华为面试、笔试题汇总

（适用于：软件开发工程师）

予人玫瑰，手有余香，九度互动社区伴你一路同行！

感谢九度互动社区网友手工录入了这份试卷，任何组织和个人无权将其用于任何商业赢利为目的的活动！

整理人：浩帆

发布时间：2010.11.14

（1） 什么是预编译，何时需要预编译：

答案：

1、总是使用不经常改动的大型代码体。

2、程序由多个模块组成，所有模块都使用一组标准的包含文件和相同的编译选项。在这种情况下，可以将所有包含文件预编译为一个预编译头。

（2） char * const p char const * p const char *p 上述三个有什么区别？

答案：

char * const p; //常量指针，p 的值不可以修改

char const * p; //指向常量的指针，指向的常量值不可以改 const char *p; //和 char const *p

（3）

```
char str1[] = "abc";
char str2[] = "abc";
const char str3[] = "abc";
const char str4[] = "abc";
const char *str5 = "abc";
const char *str6 = "abc";
char *str7 = "abc";
char *str8 = "abc";
cout < < ( str1 == str2 ) < < endl;
cout < < ( str3 == str4 ) < < endl;
cout < < ( str5 == str6 ) < < endl;
cout < < ( str7 == str8 ) < < endl;
```

结果是：0 0 1 1 str1,str2,str3,str4 是数组变量，它们有各自的内存空间；而 str5,str6,str7,str8 是指针，它们指向相同的常量区域。

（4） 以下代码中的两个 sizeof 用法有问题吗？

```
void UpperCase( char str[] ) // 将 str 中的小写字母转换成大写字母
{
    for( size_t i=0; i < sizeof(str)/sizeof(str[0]); ++i )
        if( 'a' <=str[i] && str[i] <='z' )
            str[i] -= ('a'-'A' );
}
char str[] = "aBcDe";
cout < < "str 字符长度为: " < < sizeof(str)/sizeof(str[0]) < < endl;
UpperCase( str );
cout < < str < < endl;
```

答案：函数内的 sizeof 有问题。根据语法，**sizeof 如用于数组，只能测出静态数组的大小**，无法检测动态分配的或外部数组大小。函数外的 str 是一个静态定义的数组，因此其大小为 6，因为还有'\0'，函数内的 str 实际只是一个指向字符串的指针，没有任何额外的与数组相关的信息，因此 sizeof 作用于上只将其当指针看，一个指针为 4 个字节，因此返回 4。

（5） 一个 32 位的机器,该机器的指针是多少位答案：

指针是多少位只要看地址总线的位数就行了。80386 以后的机子都是 32 的地址总线。所以指针的位数就是 4 个字节了。

（6）

```
main()
{
    int a[5]={1,2,3,4,5};
    int *ptr=(int *)(&a+1);
    printf("%d,%d",*(a+1),*(ptr-1));
}
```

答案：2，5。

(a+1）就是 a[1]，(ptr-1)就是 a[4],执行结果是 2，5。&a+1 不是首地址+1，系统会认为加一个 a 数组的偏移，是偏移了一个数组的大小（本例是 5 个 int）。

int *ptr=(int *)(&a+1)；则 ptr 实际是&(a[5]),也就是 a+5 原因如下： &a 是数组指针，其类型为 int (*)[5]；而指针加 1 要根据指针类型加上一定的值，不同类型的指针+1 之后增加的大小不同 a 是长度为 5 的 int 数组指针，所以要加 5*sizeof(int) 所以 ptr 实际是 a[5] 但是 ptr 与(&a+1)类型是不一样的(这点很重要) 所以 prt-1 只会减去 sizeof(int*) a,&a 的地址是一样的，但意思不一样，a 是数组首地址，也就是 a[0]的地址，&a 是对象（数组）首地址，a+1 是数组下一元素的地址，即 a[1],&a+1 是下一个对象的地址，即 a[5].

（7） 请问以下代码有什么问题：

```
int  main()
{
    char a;
    char *str=&a;
    strcpy(str,"hello");
    printf(str);
    return 0;
}
```

答案：没有为 str 分配内存空间，将会发生异常问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。

（8）

```
char* s="AAA";
printf("%s",s);
s[0]='B';
printf("%s",s);
```

有什么错?

答案：**"AAA"是字符串常量**。s 是指针，指向这个字符串常量，所以声明 s 的时候就有问题。 cosnt char* s="AAA"; 然后又因为是常量，所以对是 s[0]的赋值操作是不合法的。

（9） 写一个“标准”宏，这个宏输入两个参数并返回较小的一个。

答案：#define Min(X, Y) ((X)>(Y)?(Y):(X)) //结尾没有‘；’

（10） 嵌入式系统中经常要用到无限循环，你怎么用 C 编写死循环。

答案：while(1){}或者 for(;;)

（11） 关键字 static 的作用是什么？

答案： 修饰变量：静态全局变量（本文件内可见，作用域从定义开始，文件结尾处结束。定义处之前的代码也无法使用 extern 访问它） 和 静态局部变量（只初始化一次）

修饰函数：函数只在本文件内可见。

（12） 关键字 const 有什么含意？

答案：表示常量不可以修改的变量。

（13） 关键字 volatile 有什么含意？并举出三个不同的例子？

答案：提示编译器对象的值可能在编译器未监测到的情况下改变，防止编译器优化。

```
int i = 10;
int j = i;
int k = i;
```

（14） int (*s[10])(int) 表示的是什么啊？

答案：int (*s[10])(int) 函数指针数组，每个指针指向一个 int func(int param)的函数。

（15） 有以下表达式：

```
int a=248;
int b=4;
int const *c=21;
const int *d=&a;
int *const e=&b;
int const *f const =&a;
```

请问下列表达式哪些会被编译器禁止？为什么？

```
*c=32;
d=&b;
*d=43;
e=34;
e=&a;
f=0x321f;
```

答案:*c=32 这是个什么东东，禁止

*d=43 说了是 const， 禁止

e = &a,e=34 说了是 const 禁止

const *f const =&a; 禁止

（16） 交换两个变量的值，不使用第三个变量。即 a=3,b=5,交换之后 a=5,b=3；

答案：有两种解法,

一种用算术算法,

```
a = a + b;
b = a - b;
a = a - b;
```

一种用^(异或)

```
a = a^b;// 只能对 int,char..
b = a^b;
a = a^b;
```

or

```
a ^= b ^= a;
```

（17） c 和 c++中的 struct 有什么不同？

答案：c 和 c++中 struct 的主要区别是 c 中的 struct 不可以含有成员函数，而 c++中的 struct 可以。c++中 struct 和 class 的主要区别在于默认的存取权限不同，struct 默认为 public，而 class 默认为 private

（18）

```
#include <stdio.h>
#include <stdlib.h>
void getmemory(char *p)
{
    p=(char *) malloc(100);
    strcpy(p,"hello world");
}
int main( )
{
    char *str=NULL;
    getmemory(str);
    printf("%s/n",str);
    free(str);
    return 0;
}
```

答案：程序崩溃，getmemory 中的 malloc 不能返回动态内存， free（）对 str 操作很危险

（19）

```
char szstr[10];
strcpy(szstr,"0123456789");
```

产生什么结果？为什么？

答案： 长度不一样，会造成非法的 OS

（20） 列举几种进程的同步机制，并比较其优缺点。

答案：

原子操作、信号量机制、自旋锁、管程、会合、分布式系统

（21） 进程之间通信的途径答案：

共享存储、系统消息传递、系统管道：以文件系统为基础

（22） 进程死锁的原因

答案：

（1）因为系统资源不足。

（2）进程运行推进的顺序不合适。

（3）资源分配不当等。

（23） 死锁的 4 个必要条件

答案：

（1） 互斥条件：一个资源每次只能被一个进程使用。

（2） 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

（3） 不剥夺条件:进程已获得的资源，在未使用完之前，不能强行剥夺。

（4） 循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系

（24） 死锁的处理

答案：鸵鸟策略、预防策略、避免策略、检测与解除死锁

（25） 操作系统中进程调度策略有哪几种？

答案：FCFS(先来先服务)，优先级，时间片轮转，多级反馈

（26） 类的静态成员和非静态成员有何区别？

答案：类的静态成员每个类只有一个，非静态成员每个对象一个

（27） 纯虚函数如何定义？使用时应注意什么？

答案：virtual void f()=0; 是接口，子类必须要实现

（28） 数组和链表的区别

答案： 数组：数据顺序存储，固定大小

链表：数据可以随机存储，大小可动态改变

（29） ISO 的七层模型是什么？tcp/udp 是属于哪一层？tcp/udp 有何优缺点？

答案：应用层、表示层、会话层、传输层、网络层、物理链路层、物理层

tcp /udp 属于传输层。

TCP 服务提供了数据流传输、可靠性、有效流控制、全双工操作和多路复用技术等。与 TCP 不同， UDP 并不提供对 IP 协议的可靠机制、流控制以及错误恢复功能等。由于 UDP 比较简单， UDP 头包含很少的字节，比 TCP 负载消耗少。

TCP: 提供稳定的传输服务，有流量控制，缺点是包头大，冗余性不好 udp: 不提供稳定的服务，包头小，开销小

（30） (void *)ptr 和 (*(void))ptr 的结果是否相同？其中 ptr 为同一个指针**

答案：.(void *)ptr 和 (*(void**))ptr 值是相同的

（31）

```
int main()
{
    int x=3;
    printf("%d",x);
    return 1;
}
```

问函数既然不会被其它函数调用，为什么要返回 1?

答案：main 中，c 标准认为 0 表示成功，非 0 表示错误。具体的值是某中具体出错信息 。将相关信息提供给系统。

（32） 要对绝对地址 0x100000 赋值，我们可以用 (unsigned int*)0x100000 = 1234; 那么要是想让程序跳转到绝对地址是 0x100000 去执行，应该怎么做？

批注 [DavidHoo1]: 绝对地址？程序员看得到绝对地址？

答案：*((void (*)( ))0x100000) ( );

首先要将 0x100000 强制转换成函数指针,即: (void (*)())0x100000 然后再调用它: *((void (*)())0x100000)();

用 typedef 可以看得更直观些:

```
typedef void(*)() voidFuncPtr;
*((voidFuncPtr)0x100000)();
```

（33） 已知一个数组 table，用一个宏定义，求出数据的元素个数

答案： #define NTBL (sizeof(table)/sizeof(table[0]))

（34） 线程与进程的区别和联系？线程是否具有相同的堆栈？dll 是否有独立的堆栈？

答案：进程是死的，只是一些资源的集合，真正的程序执行都是线程来完成的，程序启动的时候操作系统就帮你创建了一个主线程。每个线程有自己的堆栈。

DLL 中有没有独立的堆栈，这个问题不好回答，或者说这个问题本身是否有问题。因为 DLL 中的代码是被某些线程所执行,只有线程拥有堆栈,如果 DLL 中的代码是 EXE 中的线程所调用,那么这个时候是不是说这个 DLL 没有自己独立的堆栈？如果 DLL 中的代码是由 DLL 自己创建的线程所执行，那么是不是说 DLL 有独立的堆栈？以上讲的是堆栈，如果对于堆来说，每个 DLL 有自己的堆，所以如果是从 DLL 中动态分配的内存，最好是从 DLL 中删除，如果你从 DLL 中分配内存，然后在 EXE 中，或者另外一个 DLL 中删除，很有可能导致程序崩溃

批注 [DavidHoo2]: 不懂

（35）

```
unsigned short A = 10;
printf("~A = %u\n", ~A);
char c=128;
printf("c=%d\n",c);
```

输出多少？并分析过程

答案：

第一题，～A ＝0xfff5,int 值 为－11，但输出的是 uint。所以输出 4294967285

第二题，c＝0x10,输出的是 int，最高位为 1，是负数，所以它的值就是 0x00 的补码就是 128，所以输出－128。这两道题都是在考察二进制向 int 或 uint 转换时的最高位处理。

（二）

1. -1,2,7,28,,126 请问 28 和 126 中间那个数是什么？为什么？

答案：第一题的答案应该是 4^3-1=63 规律是 n^3-1(当 n 为偶数 0，2，4) n^3+1(当 n 为奇数 1，3，5)

2.用两个栈实现一个队列的功能？要求给出算法和思路！

答案：设 2 个栈为 A,B, 一开始均为空.

入队：将新元素 push 入栈 A;

出队:判断栈 B 是否为空：

(1)如果为空，则将栈 A 中所有元素依次 pop 出并 push 到栈 B，再将栈 B 的栈顶元素 pop 出；

(2)如果不为空，则直接将栈 B 的栈顶元素 pop 出；

这样实现的队列入队和出队的平摊复杂度都还是 O(1)，比上面的几种方法要好。

3.在 c 语言库函数中将一个字符转换成整型的函数是 atol()吗，这个函数的原型是什么？

答案：函数名：atol 功 能：把字符串转换成长整型数 用 法：long atol(const char *nptr);

程序

例:

```
#include <stdlib.h>
#include <stdio.h>
int main(void)
```

```
{
    long l;
    char *str = "98765432";
    l = atol(lstr);
    printf("string = %s integer = %ld\n", str, l);
    return(0);
}
```

4。对于一个频繁使用的短小函数,在 C 语言中应用什么实现,在 C++中应用什么实现?

答案：c 用宏定义，c++用 inline

5。直接链接两个信令点的一组链路称作什么?

答案：PPP 点到点连接

7。软件测试都有那些种类?

答案：黑盒：针对系统功能的测试　　白盒：测试函数功能，各函数接口

8。确定模块的功能和模块的接口是在软件设计的那个阶段完成的?

答案：概要设计阶段

9。enum string　{　x1,　x2,　x3=10,　x4,　x5,　}x;　问 x;

答案：取值在 0。1。10。11。12 中的一个

10。

```
unsigned char *p1;
unsigned long *p2;
p1=(unsigned char *)0x801000;
p2=(unsigned long *)0x810000;
请问 p1+5=  ;
     p2+5=  ;
```

答案：0x801005；　0x810014。不要忘记了这个是 16 进制的数字，p2 要加 20 变为 16 进制就是 14

选择题:

1.Ethternet 链接到 Internet 用到以下那个协议?

A.HDLC;B.ARP;C.UDP;D.TCP;E.ID

2.属于网络层协议的是:

A.TCP;B.IP;C.ICMP;D.X.25

3.Windows 消息调度机制是:　A.指令队列;B.指令堆栈;C.消息队列;D.消息堆栈;

答案：b，a，c

四.找错题:

1.请问下面程序有什么错误?

```
int a[60][250][1000],i,j,k;
for(k=0;k <=1000;k++)
```

```
    for(j=0;j <250;j++)
        for(i=0;i <60;i++)
            a[i][j][k]=0;
```

答案：把循环语句内外换一下

批注 [DavidHoo3]: 程序本身没有错，但是因为 C 语言是按行序存储的，因此应该先从最左边的开始循环效率更高！

2。以下是求一个数的平方的程序,请找出错误:

```
#define SQUARE(a)  ((a)*(a))
int a=5;
int b;
b=SQUARE(a++);
```

答案：这个没有问题，SQUARE（a＋＋），就是（（a＋＋）×（a＋＋））唯一要注意的就是计算后 a＝7 了

3。

```
typedef unsigned char BYTE
int examply_fun(BYTE gt_len; BYTE *gt_code)
{
    BYTE *gt_buf;
    gt_buf=(BYTE*)MALLOC(Max_GT_Length);
    ......
    if(gt_len>Max_GT_Length)
    {
        return GT_Length_ERROR;
    }
    .......
}
```

答案：要释放内存

问答题:

1.IP Phone 的原理是什么?

答案：IPV6

2.TCP/IP 通信建立的过程怎样，端口有什么作用?

答案：三次握手，确定是哪个应用程序使用该协议

（三）

1、局部变量能否和全局变量重名?

答案：能，局部会屏蔽全局。要用全局变量，需要使用"::" 局部变量可以与全局变量同名，在函数内引用这个变量时，会用到同名的局部变量，而不会用到全局变量。对于有些编译器而言，在同一个函数内可以定义多个同名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内

2、如何引用一个已经定义过的全局变量?

答案：extern 可以用引用头文件的方式，也可以用 extern 关键字，如果用引用头文件方式来引用某个在头文件中声明的全局变理，假定你将那个变写错了，那么在编译期间会报错，如果你用 extern 方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在连接期间报错

3、全局变量可不可以定义在可被多个.C 文件包含的头文件中？为什么？

答案：可以，在不同的 C 文件中以 static 形式来声明同名全局变量。可以在不同的 C 文件中声明同名的全局变量，前提是其中只能有一个 C 文件中对此变量赋初值，此时连接不会出错

4、语句 for(；1 ；)有什么问题？它是什么意思?

答案：和 while(1)相同。

5、do……while 和 while……do 有什么区别？

答案：前一个循环一遍再判断，后一个判断以后再循环。

6、请写出下列代码的输出内容

```
#include <stdio.h>
main()
{
    int a,b,c,d;
    a=10;
    b=a++;
    c=++a;
    d=10*a++;
    printf("b, c, d: %d, %d, %d", b, c, d) ;
    return 0;
}
```

答案：10，12，120 a=10； b=a++;//a=11 b=10 c=++a;//a=12 c=12 d=10*a++;//a=13 d=120

高级题

1、static 全局变量与普通的全局变量有什么区别？static 局部变量和普通局部变量有什么区别？static 函数与普通函数有什么区别？

答案：全局变量(外部变量)的说明之前再冠以 static 就构成了静态的全局变量。全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序，当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。而静态全局变量则限制了其作用域，即只在定义该变量的**源文件**内有效，在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，因此可以避免在其它源文件中引起错误。

把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域，限制了它的使用范围。

static 函数与普通函数作用域不同。仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件

static 全局变量与普通的全局变量有什么区别：static 全局变量只初始化一次，防止在其他文件单元中被引用；

static 局部变量和普通局部变量有什么区别：static 局部变量只被初始化一次，下一次依据上一次结果值；

static 函数与普通函数有什么区别：static 函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

2、程序的局部变量存在于（）中，全局变量存在于（）中，动态申请数据存在于（ ）中。

答案：栈；静态区；堆

3、设有以下说明和定义：

```
typedef union
{
    long i;
    int k[5];
    char c;
} DATE;
```

```
struct data
{
    int cat;
    DATE cow;
    double dog;
} too;
DATE max;
```

则语句 printf("%d",sizeof(too)+sizeof(max));的执行结果是：_____

答案：DATE 是一个 union, 变量公用空间. 里面最大的变量类型是 int[5], 占用 20 个字节. 所以它的大小是 20 data 是一个 struct, 每个变量分开占用空间. 依次为 int4 + DATE20 + double8 = 32. 所以结果是 20 + 32 = 52. 当然...在某些 16 位编辑器下, int 可能是 2 字节,那么结果是 int2 + DATE10 + double8 = 20

4、队列和栈有什么区别？

答案：队列先进先出，栈后进先出

5、这道题目出错了，这里就不写上了。

6、已知一个单向链表的头，请写出删除其某一个结点的算法，要求，先找到此结点，然后删除。

7、请找出下面代码中的所以错误说明：以下代码是把一个字符串倒序，如“abcd”倒序后变为“dcba”

```
1、#include"string.h"
2、main()
3、{
4、  char*src="hello,world";
5、  char* dest=NULL;
6、  int len=strlen(src);
7、  dest=(char*)malloc(len);
8、  char* d=dest;
9、  char* s=src[len];
10、 while(len--!=0)
11、 d++=s--;
12、 printf("%s",dest);
13、 return 0;
14、}
```

答案：还要加上#include <stdio.h>

```
int main()
{
    char* src = "hello,world";
    int len = strlen(src);
    char* dest = (char*)malloc((len+1)*sizeof(char));  //要为\0 分配一个空间
    char* d = dest;
    char* s = &src[len-1]; //指向最后一个字符
    while( len-- != 0 )
        *d++=*s--;
        *d = 0; //尾部要加\0
    printf("%s\n",dest);
```

```
    free(dest);// 使用完，应当释放空间，以免造成内存汇泄露
    return 0;
}
```

华为笔试题（3） 2006-09-29 19:41

一、判断题（对的写 T，错的写 F 并说明原因，每小题 4 分，共 20 分）

1、有数组定义 int a[2][2]={{1},{2,3}};则 a[0][1]的值为 0。（正确）

2、int (*ptr) (),则 ptr 是一维数组的名字。（错误 int (*ptr) ();定义一个指向函数的指针变量）

3、指针在任何情况下都可进行>, <,>=, <=,==运算。（错误）

4、switch(c) 语句中 c 可以是 int ,long,char ,float ,unsigned int 类型。（ 错，不能用浮点数）

二、填空题（共 30 分）

1、在 windows 下，写出运行结果，每空 2 分，共 10 分。

```
char str[ ]= "Hello";
char *p=str;
int n=10;
sizeof(str)=(      )
sizeof(p)=(      )
sizeof(n)=(      )
void func(char str[100]){    }
sizeof(str)=(    )
```

答案：6，4，4，4，具体解释请参看我的空间里的“C/C++程序员应聘试题剖析”

2、

```
void getmemory(char **p, int num)
{
    *p=(char *) malloc(num);
}
void test(void)
{
    char *str=NULL;
    getmemory(&str,100);
    strcpy(str,"hello");
    printf(str);
}
```

运行 test 函数有什么结果？（ ）10 分

答案：输出 hello，但是发生内存泄漏。 多分配了很大一部分空间。

3、设

```
int arr[]={6,7,8,9,10};
int *ptr=arr;
*(ptr++)+=123;
printf("%d,%d",*ptr,*(++ptr)); (                    ) 10 分
```

答案：8，8。

这道题目的意义不大，因为在不同的编译器里 printf 的参数的方向是不一样的，在 vc6.0 下是从右到左，这里先*(++ptr)后*ptr，于是结果为 8，8

三、编程题（第一小题 20，第二小题 30 分）

1、 不使用库函数，编写函数 int strcmp(char *source, char *dest) 相等返回 0，不等返回-1；

答案：一、

```
int strcmp(char  *source, char *dest)
{
    assert((source!=NULL)&&(dest!=NULL));
    int i,j;
    for(i=0; source[i]==dest[i]; i++)
    {
        if(source[i]=='\0' && dest[i]=='\0')
            return 0;
    }
    return -1;
}
```

答案：二、

```
int strcmp(char *source, char *dest)
{
    while ( (*source != '\0') && (*source == *dest))
    {
        source++;
        dest++;
    }
    return ( (*source) - (*dest) ) ? -1 : 0;
}
```

2、 写一函数 int fun(char *p)判断一字符串是否为回文，是返回 1，不是返回 0，出错返回-1

答案：

一、

```
int fun(char *p)
{
    if(p==NULL)
        return -1;
    else
    {
        int length = 0;
        int i = 0;
        int judge = 1;
        length = strlen(p);
        for(i=0; i <length/2; i++)
        {
            if(p[i]!=p[length-1-i])
            judge = 0;
            break;
        }
        if(judge == 0)
            return 0;
        else
            return 1;
```

```
        }
    }
```

答案：

二、

```
int fun(char *p)
{
    int len = strlen(p) - 1;
    char *q = p + len;
    if (!p)
        return -1;
    while (p < q)
    {
        if ((*p++) != (*q--))
            return 0;
    }
    return 1;
}
```

华为笔试网络题（3） 2006-09-30 12:48

1.在 OSI 7 层模型中,网络层的功能有( )

A.确保数据的传送正确无误 B.确定数据包如何转发与路由 C.在信道上传送比特流 D.纠错与流控

2.FDDI 使用的是___局域网技术。( )

A.以太网； B.快速以太网； C.令牌环； D.令牌总线。

3.下面那种 LAN 是应用 CSMA/CD 协议的 （）

A.令牌环 B.FDDI C.ETHERNET D.NOVELL

4.TCP 和 UDP 协议的相似之处是 ( )

A.面向连接的协议 B.面向非连接的协议 C.传输层协议 D.以上均不对

5.应用程序 PING 发出的是___报文.( )

A.TCP 请求报文。 B.TCP 应答报文。 C.ICMP 请求报文。 D.ICMP 应答报文。

6.以下说法错误的是(多) ( )

A.中继器是工作在物理层的设备 B.集线器和以太网交换机工作在数据连路层 C.路由器是工作在网络层的设备 D.桥能隔离网络层广播

7.当桥接收的分组的目的 MAC 地址在桥的映射表中没有对应的表项时,采取的策略是( )

A.丢掉该分组 B.将该分组分片 C.向其他端口广播该分组 D.以上答案均不对

8.LAN Switch 在网络层次模型中的地位( )

A.物理层 B.链路层 C.网络层 D.以上都不是

9.小于___的 TCP/UDP 端口号已保留与现有服务一一对应,此数字以上的端口号可自由分配。( )

A.199 B.100 C.1024 D.2048

10.当一台主机从一个网络移到另一个网络时,以下说法正确的是 ( )

A.必须改变它的 IP 地址和 MAC 地址

B.必须改变它的 IP 地址,但不需改动 MAC 地址

C.必须改变它的 MAC 地址,但不需改动 IP 地址

D.MAC 地址.IP 地址都不需改动

答案：1.B; 2.C; 3.C; 4.C; 5.C; 6.BD; 7.C; 8.B; 9.C; 10.B.

华为笔试题（4） 2006-09-30 13:00

1. 找错

```
void test1()
{
    char string[10];
    char* str1="0123456789";
    strcpy(string, str1);
}
```

答：表面上并且编译都不会错误。但如果 string 数组原意表示的是字符串的话，那这个赋值就没有达到意图。最好定义为 char string[11]，这样最后一个元素可以存储字符串结尾符'\0'；

```
void test2()
{
    char string[10], str1[10];
    for(int i=0; i <10;i++)
    {
        str1[i] ='a';
    }
    strcpy(string, str1);
}
```

答：strcpy 使用错误，strcpy 只有遇到字符串末尾的'\0'才会结束，而 str1 并没有结尾标志，导致 strcpy 函数越界访问，不妨让 str1[9]='\0'，这样就正常了。

```
void test3(char* str1)
{
    char string[10];
    if(strlen(str1) <=10)
    {
        strcpy(string, str1);
    }
}
```

答：这又会出现第一道改错题的错误了。strlen(str1)算出来的值是不包含结尾符'\0'的，如果 str1 刚好为 10 个字符＋1 结尾符，string 就得不到结尾符了。可将 strlen(str1) <=10 改为 strlen(str1) <10。

2. 找错

```
#define MAX_SRM 256
DSN get_SRM_no()
{
    static int SRM_no;
```

```
    int I;
    for(I=0;I <MAX_SRM;I++,SRM_no++)
    {
        SRM_no %= MAX_SRM;
        if(MY_SRM.state==IDLE)
        {
            break;
        }
    }
    if(I>=MAX_SRM)
        return (NULL_SRM);
    else
        return SRM_no;
}
```

答：我不知道这段代码的具体功能，但明显有两个错误

1，SRM_no 没有赋初值

2，由于 static 的声明，使该函数成为不可重入（即不可预测结果）函数，因为 SRM_no 变量放在程序的全局存储区中，每次调用的时候还可以保持原来的赋值。这里应该去掉 static 声明。

3. 写出程序运行结果

```
int sum(int a)
{
    auto int c=0;
    static int b=3;
    c+=1;
    b+=2;
    return(a+b+c);
}
void main()
{
    int I;
    int a=2;
    for(I=0;I <5;I++)
    {
        printf("%d,", sum(a));
    }
}
```

答：8,10,12,14,16 该题比较简单。只要注意 b 声明为 static 静态全局变量，其值在下次调用时是可以保持住原来的赋值的就可以。

4.

```
int func(int a)
{
    int b;
    switch(a)
    {
```

```
            case 1: b=30;
            case 2: b=20;
            case 3: b=16;
            default: b=0;
        }
        return b;
    } 则 func(1)=?
```

答：func(1)=0，因为没有 break 语句，switch 中会一直计算到 b=0。这是提醒我们不要忘了 break。呵呵。

5:

```
    int a[3];
    a[0]=0;
    a[1]=1;
    a[2]=2;
    int *p, *q;
    p=a;
    q=&a[2];
    则 a[q-p]=?
```

答：a[q-p]=a[2]=2;这题是要告诉我们指针的运算特点

6. 定义 int **a[3][4], 则变量占有的内存空间为：_____

答：此处定义的是指向指针的指针数组，对于 32 位系统，指针占内存空间 4 字节，因此总空间为 3×4×4＝48。

7. 编写一个函数，要求输入年月日时分秒，输出该年月日时分秒的下一秒。如输入 2004 年 12 月 31 日 23 时 59 分 59 秒，则输出 2005 年 1 月 1 日 0 时 0 分 0 秒。

答： /*输入年月日时分秒，输出年月日时分秒的下一秒，输出仍然在原内存空间*/