```
 1: // $Id: prthexaddr.c,v 1.4 2012-02-09 19:05:55-08 - - $
 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", (void*) 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:    3.14159265358979323846264338327950288419716939 9,
21:    2.71828182845904523536028747135266249775724709 3,
22:    0.30102999566398119521373889472449302676818988 1,
23:    1.41421356237309504880168872420969807856967187 5,
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: void print_uname (void) {
50:    struct utsname name;
51:    int rc = uname (&name);
52:    if (rc < 0) {
53:       printf ("uname: %s\n", strerror (errno));
54:       return;
55:    }
56:    printf ("sysname  = \"%s\"\n", name.sysname );
57:    printf ("nodename = \"%s\"\n", name.nodename);
58:    printf ("release  = \"%s\"\n", name.release );
59:    printf ("version  = \"%s\"\n", name.version );
60:    printf ("machine  = \"%s\"\n", name.machine );
61: }
```

```
 62:
 63: int main (int argc, char **argv) {
 64:    print_uname ();
 65:    int main_local;
 66:    printf ("\n");
 67:    PRINT (NULL, "null pointer");
 68:
 69:    printf ("\nAddresses of some local variables:\n");
 70:    stack (1);
 71:    PRINT (&main_local, "address of a local variable in main");
 72:    PRINT (&argc, "address of argc");
 73:    PRINT (&argv, "address of argv");
 74:    PRINT (argv, "address of arg vector");
 75:    PRINT (environ, "address of environ vector");
 76:    for (int argi = 0; argi < argc; ++argi) {
 77:       printf ("%16p: argv[%2d] = \"%s\"\n",
 78:               argv[argi], argi, argv[argi]);
 79:    }
 80:    PRINT (stack_bottom (environ), "byte at bottom of stack");
 81:
 82:    printf ("\nAddresses of some static variables:\n");
 83:    PRINT (printf, "(text) address of the printf() function");
 84:    PRINT (&_start, "start of program text");
 85:    PRINT (main, "(text) address of the main() function");
 86:    PRINT (&_etext, "end of program text");
 87:    PRINT (&init_var, "address of an init static variable");
 88:    PRINT (&_edata, "end of init data segment");
 89:    PRINT (&uninit_var1, "address of an uninit static variable1");
 90:    PRINT (&uninit_var2, "address of an uninit static variable2");
 91:    PRINT (&_end, "end of uninit data segment");
 92:
 93:    printf ("\nAddresses of some heap variables:\n");
 94:    for (int heap_count = 0; heap_count < 10; ++heap_count) {
 95:       void *heap_variable = calloc (1000, sizeof (int));
 96:       assert (heap_variable != NULL);
 97:       char *message = fmt ("heap variable ", heap_count);
 98:       PRINT (heap_variable, message);
 99:       free (message);
100:    }
101:
102:    return EXIT_SUCCESS;
103: }
104:
105: //TEST// ./prthexaddr hello world >prthexaddr.list
106: //TEST// mkpspdf prthexaddr.ps prthexaddr.c* prthexaddr.lis*
107:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting prthexaddr.c
2: prthexaddr.c: $Id: prthexaddr.c,v 1.4 2012-02-09 19:05:55-08 - - $
3: gcc -g -O0 -Wall -Wextra -std=gnu99 prthexaddr.c -o prthexaddr -lm
4: rm -f prthexaddr.o
5: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished prthexaddr.c
```

```
 1: sysname  = "Linux"
 2: nodename = "unix2.ic.ucsc.edu"
 3: release  = "2.6.32-220.23.1.el6.x86_64"
 4: version  = "#1 SMP Mon Jun 18 18:58:52 BST 2012"
 5: machine  = "x86_64"
 6:
 7:            (nil): NULL - null pointer
 8:
 9: Addresses of some local variables:
10:   0x7fff2f21b60c: &level - address of a stack variable at level 5
11:   0x7fff2f21b64c: &level - address of a stack variable at level 4
12:   0x7fff2f21b68c: &level - address of a stack variable at level 3
13:   0x7fff2f21b6cc: &level - address of a stack variable at level 2
14:   0x7fff2f21b70c: &level - address of a stack variable at level 1
15:   0x7fff2f21b754: &main_local - address of a local variable in main
16:   0x7fff2f21b74c: &argc - address of argc
17:   0x7fff2f21b740: &argv - address of argv
18:   0x7fff2f21b868: argv - address of arg vector
19:   0x7fff2f21b888: environ - address of environ vector
20:   0x7fff2f21d1cd: argv[ 0] = "./prthexaddr"
21:   0x7fff2f21d1da: argv[ 1] = "hello"
22:   0x7fff2f21d1e0: argv[ 2] = "world"
23:   0x7fff2f21dfea: stack_bottom (environ) - byte at bottom of stack
24:
25: Addresses of some static variables:
26:         0x4006f8: printf - (text) address of the printf() function
27:         0x4007d0: &_start - start of program text
28:         0x400aaf: main - (text) address of the main() function
29:         0x400f06: &_etext - end of program text
30:         0x6016c0: &init_var - address of an init static variable
31:         0x6016e0: &_edata - end of init data segment
32:         0x601700: &uninit_var1 - address of an uninit static variable1
33:         0x602700: &uninit_var2 - address of an uninit static variable2
34:         0x603700: &_end - end of uninit data segment
35:
36: Addresses of some heap variables:
37:         0x856010: heap_variable - heap variable  0
38:         0x856fc0: heap_variable - heap variable  1
39:         0x857f70: heap_variable - heap variable  2
40:         0x858f20: heap_variable - heap variable  3
41:         0x859ed0: heap_variable - heap variable  4
42:         0x85ae80: heap_variable - heap variable  5
43:         0x85be30: heap_variable - heap variable  6
44:         0x85cde0: heap_variable - heap variable  7
45:         0x85dd90: heap_variable - heap variable  8
46:         0x85ed40: heap_variable - heap variable  9
```