C++的placement new操作

CodingCode 关注

2018.11.27 14:05:13 字数 458 阅读 806

C++的placement new操作

什么是placement new操作
我们知道C++的new操作符会创建一个对象，他完成两步操作：

1. 分配对象内存。
2. 调用对象类的构造函数创建对象。

通常分配的内存是在堆中。

但是有些场景下，我们预先已经分配了内存，想要在已知的内存上创建对象怎么办呢？就是说我就要一个对象创建在这个内存地址，placement new就是实现这个目的的。其语法：

```
Object * p = new (address) ClassConstruct(...)
```

应用场景
在进程间使用共享内存的时候，C++的placement new经常被用到。例如主进程分配共享内容，然后在共享内存上创建C++类对象，然后从进程直接attach到这块共享内容，拿到类对象，直接访问类对象的变量和函数。

通过下面的例子来说明：

1. 主进程以server的方式启动
   - 分配共享内存
   - 在共享内存上通过placement new创建对象SHMObj
2. 从进程以普通方式启动
   - attach到主进程的共享内存
   - 拿到代表SHMObj对象的指针。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

```

```
#include <sys/ipc.h>
#include <sys/shm.h>

#include <new>

#define SHM_KEY 0x3887

class SHMObj {
private:
    int count;
public:
    SHMObj() : count(100) {}
    void increase() { count++; print(); }
    void decrease() { count--; print(); }
    void print() { printf("count=[%d]\n", count); }
};

static int      shm_id = 0;
static void *   shm_addr = NULL;
static SHMObj * shm_object = NULL;

static int attach(bool server) {
    shm_id = shmget(SHM_KEY, sizeof(SHMObj), server ? (IPC_CREAT | 0660) : 0660);
    if (shm_id < 0) {
        printf("ERROR: attach(%d), errno=[%d],strerror=[%s]\n", server, errno, strerror(errno));
        return -1;
    }

    if ((shm_addr = (struct shm_content *)shmat(shm_id, (void *)0, 0)) == NULL) {
        printf("ERROR: attach(%d), errno=[%d],strerror=[%s]\n", server, errno, strerror(errno));
        return -1;
    }

    printf("SUCC: attach(%d), key=[0x%x],id=[0x%x],address=[0x%x],object=[0x%x]\n", server, SHM
}

static int dettach() {
    if (shmdt(shm_addr) != 0) {
        printf("ERROR: dettach(), errno=[%d],strerror=[%s]\n", errno, strerror(errno));
        return -1;
    }

    if (shmctl(shm_id, IPC_RMID, NULL) < 0) {
        printf("ERROR: dettach(), errno=[%d],strerror=[%s]\n", errno, strerror(errno));
        return -1;
    }

    return 0;
}

static int create(bool server) {
    if (server) {
        if ((shm_object = new (shm_addr) SHMObj()) == NULL) {
            printf("ERROR: attach(), errno=[%d],strerror=[%s]\n", errno, strerror(errno));
            return -1;
        }
    }
    else {
```

```
64          //shm_object = (SHMObj *)(shm_addr);
65          shm_object = reinterpret_cast<SHMObj *>(shm_addr);
66      }
67      printf("SUCC: create(%d), address=[0x%x],object=[0x%x]\n", server, shm_addr, shm_object);
68      return 0;
69  }
70
71  static int increase() {
72      if (shm_object != NULL) {
73          shm_object->increase();
74      }
75      else {
76          printf("ERROR: increase(), call attach firstly\n");
77      }
78  }
79
80  static int decrease() {
81      if (shm_object != NULL) {
82          shm_object->decrease();
83      }
84      else {
85          printf("ERROR: increase(), call attach firstly\n");
86      }
87  }
88
89  static int print() {
90      if (shm_object != NULL) {
91          shm_object->print();
92      }
93      else {
94          printf("ERROR: print(), call attach firstly\n");
95      }
96  }
97
98  void help() {
99      printf("attach  : \n");
100     printf("dettach : \n");
101     printf("create  : \n");
102     printf("increase: \n");
103     printf("decrease: \n");
104     printf("print   : \n");
105     printf("quit    : quit program\n");
106 }
107
108 int parseCommand(char * cmd, char * argv[]) {
109    const char sep[3] = " \n";
110    char *token = strtok(cmd, sep);
111
112    int i = 0;
113    while (token != NULL) {
114       argv[i++] = token;
115       token = strtok(NULL, sep);
116    }
117     return i;
118 }
119
120 int main(int argc, char * argv[]) {
121     char cmdbuffer[1024];
```

```
    char * cmds[10];    /** max parameters count */
    int i = 0;

    while (1) {
        printf("CMD> ");
        fgets(cmdbuffer, 1024, stdin);
        i = parseCommand(cmdbuffer, cmds);
        if (i > 0) {
            if (strcmp(cmds[0], "quit") == 0) {
                break;
            }
            else if (strcmp(cmds[0], "attach") == 0) {
                    attach(argc > 1 && strcmp(argv[1],"server") == 0);
            }
            else if (strcmp(cmds[0], "dettach") == 0) {
                    dettach();
            }
            else if (strcmp(cmds[0], "create") == 0) {
                    create(argc > 1 && strcmp(argv[1],"server") == 0);
            }
            else if (strcmp(cmds[0], "increase") == 0) {
                    increase();
            }
            else if (strcmp(cmds[0], "decrease") == 0) {
                    decrease();
            }
            else if (strcmp(cmds[0], "print") == 0) {
                    print();
            }
            else if (strcmp(cmds[0], "help") == 0) {
                help();
            }
            else {
                printf("unknown command: %s\n", cmds[0]);
            }
        }
    }
    return 0;
}
```

运行

主进程:

```
$ ./main server
CMD> attach
SUCC: attach(1), key=[0x3887],id=[0x1a8004],address=[0xe7b59000],object=[0x0]
CMD> create
SUCC: create(1), address=[0xe7b59000],object=[0xe7b59000]
CMD> print
count=[100]
CMD> increase
count=[101]
CMD> increase
count=[102]
```

```
CMD> print
count=[102]
CMD>
```

从进程：

```
$ ./main
CMD> attach
SUCC: attach(0), key=[0x3887],id=[0x1a8004],address=[0xa1126000],object=[0x0]
CMD> create
SUCC: create(0), address=[0xa1126000],object=[0xa1126000]
CMD> print
count=[102]
CMD>
```

从这个例子我们看到对象shm_object在主进程里面被创建(placement new)，但是在从进程里面并没有创建，而是直接从共享内存里面解析出来，然后直接访问类成员和函数。
需要注意的是，创建出来的对象的地址就是共享内存的地址，就是基于这个属性，我们的功能才能实现。也就是说：

```
Object * p = new (address) ClassConstruct(...)
```

返回p的值，和输入地址address的值是相同的。