```
File - D:\cpl\2023-cpl-coding-0\12-struct\musician.c
 1 //
 2 // Created by hfwei on 2022/12/8.
 3 //
 4
 5 #include <stdio.h>
 6 #include <string.h>
 7 #include <stdlib.h>
 8 #include <stddef.h>
 9 #include <time.h>
10
11 typedef enum gender {
12
     MALE,
13
     FEMALE,
14 } Gender;
15
16 typedef struct score {
17
     int c_score;
     int java_score;
18
19
     int python_score;
20 } Score;
21
22 typedef struct musician {
    char *name;
24
     // char gender;
25
    Gender gender;
    struct tm birth;
26
27
28
     char *album;
29
30
     Score score;
31 } Musician;
33 // void PrintMusician(const Musician m);
34 void PrintMusician(const Musician *m);
35 int CompareMusician(const void *m1, const void *m2);
36
37 int main() {
     printf("sizeof(Score) = %zu\n", sizeof(Score));
     printf("sizeof(Musician) = %zu\n", sizeof(Musician));
     printf("offsetof(Musician, name) = %zu\n", offsetof(Musician, name)
   ));
    printf("offsetof(Musician, gender) = %zu\n", offsetof(Musician,
41
   gender));
42
    printf("offsetof(Musician, album) = %zu\n", offsetof(Musician, album)
   ));
     printf("offsetof(Musician, score) = %zu\n", offsetof(Musician, score
   ));
44
     Musician luo = {
45
46
         .name = "Luo Dayou",
47
          .gender = MALE,
48
          .birth = {
49
              .tm_year = 1954 - 1900,
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\musician.c
 50
               .tm_{mon} = 7 - 1,
 51
               .tm_mday = 20,
 52
               .tm_wday = 2, // Tuesday
 53
           },
 54
           .album = "ZhiHuZheYe",
 55
           .score = {
 56
               .c_{score} = 0,
 57
               .java_score = 10,
 58
               .python_score = 20,
 59
           },
       };
 60
 61
       Musician cui = {
 62
 63
           .name = "Cui Jian",
 64
           .gender = MALE,
 65
           .birth = {
 66
               .tm_year = 1961 - 1900,
 67
               .tm_{mon} = 8 - 1,
 68
               .tm_mday = 2,
               .tm_wday = 3, // Wednesday
 69
 70
           },
           .album = "XinChangZhengLuShangDeYaoGun",
 71
 72
           .score = {
 73
               .c_score = 10,
 74
               .java_score = 20,
 75
               .python_score = 30,
 76
           },
 77
       };
 78
 79
       char album[] = "YiKeBuKenMeiSuDeXin";
       Musician zhang = {
 80
 81
           .name = "Zhang Chu",
 82
           .gender = MALE,
 83
           .birth = {
 84
               .tm_year = 1968 - 1900,
 85
               .tm_{mon} = 11 - 1,
 86
               .tm_mday = 17,
 87
               .tm_wday = 0, // Sunday
 88
           },
 89
           // .album = "YiKeBuKenMeiSuDeXin",
 90
           .album = album,
 91
           .score = {
 92
               .c_score = 20,
 93
               .java_score = 30,
 94
               .python_score = 40,
 95
           },
       };
 96
 97
 98
       Musician guo = zhang;
 99
       guo.name = "Guo Fucheng";
100
       strcpy(guo.album, "YiKeJiuMeiSuDeXin");
101
      // PrintMusician(guo);
102
      // PrintMusician(zhang);
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\musician.c
103
      PrintMusician(&guo);
104
      PrintMusician(&zhang);
105
106
      Musician musicians[] = { luo, cui, zhang, };
      int len = sizeof musicians / sizeof *musicians;
107
108
      for (int i = 0; i < len; ++i) {
109
        // PrintMusician(musicians[i]);
110
        PrintMusician(&musicians[i]);
      }
111
112
113
      qsort(musicians, len,
114
             sizeof *musicians,
115
             CompareMusician);
116
117
      for (int i = 0; i < len; ++i) {
118
        // PrintMusician(musicians[i]);
119
        PrintMusician(&musicians[i]);
120
      }
121
122
      return 0;
123 }
124
125 // void PrintMusician(const Musician m) {
126 //
        printf("\n%s\t%d\t%s\t%d\t%d\t%d\n",
127 //
                m.name,
128 //
                 m.gender,
129 //
                m.album,
130 //
                m.score.c_score,
131 //
                m.score.java_score,
132 //
                m.score.python_score);
133 // }
134
135 void PrintMusician(const Musician *m) {
136
      printf("\n%s\t%d\t%s\t%d\t%d\t%d\n",
137
              m->name,
138
              m->gender,
              asctime(&m->birth),
139
140
              m->album,
141
              m->score.c_score,
142
              m->score.java_score,
143
              m->score.python_score);
144 }
145
146 int CompareMusician(const void *m1, const void *m2) {
      const Musician *m_left = m1;
147
148
      const Musician *m_right = m2;
149
150
      return strcmp(m_left->album, m_right->album);
151 }
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\padding.c
 1 //
 2 // Created by hfwei on 2023/12/18.
 3 //
 5 #include <stdio.h>
 6 #include <stddef.h>
 8 typedef struct abc {
 9 char a;
10
     int b;
11 char c;
12 } ABC;
13
14 int main(void) {
     printf("sizeof(ABC) = %zu\n", sizeof(ABC));
     printf("offsetof(ABC, a) = %zv\n", offsetof(ABC, a));
16
     printf("offsetof(ABC, b) = %zv\n", offsetof(ABC, b));
17
18
     printf("offsetof(ABC, c) = %zu\n", offsetof(ABC, c));
19
20 return 0;
21 }
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\sds.c
 1 //
 2 // Created by hfwei on 2023/12/19.
 3 //
 4 // sds.h: https://github.com/huangz1990/redis-3.0-annotated/blob/
   unstable/src/sds.h
 5 // sds.c: https://github.com/huangz1990/redis-3.0-annotated/blob/
   unstable/src/sds.c
 6 //
 7
 8 #include <stdio.h>
 9 #include <string.h>
10 #include <stdlib.h>
11 #include <assert.h>
12
13 typedef char *sds;
15 struct sdshdr {
16
    int len;
17
    int free;
18 char buf[];
19 };
20
21 static inline size_t sdslen(const sds s) {
     struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
23
    return sh->len;
24 }
25
26 static inline size_t sdsavail(const sds s) {
     struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
28
     return sh->free;
29 }
30
31 sds sdsnewlen(const void *init, size_t initlen);
32 // sds sdsnew(const char *init);
33
34 void sdsfree(sds s);
36 sds sdsMakeRoomFor(sds s, size_t addlen);
37 sds sdscatlen(sds s, const void *t, size_t len);
38 sds sdscpylen(sds s, const char *t, size_t len);
39
40 int main(void) {
41
    struct sdshdr *sh;
42
43
   sds x = sdsnewlen("foo", 3);
     assert(sdslen(x) == 3);
44
45
     // adding test-case for sdscatlen
46
     x = sdscatlen(x, "bar", 3);
47
48
     assert(sdslen(x) == 6);
49
     assert(strcmp(x, "foobar") == 0);
50
51
     // adding test-case for sdscpylen
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\sds.c
      x = sdscpylen(x, "a", 1);
 53
     assert(sdslen(x) == 1);
 54
      assert(strcmp(x, "a") == \theta);
 55
 56
     return 0;
 57 }
 58
 59 sds sdsnewlen(const void *init, size_t initlen) {
      struct sdshdr *sh;
 60
 61
      sh = malloc(sizeof(struct sdshdr) + initlen + 1);
 62
 63
      if (sh == NULL) {
 64
       return NULL;
      }
 65
 66
 67
      sh->len = initlen;
 68
      sh->free = 0;
 69
 70
      if (initlen && init) {
 71
       memcpy(sh->buf, init, initlen);
 72
 73
 74
    sh->buf[initlen] = '\0';
 75
 76
      return (char *) sh->buf;
 77 }
 78
 79 void sdsfree(sds s) {
      if (s == NULL) {
 80
 81
       return;
      }
 82
 83
     free(s - sizeof(struct sdshdr));
 85 }
 86
 87 sds sdsMakeRoomFor(sds s, size_t addlen) {
 88 struct sdshdr *sh, *newsh;
      size_t free = sdsavail(s);
 89
 90
      size_t len, newlen;
 91
 92
      if (free >= addlen) {
 93
       return s;
 94
      }
 95
 96
      len = sdslen(s);
 97
      sh = (void *) (s - sizeof(struct sdshdr));
 98
      newlen = (len + addlen) * 2;
 99
      newsh = realloc(sh, sizeof(struct sdshdr) + newlen + 1);
100
      if (newsh == NULL) {
101
       return NULL;
102
      }
103
      newsh->free = newlen - len;
104
```

```
File - D:\cpl\2023-cpl-coding-0\12-struct\sds.c
     return newsh->buf;
106 }
107
108 sds sdscatlen(sds s, const void *t, size_t len) {
109
     struct sdshdr *sh;
110
      size_t curlen = sdslen(s);
111
112
     s = sdsMakeRoomFor(s, len);
113
      if (s == NULL) {
114
       return NULL;
115
116
      sh = (void *) (s - sizeof(struct sdshdr));
117
118
      memcpy(s + curlen, t, len);
119
      sh->len = curlen + len;
120
      sh->free = sh->free - len;
121
      s[curlen + len] = ' \ 0';
122
123
      return s;
124 }
125
126 sds sdscpylen(sds s, const char *t, size_t len) {
      struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
128
      size_t totlen = sh->free + sh->len;
129
130
     if (totlen < len) {</pre>
       s = sdsMakeRoomFor(s, len - sh->len);
131
132
        if (s == NULL) {
133
          return NULL;
134
135
        sh = (void *) (s - sizeof(struct sdshdr));
136
        totlen = sh->free + sh->len;
137
138
139
      memcpy(s, t, len);
140
      s[len] = ' \ 0';
141
      sh->len = len;
142
      sh->free = totlen - len;
143
144
      return s;
145 }
```