

```

1 ; Name      : tree.asm
2 ;
3 ;build      : aclocal && autoconf && automake --add-missing --foreign
4 ;
5 ;           mkdir build
6 ;           cd build
7 ;           ../configure
8 ;           make
9 ;description : an example of glib 2.0 balanced binary trees
10 ;
11 ;source      : https://github.com/steshaw/gtk-examples
12
13 bits 64
14
15 [list -]
16     extern  g_tree_destroy
17     extern  g_tree_foreach      ;g_tree_traverse is deprecated since 2.2
18     extern  g_tree_height
19     extern  g_tree_insert
20     extern  g_tree_lookup
21     extern  g_tree_new
22     extern  g_tree_nnodes
23     extern  g_print
24     extern  strcmp
25     extern  exit
26 [list +]
27
28 %define NNODES      4              ;nodes to show in g_tree_foreach
29
30 section .rodata
31     names:
32     .fred:          db      "Fred",0
33     .mary:          db      "Mary",0
34     .sue:           db      "Sue",0
35     .john:          db      "John",0
36     .shelley:       db      "Shelley",0
37     .mark:          db      "Mark",0
38     .renato:        db      "Renato",0
39     properties:
40     .loud:          db      "Loud",0
41     .obnoxious:     db      "Obnoxious",0
42     .drunk:         db      "Drunk",0
43     .quiet:         db      "Quiet",0
44     .civil:         db      "Civil",0
45     .strange:       db      "Strange",0
46     .mighty:        db      "Mighty",0
47     messages:
48     .lookup:        db      "Looking up %s => value %s",10,0
49     .height:        db      "Tree height: %d",10,0
50     .nodes:         db      "Tree nodes: %d",10,0
51     .tree:          db      "Tree:",10,0
52     .node:          db      "key: %s %s value: %s",10,0
53     userdata:       db      "=>",0
54
55 section .data
56
57     tree:           dq      0              ;start of the tree
58     flag:           db      0
59
60 section .text
61 global _start
62
63 _start:
64     ;create tree with compare function
65     mov     rdi,compare
66     call    g_tree_new
67     mov     [tree],rax
68     ;insert the key/value pairs
69     mov     rdi,[tree]
70     mov     rsi,names.fred
71     mov     rdx,properties.loud
72     call    g_tree_insert
73

```

```

74     mov     rdi,[tree]
75     mov     rsi,names.mary
76     mov     rdx,properties.obnoxious
77     call    g_tree_insert
78
79     mov     rdi,[tree]
80     mov     rsi,names.sue
81     mov     rdx,properties.drunk
82     call    g_tree_insert
83
84     mov     rdi,[tree]
85     mov     rsi,names.john
86     mov     rdx,properties.quiet
87     call    g_tree_insert
88
89     mov     rdi,[tree]
90     mov     rsi,names.shelley
91     mov     rdx,properties.civil
92     call    g_tree_insert
93
94     mov     rdi,[tree]
95     mov     rsi,names.mark
96     mov     rdx,properties.strange
97     call    g_tree_insert
98
99     mov     rdi,[tree]
100    mov     rsi,names.renato
101    mov     rdx,properties.mighty
102    call    g_tree_insert
103    ;search if Fred is in the list and print result
104    mov     rdi,[tree]
105    mov     rsi,names.fred
106    call    g_tree_lookup
107    mov     rdx,rax
108    mov     rdi,messages.lookup
109    xor     rax,rax
110    call    g_print
111    ;get tree height and print result
112    mov     rdi,[tree]
113    call    g_tree_height
114    mov     rsi,rax
115    mov     rdi,messages.height
116    xor     rax,rax
117    call    g_print
118    ;get tree nodes and print result
119    mov     rdi,[tree]
120    call    g_tree_nnodes
121    mov     rsi,rax
122    mov     rdi,messages.nodes
123    xor     rax,rax
124    call    g_print
125    ;print nodes 0 to NNODES
126    mov     rdi,messages.tree
127    xor     rax,rax
128    call    g_print
129    mov     rdi,[tree]
130    mov     rsi,display
131    mov     rdx,userdata
132    call    g_tree_foreach
133    ;destroy our tree and the pointer
134    mov     rdi,[tree]
135    call    g_tree_destroy
136    xor     rdi,rdi
137    mov     [tree],rdi                ;destroy pointer too
138    ;exit the program
139    xor     rdi,rdi
140    call    exit
141
142 display:
143     push    rbp
144     mov     rbp,rsp
145     mov     rcx,rsi
146     mov     rsi,rdi

```

```

147     mov     rdi,messages.node
148     xor     rax,rax
149     call    g_print
150     xor     rax,rax
151     inc     byte[flag]
152     cmp     byte[flag],NNODES           ;stop after n nodes
153     jl      .exit
154     inc     rax                         ;return TRUE
155 .exit:
156     mov     rsp,rbp
157     pop     rbp
158     ret
159
160 compare:
161     ;compare strings in rdi and rsi, returning
162     ;-1 when string rdi comes before string rsi
163     ; 0 when both strings are equal
164     ; 1 when string in rsi comes after string rdi
165     call    strcmp
166     ret
167

```