笔试时间：90分钟

〇、准备工作

1. 请问您最熟悉的编程语言是什么？ \_\_\_\_\_\_\_\_\_\_\_
2. 除改错题外，其他所有涉及编程的题目，请使用您最熟悉的语言答题。

一、选择题 (每题4分，共5题，总计20分)

1. 以下进程之间共享数据的方式，错误的是：

A. 命名管道

B. 动态链接库

C. 剪贴板

D. 信号量

答案：B

2. 随着IP网络的发展，为了节省可分配的注册IP地址，有一些地址被拿出来用于私有IP地址，以下不属于私有IP地址范围的是：

A. 10.6.207.84

B. 100.64.0.100

C. 172.32.50.80

D. 192.168.1.100

答案：B

3. 以下关于快速排序，说法错误的是：

A. 稳定排序

B. 最佳时间复杂度O(nlogn)

C. 最坏时间复杂度为O(n\*2)

D. 数据移动少

答案：A

4. 在linux系统中进行下列操作，哪个操作必须要root权限：

A. make

B. ldconfig

C. sudo

D. kill

答案：B

5. 在区间[-2, 2]里任取两个实数，它们的和>1的概率是：

A. 3/8 B. 3/16 C. 9/32 D. 9/64

答案：C

二、改错题 (20分)

以下是运行在某服务器上的代码段，是为了实现抢红包的功能。代码运行时，通过实例化Hongbao对象，一旦用户抢到红包，就会调用Process方法，进入发放红包流程。虽然这段代码看上去没有逻辑问题，但是其中却隐藏了一些缺陷。请从中找出你觉得是缺陷的地方，并指出修改方法：

class Hongbao

{

static int num = 100; // 总共有100个红包

bool CheckUser(User u) {

... // 此处做一些用户条件判断，例如判断该用户是否抢过红包

}

bool Deliver(User u) {

... // 此处会发放红包给用户，如果发放失败返回false

}

void Process(User u) {

if (num != 0) { // 当红包数量不为0时，进入发红包流程

if (!CheckUser(u))

return; // 如果用户未通过检查则返回

num--; // 开始发红包，红包数量减1

Deliver(u); // 发放红包给用户

}

}

}

答案：

1）if(num!=0) 这个条件判断有并发风险

num是静态变量，也就是所有Hongbao对象的共享变量。如果有两个或者更多的线程同时进入Process()，假设此时num=1，则两个现场均可以通过条件判断进入逻辑，这样就会同时发放多个红包，即使num=1（红包数量为1）。更危险的是，在此之后，num=-1，则永远可以通过if(num!=0)的判断，红包将会发出无限多个。

修改方法：给num加锁

2）num-- 不应该在Deliver之前

由于Deliver可能失败或者出现Exception，所以num--应该放到Deliver成功之后。

三、简答题 (共20分)

1. 在Linux系统里，常见的进程间通讯的方法有哪些？并给每个通讯方法举一个例子，用以说明该方法的典型应用场景（本题10分）。

评分标准：每提到一个正确的进程间通讯术语1分（答错则反扣1分），场景描述正确再加1分。累加直到满分为止。

1. 信号量(semaphore)
2. 信号(signal)
3. 管道和命名管道(pipe & named pipe)
4. 消息队列(message queue)
5. 共享内存(share memory)
6. 文件(file)
7. 套接字(socket)
8. unix域(unix domain socket)

2. HTTP请求报文由哪些部分组成？举例说明各个部分至少3个典型数据（尽可能不要举雷同的例子，本题10分）。

答案：

一个HTTP请求报文由请求行（request line）、多个请求头部行（header）、空行和正文（request body）部分等4个部分组成。

请求行（request line）的典型数据：

1. GET /articles?page=2 HTTP/1.1
2. POST /query HTTP/1.1
3. DELETE /article?id=123 HTTP/1.1

头部行（header）的典型数据：

1. User-Agent: Mozilla/7.0
2. Host: [www.qiniu.com](http://www.qiniu.com)
3. Content-Type: application/x-www-form-urlencoded

正文（request body）的典型数据：

1. 无正文 (Content-Length=0)
2. Form格式：query=abc&page=2
3. Json格式：{“query”: “abc”, “page”: 2}

四、编程题 (共40分)

1. 实现一个函数，输入一个单向链表，输出该链表中倒数第k个结点，请尽可能减少链表的遍历次数（本题20分）。

评分标准：

1. 考察函数原型抽象的合理性（3分）。
2. 实现常规逻辑的正确性（2分）。
3. 实现逻辑的完备性（比如在链表元素不足k个的时候应该怎么办，2分）。
4. 实现的高效性（3分）。

分析：为了得到倒数第k个结点，很自然的想法是先走到链表的尾端，再从尾端回溯k步。可是输入的是单向链表，只有从前往后的指针而没有从后往前的指针。因此我们需要打开我们的思路。既然不能从尾结点开始遍历这个链表，我们还是把思路回到头结点上来。假设整个链表有n个结点，那么倒数第k个结点是从头结点开始的第n-k-1个结点（从0开始计数）。如果我们能够得到链表中结点的个数n，那我们只要从头结点开始往后走n-k-1步就可以了。如何得到结点数n？这个不难，只需要从头开始遍历链表，每经过一个结点，计数器加一就行了。这种思路的时间复杂度是O(n)，但需要遍历链表两次。第一次得到链表中结点个数n，第二次得到从头结点开始的第n­-k-1个结点即倒数第k个结点。如果链表的结点数不多，这是一种很好的方法。但如果输入的链表的结点个数很多，能不能把链表遍历的次数减少到1？一个方法是：如果我们在遍历时维持两个指针，第一个指针从链表的头指针开始遍历，在第k-1步之前，第二个指针保持不动；在第k-1步开始，第二个指针也开始从链表的头指针开始遍历。由于两个指针的距离保持在k-1，当第一个（走在前面的）指针到达链表的尾结点时，第二个指针（走在后面的）指针正好是倒数第k个结点。

2. 写一个函数，输入一个整数数组，在屏幕上打印该数组的全排列（本题20分）。例如：

输入为：1,2,3

输出为:

1,2,3

3,2,1

2,1,3

...

评分标准：

1. 考察函数原型抽象的合理性（1分）：func XXX(array []int)
2. 实现逻辑的正确性（9分）。

五、表达能力考察（共20分）

如何评价AlphaGo与李世石之间的世纪之战，你如何看待AI的未来？