2020/04/27_第18课_结构体

笔记本:	C
创建时间:	2020/4/27 星期一 15:12
作者:	ileemi
标签:	结构体, 内存对齐

模仿标准库函数，不做检查，检查交给调用方
特点：兼容多个平台
0地址不能读写时Windows系统规定的
比如单片机的0地址可以读写

void * 没有解释方式的指针，也就是地址
特点：

```
void *pv = NULL; //没有解释方式的指针，也就是地址
int *pn = &argc;

pv = pn; //ok，pv需要一个地址类型就好，任意类型的指针就行
pn = pv; //error，pn需要一个解释方式为整型的地址
*pv = 6; //error，void *没有解释方式，所以不支持*运算
pv[] = 6; //error，void *没有解释方式，所以不支持[]运算
pv = pv + 1; //error，void *没有解释方式，所以不支持 +,-,++,--运算
```

结构体

基本语法
内存结构
设计（难点）

微软MSG
结构体名称 tag开头

用途(函数定义)+数据 才能完整的描述一个事物

数据不能识别一个事物，行为可以识别一个事物

结构体的设计 应该重点设计 用途，行为

数据关系的整理中：不应该出现身高、年龄、体重、血压心跳等信息与你个人信息无关，这属于检查信息，与检查有关，应该另外设置一个检查信息表(检查信息)

.cpp为结构体声明类型时 可以不添加struct

性别：F(Female) M(Male) U(Unknown) O(Other) 应该定义 char

空结构体占用1字节内存，允许空结构体定义变量，可以为其取地址，所以需要占用空间

空结构体表示极度抽象事物

内存对齐

结构体成员变量的内存对齐原则：

设置编译对齐值为 Zp /(对齐值)
设成员变量在结构体定义中的偏移量为 offset
offset 必须满足
offset % min(Zp, sizeof(member type)) == 0

```
/*
设编译对齐值为 Zp
设成员变量在结构体定义中的偏移量为 offset

offset必须满足
offset % min(Zp, sizeof(member type)) == 0
*/
struct tagPerson
{
  int nID;                  // +0
  double dblScore;          // +8
  char szName[5];           // +16 % min(8, sizeof(char))
  unsigned short int nAge;
  char cGender;
  float fHeight;
  float fWeight;
};
```

设结构体变量自身对齐值为 Align
Align必须满足：
Align = max(取最大值)(sizeof(member1 type), sizeof(member2 type), ...,sizeof(memberN type))
Align = min(Zp, Align)

设结构体变量的占用空间大小为size
size必须满足：
size % Align == 0

结构体的嵌套 不允许嵌套结构体自己的类型名

```c
#include <stdio.h>
#include <stdlib.h>

/*
规则1
设编译对齐值为Zp
设成员变量在结构体中的偏移量为 offset

offset 必须满足:
offset % min(Zp, sizeof(member type)) == 0
```

```
规则2
设结构体变量自身对齐值为：Align

Align 必须满足:
Align = (max)(sizeof(member1 type), sizeof(member2
type),...,sizeof(memberN type))
Align = min(Zp, Align)

Align = (max)(sizeof(int), sizeof(char),sizeof(double), sizeof(unsigned
short int), sizeof(float))
Align = (max)(4, 1, 8, 2, 4)
      = 8

Align = min(8, 8)
      = 8

规则3
设结构体变量占用空间大小为： size
size必须满足：
size % Align == 0
*/

struct tagDataOfBirth
{
    int nYear; //0
    short int wMonth; //4 4%2 == 0
    char cDay; //4+2 6%1 == 0
};  // 6 + 1  == 7 7%4 != 0 7+1 == 8 size:8

struct tagPerson
{
    int nID; // 0
    double dblScore; // 8   4 % 8 != 0
    char szName[5]; // 16  16 % 1 == 0
    unsigned short int nAge; // 22  21 % 2 != 0
    struct tagDataOfBirth DOB; // 24  24 % min(8, 4)== 0
    char cGender; // 32  24 + 8 == 32 % 1 == 0
    float fHeight; // 36  32 + 1 == 33 % 4 != 0
    float fWeight; // 40  36 + 4 == 36 % 4 == 0
}; // 36 + 4 = 44 % min(8, 8)  size: 48

struct tagTest
{
    char *name; //0
    int age; //4
    float scores; //4
```

```
}; //size 8 + 4 = 12

int main(int argc, char* argv[])
{
    struct tagPerson person = {
        001,
        89.5,
        "Tom",
        20,
        {
            1999,
            12,
            30
        },
        'M',
        175.f,
        70.f

    };
    /*
    struct Type tag = ...;
    tag.member address is:
        (int)&tag + member offset
    tag.member == *(memeber type *)((int)&tag + member offset)
    */

    printf("%d\r\n", sizeof(tagPerson));


    struct tagPerson *pPer = &person;
    printf("%f\r\n", person.fHeight);
    printf("%f\r\n", *(float *)((int)&person + 28));

    printf("%f\r\n", pPer->fWeight);
    printf("%f\r\n", *(float *)((int)pPer + 32));


    pPer = NULL;
    printf("%d\r\n", &pPer->fHeight);
    printf("%d\r\n", (float *)((int)pPer + 28));

    system("pause");
    return 0;
}
```

对齐系数为1 不进行字节对齐

两个结构体分别设置对齐系数
#pragma pack(1) 对齐系数为1

#pragma pack(8) 对齐系数为8

#pragma pack(push) 保存原来的对齐值
#pragma pack(1) 设置当前结构体的对齐系数为1
#pragma pack(pop) 在结构体末尾恢复原对齐值

```
#pragma pack(push)      //保存原来的对齐系数
#pragma pack(1)         //设置当前结构体的系数为1
struct tagDataOfBirth
{
    int nYear;              //0
    short int wMonth;       //4 4%2 == 0
    char cDay;              //4+2 6%1 == 0
};  // 6 + 1  == 7 7 % min(1, 4) == 0 size: 7
#pragma pack(pop)       //在该结构体的末尾恢复原结构体的对齐值
```

网络通讯时需要首先定义协议

结构体成员的寻址方式：结构体类型的首地址加偏移量 基址(变址)相对寻址方式(决定结构体访问成员的效率)

```
    struct Type tag = ...;
    tag.member address is:
        (int)&tag + member offset       结构体类型的首地址视为整型 + 成员偏移量
    tag.member == *(memeber type *)((int)&tag + member offset)
    *(memeber type *) 解释为memeber type的指针 再取内容
```

结构体成员名经过编译器处理后都是符号化的偏移量

以0地址作为结构体类型的首地址，然后访问结构体中的成员，并取地址，就可以求出其成员的偏移量
示例:

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>

struct tagPerson
{
    int nAge; // 0
    char szName[16]; // 4
    int nLen; // 4 + 16 == 20
};  //size 20 + 4
int main(int argc, char* argv[])
```

```
{
    struct tagPerson per = {
        20,
        "Tom",
        99
    };
    /*
    struct Type tag = ...;
    tag.member address is:
        (int)&tag + member offset  结构体类型的首地址视为整型 + 成员偏移量
    tag.member == *(memeber type *)((int)&tag + member offset)
    */
    printf("结构体总字节大小：%d\r\n", sizeof(tagPerson));
        struct tagPerson *pPer = &per;
//      printf("%d\r\n", per.nAge);
//      printf("%d\r\n", *(int *)((int)&per + 0));

//      pPer = NULL;
//      printf("%d\r\n", &pPer->nLen);
//      printf("%d\r\n", (int *)((int)pPer + 20));


    /*
    s 结构体类型名
    m 结构体成员名
    */
    //该宏可用于取结构体内成员的的偏移量
#define GetOffset(s, m) (size_t)&(((s *)NULL)->m)

    pPer = NULL;
    printf("结构体成员 nAge 的偏移量为：");
    printf("%d ", &pPer->nAge);          //0
    printf("%d ", (int *)((int)pPer + 0));//0
    printf("%d\r\n", GetOffset(tagPerson, nAge));//0

    printf("结构体成员 szName 的偏移量为：");
    printf("%d ", &pPer->szName);           //4
    printf("%d ", (char *)((int)pPer + 4));     //4
    printf("%d\r\n", GetOffset(tagPerson, szName)); //4

    printf("结构体成员 nLen 的偏移量为：");
    printf("%d ", &pPer->nLen);             //20
    printf("%d ", (int *)((int)pPer + 20)); //20
    printf("%d\r\n\n", GetOffset(tagPerson, nLen)); //20
```

```
    printf("成员 nAge 在当前结构体内的偏移量为：%d 字节\r\n",
GetOffset(tagPerson, nAge));//0
    printf("成员 szName 在当前结构体内的偏移量为：%d 字节\r\n",
GetOffset(tagPerson, szName));//4
    printf("成员 nLen 在当前结构体内的偏移量为：%d 字节\r\n\n",
GetOffset(tagPerson, nLen));//20

    printf("成员 nAge 在当前结构体内的偏移量为：%d 字节\r\n",
offsetof(tagPerson, nAge));//0
    printf("成员 szName 在当前结构体内的偏移量为：%d 字节\r\n",
offsetof(tagPerson, szName));//4
    printf("成员nLen 在当前结构体内的偏移量为：%d 字节\r\n",
offsetof(tagPerson, nLen));//20

    system("pause");
    return 0;
}
```