嵌入式开发八股文整理

苏嵌教育
已认证账号

关注

40 人赞同了该文章

1、C/C++

1.1 关键字

volatile

当声明指向设备寄存器的指针时一定要用volatile，它会告诉编译器不要对存储在这个地

址的数据进行假设。中断服务程序中修改的供其他程序检测的变量。 中断中直接从变量地址中读取数据，而不是从寄存器中读取。多线程应用中被几个任务共享的变量。

static

1、函数体内的变量，这个变量只被声明一次。

2、在模块内的变量，表示只能被模块内函数使用，不能被模块外函数访问，表示本地全局变量

3、模块内的函数，限制在模块内使用，同上。

extern

1、引用同一文件变量使用在声明之前时，可以使用关键字extern，让声明在程序任意位置。

2、引用另一个文件中的变量extern可以引用其他文件中的全局变量，而且extern只需要指明数据类型和extern int num=4; 这样不行。

3、引用另一个文件中的函数可以不用包含头文件引用函数。

new/delete malloc/free

1、new/delete是操作符，malloc/free是库函数

2、new/delete可以调用构造函数/析构函数，m/f 只是分配内存。

struct 和 union区别

1、联合体公用一块地址空间，联合体变量长度等于最长的成员的长度

2、对不同成员赋值，会将其他成员重写。

const

1、定义变量为常量

2、修饰参数为常量

3、修饰返回值为常量

sizeof和strlen

1、sizeof是运算符，strlen是函数

2、sizeof可以用类型、函数作为参数，strlen只能计算char*，还必须以/0结尾

3、sizeof编译的

1、都是替对象去一个别名，增强程序的可读性

2、define为预处理指令，不做正确性检查，只有带入之后才能发现

3、typedef用来定义类型别名，不止包含内部类型还包含自定义类型（与机器无关），方便记忆

4、define不仅可以给类型取别名，还能定义常量、变量、编译开关。

5、define没有作用域限制，typedef有。

 # define还是 const，谁定义常量最好

1、define只是文本替换，声明周期止于编译期，不分配内存空间，存在于代码段。const常量存在于数据段，堆栈中分配了空间。

2、const有数据类型，编译器可以对const进行安全检查。

3、const有保护常量不被修改的作用，提高程序的健壮性。

总结：一般倾向于用const定义常量

1.2 内存

C语言内存分配方式

1、静态储存区分配

2、栈上分配

3、堆上分配

C++内存管理是怎样的

分为代码段、数据段、BSS段、堆区、栈区、文件映射区

代码段：分为只读区和文本区，只读取储存字符串常量，文本区储存机器代码。

数据段：储存以及初始化的全局变量和静态变量

BSS段：储存未初始化的全局变量和静态变量，以及初始化为0的全局和静态。

堆区：手动分配的内存

栈：局部变量参数返回值等

映射区：储存动态链接库，mmap函数的文件映射

堆和栈的区别

1、申请方式。 栈为操作系统自动分配/释放，堆为手动

2、申请大小，栈空间有限，向低地址拓展的连续区域，堆是向高地址拓展的不连续区域，链表储存的空闲地址。

3、申请效率，栈是系统自动分配，速度快，不可控。堆是由new分配，速度比较慢，容易产生内存碎片。

栈的作用

1、储存临时变量

2、多线程编程的

申请了没有释放，由程序申请的一块内存，没有任何指针指向它，这个内存就泄露了。

避免内存泄漏方法

1、分配的内存以链表管理，使用完毕后从链表删除，程序结束的时候检查链表

2、良好的编程习惯，在设计内存的程序段，检验出内存泄漏，使用了内存分配的函数，使用完毕后将使用的相应函数释放掉

3、smart pointer

指针

数组指针和指针数组

int （*p）[20]; 数组指针，本质是一个指针，指向一个数组

int *p[20]; 指针数组，本质是一个数组，里面装的是指针。

函数指针和指针函数

1、函数指针 int（*p）(int,int);本身是一个指针，指向一个函数的地址

2、指针函数 int *p(int,int); 指针函数表示一个函数，返回数是指针。

数组名和指针区别

1、指针保存的是地址，数组保存的是数据，单数组名是第一个元素的地址

2、指针间接访问，数组直接下标或者偏移量

3、sizeof 有区别，指针为指针大小，数组为全体数据大

指针常量，常量指针、指向常量的指针

1、int *const p 指向地址不变，地址值可变

2 int const *p 指向地址可变，地址值不能边

3、const int * const p 都不能变

指针与引用区别

1、都是地址，指针是地址，应用是别名

2、引用本质是指针常量，对象不变，对象的值可变

3、++不同，指针是地址自增，引用是对象自增

4、指针需要解引用

5、指针可为空，引用不行

6、sizeof不同 一个是指针大小一个是对象大小

野指针

1、指向不可用内存的指针，指针被创建时如果没有初始化就是野指针

2、指针被free、delete时没有指向NULL就是野指针

3、指针超出了变

C++智能指针是指一个类，用来存储指针

1.3 预处理

预处理器标识#error的目的是什么?

1、遇到#error就会生成一个编译错误提示信息，并停止编译

define声明一年多少秒

#define SECOND_OF_PER_YEAR (3652460*60)UL

#include"" 和 include<>区别

<>号先搜索标准库搜索系统文件比较快， “” 号先搜索工作路径搜索自定义文件比较快

头文件作用

1、通过文件调用库功能，源码保护

2、头文件加强类型安全检查，编译器报错

头文件定义静态变量

1、资源浪费，每个头文件都会单独存在一个静态变量

不使用流程控制语句，打印1~1000数字

```
#include<stdio.h>
#define A(x) x;x;x;x;x;x;x;x;x;x;
int main()
{
        int n=1;
        A(A(A(printf("%d",n++))));
        return 0;
}
```

1.4 变量

全局变量和静态变量

1、全局变量作用域为程序块，局部变量为当前函数

2、全局变量储存在静态区，后者为栈

3、全局变量生命周期为主函数，局部变量生命周期在局部函数中，甚至循环体内

1.5 函数

写个函数在main函数执行前执行

1、attribute可以设置函数属性

```
#include <stdio.h>
void before() __attribute__((constructor));
void after() __attribute__((destructor));
void before() {
  printf("this is function %s\n",__func__);
  return;
}
void after(){
```

```
}
int main(){
  printf("this is function %s\n",__func__);
  return 0;
}
// 输出结果
// this is function before
// this is function main
// this is function after
```

为什么析构函数必须是虚函数

1、基类指针指向子类时，释放基类指针也能释放掉子类的空间，防止内存泄漏。

2、最好是作为父类的类的析构函数作为虚函数

为什么C++默认的析构函数不是虚函数?

1、虚函数有额外的虚函数表和虚指针表，占用额外的内存，对于那些不会被继承的类当然也不需要虚函数作为析构函数。

静态函数和虚函数的区别?

1、静态函数编译时确定运行时机

2、虚函数运行时动态绑定，并且使用虚函数表，内存开销增加

重载与覆盖

1、覆盖是子类和父类的关系，垂直关系，重载是一个类之间的关系，水平关系

2、覆盖一对一，重载多个方法

3、覆盖由对象类型决定，重载根据调用的参数表决定

虚函数表实现多态方法

原理:

虚函数表示一个类的地址表，子类创建时，按照函数声明吮吸会将函数的地址存在虚函数表中。子类重写父类虚函数的时候，父类虚函数表中的位置会被子类虚函数地址覆盖。

C语言函数调用方法

1、使用栈来支持函数调用操作，栈被用来传递参数，返回值，局部变量等。

2、函数调用主要操作栈帧结构

select函数

```
nt select(int maxfdp,fd_set *readfds,fd_set *writefds,fd_set *errorfds,struct timeval *
```

fork wait exec函数

1、附近产生的子进程使用fork拷贝出一个父进程的副本

数组的下标可以为负数吗?

可以，数组下标指地址偏移量，根据偏移量能定位得到目标地址。

inline函数和宏定

2、在编译的时候，内联函数直接被嵌入到目标代码中去，而宏只是一个简单的文本替换。

3、内联函数可以进行诸如类型安全检查、语句是否正确等编译功能，宏不具有这样的功能。

4、宏不是函数，而inline是函数。

5、宏在定义时要小心处理宏参数，一般用括号括起来，否则容易出现二义性。而内联函数不会出现二义性。

6、inline可以不展开，宏一定要展开。因为inline指示对编译器来说，只是一个建议，编译器可以选择忽略该建议，不对该函数进行展开。

7、宏定义在形式上类似于一个函数，但在使用它时，仅仅只是做预处理器符号表中的简单替换，因此它不能进行参数有效性的检测，也就不能享受C++编译器严格类型检查的好处，另外它的返回值也不能被强制转换为可转换的合适的类型，这样，它的使用就存在着一系列的隐患和局限性。

宏和函数的优缺点

1、函数调用时，先求出实参表达式的值，然后带入形参。而使用带参数的函数只是进行简单的字符替换

2、函数调用实在程序运行时处理的，分配的临时的内存单元；而宏展开则是在编译时进行的，在展开时不分配i内存单元，不进行值的传递，也没有"返回值的概念"

3、函数实参形参都要定义类型，二者要求一致 ，宏不存在类型问题，宏没有类型，宏的参数只是一个符号代表，展开时代入指定的字符就行，宏定义时字串可以是任意内心的数据

4、函数只可以得到一个返回值，宏可以设法得到多个

5、使用宏次数多时，展开后源程序长，每次展开都使程序增长，而函数调用不使源程序变长。

6、宏的替换不占用时间，只占用编译时间，函数调用占用运行时间。

简单回答：宏由编译计算，增加编译时间，函数运行的时候计算，增加运行时间；函数的返回值入口参数有数据类型，宏只是简单的符号加减。

ASSERT（）作用

ASSERT()是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。

strcpy()和memcpy()的区别

1、复制的内容不同。strcpy只能复制字符串，而memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。

2、复制的方法不同。strcpy不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy则是根据其第3个参数决定复制的长度。

3、用途不同。通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy

1.6 位操作

求解整型类二进制表示1的个数

```
int func(int x)
{
        int countx = 0;
    while(x)
    {
        countx+
        x = x&(
```

```
}
```

求解整型类二进制表示0的个数

```
int CountZeroBit(int num)
{
    int count = 0;
        while (num + 1)
        {
            count++;
            num |= (num + 1); //算法转换
        }
    return count;
}
```

给定一个整型变量a，写两段代码，第一个设置a的bit 3，第二个清除a 的bit 3。在

以上两个操作中，要保持其它位不变。

```
void clearbit3(int a)
{
        a&=~(1<<3);
}
void setbit3(int a)
{
        a|=1<<3;
}
```

1.7 容器与算法

map与set区别和底层实现

1、底层实现都是红黑树

2、map是键值对，关键字起到索引作用，值表示与索引相关联的数据，set是关键字的集合并且每个元素只包含一个关键字。

3、set迭代器是const不能修改元素值，map允许修改value不能修改key

4、map支持下标操作，set不支持，map可以用key作为下标，set用find

STL的allocator有什么作用?

1、内存配置有alloc::allocate()负责，内存释放由alloc::deallocate()负责；对象构造由::construct()负责，对象析构由::destroy()负责。

2、提升内存管理效率， STL采用了两级配置器，当分配的空间大小超过128B时，会使用第一级空间配置器；当分配的空间大小小于128B时，将使用第二级空间配置器。第一级空间配置器直接使用malloc()、realloc()、free()函数进行内存空间的分配和释放，而第二级空间配置器采用了内存池技术，通过空闲链表来管理内存。

STL迭代器如何删除元素?

对于序列容器vector,deque来说，使用erase(itertor)后，后边的每个元素的迭代器都会失效，但是后边

每个元素都会往前移动一个位置，但是erase会返回下一个有效的迭代器；

对于关联容器map set来说，使用了erase(iterator)后，当前元素的迭代器失效，但是其结构是红黑树，

可。

对于list来说，它使用了不连续分配的内存，并且它的erase方法也会返回下一个有效的iterator，因此上

面两种正确的方法都可以使用

STL中map与unordered_map有什么区别?

1、map底层红黑树实现，unordered_map采用hash表实现'

2、map中序遍历有序，un——map无序

vector和list的区别是什么

1、vector为数组实现，list为双向链表

2、vector支持随机访问，list不行

3、vector顺序储存，list随机

4、vector一次性分配内存，不够才二倍扩容，list一个个分配

5、vector随机访问性能好，插入删除比较慢，list反之

迭代器与指针

1、迭代器又名游标模式，提供一种顺序访问一个聚合对象中各个元素，但又不暴露该对象的内部表示。

2、迭代器是类模板，表现得象指针，重载了指针一些操作，封装了指针，指针的++只是递增地址，但是不能对list生效，迭代器可以。

3、迭代器有着更良好的用法begin，end等不用担心越界

STL里resize和reserve的区别是什么?

1、resize改变当前容器内含有元素的数量，会新增元素0，reserve只是增加空间，不新增元素。

1.8 类和数据抽象

c++类成员访问权限

1、C++通过 public、protected、private 三个关键字来控制成员变量和成员函数的访问权限

2、类内随便访问，类外通过对象访问，且只能访问public成员

引用与指针的区别

1、引用必须被初始化，指针不必。

2、引用初始化以后不能被改变，指针可以改变所指的对象。

3、不存在指向空值的引用，但是存在指向空值的指针。

struct与class区别

1、c++中两者都可以定义类，但是struct没有权限，默认public

面对对象和泛型编

2、泛型编程让类型参数化，使程序可以从逻辑功能上抽象。吧处理的对象当成参数来传递

右值引用，和左值的区别。

左值可以寻址，而右值不可以。

左值可以被赋值，右值不可以被赋值，可以用来给左值赋值。

左值可变,右值不可变（仅对基础类型适用，用户自定义类型右值引用可以通过成员函数改变）。
C++的类和C里面的struct有什么区别?

析构函数可以为 virtual 型，构造函数则不能，为什么?

1、虚函数主要是用作多态，如果构造函数也用了，那么派生类必须在初始化列表给基类参数初始化

2、构造函数运行的时候对象的动态类型还不完整，没法确定没所以不能动态绑定。

C++的类和C里面的struct有什么区别?

c++中的类具有成员保护功能，并且具有继承，多态这类特点，而c里的struct没有

c里面的struct没有成员函数,不能继承,派生等等.

C++中空类默认产生哪些类成员函数?

1、构造函数

2、拷贝构造

3、析构函数

4、赋值运算符重载函数

5、取值运算符重载函数

6、const取址运算符重载函数

静态成员函数与非静态成员函数的区别

前者没有 this 指针,后者有 this 指针。

静态成员函数只要用来访问静态数据成员,而不访问非静态成员

1.9 面对对象

面向对象和面向过程有什么区别?

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了；面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

1、面对对象以对象为中心，面向过程以过程为中心

2、面对对象把代码封装成一个整体，其他对象不能直接修改其数据。面向过程直接使用程序来处理数据，各模块存在控制与被控制的关系。

3、面对对象是将问题分为不同的对象，给予对象赋予属性和行为。面对过程则是将事件分为不同的步骤，按照步骤完成编程。

面对对象的基本特

2、继承：子类继承父类的功能

3、多态：不同的对象对从父类继承的同一动作做出不同的反应，

4、抽象:不打算了解问题全部，只关注当前目标。过程抽象和数据抽象。过程抽象是指任何操作都被当成实体看待，不在乎它是不是由其他子函数完成。

什么是深拷贝？什么是浅拷贝？

1、深拷贝复制一份

2、浅拷贝哟与可能共享成员变量

友元

1、友元函数：普通函数对一个访问某个类中的私有或者保护成员

2、友元类：类A中的成员函数访问类B中的私有或保护成员。

初始化列表和构造函数初始化的区别?

```
Example::Example() : ival(0), dval(0.0) {} //初始化列表的构造函数
Example::Example()   //构造函数
{
ival = 0;
dval = 0.0;
```

结果是一样的，使用初始化列表的构造函数表示 初始化类的成员，使用初始化列表的构造函数是对类成员的赋值，而不是初始化。所以一下情况需要对成员初始化所以必须用初始化列表的方法。

1、成员类型为没有默认构造函数的类

2、const成员或引用类型的成员

类的成员变量的初始化顺序是什么?

1、成员变量在使用初始化列表初始化时，与构造函数中初始化成员列表的顺序无关，只与定义成员变量的顺序有关。

2、不使用初始化列表的话就与构造函数有关。

Public继承、protected继承、private继承的区别?

1、public继承就是公有继承完还是公有，保护还是保护，私有还是私有

2、protected继承就是公有变保护，保护还是保护，私有还是私有

3、private继承就是所有变成私有

1.10 虚函数

虚函数注意内容

1、只需要在声明的函数体中使用关键字virtual将函数声明为虚函数，定义中不需要

2、基类某一成员为虚函数之后，派生类中的同名函数自动成为虚函数

3、非类的成员函数不能定义为虚函数，全局函数以及类的成员函数和构造函数也不能定义为虚函数，可以将析构函数定义为虚函数

什么函数不能声明

1.11数据结构

链表和数组的区别

数组在内存中栈上按顺序存储的，而链表是在堆上随机存储的。

要访问数组中的元素可以按下标索引来访问，速度比较快，如果对他进行插入操作的话，就得移动很多元素，所以对数组进行插入操作效率很低. 由于连表是随机存储的，链表在插入，删除操作上有很高的效率（相对数组）

如果要访问链表中的某个元素的话，那就得从链表的头逐个遍历，直到找到所需要的元素为止，所以链表的随机访问的效率就比数组要低 。

2、ARM体系与架构

2.1 硬件基础

NAND FLASH 和NOR FLASH异同?

类别 读 写 擦除 可靠性 容量 用途 价格

NOR 快 慢 非常慢 比较高 小 保存代码 高

NAND 快 快 快 低 大 保存数据 低

CPU,MPU,MCU,SOC,SOPC联系与差别?

1、CPU：是一台计算机的运算核心和控制核心

2、MPU: 微处理器稍强的CPU

3、MCU：将计算机的CPU、RAM、ROM、定时计数器和多种I/O接口集成在一片芯片上。

4、SOC： 系统级芯片不单单是放简单的代码，可以放系统级的代码，也就是说可以运行操作系统

CPU中cache的作用？cache的基本组织结构?

（1）高速缓冲存储器Cache是位于CPU与内存之间的临时存储器，它的容量比内存小但交换速度快。

在Cache中的数据是内存中的一小部分，但这一小部分是短时间内CPU即将访问的，当CPU调用大量数据时，就可避开内存直接从Cache中调用，从而加快读取速度。由此可见，在CPU中加入Cache是一种高效的解决方案，这样整个内存储器（Cache+内存）就变成了既有Cache的高速度，又有内存的大容量的存储系统了。

（2）全相连映射，直接映射，组相连映射

交叉编译

在一种计算机环境中运行的编译程序，能编译出在另外一种环境下运行的代码，我们就称这种编译器支持交叉编译。这个编译过程就叫交叉编译。

C/C++的编译包括几个部分

1、预编译：预处理器对c程序进行一些预处理工作，例如对宏定义的变量进行替换；

1）将所有的#define删除，并展开所有的宏定义；

2）处理所有的预编译指令，例如：#if #elif #else #endif;

4）添加行号信息文件名信息，便于调试；

5）删除所有的注释：// /**/;

6）保留所有的#pragma编译指令，因为在编写程序的时候，我们经常要用到#pragma指令来设定编译器的状态或者是指示编译器完成一些特定的动作；

最后生成.i文件；

总的来说，包括（1）去注释 （2）宏替换 （3）头文件展开 （4）条件编译

2、编译：编译器将c语言程序翻译成汇编语言程序；

1）扫描，语法分析，语义分析，源代码优化，目标代码生成，目标代码优化；

2）生成汇编代码；

3）汇总符号；

4）生成.s文件；

3、汇编：汇编语言通过汇编器编译成可重定位目标程序.o，与之相反称为反汇编；

1）根据汇编指令和特定平台，把汇编指令翻译成二进制形式；

2）合并各个section，合并符号表；

3）生成.o文件；

4、链接：将目标文件和所需的库函数用链接器进行链接，常见的链接器有Unix；

1）合并各个.obj文件的section，合并符号表，进行符号解析；

2）符号地址重定位；

3）生成可执行文件；

描述一下嵌入式基于ROM的运行方式和基于RAM的运行方式有什么区别?

基于RAM：

1、将硬盘或者其介质的代码加载到ram中。

2、速度快但是可用RAM少，因为自身的空间要存一部分代码

基于ROM:

1、将部分代码搬到RAM中去，所以可用RAM资源比基于RAM的多。

2.2 中断与异常

中断与异常区别

1、中断是指外部硬件产生的一个电信号从CPU的中断引脚进入，打断CPU的运行，异常是指软件运行过程中发生了一些必须作出处理的事件，CPU自动产生一个陷入来打断CPU的运行。

2、异常处理的时候要考虑与处理器的时钟同步，异常被称为同步中断

中断能不能睡眠 为什么?

1、一般说中断上下文中不能睡眠,这个中断是指硬件事件发生,触发CPU停止当前活动转而去处理硬件请求.

包了,把数据包从网卡缓存拷贝到主存(可以由DMA完成,但寄存器的修改以及资源设定还是要由cpu去做)的逻辑就需要cpu立即去做,不做的话,网络新来的数据包就可能丢失.所以这些紧要操作逻辑为硬中断处理.

3、下半部有很多种机制,其中就包括软中断,还有tasklet,workqueue等,软中断只是其中的一种,由于历史的原因,有时候是混淆称呼下半部和软中断的.

4、而可以看到软中断逻辑不属于任何进程,所以才不能睡眠,因为一旦睡眠,cpu切换出去,就切不回来了。

简单说就是：唤醒函数针对进程而言的，下半部的中断不属于进程，所以无法被唤醒

中断的响应执行流程是什么?

cpu接受中断->保存中断上下文跳转到中断处理历程->执行中断上半部->执行中断下半 部->恢复中断上下文。

写一个中断服务需要注意哪些？如果中断产生之后要做比较多的事情你是怎么做的?

1、快进快出，在中断服务函数里尽量快速采集信息。

2、中断中不能有阻塞操作

3、中断服务函数注意返回值，使用操作系统定义的宏，而不是自己定义的。

4、做的事情较多，将这些任务放在后半段tasklet处理。

中断和轮询哪个效率高？怎样决定是采用中断方式还是采用轮询方式去实现驱动?

1、中断是CPU处于被动状态下来接受设备的信号，而轮询是CPU主动去查询该设备是否有请求。

2、请求设备是一个频繁请求cpu的设备，或者有大量数据请求的网络设备，那么轮询的效率是比中断高。

3、如一般设备，并且该设备请求cpu的频率比较低，则用中断效率要高一些。主要是看请求频率。

2.3 通讯协议

异步传输与同步传输?

异步传输：是一种典型的基于字节的输入输出，数据按每次一个字节进行传输，其传输速度低。

同步传输：需要外界的时钟信号进行通信，是把数据字节组合起来一起发送，这种组合称之为帧，其传输速度比异步传输快。

RS232和RS485区别?

传输方式不同。 RS232采取不平衡传输方式，即所谓单端通讯。 而RS485则采用平衡传输，即差分传输方式。

传输距离不同。RS232适合本地设备之间的通信，传输距离一般不超过20m。而RS485的传输距离

为几十米到上千米。

设备数量。RS232 只允许一对一通信，而RS485 接口在总线上是允许连接多达128个收发器。

连接方式。RS232，规定用电平表示数据，因此线路就是单线路的，用两根线才能达到全双工的目的；而RS485， 使用差分电平表示数据，因此，必须用两根线才能达到传输数据的基本要求，要实现全双工，必需用4根线。

SPI协议

接口：输出线、输入线、时钟线、片选信号线

1、片选信号线由高到低是SPI的起始信号，从机检测到自己的NSS线起始信号之后就知道自己被选中了，然后由低到高是停止信号。

2、SPI 使用 MOSI 及 MISO 信号线来传输数据，使用 SCK 信号线进行数据同步。 在时钟线上升沿触发输出，在下降沿被采样。

IIC协议

1、IIC协议是由数据线SDA和时钟SCL构成的串行总线，可发送和接收数据,是一个多主机的半双工通信方式

2、空闲状态

SDA与SCL都处于高电平，就是空闲状态。

2、起始信号

时钟线为高，数据线由高到低就是启动信号，只能由主机发起空闲状态下才能启动该信号

3、停止信号

时钟为高，数据线由低到高就是停止信号

4、传输数据格式

SCL为高就会获取SDA数据值，SDA在这期间必须稳定

SCL为低便是SDA电平变化状态，在此期间SDA可以自由变化

可以主动拉低SCL让IIC进入等待状态知道处理结束再释放SCL数据传输会继续

5、ACK应答信号

发送方在第9个时钟脉冲奇迹爱你释放SDA数据，当接收方接收成功时，会输出一个应答信号，低电平有效

6、写操作

start信号–设备地址–方向（读、写）。回应（确定这个设备是否存在）–发送数据–回应–发送完之后主芯片发送一个停止信号。

白色主到从、灰色从到主。

7、读操作

除了数据需要主到从，其余差不多。

嵌入式编程中，什么是大端？什么是小端？

大端模式：低位字节存在高地址上，高位字节存在低地址上。

小端模式：高位字节存在高地址上，低位字节存在低地址上。

```
//第一种
union w{
    int a;
    char b;
}c;
c.a = 1;
if(c.b==1) prir
```

```
int a = 0x12345678;
char *p = (char *)&a;
if(*p==0x78)printf("小端");
else printf("大端");
```

3、Linux驱动

3.1 指令

Linux指令

查看当前进程 ps；

执行退出 exit；

查看当前路径 pwd；

查看目录 ls -a显示所有文件及目录，-l详细列出

创建目录 mkdir；

创建文件 vi 、 touch；

查看文件内容 vi，cat ；

屏幕输出 echo；

常用的GCC命令

gcc -E test.c -o test.i #把预处理的结果导出到test.i文件

gcc -S test.i -o test.s #编译器将test.i翻译成汇编语言，并将结果存储在test.s文件中。

gcc -c test.s -o test.o #将汇编代码编译为目标文件（.o）但不链接

gcc test.o -o test #将生成的目标文件test.o生成最终的可执行文件test

gcc test.c -o test #将源文件test.c编译链接为可执行文件test

gcc test1.c test2.c -o test 多文件编译

常用的GDB调试指令

gcc -g test.c -o test #编译时生成debug有关的程序信 就是说正常编译不能使用GDB

list 查看源码

next #单步调试（逐过程，函数直接执行）,简写n

step #单步调试（逐语句：跳入自定义函数内部执行）,简写s

run #运行程序

break + num #设置第num行 为断点

continue #继续运行到下一个断点。

display 追踪具体变量值

delete breakpoints num #删除第num个断点

insmod\modprobe 加载驱动

rmmod #卸载驱动

lsmod #查看已有的字符设备信息

cat /proc/interrupt #查看已有的中断号

Makefile

经典malefile main包含了input、calcu的头文件

```
objects = main.o input.o calcu.o
main: $(objects)
        gcc -o main $(objects)
%.o : %.c
        gcc -c $<
clean:
        rm *.o
        rm main
```

shell相关操作

要求

A、在Linux操作系统启动的时候，自动加载/mnt/test/test程序。

B、当test异常退出之后，自动重新启动。

C、当test程序重启次数超过100次，自动复位操作系统。

假设你所拥有的资源：

A、目标机器是一台具有标准shell的嵌入式计算机，CPU为ARM7 56MB，内存16MB，软件环境基于Linux2.6.11和BusyBox1.2构建。

B、当前已有11个用户进程在运行，占用了大部分的CPU时间和内存，你可使用的内存只有2MB左右，CPU时间由系统分派。

```
#!/bin/sh

#load *.so that may need
if [ -r /sbin/ldconfig ]; then
ldconfig
fi

#add the libs PATH that may need
export LD_LIBRARY_PATH="/lib"

#count is the counter of test started times
count=0

#main loop
while [ 1 ] ;do
#add execute property for /mnt/test/test
chmod +x /mnt/test/test
#start test
/mnt/test/test
#the running times counter
let count=count+1
echo "test running times is $count"
#Is test runnir
if [ "$count" -
```

```
fi
#wait for test stoping...
sleep 3
done
```

3.2 uboot

bootloader

1、Linux启动需要一个bootloader程序，初始化时钟、中断或者其他外设，然后将Linux内核从flash拷贝到SDRAM中，最后启动Linux内核。

2、Bootloader就是一小段程序，它在系统上电时开始执行，初始化硬件设备、准备好软件环境，最后调用操作系统内核。

uboot启动流程

u-boot系统启动流程 ,大多数bootloader都分为stage1和stage2两部分, u-boot也不例外。

依赖于CPU体系结构的代码（如设备初始化代码等）通常都放在stage1且可以用汇编语言来实现，而stage2则通常用C语言来实现，这样可以实现复杂的功能，而且有更好的可读性和移植性。

1、Stage1 start.S代码结构 u-boot的stage1代码通常放在start.S文件中，他用汇编语言写成，其主要代码部分如下

（1） 定义入口:该工作通过修改连接器脚本来完成。

（2）设置异常向量（Exception Vector）。

（3）设置CPU的速度、时钟频率及终端控制寄存器。

（4）初始化内存控制器。

（5）将ROM中的程序复制到RAM中。

（6）初始化堆栈。

（7）转到RAM中执行，该工作可使用指令ldr pc来完成

2、Stage2

C语言代码部分 lib_arm/board.c中的start arm boot是C语言开始的函数也是整个启动代码中C语言的主函数，同时还是整个u-boot（armboot）的主函数，该函数只要完成如下操作：

（1）调用一系列的初始化函数。

（2）初始化Flash设备。

（3）初始化系统内存分配函数。

（4）如果目标系统拥有NAND设备，则初始化NAND设备。

（5）如果目标系统有显示设备，则初始化该类设备。

（6）初始化相关网络设备，填写IP、MAC地址等。

（7）进去命令循环（即整个boot的工作循环），接受用户从串口输入的命令，然后进行相应的工作

uboot启动过程中

2、初始化一个串口，检测系统内存映射，将内核映象和根文件系统映象从 Flash上读到SDRAM空间中，为内核设置启动参数，调用内核。

uboot和内核如何完成参数传递?

1、完成相关设置：CPU寄存器设置 R0=0 R1=机器类型ID R2=启动参数标记列表在RAM中起始基地址，设置禁止中断，SVC模式（超级用户模式，有利于硬件初始化）MMU关闭

2、uboot把机器ID通过R1传递给内核，R2存放块内存的基地址，这块内存主要存放uboot给Linux内核的其他参数，参数很多所有需要按规定存放，标记是一种数据结构。

3、标记的数据结构为tag，它由一个tag_header结构和一个联合（union）组成

为什么uboot要关掉caches?

caches是cpu内部的一个2级缓存，它的作用是将常用的数据和指令放在cpu内部。caches是通过CP15

管理的，刚上电的时候，cpu还不能管理caches。上电的时候指令cache可关闭，也可不关闭，但数据

cache一定要关闭，否则可能导致刚开始的代码里面，去取数据的时候，从cache里面取，而这时候RAM中数据还没有caches过来，导致数据预取异常 。

3.3 文件系统

什么是根文件系统?

1、内核启动时所挂载（mount）的第一个文件系统，内核代码的映像文件保存在根文件系统中。

2、挂载之后会把一些初始化脚本和服务加载到内存中去运行。

根文件系统为啥这么重要?

1、根文件系统包含系统启动时所必须的目录和关键性的文件，以及使其他文件系统得以挂载（mount）所必要的文件。比如shell命令程序必须运行在根文件系统上，譬如ls、cd等命令。

2、一套linux体系，只有内核本身是不能工作的，必须要rootfs（上的etc目录下的配置文件、/bin

/sbin等目录下的shell命令，还有/lib目录下的库文件等）相配合才能工作。

3.4 中断

硬中断 / 软中断是什么？有什么区别?

1、硬中断是由硬件产生的，软中断是执行中断指令产生的。

2、硬中断可以直接中断CPU，软中断并不会直接中断CPU。也只有当前正在运行的代码（或进程）才会产生软中断。

3、硬中断可屏蔽、软中断不可屏蔽

4、硬中断又称上半部，要快速完成任务

中断为什么要区分上半部和下半部?

1、调用过程：外部中断产生->发送中断信号到中断控制器->通知处理器产生中断的中断号

2、为了能被新的中断打断，将中断处理一分为二，上半部登记新的中断，处理快速简单的任务

复杂耗时的任务给

linux中断的响应执行流程?

cpu接受中断->保存中断上下文跳转到中断处理历程->执行中断上半部->执行中断下半部->恢复中断上下文。

3.5 Linux驱动模型

字符设备驱动模型

请简述主设备号和次设备号的用途

主设备号：主设备号标识设备对应的特定的驱动程序。虽然现代的linux内核允许多个驱动程序共享主设备号，但我们看待的大多数设备仍然按照“一个主设备对应一个驱动程序”的原则组织。

次设备号：次设备号由内核使用，用于确定由主设备号对应驱动程序中的各个设备。依赖于驱动程序的编写方式，我们可以通过次设备号获得一个指向内核设备的直接指针，也可将此设备号当作设备本地数组的索引。（多个设备共用一套程序的话，主设备号代表这个驱动程序，每个设备一个次设备号）

创建设备文件

1、手动创建

mknod /dev/led c 250 0 ， 其中dev/led 为设备节点 ,c 代表字符设备, 250代表主设备号, 0代表次设备号。

设备驱动程序中如何注册一个字符设备?

1、将cdev结构嵌入到自己的设备特定结构中

```
void cdev_init(struct cdev *cdev, struct file_operations *fops)
```

2、早期注册函数

```
int register_chrdev(unsigned int major, const char *namem , struct file operations *fo
```

Linux设备中字符设备和块设备有什么主要区别?

字符设备：提供连续的数据流，应用程序可以顺序读取，通常不支持随机存取。

块设备：应用程序可以随机访问设备数据，程序可自行确定读取数据的位置。

驱动中操作物理绝对地址为什么要先ioremap?

1、ioremap是将io地址空间映射到虚拟地址空间上去，便于操作。

2、因为保护模式下的cpu只认虚拟地址，不认物理地址，所以你要操作外设上的寄存器必须先映射到虚拟内存空间，拿着虚拟地址去跟cpu对接，从而操作寄存器。

Linux移植ARM的基本步骤和完成的任务

1）首先是准备工作，包括下载源码、建立交叉编译环境等；

2）然后是配置和编译内核，必要时还要对源码做一定的修改；

3）第三步就是需要制作文件系统（如RAM disk）来挂接根文件系统；

4）最后是下载、调试内核并在fs中添加自己的应用程序

ARM-linux启动分为四个部分：引导加载程序（bootloader），Linux内核，文件系统，应用程序。

bootloader是系统启动和复位后执行的第一段代码，它主要用来初始化处理器及外设，然后调用Linux内核。Linux内核在完成系统的初始化之后需要挂载某个文件系统作为根文件系统（root filesystem）。根文件系统是Linux系统的核心组成部分，它可以作为Linux系统中文件和数据的存储区域，通常它还包括配置文件运行应用程序所需要的库。应用程序实现该嵌入式产品所要实现的目标。

4、操作系统

什么是进程？什么是线程？

进程是资源分配的基本单位，它是程序执行时的一个实例，在程序运行时创建。

线程是程序执行的最小单位，是进程的一个执行流，一个线程由多个线程组成的。

进程和线程有什么区别？

1、进程是资源分配的基本单位，线程是程序运行的基本单位

2、进程有自己的资源空间，线程是共享进程中的数据，所以进程切换开销更大一点

3、线程通讯要简单一些，因为共享全局变量等

4、线程执行开销小，进程执行开销大。

5、多线程中一个线程死掉整个进程也死了，一个进程死掉不会影响其他进程，因为它有独立的地址空间。

何时使用多进程，何时使用多线程？

对资源的管理和保护要求高，不限制开销和效率时，使用多进程。

要求效率高，频繁切换时，资源的保护管理要求不是很高时，使用多线程。

进程有几种状态？

创建状态

就绪状态

运行状态

阻塞状态

终止状态

进程间通信方式

管道(pipe)

管道这种通讯方式有两种限制，一是半双工的通信，数据只能单向流动，二是只能在具有亲缘关系的进程间使用

信号量(semophore)

信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。

消息队列是由消息组成的链表，存放在内核中并由消息队列标识符标识。

共享内存(shared memory)

共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以

访问。共享内存是最快的 IPC 方式，它是针对其他进程间通信方式运行效率低而专门设计的。它往往与

其他通信机制，如信号量，配合使用，来实现进程间的同步和通信。

套接字(socket)

套解字也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同机器间的进程通信。

线程间的通讯方式

（1）通过条件变量进行线程间的通信

（2）通过标志位来通知线程间的通信

（3）通过std::furture来进行线程间的通信

线程间同步方法有哪些?

临界区：如果有多个线程试图访问共享资源，那么当有一个线程进入后，其他试图访问共享资源的线程将会被挂起，并一直等到进入临界区的线程离开，临界在被释放后，其他线程才可以抢占。

互斥量：为协调对一个共享资源的单独访问而设计，只有拥有互斥量的线程，才有权限去访问系统

的公共资源，因为互斥量只有一个，所以能够保证资源不会同时被多个线程访问。

信号量：为控制一个具有有限数量的用户资源而设计。它允许多个线程在同一个时刻去访问同一个

资源，但一般需要限制同一时刻访问此资源的最大线程数目

事件：用来通知线程有一些事件已发生，从而启动后继任务的开始。

什么是僵尸进程，孤儿进程，守护进程?

僵尸进程是 一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子

进程的状态信息，那么子进程的进程描述符仍然保存在系统中。这种进程称之为僵死进程。

孤儿进程是因为父进程异常结束了，然后被1号进程init收养。

守护进程是创建守护进程时有意把父进程结束，然后被1号进程init收养

请你回答一下fork和vfork的区别?

fork( )的子进程拷贝父进程的数据段和代码段；vfork( )的子进程与父进程共享数据段

fork( )的父子进程的执行次序不确定；vfork( )保证子进程先运行，在调用exec或exit之前与父进程

数据是共享的，在它调用exec或exit之后父进程才可能被调度运行。

vfork( )保证子进程先运行，在它调用exec或exit之后父讲程才可能被调度运行。如果在调用这两个

当需要改变共享数据段中变量的值，则拷贝父进程。

堆和栈

什么是代码段，数据段，bss段，堆，栈?

代码段：存放程序执行代码的一块区域，通常是只读

数据段：已初始化的全局变量和已初始化为非0的静态变量

BSS段：未初始化的全局变量和未初始化的静态变量或者初始化为0的静态变量

数据段和BSS段本质上都是静态区，存放全局变量和静态变量的

堆：堆是用来存放进程中被动态分配的内存段。

栈：是用户存放程序临时创建的变量。

为什么堆的空间是不连续的?

1、堆包含一个链表来维护已用和空闲的内存块。

2、分配的空间在逻辑地址（虚拟地址）上是连续的，但在物理地址上是不连续的

什么是用户栈和内核栈?

内核栈 ： 内存中属于操作系统空间的一块区域。

作用:

1、保存中断现场

2、保存调用的参数、返回值、函数局部变量

用户栈:

用户进程空间的一块区域，用于保存用户空间子程序间调用的参数，返回值以及局部变量。

为什么不能共用一个栈:

1、系统栈（内核栈）大小有限用户程序调用次数可能很多。

2、用户栈空间不能提供相应保护措施

线程是否有相同的堆栈?

每个线程有自己的堆栈。

信号、并发和互斥

驱动里面为什么要有并发、互斥的控制？如何实现？讲个例子?

并发：多个执行单元同时对共享资源操作，容易导致竞态。

互斥：一个执行单元在访问共享资源的时候、其他执行单元都被禁止访问。访问共享资源的代码区被称为临界区，临界区需要某种互斥机制加以保护。

自旋锁是什么？信号量是什么？二者有何异同?

自旋锁：一个执行单元在操作资源时，另一个执行单元不能操作。自旋锁只能短期持有

信号量：资源标量

1、长时间持有锁使用信号量，短时间持有使用自旋锁。

2、信号量可以睡眠，其他人需要时也会进入睡眠。

3、信号量代码可以被抢占。

自旋锁和信号量可以睡眠吗？为什么？

自旋锁不能睡眠，信号量可以。

原因：

1、自旋锁自旋锁禁止处理器抢占；而信号量不禁止处理器抢占。

自旋锁和信号量可以用于中断中吗？

信号量不能用于中断中，因为信号量会引起休眠，中断不能休眠。

自旋锁可以用于中断。在获取锁之前一定要先禁止本地中断（也就是本CPU中断，对于多核SOC来说会

有多个CPU核），否则可能导致锁死现象的发生

产生死锁的原因是什么？

多个并发进程因争夺系统资源而产生相互等待的现象。即：一组进程中的每个进程都在等待某个事件发

生，而只有这组进程中的其他进程才能触发该事件，这就称这组进程发生了死锁。

原因：

1、系统资源有限

2、进程推进顺序不合理

如何避免死锁？

1、线程按一定顺序加锁

2、获取锁时加上时限，也就是说超过时间则放弃获取。

3、死锁检测

内存

在1G内存的计算机中能否malloc(1.2G)？为什么？

malloc能够申请的空间大小与物理内存的大小没有直接关系，仅与程序的虚拟地址空间相关。

内存管理的方法

1、块式管理

分成一大块一大块，只需要几个字节也给一大块。造成浪费、但方便管理。

2、页式管理

划分地址空间为若干大小区域，被称为页。优点便于管理，缺点页长与逻辑大小没有关系。

3、段式管理和段页式管理

段页式管理：

用分段方法来分配和管理虚拟存储器。用分页方法来分配和管理内存

什么是虚拟内存?

它使得应用程序认为它拥有连续可用的内存（一个连续完整的地址空间），允许程序员编写并运行比实际系统拥有的内存大得多的程序

好处：

1、扩大了地址空间

2、地址保护

3、公平分配内存：每个进程相当于有了同样大小的额外内存

解释下内存碎片，内碎片，外碎片?

1、内存碎片：内存碎片是由于多次进行内存分配造成的，空白段太小无法进行下次分配

2、内碎片：分配给程序的存储空间没有用完，有一部分是程序不使用（没用完），但其他程序也没法用的空间。

3、外碎片：空间太小，小到无法给任何程序分配（不属于任何进程）的存储空间。

解释下虚拟地址、逻辑地址、线性地址、物理地址?

1、虚拟地址、逻辑地址：由程序产生的由段选择符和段内偏移地址组成的地址这两部分组成的地址并没有直接访问物理内存，而是通过分段地址的变换处理后才会对应到相应的物理内存地址。

2、线性地址：指虚拟地址到物理地址变换之间的中间层，是处理器可寻址的内存空间（称为线性地址空间）中的地址。

3、物理地址：是指现在CPU外部地址总线上的寻址物理内存的地址信号，是地址变换的最终结果。

系统调用是什么，你用过哪些系统调用，和库函数有什么区别?

系统调用：系统调用是通向操作系统本身的接口，是面向底层硬件的。通过系统调用，可以使得用户态运行的进程与硬件设备(如CPU、磁盘、打印机等)进行交互，是操作系统留给应用程序的一个接口。

库函数：库函数（Library function）是把函数放到库里，供别人使用的一种方式。.方法是把一些常用到的函数编完放到一个文件里，供不同的人进行调用。一般放在.lib文件中。系统调用是为了方便使用操作系统的接口，而库函数则是为了人们编程的方便。

区别：

1、库函数是语言或应用程序的一部分，系统调用是内核提供的接口。

2、库函数在用户地址进行，系统调用在内核地中空间执行

3、库函数有缓冲、系统调用无缓冲。

4、系统调用依赖平台，库函数不用上下文

上下文有哪些？怎么理解?

上下文简单说来就是一个环境

寄存器上下文: 通用寄存器、程序寄存器(IP)、处理器状态寄存器(EFLAGS)、栈指针(ESP)；

系统级上下文: 进程控制块task_struct、内存管理信息(mm_struct、vm_area_struct、pgd、pte)、内核栈

为什么会有上下文这种概念?

系统调用中用户空间会传递很多数据给内核空间，保存上下文以便系统调用结束后回到用户空间继续执行。

5、网络编程

TCP/ UDP

TCP怎么保证可靠性?

1、序列号、确认应答、超时重传

2、窗口控制与高速重发控制/快速重传（重复确认应答）

窗口控制：不需要对每个没收确认的数据重发，只需要确认一个窗口是否都收齐了。

简述一下TCP建立连接和断开连接的过程。

连接三次握手

1、客户端请求，标志位SYN置为1 发送x;

2、服务端回复，标志位SYN和ACK都置为1 回复 x+1,和 y

3、客户端收到后回复 y+1，服务端检查ack是否为1，是的话就连接成功

断开

1、客户端发送 x+2 回复y+1 进入FIN_WAIT_1 状态

2、服务端回复x+3 服务器进入CLOSE_WAIT状态。客户端收到后进入FIN_WAIT_2状态

3、服务端发送完所有数据之后发送y+1 服务器进入LAST_ACK状态

4、客户端回复y+2 客户端进入TIME_WAIT状态等待2MSL（报文段最大生存时间）后关闭

为什么客户端最后还要等待2MSL?

保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度

看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文

它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，

接着给出回应报文，并且会重启2MSL计时器。

为什么是三次握手?

1、为了防止已失效的连接请求报文段突然又送到了，产生错误。

2、客户端发送请求报文滞留，后续才到达时，服务端还是同意每一次浪费资源

为什么是四次挥手？二三次能不能合并

1、TCP是全双工通信，意味着关闭是双方都需要确认的行为。

2、需要时间释放资源，一旦合并需要等很久。

3、客户端会以为自己第一次发的报文没有送到，不断尝试发送第一次的报文。

TCP/UDP

1、TCP面向连接，UDP无连接

2、TCP数据保证正确、顺序正确，UDP可能丢包

3、TCP可靠稳定，但是慢效率低，UDP快，容易丢包

TCP，UDP适用场景?

TCP应用场景

效率要求相对低，但对准确性要求相对高的场景。文件传输（准确高要求高、但是速度可以相对慢）、接受邮件、远程登录。

UDP应用场景

效率要求相对高，对准确性要求相对低的场景。QQ聊天、在线视频、网络语音电话、广播通信

TCP相比UDP为什么是可靠的?

1、确认和重传机制

2、数据排序

3、流量控制 窗口和计时器的使用

4、拥塞控制

什么是OSI七层模型和TCP/IP四层模型？每层列举2个协议。

物理层: 通过媒介传输比特,确定机械及电气规范,传输单位为bit，主要包括的协议为：IEE802.3

CLOCK RJ45

数据链路层: 将比特组装成帧和点到点的传递,传输单位为帧,主要包括的协议为MAC VLAN PPP

网络层：负责数据包从源到宿的传递和网际互连，传输单位为包,主要包括的协议为IP ARP ICMP

传输层：提供端到端的可靠报文传递和错误恢复，传输单位为报文,主要包括的协议为TCP UDP

会话层：建立、管理和终止会话，传输单位为SPDU，主要包括的协议为RPC NFS

表示层: 对数据进行翻译、加密和压缩,传输单位为PPDU，主要包括的协议为JPEG ASII

应用层: 允许访问OSI环境的手段,传输单位为APDU，主要包括的协议为FTP HTTP DNS

TCP/IP

链路层：MAC VLAN PPP

网络层：IP协议、ICMP协议、ARP协议、RARP协议。

传输层：UDP协议

协议），POP3协议（邮局协议），HTTP协议。

TCP/IP数据链路层的交互过程是怎么样的?

网络层等在数据链路层用MAC地址作为通信目标，数据包到达网络层等往数据链路层发送的时候，首先

回去ARP缓存表去查找ip对应的MAC地址，如果查到了，就将此ip对应的MAC地址封装到链路层数据包

的包头。如果缓存中没有找到，则会发起一个广播寻找目的IP的物理地址。

传递到IP层怎么知道报文该给哪个应用程序，它怎么区分UDP报文还是TCP报文?

根据端口继续区分需接受的程序;

根据ip协议头中标识字段：UDP 17 、TCP 6

端口号

熟知的端口号:

20：FTP 数据传输

21/TCP FTP 文件传输协议

23/tcp Telnet 不安全的文本传送

25/tcp SMTP Simple Mail Transfer Protocol (E-mail)

69/udp TFTP Trivial File Transfer Protocol

79/tcp finger Finger

80/tcp HTTP 超文本传送协议 (WWW)

88/tcp Kerberos Authenticating agent

110/tcp POP3 Post Office Protocol (E-mail)

113/tcp ident old identification server system

119/tcp NNTP used for usenet newsgroups

220/tcp IMAP3

443/tcp HTTPS used for securely transferring web pages

ICMP协议属于什么层

1.ICMP是基于IP协议工作的，但是它并不是传输层的功能，因此仍然把它归结为网络层协议

2.ICMP只能搭配IPv4使用，如果是IPv6的情况下, 需要是用ICMPv6

物理地址和IP的转化

地址解析协议（ARP）的作用是将IP地址转换成物理地址；反地址解析协议（RARP）则负责将物理地址转换成IP地址。

从在浏览器地址栏中输入baidu.com到看到百度首页，这个过程中间经历了什么？都涉及到哪些网络协议?

1.客户端浏览器获取用户在地址栏输入的域名。

2.客户端浏览器将域名发送给DNS域名系统，请求解析。

3.DNS解析域名得到相应的IP，返回给客户端浏览器。

4.客户端浏览器根据IP向服务器发起TCP三次握手，建立TCP连接。

5.客户端浏览器向服务器发送HTTP请求，请求百度首页。

6.服务器通过HTTP响应向客户端浏览器返回百度首页文件。

7.释放TCP连接。

8.客户端浏览器解析HTML文件，根据文件内容获取CSS、JS等资源文件，将页面渲染展示给用户。

HTTP/IP

什么是http协议?

1、HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写，是用于从万维网（WWW:World Wide Web）服务器传输超文本到本地浏览器的传送协议。

2、HTTP是一个基于TCP/IP通信协议来传递数据

ICMP协议属于什么层

http协议有什么特点?

1、简单快速 客户向服务器请求服务时，只需传送请求方法和路径

2、无连接 无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。

3、基于TCP协议

4、默认端口80

https建立连接过程是什么?

客户端连接到Web服务器 一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。

发送HTTP请求

通过TCP套接字，客户端向Web服务器发送一个文本的请求报文

服务器接受请求并返回HTTP响应

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取

4．释放连接TCP连接
4．客户端浏览器解析HTML内容

http和https的区别是什么？https有什么优缺点?

区别：

1、HTTP协议是以明文的方式在网络中传输数据，而HTTPS协议传输的数据则是经过TLS加密后的，安全性高

3、HTTPS协议需要服务端申请证书，浏览器端安装对应的根证书

4、HTTP协议端口是80，HTTPS协议端口是443

HTTPS优缺点

1、握手延时 高

2、部署成本高

请你说一说IP地址作用，以及MAC地址作用

MAC地址是一个硬件地址，用来定义网络设备的位置，主要由数据链路层负责。而IP地址是IP协议提供

的一种统一的地址格式，为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地

址的差异。

具体网络层的操作该怎么做?

服务端：socket-bind-listen-accept

客户端：socket-connect

请你来说一下socket编程中服务器端和客户端主要用到哪些函数?

基于TCP的socket

服务器端程序

（1）、创建一个socket，用函数socket()

（2）、绑定IP地址、端口等信息到socket上，用函数bind()

（3）、设置允许的最大连接数，用函数listen()

（4）、接收客户端上来的连接，用函数accept()

（5）、收发数据，用函数send()和recv()，或者read()和write()

（6）、关闭网络连接

客户端程序：

（1）、创建一个socket，用函数socket()

（2）、设置要连接的对方的IP地址和端口等属性

（3）、连接服务器，用函数connect()

（4）、收发数据，用函数send()和recv()，或read()和write()

（5）、关闭网络连接

基于UDP的socket

服务器端流程

（1）、建立套接

（3）、绑定侦听端口，使用bind()函数，将套接字文件描述符和一个地址类型变量进行绑定。

（4）、接收客户端的数据，使用recvfrom()函数接收客户端的网络数据。

（5）、向客户端发送数据，使用sendto()函数向服务器主机发送数据。

（6）、关闭套接字，使用close()函数释放资源。UDP协议的客户端流程

客户端流程

（1）、建立套接字文件描述符，socket()。

（2）、设置服务器地址和端口，struct sockaddr。

（3）、向服务器发送数据，sendto()。

（4）、接收服务器的数据，recvfrom()。

（5）、关闭套接字，close()。

1、C/C++

1.1 关键字

（参考”嵌入式及Linux那些事“以及众多帖子汇总而成）

volatile

当声明指向设备寄存器的指针时一定要用volatile，它会告诉编译器不要对存储在这个地

址的数据进行假设。

中断服务程序中修改的供其他程序检测的变量。 中断中直接从变量地址中读取数据，而不是从寄存器中读取。

多线程应用中被几个任务共享的变量。

static

1、函数体内的变量，这个变量只被声明一次。

2、在模块内的变量，表示只能被模块内函数使用，不能被模块外函数访问，表示本地全局变量

3、模块内的函数，限制在模块内使用，同上。

extern

1、引用同一文件变量

使用在声明之前时，可以使用关键字extern，让声明在程序任意位置。

2、引用另一个文件中的变量

extern可以引用其他文件中的全局变量，而且extern只需要指明数据类型和

extern int num=4; 这样不行。

3、引用另一个文件中的函数

可以不用包含头文

1、new/delete是操作符，malloc/free是库函数

2、new/delete可以调用构造函数/析构函数，m/f 只是分配内存。

struct 和 union区别

1、联合体公用一块地址空间，联合体变量长度等于最长的成员的长度

2、对不同成员赋值，会将其他成员重写。

const

1、定义变量为常量

2、修饰参数为常量

3、修饰返回值为常量

总结：只读

sizeof和strlen

1、sizeof是运算符，strlen是函数

2、sizeof可以用类型、函数作为参数，strlen只能计算char*，还必须以/0结尾

3、sizeof编译的时候计算，strlen是运行期计算，表示字符串长度，不是内存大小。

typedef和 define

1、都是替对象去一个别名，增强程序的可读性

2、define为预处理指令，不做正确性检查，只有带入之后才能发现

3、typedef用来定义类型别名，不止包含内部类型还包含自定义类型（与机器无关），方便记忆

4、define不仅可以给类型取别名，还能定义常量、变量、编译开关。

5、define没有作用域限制，typedef有。

define还是 const ，谁定义常量最好

1、define只是文本替换，声明周期止于编译期，不分配内存空间，存在于代码段。const常量存在于数据段，堆栈中分配了空间。

2、const有数据类型，编译器可以对const进行安全检查。

3、const有保护常量不被修改的作用，提高程序的健壮性。

总结：一般倾向于用const定义常量

1.2 内存

C语言内存分配方式

1、静态储存区分配

2、栈上分配

3、堆上分配

C++内存管理是怎

代码段：分为只读区和文本区，只读取储存字符串常量，文本区储存机器代码。

数据段：储存以及初始化的全局变量和静态变量

BSS段：储存未初始化的全局变量和静态变量，以及初始化为0的全局和静态。

堆区：手动分配的内存

栈：局部变量参数返回值等

映射区：储存动态链接库，mmap函数的文件映射

堆和栈的区别

1、申请方式。 栈为操作系统自动分配/释放，堆为手动

2、申请大小，栈空间有限，向低地址拓展的连续区域，堆是向高地址拓展的不连续区域，链表储存的空闲地址。

3、申请效率，栈是系统自动分配，速度快，不可控。堆是由new分配，速度比较慢，容易产生内存碎片。

栈的作用

1、储存临时变量

2、多线程编程的基础。每个线程至少有一个栈用来存储临时变量和返回的值。

内存泄漏

申请了没有释放，由程序申请的一块内存，没有任何指针指向它，这个内存就泄露了。

避免内存泄漏方法

1、分配的内存以链表管理，使用完毕后从链表删除，程序结束的时候检查链表

2、良好的编程习惯，在设计内存的程序段，检验出内存泄漏，使用了内存分配的函数，使用完毕后将使用的相应函数释放掉

3、smart pointer

指针

数组指针和指针数组

int （*p）[20]; 数组指针，本质是一个指针，指向一个数组

int *p[20]; 指针数组，本质是一个数组，里面装的是指针。

函数指针和指针函数

1、函数指针 int（*p）(int,int);本身是一个指针，指向一个函数的地址

2、指针函数 int *p(int,int); 指针函数表示一个函数，返回数是指针。

数组名和指针区别

1、指针保存的是地址，数组保存的是数据，单数组名是第一个元素的地址

2、指针间接访问，数组直接下标或者偏移量

3、sizeof 有区别，

1、int *const p 指向地址不变，地址值可变

2 int const *p 指向地址可变，地址值不能边

3、const int * const p 都不能变

指针与引用区别

1、都是地址，指针是地址，应用是别名

2、引用本质是指针常量，对象不变，对象的值可变

3、++不同，指针是地址自增，引用是对象自增

4、指针需要解引用

5、指针可为空，引用不行

6、sizeof不同 一个是指针大小一个是对象大小

野指针

1、指向不可用内存的指针，指针被创建时如果没有初始化就是野指针

2、指针被free、delete时没有指向NULL就是野指针

3、指针超出了变量的地址范围

智能指针

C++智能指针是指一个类，用来存储指针

1.3 预处理

预处理器标识#error的目的是什么?

1、遇到#error就会生成一个编译错误提示信息，并停止编译

define声明一年多少秒

#define SECOND_OF_PER_YEAR (3652460*60)UL

#include"" 和 include<>区别

<>号先搜索标准库搜索系统文件比较快， “” 号先搜索工作路径搜索自定义文件比较快

头文件作用

1、通过文件调用库功能，源码保护

2、头文件加强类型安全检查，编译器报错

头文件定义静态变量

1、资源浪费，每个头文件都会单独存在一个静态变量

不使用流程控制语句，打印1~1000数字

```
#include<stdio.h>
#define A(x) x;x;x;x;x;x;x;x;x;x;
int main()
{
        int n=1
```

```
}
```

1.4 变量

全局变量和静态变量

1、全局变量作用域为程序块，局部变量为当前函数

2、全局变量储存在静态区，后者为栈

3、全局变量生命周期为主函数，局部变量生命周期在局部函数中，甚至循环体内

1.5 函数

写个函数在main函数执行前执行

1、attribute可以设置函数属性

```
#include <stdio.h>
void before() __attribute__((constructor));
void after() __attribute__((destructor));
void before() {
  printf("this is function %s\n",__func__);
  return;
}
void after(){
  printf("this is function %s\n",__func__);
  return;
}
int main(){
  printf("this is function %s\n",__func__);
  return 0;
}
// 输出结果
// this is function before
// this is function main
// this is function after
```

为什么析构函数必须是虚函数

1、基类指针指向子类时，释放基类指针也能释放掉子类的空间，防止内存泄漏。

2、最好是作为父类的类的析构函数作为虚函数

为什么C++默认的析构函数不是虚函数?

1、虚函数有额外的虚函数表和虚指针表，占用额外的内存，对于那些不会被继承的类当然也不需要虚函数作为析构函数。

静态函数和虚函数的区别?

1、静态函数编译时确定运行时机

2、虚函数运行时动态绑定，并且使用虚函数表，内存开销增加

1、覆盖是子类和父类的关系，垂直关系，重载是一个类之间的关系，水平关系

2、覆盖一对一，重载多个方法

3、覆盖由对象类型决定，重载根据调用的参数表决定

虚函数表实现多态方法

原理：

虚函数表示一个类的地址表，子类创建时，按照函数声明吮吸会将函数的地址存在虚函数表中。子类重写父类虚函数的时候，父类虚函数表中的位置会被子类虚函数地址覆盖。

C语言函数调用方法

1、使用栈来支持函数调用操作，栈被用来传递参数，返回值，局部变量等。

2、函数调用主要操作栈帧结构

select函数

int select(int maxfdp,fd_set *readfds,fd_set *writefds,fd_set *errorfds,struct timeval *timeout);

1

fork wait exec函数

1、附近产生的子进程使用fork拷贝出一个父进程的副本

数组的下标可以为负数吗?

可以，数组下标指地址偏移量，根据偏移量能定位得到目标地址。

inline函数和宏定义的区别

1、内联函数在编译时展开，而宏在预编译时展开。

2、在编译的时候，内联函数直接被嵌入到目标代码中去，而宏只是一个简单的文本替换。

3、内联函数可以进行诸如类型安全检查、语句是否正确等编译功能，宏不具有这样的功能。

4、宏不是函数，而inline是函数。

5、宏在定义时要小心处理宏参数，一般用括号括起来，否则容易出现二义性。而内联函数不会出现二义性。

6、inline可以不展开，宏一定要展开。因为inline指示对编译器来说，只是一个建议，编译器可以选择忽略该建议，不对该函数进行展开。

7、宏定义在形式上类似于一个函数，但在使用它时，仅仅只是做预处理器符号表中的简单替换，因此它不能进行参数有效性的检测，也就不能享受C++编译器严格类型检查的好处，另外它的返回值也不能被强制转换为可转换的合适的类型，这样，它的使用就存在着一系列的隐患和局限性。

1、函数调用时，先求出实参表达式的值，然后带入形参。而使用带参数的函数只是进行简单的字符替换

2、函数调用实在程序运行时处理的，分配的临时的内存单元；而宏展开则是在编译时进行的，在展开时不分配i内存单元，不进行值的传递，也没有"返回值的概念"

3、函数实参形参都要定义类型，二者要求一致 ，宏不存在类型问题，宏没有类型，宏的参数只是一个符号代表，展开时代入指定的字符就行，宏定义时字串可以是任意内心的数据

4、函数只可以得到一个返回值，宏可以设法得到多个

5、使用宏次数多时，展开后源程序长，每次展开都使程序增长，而函数调用不使源程序变长。

6、宏的替换不占用时间，只占用编译时间，函数调用占用运行时间。

简单回答：宏由编译计算，增加编译时间，函数运行的时候计算，增加运行时间；函数的返回值入口参数有数据类型，宏只是简单的符号加减。

ASSERT（）作用

ASSERT()是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。

strcpy()和memcpy()的区别

1、复制的内容不同。strcpy只能复制字符串，而memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。

2、复制的方法不同。strcpy不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy则是根据其第3个参数决定复制的长度。

3、用途不同。通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy

1.6 位操作

求解整型类二进制表示1的个数

```
int func(int x)
{
        int countx = 0;
    while(x)
    {
        countx++;
        x = x&(x-1);
    }
        return countx;
}
```

求解整型类二进制表示0的个数

```
int CountZeroBit(int num)
{
    int count = 0;
        while (num + 1)
        {
            count++;
            num |= (num + 1); //算法转换
        }
```

给定一个整型变量a，写两段代码，第一个设置a的bit 3，第二个清除a 的bit 3。在

以上两个操作中，要保持其它位不变。

```
void clearbit3(int a)
{
        a&=~(1<<3);
}
void setbit3(int a)
{
        a|=1<<3;
}
```

1.7 容器与算法

map与set区别和底层实现

1、底层实现都是红黑树

2、map是键值对，关键字起到索引作用，值表示与索引相关联的数据，set是关键字的集合并且每个元素只包含一个关键字。

3、set迭代器是const不能修改元素值，map允许修改value不能修改key

4、map支持下标操作，set不支持，map可以用key作为下标，set用find

STL的allocator有什么作用?

1、内存配置有alloc::allocate()负责，内存释放由alloc::deallocate()负责；对象构造由::construct()负责，对象析构由::destroy()负责。

2、提升内存管理效率， STL采用了两级配置器，当分配的空间大小超过128B时，会使用第一级空间配置器；当分配的空间大小小于128B时，将使用第二级空间配置器。第一级空间配置器直接使用malloc()、realloc()、free()函数进行内存空间的分配和释放，而第二级空间配置器采用了内存池技术，通过空闲链表来管理内存。

STL迭代器如何删除元素?

对于序列容器vector,deque来说，使用erase(itertor)后，后边的每个元素的迭代器都会失效，但是后边

每个元素都会往前移动一个位置，但是erase会返回下一个有效的迭代器；

对于关联容器map set来说，使用了erase(iterator)后，当前元素的迭代器失效，但是其结构是红黑树，

删除当前元素的，不会影响到下一个元素的迭代器，所以在调用erase之前，记录下一个元素的迭代器即

可。

对于list来说，它使用了不连续分配的内存，并且它的erase方法也会返回下一个有效的iterator，因此上

面两种正确的方法都可以使用

1、map底层红黑树实现，unordered_map采用hash表实现’

2、map中序遍历有序，un——map无序

vector和list的区别是什么

1、vector为数组实现，list为双向链表

2、vector支持随机访问，list不行

3、vector顺序储存，list随机

4、vector一次性分配内存，不够才二倍扩容，list一个个分配

5、vector随机访问性能好，插入删除比较慢，list反之

迭代器与指针

1、迭代器又名游标模式，提供一种顺序访问一个聚合对象中各个元素，但又不暴露该对象的内部表示。

2、迭代器是类模板，表现得象指针，重载了指针一些操作，封装了指针，指针的++只是递增地址，但是不能对list生效，迭代器可以。

3、迭代器有着更良好的用法begin，end等不用担心越界

STL里resize和reserve的区别是什么?

1、resize改变当前容器内含有元素的数量，会新增元素0，reserve只是增加空间，不新增元素。

1.8 类和数据抽象

c++类成员访问权限

1、C++通过 public、protected、private 三个关键字来控制成员变量和成员函数的访问权限

2、类内随便访问，类外通过对象访问，且只能访问public成员

引用与指针的区别

1、引用必须被初始化，指针不必。

2、引用初始化以后不能被改变，指针可以改变所指的对象。

3、不存在指向空值的引用，但是存在指向空值的指针。

struct与class区别

1、c++中两者都可以定义类，但是struct没有权限，默认public

面对对象和泛型编程

1、面对对象是一种程序设计思想，把对象作为程序的基本单元，一个对象包括了数据和操作数据的函数

2、泛型编程让类型参数化，使程序可以从逻辑功能上抽象，把处理的对象当成参数来传递

左值可以寻址，而右值不可以。

左值可以被赋值，右值不可以被赋值，可以用来给左值赋值。

左值可变,右值不可变（仅对基础类型适用，用户自定义类型右值引用可以通过成员函数改变）。
C++的类和C里面的struct有什么区别?

析构函数可以为 virtual 型，构造函数则不能，为什么?

1、虚函数主要是用作多态，如果构造函数也用了，那么派生类必须在初始化列表给基类参数初始化

2、构造函数运行的时候对象的动态类型还不完整，没法确定没所以不能动态绑定。

C++的类和C里面的struct有什么区别?

c++中的类具有成员保护功能，并且具有继承，多态这类特点，而c里的struct没有

c里面的struct没有成员函数,不能继承,派生等等.

C++中空类默认产生哪些类成员函数?

1、构造函数

2、拷贝构造

3、析构函数

4、赋值运算符重载函数

5、取值运算符重载函数

6、const取址运算符重载函数

静态成员函数与非静态成员函数的区别

前者没有 this 指针,后者有 this 指针。

静态成员函数只要用来访问静态数据成员,而不访问非静态成员

1.9 面对对象

面向对象和面向过程有什么区别?

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了；面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

1、面对对象以对象为中心，面向过程以过程为中心

2、面对对象把代码封装成一个整体，其他对象不能直接修改其数据。面向过程直接使用程序来处理数据，各模块存在控制与被控制的关系。

3、面对对象是将问题分为不同的对象，给予对象赋予属性和行为。面对过程则是将事件分为不同的步骤，按照步骤完成编程。

面对对象的基本特

2、继承：子类继承父类的功能

3、多态：不同的对象对从父类继承的同一动作做出不同的反应，

4、抽象:不打算了解问题全部，只关注当前目标。过程抽象和数据抽象。过程抽象是指任何操作都被当成实体看待，不在乎它是不是由其他子函数完成。

什么是深拷贝？什么是浅拷贝？

1、深拷贝复制一份

2、浅拷贝哟与可能共享成员变量

友元

1、友元函数：普通函数对一个访问某个类中的私有或者保护成员

2、友元类：类A中的成员函数访问类B中的私有或保护成员。

初始化列表和构造函数初始化的区别？

```
Example::Example() : ival(0), dval(0.0) {} //初始化列表的构造函数
Example::Example()   //构造函数
{
ival = 0;
dval = 0.0;
}
```

结果是一样的，使用初始化列表的构造函数表示 初始化类的成员，使用初始化列表的构造函数是对类成员的赋值，而不是初始化。所以一下情况需要对成员初始化所以必须用初始化列表的方法。

1、成员类型为没有默认构造函数的类

2、const成员或引用类型的成员

类的成员变量的初始化顺序是什么？

1、成员变量在使用初始化列表初始化时，与构造函数中初始化成员列表的顺序无关，只与定义成员变量的顺序有关。

2、不使用初始化列表的话就与构造函数有关。

Public继承、protected继承、private继承的区别？

1、public继承就是公有继承完还是公有，保护还是保护，私有还是私有

2、protected继承就是公有变保护，保护还是保护，私有还是私有

3、private继承就是所有变成私有

1.10 虚函数

虚函数注意内容

1、只需要在声明

3、非类的成员函数不能定义为虚函数，全局函数以及类的成员函数和构造函数也不能定义为虚函数，可以将析构函数定义为虚函数

什么函数不能声明为虚函数

主要有：普通函数（非成员函数）；静态成员函数；类联成员函数；构造函数：友元函数。

1.11数据结构

链表和数组的区别

数组在内存中栈上按顺序存储的，而链表是在堆上随机存储的。

要访问数组中的元素可以按下标索引来访问，速度比较快，如果对他进行插入操作的话，就得移动很多元素，所以对数组进行插入操作效率很低. 由于连表是随机存储的，链表在插入，删除操作上有很高的效率（相对数组）

如果要访问链表中的某个元素的话，那就得从链表的头逐个遍历，直到找到所需要的元素为止，所以链表的随机访问的效率就比数组要低 。

2、ARM体系与架构

2.1 硬件基础

NAND FLASH 和NOR FLASH异同?

类别 读 写 擦除 可靠性 容量 用途 价格

NOR 快 慢 非常慢 比较高 小 保存代码 高

NAND 快 快 快 低 大 保存数据 低

CPU,MPU,MCU,SOC,SOPC联系与差别?

1、CPU：是一台计算机的运算核心和控制核心

2、MPU: 微处理器稍强的CPU

3、MCU：将计算机的CPU、RAM、ROM、定时计数器和多种I/O接口集成在一片芯片上。

4、SOC： 系统级芯片不单单是放简单的代码，可以放系统级的代码，也就是说可以运行操作系统

CPU中cache的作用？cache的基本组织结构?

（1）高速缓冲存储器Cache是位于CPU与内存之间的临时存储器，它的容量比内存小但交换速度快。

在Cache中的数据是内存中的一小部分，但这一小部分是短时间内CPU即将访问的，当CPU调用大量数据时，就可避开内存直接从Cache中调用，从而加快读取速度。由此可见，在CPU中加入Cache是一种高效的解决方案，这样整个内存储器（Cache+内存）就变成了既有Cache的高速度，又有内存的大容量的存储系统了。

（2）全相连映射，直接映射，组相连映射

交叉编译

C/C++的编译包括几个部分

1、预编译：预处理器对c程序进行一些预处理工作，例如对宏定义的变量进行替换；

1）将所有的#define删除，并展开所有的宏定义；

2）处理所有的预编译指令，例如：#if,#elif,#else,#endif;

3）处理#include预编译指令，将被包含的文件插入到预编译指令的位置；

4）添加行号信息文件名信息，便于调试；

5）删除所有的注释：// /**/;

6）保留所有的#pragma编译指令，因为在编写程序的时候，我们经常要用到#pragma指令来设定编译器的状态或者是指示编译器完成一些特定的动作；

最后生成.i文件；

总的来说，包括（1）去注释 （2）宏替换 （3）头文件展开 （4）条件编译

2、编译：编译器将c语言程序翻译成汇编语言程序；

1）扫描，语法分析，语义分析，源代码优化，目标代码生成，目标代码优化；

2）生成汇编代码；

3）汇总符号；

4）生成.s文件；

3、汇编：汇编语言通过汇编器编译成可重定位目标程序.o，与之相反称为反汇编；

1）根据汇编指令和特定平台，把汇编指令翻译成二进制形式；

2）合并各个section，合并符号表；

3）生成.o文件；

4、链接：将目标文件和所需的库函数用链接器进行链接，常见的链接器有Unix；

1）合并各个.obj文件的section，合并符号表，进行符号解析；

2）符号地址重定位；

3）生成可执行文件；

描述一下嵌入式基于ROM的运行方式和基于RAM的运行方式有什么区别?

基于RAM：

1、将硬盘或者其介质的代码加载到ram中。

2、速度快但是可用RAM少，因为自身的空间要存一部分代码

基于ROM:

2.2 中断与异常

中断与异常区别

1、中断是指外部硬件产生的一个电信号从CPU的中断引脚进入，打断CPU的运行，异常是指软件运行过程中发生了一些必须作出处理的事件，CPU自动产生一个陷入来打断CPU的运行。

2、异常处理的时候要考虑与处理器的时钟同步，异常被称为同步中断

中断能不能睡眠 为什么?

1、一般说中断上下文中不能睡眠,这个中断是指硬件事件发生,触发CPU停止当前活动转而去处理硬件请求.

2、根据硬件请求响应处理逻辑的实时紧要与否,将整个中断处理过程分为上半部和下半部.上半部也就是所谓的硬中断处理逻辑,其要求cpu在收到硬件请求后必须马上处理的事情,比如网卡收到数据包了,把数据包从网卡缓存拷贝到主存(可以由DMA完成,但寄存器的修改以及资源设定还是要由cpu去做)的逻辑就需要cpu立即去做,不做的话,网络新来的数据包就可能丢失.所以这些紧要操作逻辑为硬中断处理.

3、下半部有很多种机制,其中就包括软中断,还有tasklet,workqueue等,软中断只是其中的一种,由于历史的原因,有时候是混淆称呼下半部和软中断的.

4、而可以看到软中断逻辑不属于任何进程,所以才不能睡眠,因为一旦睡眠,cpu切换出去,就切不回来了。

简单说就是：唤醒函数针对进程而言的，下半部的中断不属于进程，所以无法被唤醒

中断的响应执行流程是什么?

cpu接受中断->保存中断上下文跳转到中断处理历程->执行中断上半部->执行中断下半 部->恢复中断上下文。

写一个中断服务需要注意哪些？如果中断产生之后要做比较多的事情你是怎么做的?

1、快进快出，在中断服务函数里尽量快速采集信息。

2、中断中不能有阻塞操作

3、中断服务函数注意返回值，使用操作系统定义的宏，而不是自己定义的。

4、做的事情较多，将这些任务放在后半段tasklet处理。

中断和轮询哪个效率高？怎样决定是采用中断方式还是采用轮询方式去实现驱动?

1、中断是CPU处于被动状态下来接受设备的信号，而轮询是CPU主动去查询该设备是否有请求。

2、请求设备是一个频繁请求cpu的设备，或者有大量数据请求的网络设备，那么轮询的效率是比中断高。

3、如一般设备，并且该设备请求cpu的频率比较低，则用中断效率要高一些。主要是看请求频率。

2.3 通讯协议

异步传输与同步传输?

异步传输：是一种

RS232和RS485区别?

传输方式不同。 RS232采取不平衡传输方式，即所谓单端通讯。 而RS485则采用平衡传输，即差分传输方式。

传输距离不同。RS232适合本地设备之间的通信，传输距离一般不超过20m。而RS485的传输距离为几十米到上千米。

设备数量。RS232 只允许一对一通信，而RS485 接口在总线上是允许连接多达128个收发器。

连接方式。RS232，规定用电平表示数据，因此线路就是单线路的，用两根线才能达到全双工的目的；而RS485， 使用差分电平表示数据，因此，必须用两根线才能达到传输数据的基本要求，要实现全双工，必需用4根线

SPI协议

SPI：高速全双工串行总线。

接口：输出线、输入线、时钟线、片选信号线

1、片选信号线由高到低是SPI的起始信号 ，从机检测到自己的NSS线起始信号之后就知道自己被选中了，然后由低到高是停止信号。

2、SPI 使用 MOSI 及 MISO 信号线来传输数据，使用 SCK 信号线进行数据同步。 在时钟线上升沿触发输出，在下降沿被采样。

IIC协议

1、IIC协议是由数据线SDA和时钟SCL构成的串行总线，可发送和接收数据,是一个多主机的半双工通信方式

2、空闲状态

SDA与SCL都处于高电平，就是空闲状态。

2、起始信号

时钟线为高，数据线由高到低就是启动信号，只能由主机发起空闲状态下才能启动该信号

3、停止信号

时钟为高，数据线由低到高就是停止信号

4、传输数据格式

SCL为高就会获取SDA数据值，SDA在这期间必须稳定

SCL为低便是SDA电平变化状态，在此期间SDA可以自由变化

可以主动拉低SCL让IIC进入等待状态知道处理结束再释放SCL数据传输会继续

5、ACK应答信号

发送方在第9个时钟脉冲奇迹爱你释放SDA数据，当接收方接收成功时，会输出一个应答信号，低电平有效

6、写操作

白色主到从、灰色从到主。

7、读操作

除了数据需要主到从，其余差不多。

嵌人式编程中，什么是大端？什么是小端？

大端模式：低位字节存在高地址上，高位字节存在低地址上。

小端模式：高位字节存在高地址上，低位字节存在低地址上。

```
//第一种
union w{
    int a;
    char b;
}c;
c.a = 1;
if(c.b==1) printf("小端");
else printf("大端");
//第二种
int a = 0x12345678;
char *p = (char *)&a;
if(*p==0x78)printf("小端");
else printf("大端");
```

3、Linux驱动

3.1 指令

Linux指令

查看当前进程 ps;

执行退出 exit;

查看当前路径 pwd;

查看目录 ls -a显示所有文件及目录，-l详细列出

创建目录 mkdir;

创建文件 vi 、 touch;

查看文件内容 vi，cat ;

屏幕输出 echo;

常用的GCC命令

gcc -E test.c -o test.i #把预处理的结果导出到test.i文件

gcc -S test.i -o test.s #编译器将test.i翻译成汇编语言，并将结果存储在test.s文件中。

gcc -c test.s -o test.o #将汇编代码编译为目标文件（.o）但不链接

gcc test.o -o test #将生成的目标文件test.o生成最终的可执行文件test

gcc test.c -o test #将源文件test.c编译链接为可执行文件test

gcc test1.c test2.

gcc -g test.c -o test #编译时生成debug有关的程序信 就是说正常编译不能使用GDB

list 查看源码

next #单步调试（逐过程，函数直接执行）,简写n

step #单步调试（逐语句：跳入自定义函数内部执行）,简写s

run #运行程序

break + num #设置第num行 为断点

continue #继续运行到下一个断点。

display 追踪具体变量

delete breakpoints num #删除第num个断点

常用的驱动开发指令

insmod\modprobe 加载驱动

rmmod #卸载驱动

lsmod #查看已有的字符设备信息

cat /proc/interrupt #查看已有的中断号

Makefile

经典malefile main包含了input、calcu的头文件

```
objects = main.o input.o calcu.o
main: $(objects)
        gcc -o main $(objects)
%.o : %.c
        gcc -c $<
clean:
        rm *.o
        rm main
```

shell相关操作

要求

A、在Linux操作系统启动的时候，自动加载/mnt/test/test程序。

B、当test异常退出之后，自动重新启动。

C、当test程序重启次数超过100次，自动复位操作系统。

假设你所拥有的资源：

A、目标机器是一台具有标准shell的嵌入式计算机，CPU为ARM7 56MB，内存16MB，软件环境基于Linux2.6.11和BusyBox1.2构建。

B、当前已有11个用户进程在运行，占用了大部分的CPU时间和内存，你可使用的内存只有2MB左右，CPU时间由系统分派。

```
#load *.so that may need
if [ -r /sbin/ldconfig ]; then
ldconfig
fi

#add the libs PATH that may need
export LD_LIBRARY_PATH="/lib"

#count is the counter of test started times
count=0

#main loop
while [ 1 ] ;do
#add execute property for /mnt/test/test
chmod +x /mnt/test/test
#start test
/mnt/test/test
#the running times counter
let count=count+1
echo "test running times is $count"
#Is test running too many times?
if [ "$count" -gt 100 ]; then
echo "Will reboot because of test running too many times"
reboot
fi
#wait for test stoping...
sleep 3
done
```

3.2 uboot

bootloader

1、Linux启动需要一个bootloader程序，初始化时钟、中断或者其他外设，然后将Linux内核从flash拷贝到SDRAM中，最后启动Linux内核。

2、Bootloader就是一小段程序，它在系统上电时开始执行，初始化硬件设各、准备好软件环境，最后调用操作系统内核。

uboot启动流程

u-boot系统启动流程 ,大多数bootloader都分为stage1和stage2两部分, u-boot也不例外。

依赖于CPU体系结构的代码（如设备初始化代码等）通常都放在stage1且可以用汇编语言来实现，而stage2则通常用C语言来实现，这样可以实现复杂的功能，而且有更好的可读性和移植性。

1、Stage1 start.S代码结构 u-boot的stage1代码通常放在start.S文件中，他用汇编语言写成，其主要代码部分如下

（1） 定义入口:

该工作通过修改连接器脚本来完成。

（2）设置异常向量（Exception Vector）。

（3）设置CPU的速度、时钟频率及终端控制寄存器。

（4）初始化内存控制器。

（5）将ROM中的程序复制到RAM中。

（6）初始化堆栈。

2、Stage2

C语言代码部分 lib_arm/board.c中的start arm boot是C语言开始的函数也是整个启动代码中C语言的主函数，同时还是整个u-boot（armboot）的主函数，该函数只要完成如下操作：

（1）调用一系列的初始化函数。

（2）初始化Flash设备。

（3）初始化系统内存分配函数。

（4）如果目标系统拥有NAND设备，则初始化NAND设备。

（5）如果目标系统有显示设备，则初始化该类设备。

（6）初始化相关网络设备，填写IP、MAC地址等。

（7）进去命令循环（即整个boot的工作循环），接受用户从串口输入的命令，然后进行相应的工作

uboot启动过程中做了那些事?

1、初始化时钟，关闭看门狗，关中断，启动ICACHE，关闭DCACHE和TLB，关闭MMU，初始化SDRAM，初始化NAND FLASH，重定位。

2、初始化一个串口，检测系统内存映射，将内核映象和根文件系统映象从 Flash上读到SDRAM空间中，为内核设置启动参数，调用内核。

uboot和内核如何完成参数传递?

1、完成相关设置：CPU寄存器设置 R0=0 R1=机器类型ID R2=启动参数标记列表在RAM中起始基地址，设置禁止中断，SVC模式（超级用户模式，有利于硬件初始化）MMU关闭

2、uboot把机器ID通过R1传递给内核，R2存放块内存的基地址，这块内存主要存放uboot给Linux内核的其他参数，参数很多所有需要按规定存放，标记是一种数据结构。

3、标记的数据结构为tag，它由一个tag_header结构和一个联合（union）组成

为什么uboot要关掉caches?

caches是cpu内部的一个2级缓存，它的作用是将常用的数据和指令放在cpu内部。caches是通过CP15

管理的，刚上电的时候，cpu还不能管理caches。上电的时候指令cache可关闭，也可不关闭，但数据

cache一定要关闭，否则可能导致刚开始的代码里面，去取数据的时候，从cache里面取，而这时候RAM中数据还没有caches过来，导致数据预取异常 。

3.3 文件系统

什么是根文件系统?

1、内核启动时所挂载（mount）的第一个文件系统，内核代码的映像文件保存在根文件系统中。

2、挂载之后会把一些初始化脚本和服务加载到内存中去运行。

根文件系统为啥这

2、一套linux体系，只有内核本身是不能工作的，必须要rootfs（上的etc目录下的配置文件、/bin /sbin等目录下的shell命令，还有/lib目录下的库文件等）相配合才能工作。

3.4 中断

硬中断 / 软中断是什么？有什么区别?

1、硬中断是由硬件产生的，软中断是执行中断指令产生的。

2、硬中断可以直接中断CPU，软中断并不会直接中断CPU。也只有当前正在运行的代码（或进程）才会产生软中断。

3、硬中断可屏蔽、软中断不可屏蔽

4、硬中断又称上半部，要快速完成任务

中断为什么要区分上半部和下半部?

1、调用过程：外部中断产生->发送中断信号到中断控制器->通知处理器产生中断的中断号

2、为了能被新的中断打断。将中断处理一分为二，上半部登记新的中断，处理快速简单的任务，复杂耗时的任务给下半段处理，所以下半段可以被打断。

3、中断下半部一般使用tasklet或工作队列实现

linux中断的响应执行流程?

cpu接受中断->保存中断上下文跳转到中断处理历程->执行中断上半部->执行中断下半部->恢复中断上下文。

3.5 Linux驱动模型

字符设备驱动模型

请简述主设备号和次设备号的用途

主设备号：主设备号标识设备对应的特定的驱动程序。虽然现代的linux内核允许多个驱动程序共享主设备号，但我们看待的大多数设备仍然按照“一个主设备对应一个驱动程序”的原则组织。

次设备号：次设备号由内核使用，用于确定由主设备号对应驱动程序中的各个设备。依赖于驱动程序的编写方式，我们可以通过次设备号获得一个指向内核设备的直接指针，也可将此设备号当作设备本地数组的索引。（多个设备共用一套程序的话，主设备号代表这个驱动程序，每个设备一个次设备号）

创建设备文件

1、手动创建

mknod /dev/led c 250 0 ， 其中dev/led 为设备节点 ,c 代表字符设备, 250代表主设备号, 0代表次设备号。

设备驱动程序中如何注册一个字符设备?

1、将cdev结构嵌

2、早期注册函数

int register_chrdev(unsigned int major, const char *namem , struct file operations *fopen);

Linux设备中字符设备和块设备有什么主要区别?

字符设备：提供连续的数据流，应用程序可以顺序读取，通常不支持随机存取。

块设备：应用程序可以随机访问设备数据，程序可自行确定读取数据的位置。

驱动中操作物理绝对地址为什么要先ioremap?

1、ioremap是将io地址空间映射到虚拟地址空间上去，便于操作。

2、因为保护模式下的cpu只认虚拟地址，不认物理地址，所以你要操作外设上的寄存器必须先映射到虚拟内存空间，拿着虚拟地址去跟cpu对接，从而操作寄存器。

Linux移植ARM的基本步骤和完成的任务

1）首先是准备工作，包括下载源码、建立交叉编译环境等；

2）然后是配置和编译内核，必要时还要对源码做一定的修改；

3）第三步就是需要制作文件系统（如RAM disk）来挂接根文件系统；

4）最后是下载、调试内核并在fs中添加自己的应用程序。

ARM-linux启动分几部分，简述流程：

ARM-linux启动分为四个部分：引导加载程序（bootloader），Linux内核，文件系统，应用程序。

bootloader是系统启动和复位后执行的第一段代码，它主要用来初始化处理器及外设，然后调用Linux内核。Linux内核在完成系统的初始化之后需要挂载某个文件系统作为根文件系统（root filesystem）。根文件系统是Linux系统的核心组成部分，它可以作为Linux系统中文件和数据的存储区域，通常它还包括配置文件运行应用程序所需要的库。应用程序实现该嵌入式产品所要实现的目标。

4、操作系统

什么是进程？什么是线程?

进程是资源分配的基本单位，它是程序执行时的一个实例，在程序运行时创建。

线程是程序执行的最小单位，是进程的一个执行流，一个线程由多个线程组成的。

进程和线程有什么区别?

1、进程是资源分配的基本单位，线程是程序运行的基本单位

2、进程有自己的资源空间，线程是共享进程中的数据，所以进程切换开销更大一点

3、线程通讯要简单一些，因为共享全局变量等

4、线程执行开销小，进程执行开销大。

5、多线程中一个线程死掉整个进程也死了，一个进程死掉不会影响其他进程，因为它有独立的地址空间。

对资源的管理和保护要求高，不限制开销和效率时，使用多进程。

要求效率高，频繁切换时，资源的保护管理要求不是很高时，使用多线程。

进程有几种状态?

创建状态

就绪状态

运行状态

阻塞状态

终止状态

进程间通信方式

管道(pipe)

管道这种通讯方式有两种限制，一是半双工的通信，数据只能单向流动，二是只能在具有亲缘关系的进程间使用

信号量(semophore)

信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。

消息队列(message queue)

消息队列是由消息组成的链表，存放在内核中并由消息队列标识符标识。

共享内存(shared memory)

共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以

访问。共享内存是最快的 IPC 方式，它是针对其他进程间通信方式运行效率低而专门设计的。它往往与

其他通信机制，如信号量，配合使用，来实现进程间的同步和通信。

套接字(socket)

套解字也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同机器间的进程通信。

线程间的通讯方式

1）通过条件变量进行线程间的通信

（2）通过标志位来通知线程间的通信

（3）通过std::furture来进行线程间的通信

线程间同步方法有哪些?

临界区：如果有多个线程试图访问共享资源，那么当有一个线程进入后，其他试图访问共享资源的线程将会被挂

互斥量：为协调对一个共享资源的单独访问而设计，只有拥有互斥量的线程，才有权限去访问系统

的公共资源，因为互斥量只有一个，所以能够保证资源不会同时被多个线程访问。

信号量：为控制一个具有有限数量的用户资源而设计。它允许多个线程在同一个时刻去访问同一个

资源，但一般需要限制同一时刻访问此资源的最大线程数目

事件：用来通知线程有一些事件已发生，从而启动后继任务的开始。

什么是僵尸进程，孤儿进程，守护进程?

僵尸进程是 一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子

进程的状态信息，那么子进程的进程描述符仍然保存在系统中。这种进程称之为僵死进程。

孤儿进程是因为父进程异常结束了，然后被1号进程init收养。

守护进程是创建守护进程时有意把父进程结束，然后被1号进程init收养

请你回答一下fork和vfork的区别?

fork( )的子进程拷贝父进程的数据段和代码段；vfork( )的子进程与父进程共享数据段

fork( )的父子进程的执行次序不确定；vfork( )保证子进程先运行，在调用exec或exit之前与父进程

数据是共享的，在它调用exec或exit之后父进程才可能被调度运行。

vfork( )保证子进程先运行，在它调用exec或exit之后父进程才可能被调度运行。如果在调用这两个

函数之前子进程依赖于父进程的进一步动作，则会导致死锁。

当需要改变共享数据段中变量的值，则拷贝父进程。

堆和栈

什么是代码段，数据段，bss段，堆，栈?

代码段：存放程序执行代码的一块区域，通常是只读

数据段：已初始化的全局变量和已初始化为非0的静态变量

BSS段：未初始化的全局变量和未初始化的静态变量或者初始化为0的静态变量

数据段和BSS段本质上都是静态区，存放全局变量和静态变量的

堆：堆是用来存放进程中被动态分配的内存段。

栈：是用户存放程序临时创建的变量。

为什么堆的空间是不连续的?

1、堆包含一个链表来维护已用和空闲的内存块。

2、分配的空间在逻辑地址（虚拟地址）上是连续的，但在物理地址上是不连续的

内核栈：内存中属于操作系统空间的一块区域。

作用：

1、保存中断现场

2、保存调用的参数、返回值、函数局部变量

用户栈：

用户进程空间的一块区域，用于保存用户空间子程序间调用的参数，返回值以及局部变量。

为什么不能共用一个栈：

1、系统栈（内核栈）大小有限用户程序调用次数可能很多。

2、用户栈空间不能提供相应保护措施

线程是否有相同的堆栈？

每个线程有自己的堆栈。

信号、并发和互斥

驱动里面为什么要有并发、互斥的控制？如何实现？讲个例子？

并发：多个执行单元同时对共享资源操作，容易导致竞态。

互斥：一个执行单元在访问共享资源的时候、其他执行单元都被禁止访问。访问共享资源的代码区被称为临界区，临界区需要某种互斥机制加以保护。

自旋锁是什么？信号量是什么？二者有何异同？

自旋锁：一个执行单元在操作资源时，另一个执行单元不能操作。自旋锁只能短期持有

信号量：资源标量，使用完了就不允许操作了。会有信号告诉需要等多久。适合长期持有的时候用

区别：

1、长时间持有锁使用信号量，短时间持有使用自旋锁。

2、信号量可以睡眠，其他人需要时也会进入睡眠。

3、信号量代码可以被抢占。

自旋锁和信号量可以睡眠吗？为什么？

自旋锁不能睡眠，信号量可以。

原因：

1、自旋锁自旋锁

自旋锁和信号量可以用于中断中？

信号量不能用于中断中，因为信号量会引起休眠，中断不能休眠。

自旋锁可以用于中断。在获取锁之前一定要先禁止本地中断（也就是本CPU中断，对于多核SOC来说会

有多个CPU核），否则可能导致锁死现象的发生

产生死锁的原因是什么?

多个并发进程因争夺系统资源而产生相互等待的现象。即：一组进程中的每个进程都在等待某个事件发

生，而只有这组进程中的其他进程才能触发该事件，这就称这组进程发生了死锁。

原因：

1、系统资源有限

2、进程推进顺序不合理

如何避免死锁?

1、线程按一定顺序加锁

2、获取锁时加上时限，也就是说超过时间则放弃获取。

3、死锁检测

内存

在1G内存的计算机中能否malloc(1.2G)？为什么?

malloc能够申请的空间大小与物理内存的大小没有直接关系，仅与程序的虚拟地址空间相关。

内存管理的方法

1、块式管理

分成一大块一大块，只需要几个字节也给一大块。造成浪费、但方便管理。

2、页式管理

划分地址空间为若干大小区域，被称为页。优点便于管理，缺点页长与逻辑大小没有关系。

3、段式管理和段页式管理

按照程序的自然分界划分的并且长度可以动态改变的区域，每段可以定义一组相对完整的逻辑信息。段与段在内存中可以不相邻接，也实现了离散分配。

段页式管理：

用分段方法来分配和管理虚拟存储器。用分页方法来分配和管理内存

什么是虚拟内存?

好处：

1、扩大了地址空间

2、地址保护

3、公平分配内存：每个进程相当于有了同样大小的额外内存

解释下内存碎片，内碎片，外碎片？

1、内存碎片：内存碎片是由于多次进行内存分配造成的，空白段太小无法进行下次分配

2、内碎片：分配给程序的存储空间没有用完，有一部分是程序不使用（没用完），但其他程序也没法用的空间。

3、外碎片：空间太小，小到无法给任何程序分配（不属于任何进程）的存储空间。

解释下虚拟地址、逻辑地址、线性地址、物理地址？

1、虚拟地址、逻辑地址：由程序产生的由段选择符和段内偏移地址组成的地址这两部分组成的地址并没有直接访问物理内存，而是通过分段地址的变换处理后才会对应到相应的物理内存地址。

2、线性地址：指虚拟地址到物理地址变换之间的中间层，是处理器可寻址的内存空间（称为线性地址空间）中的地址。

3、物理地址：是指现在CPU外部地址总线上的寻址物理内存的地址信号，是地址变换的最终结果。

系统调用是什么，你用过哪些系统调用，和库函数有什么区别？

系统调用：系统调用是通向操作系统本身的接口，是面向底层硬件的。通过系统调用，可以使得用户态运行的进程与硬件设备(如CPU、磁盘、打印机等)进行交互，是操作系统留给应用程序的一个接口。

库函数：库函数（Library function）是把函数放到库里，供别人使用的一种方式。.方法是把一些常用到的函数编完放到一个文件里，供不同的人进行调用。一般放在.lib文件中。系统调用是为了方便使用操作系统的接口，而库函数则是为了人们编程的方便。

区别：

1、库函数是语言或应用程序的一部分，系统调用是内核提供的接口。

2、库函数在用户地址进行，系统调用在内核地中空间执行

3、库函数有缓冲、系统调用无缓冲。

4、系统调用依赖平台，库函数不用

上下文

上下文有哪些？怎么理解？

上下文简单说来就是一个环境。

用户级上下文: 正

系统级上下文: 进程控制块task_struct、内存管理信息(mm_struct、vm_area_struct、pgd、

pte)、内核栈

为什么会有上下文这种概念?

系统调用中用户空间会传递很多数据给内核空间，保存上下文以便系统调用结束后回到用户空间继续执行。

5、网络编程

TCP/ UDP

TCP怎么保证可靠性?

1、序列号、确认应答、超时重传

2、窗口控制与高速重发控制/快速重传（重复确认应答）

窗口控制：不需要对每个没收确认的数据重发，只需要确认一个窗口是否都收齐了。

简述一下TCP建立连接和断开连接的过程。

连接三次握手

1、客户端请求，标志位SYN置为1 发送x;

2、服务端回复，标志位SYN和ACK都置为1 回复 x+1,和 y

3、客户端收到后回复 y+1，服务端检查ack是否为1，是的话就连接成功

断开

1、客户端发送 x+2 回复y+1 进入FIN_WAIT_1 状态

2、服务端回复x+3 服务器进入CLOSE_WAIT状态。客户端收到后进入FIN_WAIT_2状态

3、服务端发送完所有数据之后发送y+1 服务器进入LAST_ACK状态

4、客户端回复y+2 客户端进入TIME_WAIT状态等待2MSL（报文段最大生存时间）后关闭

为什么客户端最后还要等待2MSL?

保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度

看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文

它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，

接着给出回应报文，并且会重启2MSL计时器。

为什么是三次握手?

1、为了防止已失

3、客户端发送请求报文后掉线了，服务端还是会回复，浪费资源

为什么是四次挥手？二三次能不能合并

1、TCP是全双工通信，意味着关闭是双方都需要确认的行为。

2、需要时间释放资源，一旦合并需要等很久。

3、客户端会以为自己第一次发的报文没有送到，不断尝试发送第一次的报文。

TCP/UDP

1、TCP面向连接，UDP无连接

2、TCP数据保证正确、顺序正确，UDP可能丢包

3、TCP可靠稳定，但是慢效率低，UDP快，容易丢包

TCP，UDP适用场景？

TCP应用场景

效率要求相对低，但对准确性要求相对高的场景。文件传输（准确高要求高、但是速度可以相对慢）、接受邮件、远程登录。

UDP应用场景

效率要求相对高，对准确性要求相对低的场景。QQ聊天、在线视频、网络语音电话、广播通信

TCP相比UDP为什么是可靠的？

1、确认和重传机制

2、数据排序

3、流量控制 窗口和计时器的使用

4、拥塞控制

什么是OSI七层模型和TCP/IP四层模型？每层列举2个协议。

物理层: 通过媒介传输比特,确定机械及电气规范,传输单位为bit，主要包括的协议为：IEE802.3

CLOCK RJ45

数据链路层: 将比特组装成帧和点到点的传递,传输单位为帧,主要包括的协议为MAC VLAN PPP

网络层：负责数据包从源到宿的传递和网际互连，传输单位为包,主要包括的协议为IP ARP ICMP

传输层：提供端到端的可靠报文传递和错误恢复，传输单位为报文,主要包括的协议为TCP UDP

会话层：建立、管理和终止会话，传输单位为SPDU，主要包括的协议为RPC NFS

表示层: 对数据进行翻译、加密和压缩,传输单位为PPDU，主要包括的协议为JPEG ASII

应用层: 允许访问OSI环境的手段,传输单位为APDU，主要包括的协议为FTP HTTP DNS

链路层：MAC VLAN PPP

网络层：IP协议、ICMP协议、ARP协议、RARP协议。

传输层：UDP协议、TCP协议。

应用层：FTP（文件传送协议）、Telnet（远程登录协议）、DNS（域名解析协议）、SMTP（邮件传送

协议），POP3协议（邮局协议），HTTP协议。

TCP/IP数据链路层的交互过程是怎么样的?

网络层等在数据链路层用MAC地址作为通信目标，数据包到达网络层等往数据链路层发送的时候，首先

回去ARP缓存表去查找ip对应的MAC地址，如果查到了，就将此ip对应的MAC地址封装到链路层数据包

的包头。如果缓存中没有找到，则会发起一个广播寻找目的IP的物理地址。

传递到IP层怎么知道报文该给哪个应用程序，它怎么区分UDP报文还是TCP报文?

根据端口继续区分需接受的程序;

根据ip协议头中标识字段：UDP 17 、TCP 6

端口号

熟知的端口号:

20：FTP 数据传输

21/TCP FTP 文件传输协议

23/tcp Telnet 不安全的文本传送

25/tcp SMTP Simple Mail Transfer Protocol (E-mail)

69/udp TFTP Trivial File Transfer Protocol

79/tcp finger Finger

80/tcp HTTP 超文本传送协议 (WWW)

88/tcp Kerberos Authenticating agent

110/tcp POP3 Post Office Protocol (E-mail)

113/tcp ident old identification server system

119/tcp NNTP used for usenet newsgroups

220/tcp IMAP3

443/tcp HTTPS used for securely transferring web pages

ICMP协议属于什么

2.ICMP只能搭配IPv4使用，如果是IPv6的情况下，需要是用ICMPv6

物理地址和IP的转化

地址解析协议（ARP）的作用是将IP地址转换成物理地址；反地址解析协议（RARP）则负责将物理地址转换成IP地址。

从在浏览器地址栏中输入baidu.com到看到百度首页，这个过程中间经历了什么？都涉及到哪些网络协议？

按照时间顺序：

1.客户端浏览器获取用户在地址栏输入的域名。

2.客户端浏览器将域名发送给DNS域名系统，请求解析。

3.DNS解析域名得到相应的IP，返回给客户端浏览器。

4.客户端浏览器根据IP向服务器发起TCP三次握手，建立TCP连接。

5.客户端浏览器向服务器发送HTTP请求，请求百度首页。

6.服务器通过HTTP响应向客户端浏览器返回百度首页文件。

7.释放TCP连接。

8.客户端浏览器解析HTML文件，根据文件内容获取CSS、JS等资源文件，将页面渲染展示给用户。

HTTP/IP

什么是http协议?

1、HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写，是用于从万维网（WWW:World Wide Web）服务器传输超文本到本地浏览器的传送协议。

2、HTTP是一个基于TCP/IP通信协议来传递数据

ICMP协议属于什么层

http协议有什么特点?

1、简单快速 客户向服务器请求服务时，只需传送请求方法和路径

2、无连接 无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。

3、基于TCP协议

4、默认端口80

https建立连接过程是什么?

客户端连接到Web服务器 一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。

发送HTTP请求

服务器接受请求并返回HTTP响应

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。

4. 释放连接TCP连接

4. 客户端浏览器解析HTML内容

http和https的区别是什么？https有什么优缺点?

区别：

1、HTTP协议是以明文的方式在网络中传输数据，而HTTPS协议传输的数据则是经过TLS加密后的，安全性高

2、HTTPS在TCP三次握手阶段之后，还需要进行SSL 的handshake，协商加密使用的对称加密密钥

3、HTTPS协议需要服务端申请证书，浏览器端安装对应的根证书

4、HTTP协议端口是80，HTTPS协议端口是443

HTTPS优缺点

1、握手延时 高

2、部署成本高

请你说一说IP地址作用，以及MAC地址作用

MAC地址是一个硬件地址，用来定义网络设备的位置，主要由数据链路层负责。而IP地址是IP协议提供

的一种统一的地址格式，为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地

址的差异。

服务端：socket-bind-listen-accept

客户端：socket-connect

请你来说一下socket编程中服务器端和客户端主要用到哪些函数?

基于TCP的socket

服务器端程序

（1）、创建一个socket，用函数socket()

（2）、绑定IP地址、端口等信息到socket上，用函数bind()

（3）、设置允许的最大连接数，用函数listen()

（4）、接收客户端上来的连接，用函数accept()

（5）、收发数据，用函数send()和recv()，或者read()和write()

（6）、关闭网络

（1）、创建一个socket，用函数socket()

（2）、设置要连接的对方的IP地址和端口等属性

（3）、连接服务器，用函数connect()

（4）、收发数据，用函数send()和recv()，或read()和write()

（5）、关闭网络连接

基于UDP的socket

服务器端流程

（1）、建立套接字文件描述符，使用函数socket()，生成套接字文件描述符。

（2）、设置服务器地址和侦听端口，初始化要绑定的网络地址结构。

（3）、绑定侦听端口，使用bind()函数，将套接字文件描述符和一个地址类型变量进行绑定。

（4）、接收客户端的数据，使用recvfrom()函数接收客户端的网络数据。

（5）、向客户端发送数据，使用sendto()函数向服务器主机发送数据。

（6）、关闭套接字，使用close()函数释放资源。UDP协议的客户端流程

客户端流程

（1）、建立套接字文件描述符，socket()。

（2）、设置服务器地址和端口，struct sockaddr。

（3）、向服务器发送数据，sendto()。

（4）、接收服务器的数据，recvfrom()。

（5）、关闭套接字，close()。

原文链接：嵌入式八股文汇总_Miss shirly的博客-CSDN博客

编辑于 2023-03-09 10:31 · IP 属地江苏

嵌入式开发 嵌入式系统 嵌入式系统培训

2023-09-25 · IP 属地江苏 回复 喜欢

荣耀尽归吾王鲁鲁修
爱了呀
2023-09-05 · IP 属地湖南 回复 喜欢

苏嵌教育 作者
2023-09-06 · IP 属地江苏 回复 喜欢

文章被以下专栏收录

嵌入式人工智能开发

推荐阅读

嵌入式开发

嵌入式小强

嵌入式硬件开发的步骤

编程小霸王

嵌入式入门学习路线（更新中）

从事嵌入式相关工作整整四年，在此分享一下嵌入式入门学习路线吧。 一、linux系统操作基础《鸟哥的私房菜》 shell Makefile 二、C语言《C语言参考手册》 三、数据结构《大话数据结构》 四、...

Fartl... 发表于老程序员逆...

嵌入式学习的八大

一、嵌入式C语言 C
域最重要也是最主要
通过大量编程实例重
基础编程以及高级编
括：基本数据类型、
结构体、链表、文件

苏老师