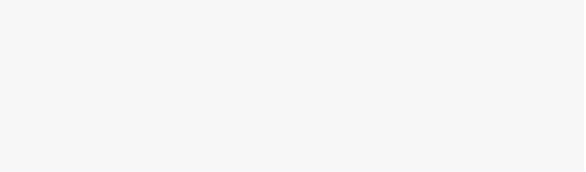
海康嵌入式笔试题

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/95030c891a364370a5fd35aa9fae2253.jpg.sub1.png/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/95030c891a364370a5fd35aa9fae2253.jpg.sub2.png

**海康嵌入式笔试题**

**1.**请问TCP/IP协议分为哪几层？FTP协议属于哪一层？

**答：可以分为 物理层，数据链路层，网络层，传输层，应用层**



**应用层：**应用程序间沟通的层，如简单电子传输（SMTP）、

文件传输协议（FTP）、网络远程访问协议（**Telnet**）等。

**传输层：**在此层中，它提供了节点间的数据传送，应用程序之间的通信服务，

主要功能是数据格式化、数据确认和丢失重传等。

如传输控制协议（TCP）、用户数据报协议（UDP）等，

TCP和UDP给数据包加入传输数据并把它传输到下一层中，这一层负责传送数据，

并且确定数据已被送达并接收。

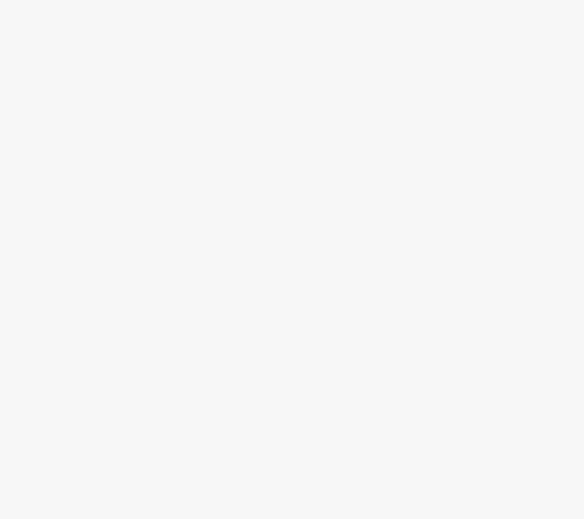
**互连网络层：负责提供基本的数据封包传送功能，让每一块数据包都能够到达目**

**的主机（但不检查是否被正确接收），如网际协议（IP）。如ARP是地址解析协议，**

**在这一层的时候吧IP地址转换为物理地址**

**网络接口层（主机-网络层）：接收IP数据报并进行传输，从网络上接收物理帧，**

**抽取IP数据报转交给下一层，对实际的网络媒体的管理，定义如何使用实际网络（如**

**Ethernet、Serial Line等）来传送数据。**

另外补充一下OSI的七层所对应的协议：

应用层（Application)：

应用程序网关（application gateway）Telnet: 远程登录

（在应用层连接两部分应用程序）

FTP（File Transfer Protocol）：文件传输协议

HTTP（Hyper Text Transfer Protocol）：超文本传输协议

SMTP（Simple Mail Transter Protocol）：简单传输协议

POP3（Post Office Ptotocol）：邮局协议

SNMP（Simple Network Mangement Protocol）简单网络管理协议

DNS（Domain Name System）：域名系统

传输层（Transport）：

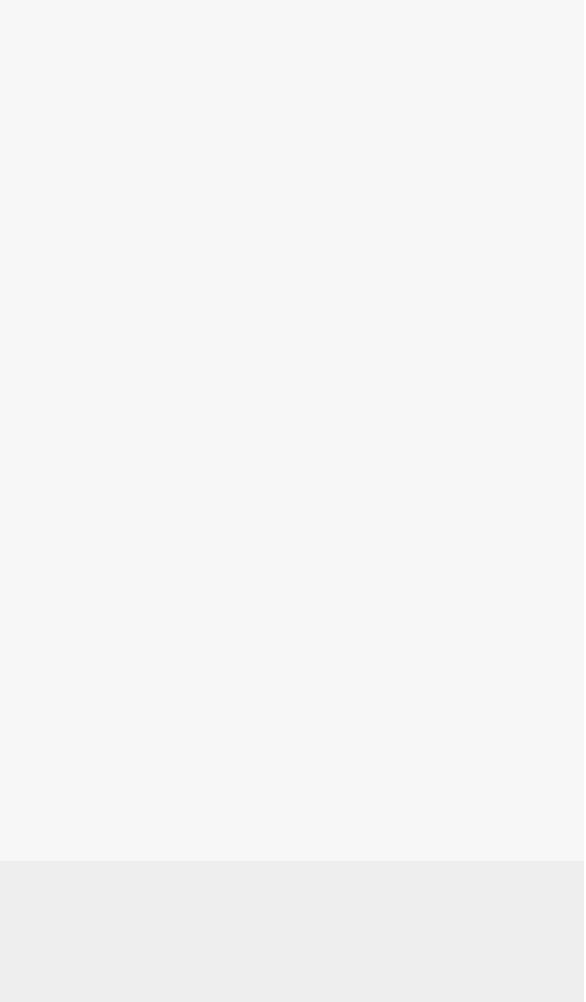
传输网关（transport gateway）

TCP（Transmission Control Potocol)：传输控制协议

（在传输层连接两个网络）

UDP（User Data Potocol）：用户数据协议

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/f2d88225442648cabeaa1c1b1a68d02e.jpg.sub1.png

网络层（Internet）：

多协议路由器（multiprotocol router）

IP（Internet Protocol）：网络协议 （在异构网络间转发分组）

ARP（Address Resolution Protocol）：地址解析协议

RARP（Reverse Address Resolution Protocol) ：逆地址解析协议

ICMP（Internet Control Message Protocol）：因特网控制消息协议

IGMP（Internet Group Manage Protocol）：因特网组管理协议

BOOTP （Bootstrap）：可选安全启动协议

数据链路层（Data Link）：

网桥（bridge）交换机（switcher）

HDLC（High Data Link Control）：高级数据链路控制

（在LAN之间存储-转发数据链路针）

SLIP（Serial Line IP）：串行线路IP

PPP（Point-to-Point Protocol）：点到点协议802.2等

物理层（Physical）：

中继器（repeater） 集线器（hub）

（放大或再生弱的信号，在两个电缆段之间复制每一个比特）

再回过头来看，ftp是用tcp写的一个文件传输协议，明显就是在应用层了

====================================================================

2.在网络应用中，函数htons，htonl，ntohs，nt

ohl的作用？

htons： 把短整型的主机字节顺序转变为网络字节顺序

其它依次类推

htons一般用来转变端口，htonl一般用来转IP

网络字节顺序是先高位字节，再低位字节；主机字节顺序则视cpu而定

干脆扩展一点

#include<unistd.h>

#include<stdio.h>

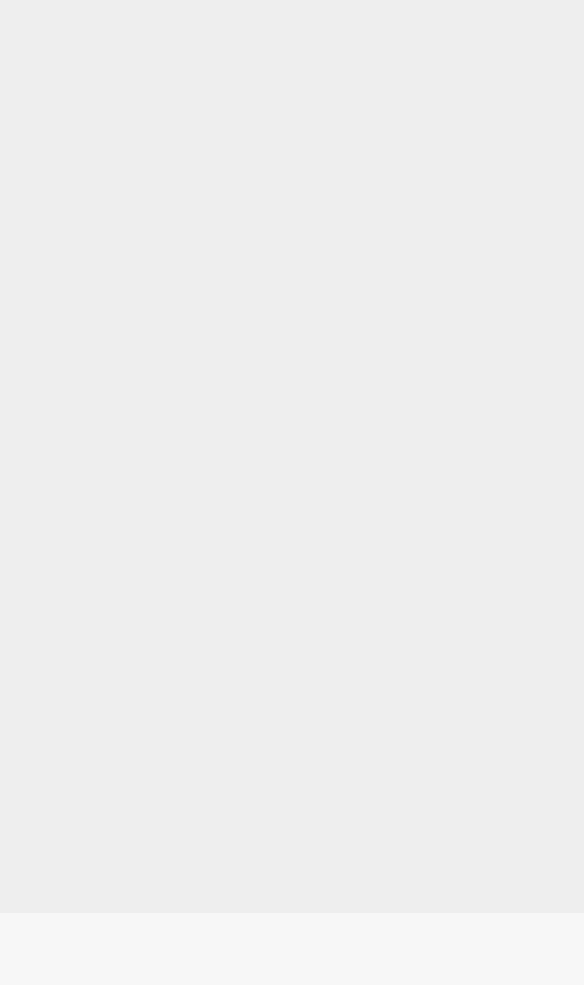
#include<stdlib.h>

#include<string.h>

#include<netinet/in.h>

#include<sys/socket.h>

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/94b64bbe520446cb8012435abe8db685.jpg.sub1.png

#include<sys/types.h>

int main()

{

char str[]="255.255.255.255";

in\_addr\_t r1,r2,r3;

struct in\_addr inp;

r1=inet\_addr(str);

if(r1==-1)

{

printf("inet\_addr return -1 when 255.255.255.25

5\n");

}

else

{

printf("inet\_addr:ip=%lu\n",ntohl(r1));

}

r2=inet\_network(str);

if(r2==-1)

{

printf("inet\_network return -1 when 255.255.255.2

55\n");

}

else

{

printf("inet\_network:ip=%lu\n",r2);

}

r3=inet\_aton(str,&inp);

if(r3==0)

{

printf("inet\_aton return -1 when 255.255.255.255\n

");

}

else

{

printf("inet\_aton:ip=%lu\n",ntohl(inp.s\_addr));

}

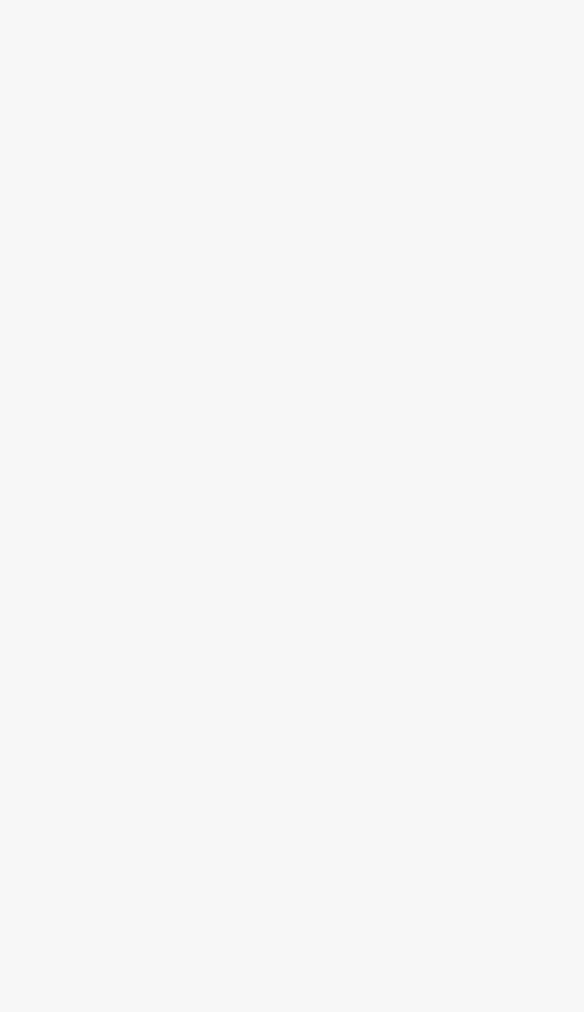
return 0;

}

====================================================================

3.在c语言中，static函数与普通函数有什么区

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/fda5e90e391e4a98902d3fbfcf46be0d.jpg.sub1.png

别？

1. 全局静态变量

在全局变量之前加上关键字static，全局变量就被定义成为一个全局静态变量。

1）存中的位置：静态存储区（静态存储区在整个程序运行期间都存在）。

2）初始化：未经初始化的全局静态变量会被程序自动初始化为0（自动对象的值是任意的，除

非他被显示初始化）。

3）作用域：全局静态变量在声明他的文件之外是不可见的。准确地讲从定义之处开始到文件结

尾。

定义全局静态变量的好处：

<1>不会被其他文件所访问，修改。

<2>其他文件中可以使用相同名字的变量，不会发生冲突。

2. 局部静态变量

在局部变量之前加上关键字static，局部变量就被定义成为一个局部静态变量。

1）存中的位置：静态存储区。

2）初始化：未经初始化的全局静态变量会被程序自动初始化为0（自动对象的值是任意的，除

非他被显示初始化）。

3）作用域：作用域仍为局部作用域，当定义它的函数或者语句块结束的时候，作用域随之结束。

注：当static用来修饰局部变量的时候，它就改变了局部变量的存储位置，从原来的栈中存放

改为静态存储区。考试,大提示局部静态变量在离开作用域之后，并没有被销毁，而是仍然驻留在存

当中，直到程序结束，只不过我们不能再对他进行访问。

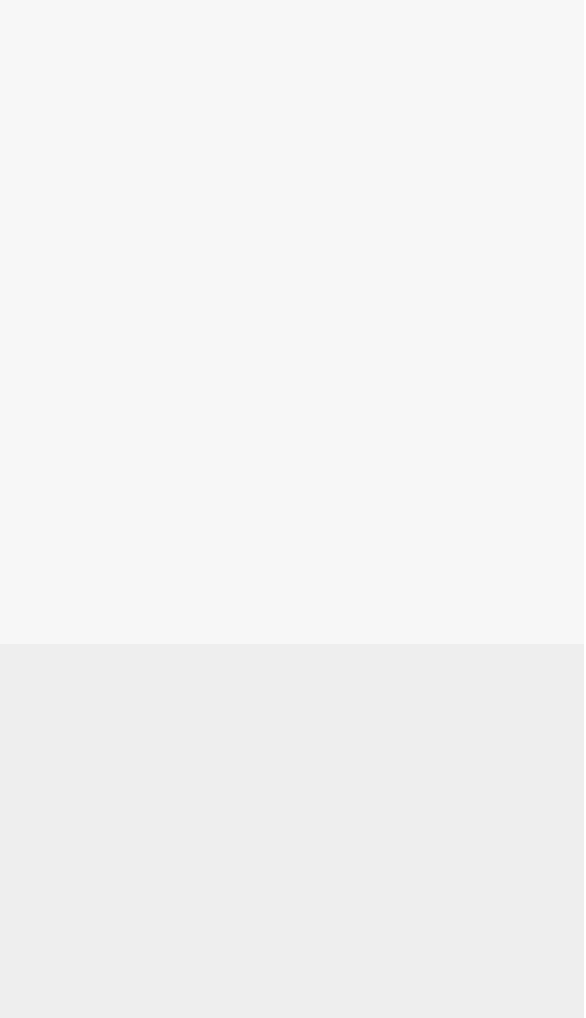
当static用来修饰全局变量的时候，它就改变了全局变量的作用域（在声明他的文件之外是不

可见的），但是没有改变它的存放位置，还是在静态存储区中。

3. 静态函数

在函数的返回类型前加上关键字static，函数就被定义成为静态函数。

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/a388898248c945168ac3027451634f07.jpg.sub1.png

函数的定义和声明默认情况下是extern的，但静态函数只是在声明他的文件当中可见，不能被

其他文件所用。

定义静态函数的好处：

<1> 其他文件中可以定义相同名字的函数，不会发生冲突。

<2> 静态函数不能被其他文件所用。

存储说明符auto，register，extern，static，对应两种存储期：自动存储期和静态存储期。

auto和register对应自动存储期。具有自动存储期的变量在进入声明该变量的程序块时被建

立，它在该程序块活动时存在，退出该程序块时撤销。

关键字extern和static用来说明具有静态存储期的变量和函数。用static声明的局部变量具

有静态存储持续期（static storage duration），或静态围（static extent）。虽然他的值在函

数调用之间保持有效，但是其名字的可视性仍限制在其局部域。静态局部对象在程序执行到该对象

的声明处时被首次初始化。

扩展分析：

术语static有着不寻常的历史.起初，在C中引入关键字static是为了表示退出一个块后仍然

存在的局部变量。随后，static C中有了第二种含义：用来表示不能被其它文件访问的全局变量和

函数。为了避免引入新的关键字，所以仍使用static关键字来表示这第二种含义。最后，C++重用

了这个关键字，并赋予它与前面不同的第三种含义：表示属于一个类而不是属于此类的任何特定对

象的变量和函数(与Java中此关键字的含义相同)。

写个例子，比对一下

#include <stdio.h>

#include <stdlib.h>

extern int a;//声明

static int b=5;

a=0;//初始化

void func1()

{

printf("a=%d;b=%d\n",a,b);

return ;

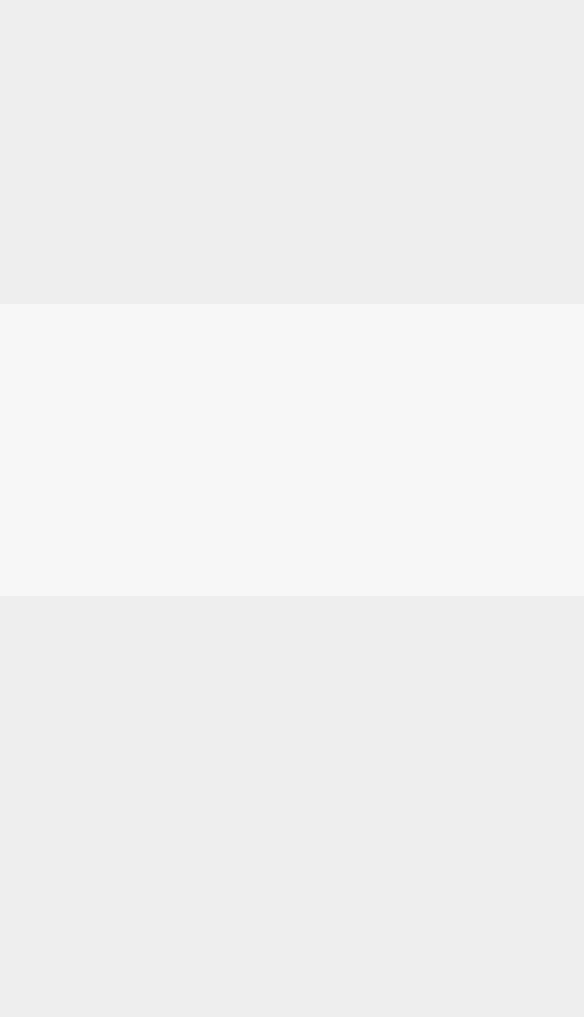
}

void func2()

{

static int i=333;

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/d0d50df236134f3098f21482c91c5629.jpg.sub1.png

i++;

printf("i=%d\n",i);

return ;

}

int main()

{

a=20;

func1();

func2();

func2();

return 0;

}

输出为：

a=20;b=5

i=334

i=335

总的来说，静态的东西就是只在本文件中可见，而且在本文件中保持，它只初始化一次，存在静态

存储区中，再对它进行初始化不会引起错误，但是没有用了

====================================================================

4.请实现存复制函数

void memcpy（void \*dst,void \*src,int size）?

网上搜了一下

void \*MyMemCopy(void \*dest,const void \*src,size\_t count)

{

char \*pDest=static\_cast<char \*>(dest);

const char \*pSrc=static\_cast<const char \*>(src);

if( pDest>pSrc && pDest<pSrc+count )

{

for(size\_t i=count-1; i<=0; ++i)

{

pDest[i]=pSrc[i];

}

}

else

{

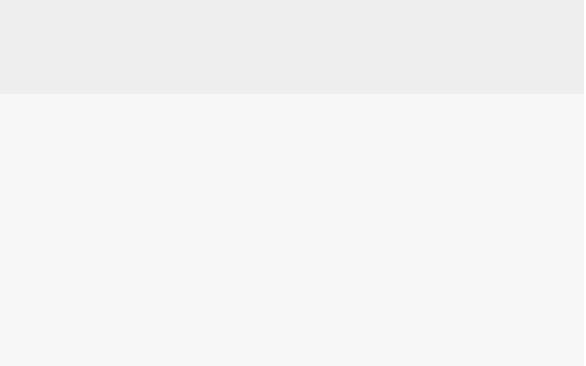
for(size\_t i=0; i<count; ++i)

{

pDest[i]=pSrc[i];

}

专业WORD.

/root/apache-tomcat-baidu_catch/baiduwenkutmp/2/24d7cc89f5ec4afe04a1b0717fd5360cba1a8d88.pc/8f3d8d3f0ebf405f94900837b1c74298.jpg.sub1.png

}

return pDest;

}

哎，搞IT真的没有前途，你看，人都这么聪明的？！

就这还考虑到了，幸好，我还不是笨到家，还能看得懂

稍微解释下：

如果是这个情况：

src --------------------

dst --------------------------

说明两个在存区有重叠的地方。

如果用dst[0]=src[0]

那么说明src的某块会被弄脏掉，怎么办？从后面拷起就不会有问题，dst[size-1]=src[size-1]

如果没有重叠的话就从头拷起

如程序所见