1. C语言

const static volatile register

#define 和typedef

递归函数、回调函数

存储模型

2.数据结构

数组和链表的区别

链表的插入与删除

链表的倒置

判断一个链表是否有环

1. IO课程以及进程线程

标准i/o和文件i/o的区别

缓存有几种类型

动态库与静态库区别

进程和程序区别

进程与线程的区别

僵尸进程和孤儿进程如何产生的

Fork与vfork

进程间通信有几种

无名管道和有名管道的区别

什么是竞态、同步与互斥

4.网络编程

Linux下C语言实现TCP服务器的初始化流程

TCP/IP模型 和 OSI模型

TCP/IP四层模型以及每层有哪些协议

TCP与UDP区别

IP地址和端口相关知识

服务器模型有哪几种

并发服务器的实现有几种机制

select poll epoll的区别

三次握手和四次挥手

部分参考答案：

C语言类：

(1) C语言中的static的作用

1.在修饰局部变量的时候，static修饰的静态局部变量初始化只执行一次，而且延长了 局部变量的生命周期，直到程序运行结束以后才释放。

2.static修饰全局变量的时候，这个全局变量只能在本文件中访问，不能在其它文件中 访问，即便是extern外部声明也不可以。

3.static修饰一个函数，则这个函数的只能在本文件中调用，不能被其他文件调用。

(2) #define 和typedef 的不同之处

1.执行时间不同

关键字typedef在编译阶段有效，由于是在编译阶段，因此typedef有类型检查的功能。

#define则是宏定义，发生在预处理阶段，也就是编译之前，它只进行简单而机械的 字符串替换，而不进行任何检查。

2.功能有差异

typedef用来定义类型的别名，定义与平台无关的数据类型，与struct的结合使用等。

#define不只是可以为类型取别名，还可以定义常量、变量、编译开关等。

3.作用域不同

#define没有作用域的限制，只要是之前预定义过的宏，在以后的程序中都可以使用。

而typedef有自己的作用域。

4.对指针的操作

二者修饰指针类型时，作用不同。

(3) c语言中使用头文件时尖括号和双引号的区别

1.用尖括号 #include <>：

一般用于包含标准的库头文件，编译器会去系统配置的库环境变量或者用户配置的路径去 搜索，而不会在项目的当前目录去查找

2.用双引号 #include ""：

一般用于包含用户自己编写的头文件，编译器会先在项目的当前目录查找，找不到后才会 去系统配置的库环境变量和用户配置的路径去搜索

1. c语言register

使用register修饰符有几点限制

（1）register变量必须是能被CPU所接受的类型。

这通常意味着register变量必须是一个单个的值，并且长度应该小于或者等于整型的长度。不过，有些机器的寄存器也能存放浮点数。

（2）因为register变量可能不存放在内存中，所以不能用“&”来获取register变量的地址。

（3）只有局部自动变量和形式参数可以作为寄存器变量，其它（如全局变量）不行。

在调用一个函数时占用一些寄存器以存放寄存器变量的值，函数调用结束后释放寄存器。此后，在调用另外一个函数时又可以利用这些寄存器来存放该函数的寄存器变量。

（4）局部静态变量不能定义为寄存器变量。不能写成：register static int a, b, c;

（5）由于寄存器的数量有限（不同的cpu寄存器数目不一），不能定义任意多个寄存器变量，而且某些寄存器只能接受特定类型的数据（如指针和浮点数），因此真正起作用的register修饰符的数目和类型都依赖于运行程序的机器，而任何多余的register修饰符都将被编译程序所忽略。

注意：

　　早期的C编译程序不会把变量保存在寄存器中，除非你命令它这样做，这时register修饰符是C语言的一种很有价值的补充。然而，随着编译程序设计技术的进步，在决定哪些变量应该被存到寄存器中时，现在的C编译环境能比程序员做出更好的决定。实际上，许多编译程序都会忽略register修饰符，因为尽管它完全合法，但它仅仅是暗示而不是命令。

博客：https://blog.csdn.net/21aspnet/article/details/257511

数据结构：

(1)请说出链表与数组的区别

从逻辑结构来看：

（1）数组必须事先定义固定的长度（元素个数），不能适应数据动态地增减的情况。

当数据增加时，可能超出原先定义的元素个数；当数据减少时，造成内存浪费；数组可以根据 下标直接存取。

（2）链表动态地进行存储分配，可以适应数据动态地增减的情况，且可以方便地插入、删除数 据项。

（数组中插入、删除数据项时，需要移动其它数据项，非常繁琐）链表必须根据next指针找到 下一个元素

从内存存储来看：

(1)(静态)数组从栈中分配空间, 对于程序员方便快速,但是自由度小

(2)链表从堆中分配空间, 自由度大但是申请管理比较麻烦

从上面的比较可以看出，如果需要快速访问数据，很少或不插入和删除元素，就应该用数组；

相反， 如果需要经常插入和删除元素就需要用链表数据结构了。

(2)链表的经典面试题：

http://blog.csdn.net/wu\_george/article/details/20793929

高级编程：

1.[设计模式之CS和BS结构的区别](http://blog.csdn.net/xiven/article/details/5279350)

<http://blog.csdn.net/xiven/article/details/5279350>

2.fork和vfork的异同

使用fork，子进程与父进程的执行先后顺序不确定，使用vfork是子进程先运行，后运行父进程，

Fork是子进程拷贝父进程空间，vfork则是子进程与父进程共享空间

两者都是被调用一次，返回两次，子进程的返回值是0，而父进程的返回值则是子进程的进程ID

1. [select、poll、epoll之间的区别](http://www.cnblogs.com/Anker/p/3265058.html)

<http://www.cnblogs.com/Anker/p/3265058.html>

4.画出七层网络参考模型

