```
 1: // $Id: addresses.c,v 1.8 2016-08-18 14:13:59-07 - - $
 2:
 3: #include <assert.h>
 4: #include <errno.h>
 5: #include <stdint.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", \
13:                    (void*) SYMBOL, #SYMBOL, DESCR); \
14:         }
15:
16: extern char _start;
17: extern char _etext;
18: extern char _edata;
19: extern char _end;
20: extern char** environ;
21: static double init_var[] = {
22:    3.14159265358979323846264338327950288419716939937,
23:    2.71828182845904523536028747135266249775724709369,
24:    0.30102999566398119521373889472449302676818988146,
25:    1.41421356237309504880168872420969807856967187537,
26: };
27: static int uninit_var1[1<<10];
28: static int uninit_var2[1<<10];
29:
30: char* fmt (char* text, int value) {
31:    char* buffer = malloc (strlen (text) + 16);
32:    sprintf (buffer, "%s %d", text, value);
33:    return buffer;
34: }
35:
36: void stack (int level) {
37:    if (level < 5) stack (level + 1);
38:    char* message = fmt ("address of a stack variable at level", level);
39:    PRINT (&level, message);
40:    free (message);
41: }
42:
43: void* stack_bottom (char** start) {
44:    for (; *start != NULL; ++start) {}
45:    --start;
46:    char* startstr = *start;
47:    while (*startstr != '\0') ++startstr;
48:    return startstr;
49: }
50:
```

```
 51:
 52: void print_uname (void) {
 53:    struct utsname name;
 54:    int rc = uname (&name);
 55:    if (rc < 0) {
 56:       printf ("uname: %s\n", strerror (errno));
 57:       return;
 58:    }
 59:    printf ("sysname  = \"%s\"\n", name.sysname );
 60:    printf ("nodename = \"%s\"\n", name.nodename);
 61:    printf ("release  = \"%s\"\n", name.release );
 62:    printf ("version  = \"%s\"\n", name.version );
 63:    printf ("machine  = \"%s\"\n", name.machine );
 64: }
 65:
 66: int main (int argc, char** argv) {
 67:    print_uname ();
 68:    printf ("sizeof (char**) = %ld\n", sizeof (char**));
 69:    printf ("sizeof (uintptr_t) = %ld, (uintptr_t) argv = %ld\n",
 70:            sizeof (uintptr_t), (uintptr_t) argv);
 71:    int main_local;
 72:    PRINT (NULL, "NULL");
 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 = malloc (1<<12);
 97:       assert (heap_variable != NULL);
 98:       char* message = fmt ("heap variable ", heap_count);
 99:       PRINT (heap_variable, message);
100:       free (message);
101:    }
102:    return EXIT_SUCCESS;
103: }
104:
105: //TEST// ./addresses >addresses.out 2>&1
106: //TEST// mkpspdf addresses.ps addresses.c* addresses.out
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting addresses.c
    2: addresses.c: $Id: addresses.c,v 1.8 2016-08-18 14:13:59-07 - - $
    3: gcc -g -O0 -std=gnu11 -rdynamic -Wall -Wextra addresses.c -o addresses -
lglut -lGLU -lGL -lX11 -lrt -lm
    4: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished addresses.c
```

```
 1: sysname  = "Linux"
 2: nodename = "unix1.lt.ucsc.edu"
 3: release  = "3.10.0-327.22.2.el7.x86_64"
 4: version  = "#1 SMP Thu Jun 23 17:05:11 UTC 2016"
 5: machine  = "x86_64"
 6: sizeof (char**) = 8
 7: sizeof (uintptr_t) = 8, (uintptr_t) argv = 140725506758248
 8:            (nil): NULL: NULL
 9:
10: Addresses of some stack variables:
11:   0x7ffd35d7546c: &level: address of a stack variable at level 5
12:   0x7ffd35d7549c: &level: address of a stack variable at level 4
13:   0x7ffd35d754cc: &level: address of a stack variable at level 3
14:   0x7ffd35d754fc: &level: address of a stack variable at level 2
15:   0x7ffd35d7552c: &level: address of a stack variable at level 1
16:   0x7ffd35d75564: &main_local: address of a local variable in main
17:   0x7ffd35d7555c: &argc: address of argc
18:   0x7ffd35d75550: &argv: address of argv
19:   0x7ffd35d75668: argv: address of arg vector
20:   0x7ffd35d75678: environ: address of environ vector
21:   0x7ffd35d76feb: stack_bottom (environ): byte at bottom of stack
22:
23: Addresses of some static variables:
24:         0x400a40: printf: (text) address of the printf() function
25:         0x400af0: &_start: start of program text
26:         0x400db7: main: (text) address of the main() function
27:         0x40112d: &_etext: end of program text
28:         0x6020a0: &init_var: address of an init static variable
29:         0x6020c0: &_edata: end of init data segment
30:         0x602100: &uninit_var1: address of an uninit static variable1
31:         0x603100: &uninit_var2: address of an uninit static variable2
32:         0x604100: &_end: end of uninit data segment
33:
34: Addresses of some heap variables:
35:         0xf0d030: heap_variable: heap variable  0
36:         0xf0e040: heap_variable: heap variable  1
37:         0xf0f050: heap_variable: heap variable  2
38:         0xf10060: heap_variable: heap variable  3
39:         0xf11070: heap_variable: heap variable  4
40:         0xf12080: heap_variable: heap variable  5
41:         0xf13090: heap_variable: heap variable  6
42:         0xf140a0: heap_variable: heap variable  7
43:         0xf150b0: heap_variable: heap variable  8
44:         0xf160c0: heap_variable: heap variable  9
```