-1)
22: #   define __(ID)          ID
23: #   define NONE__
24: #   define VOID__(ID)      void ID
25: #   define BOOL__(ID)      bool ID
26: #   define CHAR__(ID)      char ID
27: #   define INT__(ID)       int ID
28: #   define STRING__(ID)    string ID
29: #   define STRINGS__(ID)   string[] ID
30: VOID__(__assert_fail) (STRING__(expr), STRING__(file), INT__(line));
31: #endif
32:
33: VOID__(putb) (BOOL__(b));
34: VOID__(putc) (CHAR__(c));
35: VOID__(puti) (INT__(i));
36: VOID__(puts) (STRING__(s));
37: VOID__(endl) (NONE__);
38: INT__(getc) (NONE__);
39: STRING__(getw) (NONE__);
40: STRING__(getln) (NONE__);
41: STRINGS__ (getargv) (NONE__);
42: VOID__(exit) (int status);
43: #define assert(expr) \
44:         {if (! (expr)) __(__assert_fail) (#expr, __FILE__, __LINE__);}
45:
46: #endif
47:
```

```
 1: # 1 "oclib.oh"
 2: # 1 "<built-in>"
 3: # 1 "<command-line>"
 4: # 1 "oclib.oh"
 5: # 18 "oclib.oh"
 6: typedef unsigned char ubyte;
 7: void *xcalloc (int nelem, int size);
 8: # 33 "oclib.oh"
 9: void __putb (ubyte __b);
10: void __putc (ubyte __c);
11: void __puti (int __i);
12: void __puts (ubyte *__s);
13: void __endl (void);
14: int __getc (void);
15: ubyte *__getw (void);
16: ubyte *__getln (void);
17: ubyte **__getargv (void);
18: void __exit (int status);
```

```
# 1 "oclib.oh"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "oclib.oh"
# 30 "oclib.oh"
void __assert_fail (string expr, string file, int line);


void putb (bool b);
void putc (char c);
void puti (int i);
void puts (string s);
void endl ();
int getc ();
string getw ();
string getln ();
string[] getargv ();
void exit (int status);
```

```
 1: // $Id: oclib.c,v 1.45 2012-11-16 21:10:41-08 - - $
 2:
 3: #include <ctype.h>
 4: #include <libgen.h>
 5: #include <stdio.h>
 6: #include <stdlib.h>
 7: #include <string.h>
 8:
 9: #define __OCLIB_C__
10: #include "oclib.oh"
11:
12: ubyte **oc_argv;
13:
14: void ____assert_fail (char *expr, char *file, int line) {
15:    fflush (NULL);
16:    fprintf (stderr, "%s: %s:%d: assert (%s) failed.\n",
17:             basename ((char *) oc_argv[0]), file, line, expr);
18:    fflush (NULL);
19:    abort();
20: }
21:
22: void *xcalloc (int nelem, int size) {
23:    void *result = calloc (nelem, size);
24:    assert (result != NULL);
25:    return result;
26: }
27:
28: void __ocmain (void);
29: int main (int argc, char **argv) {
30:    argc = argc; // warning: unused parameter 'argc'
31:    oc_argv = (ubyte **) argv;
32:    __ocmain();
33:    return EXIT_SUCCESS;
34: }
35:
```

```
36:
37: ubyte *scan (int (*skipover) (int), int (*stopat) (int)) {
38:    int byte;
39:    do {
40:       byte = getchar();
41:       if (byte == EOF) return NULL;
42:    } while (skipover (byte));
43:    ubyte buffer[0x1000];
44:    ubyte *end = buffer;
45:    do {
46:       *end++ = byte;
47:       assert (end < buffer + sizeof buffer);
48:       *end = '\0';
49:       byte = getchar();
50:    }while (byte != EOF && ! stopat (byte));
51:    ubyte *result = (ubyte *) strdup ((char *) buffer);
52:    assert (result != NULL);
53:    return result;
54: }
55:
56: int isfalse (int byte)   { return 0 & byte; }
57: int isnl (int byte)      { return byte == '\n'; }
58: void __putb (ubyte byte) { printf ("%s", byte ? "true" : "false"); }
59: void __putc (ubyte byte) { printf ("%c", byte); }
60: void __puti (int val)    { printf ("%d", val); }
61: void __puts (ubyte *str) { printf ("%s", str); }
62: void __endl (void)       { printf ("%c", '\n'); fflush (NULL); }
63: int __getc (void)        { return getchar(); }
64: ubyte *__getw (void)     { return scan (isspace, isspace); }
65: ubyte *__getln (void)    { return scan (isfalse, isnl); }
66: ubyte **__getargv (void) { return oc_argv; }
67: void __exit (int status) { exit (status); }
68:
```

```
 1: // $Id: echoc.oc,v 1.1 2011-11-18 17:18:35-08 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: string[] argv = getargv ();
 6: int argi = 1;
 7: while (argv[argi] != null) {
 8:    if (argi > 1) putc (' ');
 9:    puts (argv[argi]);
10: }
11: endl ();
12:
```

```
 1: // $Id: facloop.oc,v 1.2 2011-11-18 17:18:35-08 - - $
 2: //
 3: // Function uses a loop to compute factorial.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fac (int n) {
 9:    int f = 1;
10:    while (n > 1) {
11:       f = f * n;
12:       n = n - 1;
13:    }
14:    return f;
15: }
16:
17: int n = 1;
18: while (n <= 5) {
19:    puti (fac (n));
20: }
21:
```

```
 1: // $Id: facrec.oc,v 1.1 2011-11-18 17:18:35-08 - - $
 2: //
 3: // Function uses a loop to compute factorial.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fac (int n) {
 9:    if (n < 2) return 1;
10:    return n * fac (n - 1);
11: }
12:
13: int n = 1;
14: while (n <= 5) {
15:    puti (fac (n));
16:    endl ();
17:    n = n + 1;
18: }
19:
```

```
 1: /*
 2:      1  // $Id: echoc.oil,v 1.1 2012-10-04 19:14:40-07 - - $
 3:      2
 4:      3  #include "oclib.oh"
 5:      4
 6:      5  string[] argv = getargv ();
 7:      6  int argi = 1;
 8:      7  while (argv[argi] != null) {
 9:      8     if (argi > 1) putc (' ');
10:      9     puts (argv[argi]);
11:     10     argi = argi + 1;
12:     11  }
13:     12  endl ();
14:     13
15: */
16:
17: #define __OCLIB_C__
18: #include "oclib.oh"
19:
20: ubyte **__argv;
21: int __argi;
22:
23: void __ocmain ()
24: {
25:         ubyte **p1 = __getargv ();
26:         __argv = p1;
27:         __argi = 1;
28: while_5_7_0:;
29:         ubyte *p2 = __argv[__argi];
30:         ubyte b3 = p2 != 0;
31:         if (!b3) goto break_5_7_0;
32:         ubyte b4 = __argi > 1;
33:         if (!b4) goto fi_5_8_3;
34:         __putc (' ');
35: fi_5_8_3:;
36:         ubyte *p4 = __argv[__argi];
37:         __puts (p4);
38:         int i5 = __argi + 1;
39:         __argi = i5;
40:         goto while_5_7_0;
41: break_5_7_0:;
42:         __endl ();
43: }
44:
```

```
 1: /*
 2:      1  // $Id: facloop.oil,v 1.1 2012-10-04 19:14:40-07 - - $
 3:      2  //
 4:      3  // Function uses a loop to compute factorial.
 5:      4  //
 6:      5
 7:      6  #include "oclib.oh"
 8:      7
 9:      8  int fac (int n) {
10:      9     int f = 1;
11:     10     while (n > 1) {
12:     11        f = f * n;
13:     12        n = n - 1;
14:     13     }
15:     14     return f;
16:     15  }
17:     16
18:     17  int n = 1;
19:     18  while (n <= 5) {
20:     19     puti (fac (n));
21:     20     endl ();
22:     21     n = n + 1;
23:     22  }
24:     23
25: */
26:
27: #define __OCLIB_C__
28: #include "oclib.oh"
29:
30: int __n;
31:
32: int __fac (
33:         int _1_n)
34: {
35:         int _2_f = 1;
36: while_5_10_3:;
37:         ubyte b1 = _1_n > 1;
38:         if (!b1) goto break_5_10_3;
39:         int i2 = _2_f * _1_n;
40:         _2_f = i2;
41:         int i3 = _1_n - 1;
42:         _1_n = i3;
43:         goto while_5_10_3;
44: break_5_10_3:;
45:         return _2_f;
46: }
47:
48: void __ocmain ()
49: {
50:         __n = 1;
51: while_5_18_0:;
52:         ubyte b4 = __n <= 5;
53:         if (!b4) goto break_5_18_0;
54:         int i5 = __fac (__n);
55:         __puti (i5);
56:         __endl ();
57:         int i6 = __n + 1;
58:         __n = i6;
59:         goto while_5_18_0;
60: break_5_18_0:;
61: }
62:
```

```
 1: /*
 2:      1  // $Id: facrec.oil,v 1.1 2012-10-04 19:14:40-07 - - $
 3:      2  //
 4:      3  // Function uses a loop to compute factorial.
 5:      4  //
 6:      5
 7:      6  #include "oclib.oh"
 8:      7
 9:      8  int fac (int n) {
10:      9     if (n < 2) return 1;
11:     10     return n * fac (n - 1);
12:     11  }
13:     12
14:     13  int n = 1;
15:     14  while (n <= 5) {
16:     15     puti (fac (n));
17:     16     endl ();
18:     17     n = n + 1;
19:     16  }
20:     17
21: */
22:
23: #define __OCLIB_C__
24: #include "oclib.oh"
25:
26: int __n;
27:
28: int __fac (
29:         int _1_n)
30: {
31:         ubyte b1 = _1_n < 2;
32:         if (!b1) goto fi_5_9_3;
33:         return 1;
34: fi_5_9_3:;
35:         int i2 = _1_n - 1;
36:         int i3 = __fac (i2);
37:         int i4 = _1_n * i3;
38:         return i4;
39: }
40:
41: void __ocmain ()
42: {
43:         __n = 1;
44: while_5_14_0:;
45:         ubyte b5 = __n <= 5;
46:         if (!b5) goto break_5_14_0;
47:         int i6 = __fac (__n);
48:         __puti (i6);
49:         __endl ();
50:         int i7 = __n + 1;
51:         __n = i7;
52:         goto while_5_14_0;
53: break_5_14_0:;
54: }
```

```
 1: /*
 2:      8  int strcmp (string s1, string s2) {
 3:      9     int index = 0;
 4:     10     bool contin = true;
 5:     11     while (contin) {
 6:     12        char s1c = s1[index];
 7:     13        char s2c = s2[index];
 8:     14        int cmp = ord s1c - ord s2c;
 9:     15        if (cmp != 0) return cmp;
10:     16        if (s1c == '\0') contin = false;
11:     17        index = index + 1;
12:     18     }
13:     19     return 0;
14:     20  }
15: */
16:
17: #define __OCLIB_C__
18: #include "oclib.oh"
19:
20: int __strcmp (
21:         ubyte *_1_s1,
22:         ubyte *_1_s2)
23: {
24:         int _2_index = 0;
25:         ubyte _2_contin = 1;
26: while_5_11_3:;
27:         if (!_2_contin) goto break_5_11_3;
28:         ubyte b1 = _1_s1[_2_index];
29:         ubyte _2_s1c = b1;
30:         ubyte b2 = _1_s2[_2_index];
31:         ubyte _2_s2c = b2;
32:         int i3 = (int) _2_s1c;
33:         int i4 = (int) _2_s2c;
34:         int i5 = i3 - i4;
35:         int _2_cmp = i5;
36:         ubyte b6 = _2_cmp != 0;
37:         if (!b6) goto fi_5_15_6;
38:         return _2_cmp;
39: fi_5_15_6:;
40:         ubyte b7 = _2_s1c == '\0';
41:         if (!b7) goto fi_5_16_6;
42:         _2_contin = 0;
43: fi_5_16_6:;
44:         int i8 = _2_index + 1;
45:         _2_index = i8;
46:         goto while_5_11_3;
47: break_5_11_3:;
48:         return 0;
49: }
50:
```