```
 1: // $Id: addresses.cc,v 1.3 2013-09-24 18:49:27-07 - - $
 2:
 3: #include <assert.h>
 4: #include <errno.h>
 5: #include <inttypes.h>
 6: #include <stdio.h>
 7: #include <stdlib.h>
 8: #include <string.h>
 9: #include <sys/utsname.h>
10:
11: #define PRINT(SYMBOL,DESCR) { \
12:         printf ("%16p: %s - %s\n", (void*) SYMBOL, #SYMBOL, DESCR); \
13:         }
14:
15: extern char _start;
16: extern char _etext;
17: extern char _edata;
18: extern char _end;
19: extern char **environ;
20: static double init_var[] = {
21:    3.14159265358979323846264338327950288419716939937,
22:    2.71828182845904523536028747135266249775724709369,
23:    0.30102999566398119521373889472449302676818988146,
24:    1.41421356237309504880168872420969807856967187537,
25: };
26: static int uninit_var1[1<<10];
27: static int uninit_var2[1<<10];
28:
29: char *fmt (char *text, int value) {
30:    char *buffer = malloc (strlen (text) + 16);
31:    sprintf (buffer, "%s %d", text, value);
32:    return buffer;
33: }
34:
35: void stack (int level) {
36:    if (level < 5) stack (level + 1);
37:    char *message = fmt ("address of a stack variable at level", level);
38:    PRINT (&level, message);
39:    free (message);
40: }
41:
42: void *stack_bottom (char **start) {
43:    for (; *start != NULL; ++start) {}
44:    --start;
45:    char *startstr = *start;
46:    while (*startstr != '\0') ++startstr;
47:    return startstr;
48: }
49:
```

```
 50:
 51: void print_uname (void) {
 52:    struct utsname name;
 53:    int rc = uname (&name);
 54:    if (rc < 0) {
 55:       printf ("uname: %s\n", strerror (errno));
 56:       return;
 57:    }
 58:    printf ("sysname  = \"%s\"\n", name.sysname );
 59:    printf ("nodename = \"%s\"\n", name.nodename);
 60:    printf ("release  = \"%s\"\n", name.release );
 61:    printf ("version  = \"%s\"\n", name.version );
 62:    printf ("machine  = \"%s\"\n", name.machine );
 63: }
 64:
 65: int main (int argc, char **argv) {
 66:    print_uname ();
 67:    printf ("sizeof (char**) = %ld\n", sizeof (char**));
 68:    printf ("sizeof (uintptr_t) = %ld, (uintptr_t) argv = %ld\n",
 69:            sizeof (uintptr_t), (uintptr_t) argv);
 70:    int main_local;
 71:    printf ("\n");
 72:    PRINT (NULL, "null pointer");
 73:
 74:    printf ("\nAddresses of some stack variables:\n");
 75:    stack (1);
 76:    PRINT (&main_local, "address of a local variable in main");
 77:    PRINT (&argc, "address of argc");
 78:    PRINT (&argv, "address of argv");
 79:    PRINT (argv, "address of arg vector");
 80:    PRINT (environ, "address of environ vector");
 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 = new int[1000];
 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// ./addresses >addresses.out 2>&1
107: //TEST// mkpspdf addresses.ps addresses.cc* addresses.out
108:
```

```
1: * @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting addresses.cc
2: * addresses.cc: $Id: addresses.cc,v 1.3 2013-09-24 18:49:27-07 - - $
3: * g++ -g -O0 -Wall -Wextra -std=gnu++0x addresses.cc -o addresses -lm
```

```
 1: sysname  = "Linux"
 2: nodename = "unix1.lt.ucsc.edu"
 3: release  = "2.6.32-358.18.1.el6.x86_64"
 4: version  = "#1 SMP Wed Aug 28 17:19:38 UTC 2013"
 5: machine  = "x86_64"
 6: sizeof (char**) = 8
 7: sizeof (uintptr_t) = 8, (uintptr_t) argv = 140734431517832
 8:
 9:            (nil): NULL - null pointer
10:
11: Addresses of some local variables:
12:   0x7fff49cc5a8c: &level - address of a stack variable at level 5
13:   0x7fff49cc5abc: &level - address of a stack variable at level 4
14:   0x7fff49cc5aec: &level - address of a stack variable at level 3
15:   0x7fff49cc5b1c: &level - address of a stack variable at level 2
16:   0x7fff49cc5b4c: &level - address of a stack variable at level 1
17:   0x7fff49cc5b88: &main_local - address of a local variable in main
18:   0x7fff49cc5b7c: &argc - address of argc
19:   0x7fff49cc5b70: &argv - address of argv
20:   0x7fff49cc5c88: argv - address of arg vector
21:   0x7fff49cc5c98: environ - address of environ vector
22:   0x7fff49cc6feb: stack_bottom (environ) - byte at bottom of stack
23:
24: Addresses of some static variables:
25:         0x400668: printf - (text) address of the printf() function
26:         0x400730: &_start - start of program text
27:         0x4009ce: main - (text) address of the main() function
28:         0x400d96: &_etext - end of program text
29:         0x601560: &init_var - address of an init static variable
30:         0x601580: &_edata - end of init data segment
31:         0x6015a0: &uninit_var1 - address of an uninit static variable1
32:         0x6025a0: &uninit_var2 - address of an uninit static variable2
33:         0x6035a0: &_end - end of uninit data segment
34:
35: Addresses of some heap variables:
36:         0xe78010: heap_variable - heap variable  0
37:         0xe78fc0: heap_variable - heap variable  1
38:         0xe79f70: heap_variable - heap variable  2
39:         0xe7af20: heap_variable - heap variable  3
40:         0xe7bed0: heap_variable - heap variable  4
41:         0xe7ce80: heap_variable - heap variable  5
42:         0xe7de30: heap_variable - heap variable  6
43:         0xe7ede0: heap_variable - heap variable  7
44:         0xe7fd90: heap_variable - heap variable  8
45:         0xe80d40: heap_variable - heap variable  9
```