

```
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:     3.141592653589793238462643383279502884197169399,
21:     2.718281828459045235360287471352662497757247093,
22:     0.301029995663981195213738894724493026768189881,
23:     1.414213562373095048801688724209698078569671875,
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
```

05/02/14
19:13:24

\$cmps012b-wm/Labs-cmps012m/lab6c-malloc-free/misc/
prthexaddr.c

3/3

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

[illegible]

```
1: sysname = "Linux"
2: nodename = "unix1.lt.ucsc.edu"
3: release = "2.6.32-431.11.2.el6.x86_64"
4: version = "#1 SMP Tue Mar 25 19:59:55 UTC 2014"
5: machine = "x86_64"
6:
7:         (nil): NULL - null pointer
8:
9: Addresses of some local variables:
10: 0x7fffd2c01b3c: &level - address of a stack variable at level 5
11: 0x7fffd2c01b6c: &level - address of a stack variable at level 4
12: 0x7fffd2c01b9c: &level - address of a stack variable at level 3
13: 0x7fffd2c01bcc: &level - address of a stack variable at level 2
14: 0x7fffd2c01bfc: &level - address of a stack variable at level 1
15: 0x7fffd2c01c34: &main_local - address of a local variable in main
16: 0x7fffd2c01c2c: &argc - address of argc
17: 0x7fffd2c01c20: &argv - address of argv
18: 0x7fffd2c01d38: argv - address of arg vector
19: 0x7fffd2c01d58: environ - address of environ vector
20: 0x7fffd2c02113: argv[ 0] = "./prthexaddr"
21: 0x7fffd2c02120: argv[ 1] = "hello"
22: 0x7fffd2c02126: argv[ 2] = "world"
23: 0x7fffd2c02fea: stack_bottom (environ) - byte at bottom of stack
24:
25: Addresses of some static variables:
26: 0x400740: printf - (text) address of the printf() function
27: 0x4007c0: &_start - start of program text
28: 0x400aa1: main - (text) address of the main() function
29: 0x400e75: &_etext - end of program text
30: 0x601680: &init_var - address of an init static variable
31: 0x6016a0: &edata - end of init data segment
32: 0x6016c0: &uninit_var1 - address of an uninit static variable1
33: 0x6026c0: &uninit_var2 - address of an uninit static variable2
34: 0x6036c0: &_end - end of uninit data segment
35:
36: Addresses of some heap variables:
37: 0x1a31010: heap_variable - heap variable 0
38: 0x1a31fc0: heap_variable - heap variable 1
39: 0x1a32f70: heap_variable - heap variable 2
40: 0x1a33f20: heap_variable - heap variable 3
41: 0x1a34ed0: heap_variable - heap variable 4
42: 0x1a35e80: heap_variable - heap variable 5
43: 0x1a36e30: heap_variable - heap variable 6
44: 0x1a37de0: heap_variable - heap variable 7
45: 0x1a38d90: heap_variable - heap variable 8
46: 0x1a39d40: heap_variable - heap variable 9
```