```
 1: // $Id: prthexaddr.c,v 1.6 2014-02-07 17:12:18-08 - - $
 2:
 3: #include <assert.h>
 4: #include <errno.h>
 5: #include <stdio.h>
 6: #include <stdlib.h>
 7: #include <string.h>
 8: #include <sys/utsname.h>
 9:
10: #define PRINT(SYMBOL,DESCR) { \
11:         printf ("%16p: %s - %s\n", SYMBOL, #SYMBOL, DESCR); \
12:         }
13:
14: extern char _start;
15: extern char _etext;
16: extern char _edata;
17: extern char _end;
18: extern char **environ;
19: static double init_var[] = {
20:    3.14159265358979323846264338327950288419716939937,
21:    2.71828182845904523536028747135266249775724709369,
22:    0.30102999566398119521373889472449302676818988146,
23:    1.41421356237309504880168872420969807856967187537,
24: };
25: static int uninit_var1[1<<10];
26: static int uninit_var2[1<<10];
27:
28: char *fmt (char *text, int value) {
29:    char *buffer = malloc (strlen (text) + 16);
30:    sprintf (buffer, "%s %d", text, value);
31:    return buffer;
32: }
33:
34: void stack (int level) {
35:    if (level < 5) stack (level + 1);
36:    char *message = fmt ("address of a stack variable at level", level);
37:    PRINT (&level, message);
38:    free (message);
39: }
40:
41: void *stack_bottom (char **start) {
42:    for (; *start != NULL; ++start) {}
43:    --start;
44:    char *startstr = *start;
45:    while (*startstr != '\0') ++startstr;
46:    return startstr;
47: }
48:
```

```
 49:
 50: void print_uname (void) {
 51:    struct utsname name;
 52:    int rc = uname (&name);
 53:    if (rc < 0) {
 54:       printf ("uname: %s\n", strerror (errno));
 55:       return;
 56:    }
 57:    printf ("sysname  = \"%s\"\n", name.sysname );
 58:    printf ("nodename = \"%s\"\n", name.nodename);
 59:    printf ("release  = \"%s\"\n", name.release );
 60:    printf ("version  = \"%s\"\n", name.version );
 61:    printf ("machine  = \"%s\"\n", name.machine );
 62: }
 63:
 64: int main (int argc, char **argv) {
 65:    print_uname ();
 66:    int main_local;
 67:    printf ("\n");
 68:    PRINT (NULL, "null pointer");
 69:
 70:    printf ("\nAddresses of some local variables:\n");
 71:    stack (1);
 72:    PRINT (&main_local, "address of a local variable in main");
 73:    PRINT (&argc, "address of argc");
 74:    PRINT (&argv, "address of argv");
 75:    PRINT (argv, "address of arg vector");
 76:    PRINT (environ, "address of environ vector");
 77:    for (int argi = 0; argi < argc; ++argi) {
 78:       printf ("%16p: argv[%2d] = \"%s\"\n",
 79:               argv[argi], argi, argv[argi]);
 80:    }
 81:    PRINT (stack_bottom (environ), "byte at bottom of stack");
 82:
 83:    printf ("\nAddresses of some static variables:\n");
 84:    PRINT (printf, "(text) address of the printf() function");
 85:    PRINT (&_start, "start of program text");
 86:    PRINT (main, "(text) address of the main() function");
 87:    PRINT (&_etext, "end of program text");
 88:    PRINT (&init_var, "address of an init static variable");
 89:    PRINT (&_edata, "end of init data segment");
 90:    PRINT (&uninit_var1, "address of an uninit static variable1");
 91:    PRINT (&uninit_var2, "address of an uninit static variable2");
 92:    PRINT (&_end, "end of uninit data segment");
 93:
 94:    printf ("\nAddresses of some heap variables:\n");
 95:    for (int heap_count = 0; heap_count < 10; ++heap_count) {
 96:       void *heap_variable = calloc (1000, sizeof (int));
 97:       assert (heap_variable != NULL);
 98:       char *message = fmt ("heap variable ", heap_count);
 99:       PRINT (heap_variable, message);
100:       free (message);
101:    }
102:
103:    return EXIT_SUCCESS;
104: }
105:
106: //TEST// ./prthexaddr hello world >prthexaddr.list
```

```
107: //TEST// mkpspdf prthexaddr.ps prthexaddr.c* prthexaddr.lis*
108:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting prthexaddr.c
2: prthexaddr.c:
3:      $Id: prthexaddr.c,v 1.6 2014-02-07 17:12:18-08 - - $
4: gcc -g -O0 -Wall -Wextra -rdynamic -std=gnu11 prthexaddr.c
5:          -o prthexaddr -lglut -lGLU -lGL -lX11 -lrt -lm
6: rm -f prthexaddr.o
7: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished prthexaddr.c
```

```
 1: sysname  = "Linux"
 2: nodename = "unix3.lt.ucsc.edu"
 3: release  = "3.10.0-327.4.5.el7.x86_64"
 4: version  = "#1 SMP Mon Jan 25 22:07:14 UTC 2016"
 5: machine  = "x86_64"
 6:
 7:            (nil): NULL - null pointer
 8:
 9: Addresses of some local variables:
10:   0x7ffd07cc7a0c: &level - address of a stack variable at level 5
11:   0x7ffd07cc7a3c: &level - address of a stack variable at level 4
12:   0x7ffd07cc7a6c: &level - address of a stack variable at level 3
13:   0x7ffd07cc7a9c: &level - address of a stack variable at level 2
14:   0x7ffd07cc7acc: &level - address of a stack variable at level 1
15:   0x7ffd07cc7b04: &main_local - address of a local variable in main
16:   0x7ffd07cc7afc: &argc - address of argc
17:   0x7ffd07cc7af0: &argv - address of argv
18:   0x7ffd07cc7c08: argv - address of arg vector
19:   0x7ffd07cc7c28: environ - address of environ vector
20:   0x7ffd07cc9b1a: argv[ 0] = "./prthexaddr"
21:   0x7ffd07cc9b27: argv[ 1] = "hello"
22:   0x7ffd07cc9b2d: argv[ 2] = "world"
23:   0x7ffd07ccafea: stack_bottom (environ) - byte at bottom of stack
24:
25: Addresses of some static variables:
26:         0x400ac0: printf - (text) address of the printf() function
27:         0x400b90: &_start - start of program text
28:         0x400e57: main - (text) address of the main() function
29:         0x4011fd: &_etext - end of program text
30:         0x6020c0: &init_var - address of an init static variable
31:         0x6020e0: &_edata - end of init data segment
32:         0x602140: &uninit_var1 - address of an uninit static variable1
33:         0x603140: &uninit_var2 - address of an uninit static variable2
34:         0x604140: &_end - end of uninit data segment
35:
36: Addresses of some heap variables:
37:        0x1e66030: heap_variable - heap variable  0
38:        0x1e66fe0: heap_variable - heap variable  1
39:        0x1e67f90: heap_variable - heap variable  2
40:        0x1e68f40: heap_variable - heap variable  3
41:        0x1e69ef0: heap_variable - heap variable  4
42:        0x1e6aea0: heap_variable - heap variable  5
43:        0x1e6be50: heap_variable - heap variable  6
44:        0x1e6ce00: heap_variable - heap variable  7
45:        0x1e6ddb0: heap_variable - heap variable  8
46:        0x1e6ed60: heap_variable - heap variable  9
```