```c
// Created by hfwei on 2024/12/11.

// sds.h: https://github.com/huangz1990/redis-3.0-annotated/blob/unstable/src/sds.h
// sds.c: https://github.com/huangz1990/redis-3.0-annotated/blob/unstable/src/sds.c

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>

typedef char *sds;

struct sdshdr {
  int len;
  int free;
  char buf[];
};

static inline size_t sdslen(const sds s) {
  struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
  return sh->len;
}

static inline size_t sdsavail(const sds s) {
  struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
  return sh->free;
}

sds sdsnewlen(const void *init, size_t initlen);
// sds sdsnew(const char *init);

void sdsfree(sds s);

sds sdsMakeRoomFor(sds s, size_t addlen);
sds sdscatlen(sds s, const void *t, size_t len);
sds sdscpylen(sds s, const char *t, size_t len);

int main(void) {
  sds x = sdsnewlen("foo", 3);
  assert(sdslen(x) == 3);
```

```c

  // adding test-case for sdscatlen
  x = sdscatlen(x, "bar", 3);
  assert(sdslen(x) == 6);
  assert(strcmp(x, "foobar") == 0);

  // adding test-case for sdscpylen
  x = sdscpylen(x, "a", 1);
  assert(sdslen(x) == 1);
  assert(strcmp(x, "a") == 0);

  return 0;
}

sds sdsnewlen(const void *init, size_t initlen) {
  struct sdshdr *sh;

  sh = malloc(sizeof(struct sdshdr) + initlen + 1);
  if (sh == NULL) {
    return NULL;
  }

  sh->len = initlen;
  sh->free = 0;

  if (initlen && init) {
    memcpy(sh->buf, init, initlen);
  }

  sh->buf[initlen] = '\0';

  return (char *) sh->buf;
}

void sdsfree(sds s) {
  if (s == NULL) {
    return;
  }

  free(s - sizeof(struct sdshdr));
}

sds sdsMakeRoomFor(sds s, size_t addlen) {
  struct sdshdr *sh, *newsh;
```

```c
  size_t free = sdsavail(s);
  size_t len, newlen;

  if (free >= addlen) {
    return s;
  }

  len = sdslen(s);
  sh = (void *) (s - sizeof(struct sdshdr));
  newlen = (len + addlen) * 2;
  newsh = realloc(sh, sizeof(struct sdshdr) + newlen + 1
);
  if (newsh == NULL) {
    return NULL;
  }

  newsh->free = newlen - len;
  return newsh->buf;
}

sds sdscatlen(sds s, const void *t, size_t len) {
  struct sdshdr *sh;
  size_t curlen = sdslen(s);

  s = sdsMakeRoomFor(s, len);
  if (s == NULL) {
    return NULL;
  }

  sh = (void *) (s - sizeof(struct sdshdr));
  memcpy(s + curlen, t, len);
  sh->len = curlen + len;
  sh->free = sh->free - len;
  s[curlen + len] = '\0';

  return s;
}

sds sdscpylen(sds s, const char *t, size_t len) {
  struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr
));
  size_t totlen = sh->free + sh->len;

  if (totlen < len) {
```

```
127     s = sdsMakeRoomFor(s, len - sh->len);
128     if (s == NULL) {
129       return NULL;
130     }
131     sh = (void *) (s - sizeof(struct sdshdr));
132     totlen = sh->free + sh->len;
133   }
134
135   memcpy(s, t, len);
136   s[len] = '\0';
137   sh->len = len;
138   sh->free = totlen - len;
139
140   return s;
141 }
```

```c
// Created by hfwei on 2024/12/10.

#include <stdio.h>
//#include <stddef.h>

// creference: https://en.cppreference.com/w/c/types/offsetof
// Magic: https://radek.io/posts/magical-container_of-macro/
// StackOverflow: https://stackoverflow.com/q/15832301/1833118

#define offsetof(TYPE, MEMBER) ((size_t) &(((TYPE *)0)->MEMBER))
//#define OffsetOf(TYPE, MEMBER) (&(((TYPE *)0)->MEMBER))
#define container_of(ptr, type, member ) ({                      \
        const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
        (type *)( (char *)__mptr - offsetof(type,member ) );})

typedef struct abc {
  char a;
  int b;
  char c;
} ABC;

int main(void) {
  printf("sizeof(ABC) = %zu\n", sizeof(ABC));
  printf("offsetof(ABC, a) = %zu\n", offsetof(ABC, a));
  printf("offsetof(ABC, b) = %zu\n", offsetof(ABC, b));
  printf("offsetof(ABC, c) = %zu\n", offsetof(ABC, c));

  ABC abc = {'a', 42, 'c'};
  const int *b_ptr = &abc.b;
  ABC *abc_ptr = container_of(b_ptr, ABC, b);
  printf("address: %p\t%p\n", abc_ptr, &abc);

  return 0;
}
```

```
 1 # `12-struct`
 2
 3 - `struct`
 4 - `struct musician`
 5 - `typedef struct musician`
 6 - **struct alignment and padding**
 7 - `typedef struct score`
 8 - `enum`
 9 - `struct` assignment
10 - `PrintMusician(const Musician m)`
11 - `PrintMusician(const Musician *m)`
12
13 ## c-reference
14
15 - `struct`
16 - `flexible array`
17
18 ## padding
19
20 ## `offsetof` and `container_of`
21
22 - [offsetof @ wikipedia](https://en.wikipedia.org/wiki/
   Offsetof)
23 - [offsetof @ cref](https://en.cppreference.com/w/c/types/
   offsetof)
24 - [offsetof @ stackoverflow](https://stackoverflow.com/q/
   26906621/1833118)
25 - [container_of @ stackoverflow](https://stackoverflow.com
   /q/15832301/1833118)
26 - [container_of @ radek.io](https://radek.io/posts/magical
   -container_of-macro/)
27 - [container_of @ Linux Kernel Monkey Log](http://www.
   kroah.com/log/linux/container_of.html)
28 - [container_of @ 0xAX](https://0xax.gitbooks.io/linux-
   insides/content/DataStructures/linux-datastructures-1.html
   )
29
30 ## Union
31
32 - [union in
33   `dictEntry` @ redis](https://github.com/redis/redis/blob
   /c51c96656bf1f1801ae90a376f71890cbcdea4b4/src/dict.c#L47-
   L134)
```

```c
// Created by hfwei on 2024/12/11.

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stddef.h>
#include <time.h>

typedef enum gender {
  MALE,
  FEMALE,
  GENDER_KINDS,
} Gender;

typedef struct score {
  int c_score;
  int java_score;
  int python_score;
} Score;

typedef struct musician {
  char *name;
  // char gender;
  Gender gender;
  struct tm birth;

  char *album;

  Score score;

  union {
    int performances;    // number of performances
    double funding;      // funding in millions
    int awards;          // number of awards won
    char *highlight;     // textual highlight of the yea
  } year_end_summary;

  enum {
    PERFORMANCES,
    FUNDING,
    AWARDS,
    HIGHLIGHT
  } summary_type;
} Musician;
```

```c

// void PrintMusician(const Musician m);
void PrintMusician(const Musician *m);
int CompareMusician(const void *m1, const void *m2);

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, gender));
  printf("offsetof(Musician, album) = %zu\n", offsetof(Musician, album));
  printf("offsetof(Musician, score) = %zu\n", offsetof(Musician, score));

  Musician luo = {
      .name = "Luo Dayou",
      .gender = MALE,
      .birth = {
          .tm_year = 1954 - 1900,
          .tm_mon = 7 - 1,
          .tm_mday = 20,
          .tm_wday = 2, // Tuesday
      },
      .album = "ZhiHuZheYe",
      .score = {
          .c_score = 0,
          .java_score = 10,
          .python_score = 20,
      },
      .year_end_summary.performances = 20,
      .summary_type = PERFORMANCES,
  };

  Musician cui = {
      .name = "Cui Jian",
      .gender = MALE,
      .birth = {
          .tm_year = 1961 - 1900,
          .tm_mon = 8 - 1,
          .tm_mday = 2,
          .tm_wday = 3, // Wednesday
```

```c
        },
        .album = "XinChangZhengLuShangDeYaoGun",
        .score = {
            .c_score = 10,
            .java_score = 20,
            .python_score = 30,
        },
        .year_end_summary.funding = 2.5,
        .summary_type = FUNDING,
    };

    char album[] = "YiKeBuKenMeiSuDeXin";
    Musician zhang = {
        .name = "Zhang Chu",
        .gender = MALE,
        .birth = {
            .tm_year = 1968 - 1900,
            .tm_mon = 11 - 1,
            .tm_mday = 17,
            .tm_wday = 0, // Sunday
        },
        // .album = "YiKeBuKenMeiSuDeXin",
        .album = album,
        .score = {
            .c_score = 20,
            .java_score = 30,
            .python_score = 40,
        },
        // https://www.bilibili.com/video/BV1f6i2YZEMJ/
        .year_end_summary.awards = 2,
        .summary_type = AWARDS,
    };

    Musician guo = zhang;
    guo.name = "Guo Fucheng";
    strcpy(guo.album, "YiKeJiuMeiSuDeXin");
    // PrintMusician(guo);
    // PrintMusician(zhang);
    PrintMusician(&guo);
    PrintMusician(&zhang);

    Musician musicians[] = {luo, cui, zhang,};
    int len = sizeof musicians / sizeof *musicians;
    for (int i = 0; i < len; ++i) {
```

```c
    // PrintMusician(musicians[i]);
    PrintMusician(&musicians[i]);
  }

  qsort(musicians, len,
        sizeof *musicians,
        CompareMusician);

  for (int i = 0; i < len; ++i) {
    // PrintMusician(musicians[i]);
    PrintMusician(&musicians[i]);
  }

  return 0;
}

// void PrintMusician(const Musician m) {
//   printf("\n%s\t%d\t%s\t%d\t%d\t%d\n",
//          m.name,
//          m.gender,
//          m.album,
//          m.score.c_score,
//          m.score.java_score,
//          m.score.python_score);
// }

void PrintMusician(const Musician *m) {
  printf("\n%s\t%d\t%s\t%s\t%d\t%d\t%d\n",
         m->name,
         m->gender,
         asctime(&m->birth),
         m->album,
         m->score.c_score,
         m->score.java_score,
         m->score.python_score);

  switch (m->summary_type) {
    case PERFORMANCES:printf("Performed %d times\n", m->
year_end_summary.performances);
      break;
    case FUNDING:printf("Secured $%.2fM in funding\n", m
->year_end_summary.funding);
      break;
    case AWARDS:printf("Won %d awards\n", m->
```

```
170 year_end_summary.awards);
171       break;
172     case HIGHLIGHT:printf("%s\n", m->year_end_summary.
    highlight);
173       break;
174   }
175 }
176
177 int CompareMusician(const void *m1, const void *m2) {
178   const Musician *m_left = m1;
179   const Musician *m_right = m2;
180
181   return strcmp(m_left->album, m_right->album);
182
183   // char *name_left = *(char **) m1;
184   // char *name_right = *(char **) m2;
185
186   // return strcmp(name_left, name_left);
187 }
```

```c
// Created by hengxin on 12/11/24.
// sds.h: https://github.com/redis/redis/blob/unstable/src/sds.h
// sds.c: https://github.com/redis/redis/blob/unstable/src/sds.c

/* SDSLib 2.0 -- A C dynamic strings library
 *
 * Copyright (c) 2006-Present, Redis Ltd.
 * All rights reserved.
 *
 * Licensed under your choice of the Redis Source Available License 2.0
 * (RSALv2) or the Server Side Public License v1 (SSPLv1).
 */

#include <stdio.h>
#include <ctype.h>
#include <stdint.h>

typedef char *sds;

struct __attribute__ ((__packed__)) sdshdr5 {
  unsigned char flags; /* 3 lsb of type, and 5 msb of string length */
  char buf[];
};

struct __attribute__ ((__packed__)) sdshdr8 {
  uint8_t len; /* used */
  uint8_t alloc; /* excluding the header and null terminator */
  unsigned char flags; /* 3 lsb of type, 5 unused bits */
  char buf[];
};

struct __attribute__ ((__packed__)) sdshdr16 {
  uint16_t len; /* used */
  uint16_t alloc; /* excluding the header and null terminator */
  unsigned char flags; /* 3 lsb of type, 5 unused bits */
  char buf[];
};

```

```c
struct unpacked_sdshdr16 {
  uint16_t len; /* used */
  uint16_t alloc; /* excluding the header and null terminator */
  unsigned char flags; /* 3 lsb of type, 5 unused bits */
  char buf[];
};

struct __attribute__ ((__packed__)) sdshdr32 {
  uint32_t len; /* used */
  uint32_t alloc; /* excluding the header and null terminator */
  unsigned char flags; /* 3 lsb of type, 5 unused bits */
  char buf[];
};

struct __attribute__ ((__packed__)) sdshdr64 {
  uint64_t len; /* used */
  uint64_t alloc; /* excluding the header and null terminator */
  unsigned char flags; /* 3 lsb of type, 5 unused bits */
  char buf[];
};

#define SDS_TYPE_5  0
#define SDS_TYPE_8  1
#define SDS_TYPE_16 2
#define SDS_TYPE_32 3
#define SDS_TYPE_64 4
#define SDS_TYPE_MASK 7
#define SDS_TYPE_BITS 3
#define SDS_HDR(T, s) ((struct sdshdr##T *)((s)-(sizeof(struct sdshdr##T))))
#define SDS_TYPE_5_LEN(f) ((f)>>SDS_TYPE_BITS)

static inline int sdsHdrSize(char type) {
  switch (type & SDS_TYPE_MASK) {
    case SDS_TYPE_5:return sizeof(struct sdshdr5);
    case SDS_TYPE_8:return sizeof(struct sdshdr8);
    case SDS_TYPE_16:return sizeof(struct sdshdr16);
    case SDS_TYPE_32:return sizeof(struct sdshdr32);
    case SDS_TYPE_64:return sizeof(struct sdshdr64);
  }
  return 0;
```

```c
}

static inline size_t sdslen(const sds s) {
  unsigned char flags = s[-1];
  switch (flags & SDS_TYPE_MASK) {
    case SDS_TYPE_5:return SDS_TYPE_5_LEN(flags);
    case SDS_TYPE_8:return SDS_HDR(8, s)->len;
    case SDS_TYPE_16:return SDS_HDR(16, s)->len;
    case SDS_TYPE_32:return SDS_HDR(32, s)->len;
    case SDS_TYPE_64:return SDS_HDR(64, s)->len;
  }
  return 0;
}

int main(void) {
  // test sdsHdrSize
  printf("%d\n", sdsHdrSize(SDS_TYPE_5));
  printf("%d\n", sdsHdrSize(SDS_TYPE_8));
  printf("%d\n", sdsHdrSize(SDS_TYPE_16));
  printf("%zu\n", sizeof(struct unpacked_sdshdr16));
  printf("%d\n", sdsHdrSize(SDS_TYPE_32));
  printf("%d\n", sdsHdrSize(SDS_TYPE_64));
  return 0;
}
```

```
1 add_executable(musician musician.c)
2 add_executable(padding offset.c)
3 add_executable(sds sds.c)
4 add_executable(sds-202412 sds-202412.c)
```