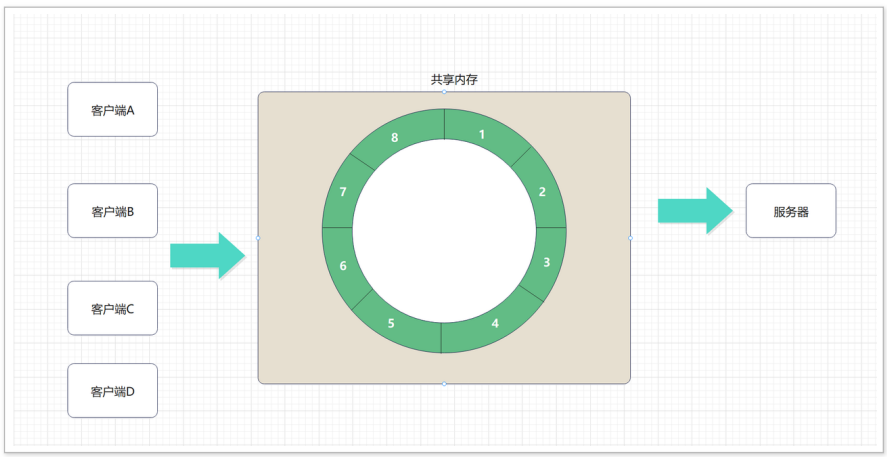
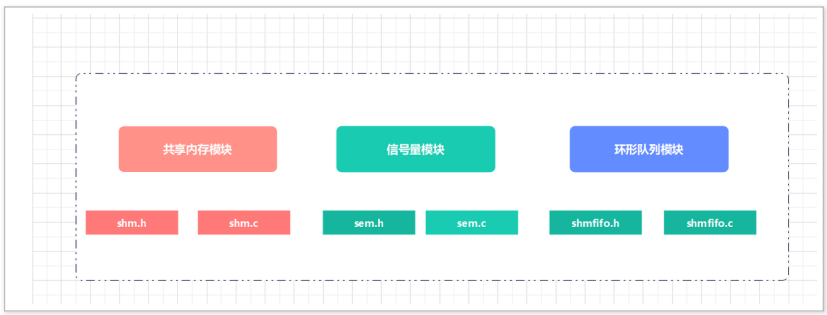
2.1 环形队列设计 (一)- 基本框架与共享内存模块设计_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.1 环形队列设计 (一)– 基本框架与共享内存模块设计涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 在本次项目中，多客户端需要频繁的将消息发布与订阅请求传送给服务器，为了提高效率，使用如下方案
    - 使用共享内存，提高进程间通讯的效率
    - 使用环形队列 (循环队列)，对相应的请求进行缓存处理，服务器依次从队列中读取数据进行处理



- 环形队列划分如下:
    - 共享内存模块：创建共享内存, 映射地址空间，删除共享内存
    - 信号量模块：用于共享内存的互斥与同步
    - 环形队列模块：用于实现环形队列的相关操作
    - 

- 共享内存模块是基于系统共享内存的接口来进行封装设计, 具体实现如下:
    - step 1：创建 shm.h shm.c
    -
    - step 2：实现 shm.h
    -
```
#ifndef __SHM_H_
#define __SHM_H_

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

```
enum shm_creat_status{
    SHM_HAS_EXIST = 0,
    SHM_CREAT_NEW,
    SHM_CREAT_ERROR,
};

extern enum shm_creat_status shm_create(size_t size,int *pshmid);

extern void *shm_at(int shmid);
extern int shm_dt(const void *shmaddr);
extern int shm_del(int shmid);

#endif
```

- step 3 : 实现 shm.c
    - shm_create 函数实现

```
#define PATHNAME "."
#define PRO_ID 101

enum shm_creat_status shm_create(size_t size,int *pshmid)
{
    key_t key;
    int shmid;

    key = ftok(PATHNAME,PRO_ID);

    shmid = shmget(key,size,0);
    if (shmid == -1){
        shmid = shmget(key,size,IPC_CREAT|0644);
        if (shmid == -1){
            perror("[ERROR] shmget(): ");
            return SHM_CREAT_ERROR;
        }

        *pshmid = shmid;
        return SHM_CREAT_NEW;

    }else{

        *pshmid = shmid;
        return SHM_HAS_EXIST;
    }
}
```

    - shm_at 函数实现

```
void *shm_at(int shmid)
{
    void *addr = NULL;

    addr = shmat(shmid,NULL,0);
    if (addr == (void *)-1){
        perror("[ERROR] shmat(): ");
        return NULL;
    }

    return addr;
}
```

    - shm_dt 函数实现

```
int shm_dt(const void *shmaddr)
 {
     return shmdt(shmaddr);
 }
```

    - shm_del 函数实现

```
int shm_del(int shmid)
{
    return shmctl(shmid,IPC_RMID,NULL);
}
```

- server.c

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

```
#include "shm.h"

#define SHM_SZ 256

int main(void)
{
    int shmid;
    enum shm_creat_status shm_status;
    void *addr = NULL;
    char buffer[16] = {0};

    shm_status = shm_create(SHM_SZ,&shmid) ;

    if (shm_status == SHM_CREAT_NEW)
        printf(" shared memory creat new.\n");
    else if (shm_status == SHM_HAS_EXIST)
        printf(" shared memory has exist.\n");

    addr = shm_at(shmid);
    if (addr == NULL){
        printf("shm at failed.\n");
        return -1;
    }

    memcpy(buffer,addr,10);

    printf("buffer : %s\n",buffer);

    shm_dt(addr);
    shm_del(shmid);
    return 0;
}
```

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

#include "shm.h"

#define SHM_SZ 256

int main(void)
{
    int shmid;
    enum shm_creat_status shm_status;
    void *addr = NULL;
    char buffer[16] = {0};

    shm_status = shm_create(SHM_SZ,&shmid) ;

    if (shm_status == SHM_CREAT_NEW)
        printf(" shared memory creat new.\n");
    else if (shm_status == SHM_HAS_EXIST)
        printf(" shared memory has exist.\n");

    addr = shm_at(shmid);
    if (addr == NULL){
        printf("shm at failed.\n");
        return -1;
    }

    memset(addr,'A',10);



    shm_dt(addr);
    return 0;
}
```

全文完