目录

Chap1 测试基础

附件 1 的 19-21 页是一个模拟的面试场景，其中有不少经典题型和问题。不过已经在网上广为流传，建议了解其中面试思路为主，题目不要照搬太多。

1. *黑盒测试和白盒测试常用的测试方法有哪些？举例说明。*

答：白盒测试：逻辑覆盖法，主要包括语句覆盖，判断覆盖，条件覆盖，判断条件覆盖，条件组合覆盖、路径覆盖。

黑盒测试：等价划分类，边界值分析，错误推测法等

2. *静态测试和动态测试的概念。*

答：静态方法是指不运行被测程序本身，仅通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性。对需求规格说明书、软件设计说明书、源程序做结构分析、流程图分析、符号执行来找错。静态方法通过程序静态特性的分析，找出欠缺和可疑之处，例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用和可疑的计算等。静态测试结果可用于进一步的查错，并为测试用例选取提供指导。

动态方法是指通过运行被测程序 ,检查运行结果与预期结果的差异 ,并分析运行效率和健壮性等性能，这种方法由三部分组成：构造测试实例、执行程序、分析程序的输出结果。所谓软件的动态测试，就是通过运行软件来检验软件的动态行为和运行结果的正确性。目前 ,动态测试也是公司的测试工作的主要方式。根据动态测试在软件开发过程中所处的阶段和作用，动态测试可分为如下几个步骤：1、单元测试 2、集成测试 3、系统测试 4、验收测试 5、回归测试。

3. *等价类有几种，含义分别是什么？*

答：等价类分为以下几类：

- 有效等价类和无效等价类
- 有效等价类就是对程序的规格说明有意义的，合理的输入数据所构成的集合，利用有效等价类可验证程序是否实现了规格说明中的功能和性能。
- 无效等价类是那些对程序的规格说明不合理或者无意义的数据所构成的，为了验证程序做其不应作的事情。

4. *等价类划分的优缺点。（答出一些使用过程中的体会即可）*

答：优点：考虑了单个数据域的各类情况，避免盲目或随机的选取输入数据的不完整性和不稳定性，同时可有效控制测试设计的数量。

缺点：对组合情况考虑不足，同时等价类划分基于等价类中的输入都能产生相同的效果，在很多情况下用例选择不当会产生问题（如边界）。

5. *边界值测试方法的优缺点。*

答：长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

不过边界值分析法与等价类划分法一样，没有考虑输入之间的组合情况，因此需要进一步结合其他测试用例设计方法。

6. *等价类划分的原则(了解大概即可，关键看是否会使用)。*

答：等价类划分的原则如下：

- 在输入条件规定了取值范围或值的个数的情况下,则可以确立一个有效等价类和两个无效等价类.
- 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下,可确立一个有效等价类和一个无效等价类.
- 在输入条件是一个布尔量的情况下,可确定一个有效等价类和一个无效等价类.
- 在规定了输入数据的一组值（假定 n 个）,并且程序要对每一个输入值分别处理的情况下,可确立 n 个有效等价类和一个无效等价类.
- 在规定了输入数据必须遵守的规则的情况下,可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）.
- 在确知已划分的等价类中各元素在程序处理中的方式不同的情况下,则应再将该等价类进一步的划分为更小的等价类.

7. *性能测试：如何评价系统的极限性能？*

答： 基本点：并发度、响应时间、单位时间吞吐量、系统稳定性、多场景。
加分点：新旧版本对比，性能瓶颈分析方法（雪崩、线性拐点等）。

8. *判断测试活动中止的条件*

答：从以下几个角度分析，包括：无新发生 bug 且严重性高的老 bug 已修复；bug 收敛；某一级别 bug 低于一定比例；时间耗尽；满足特定覆盖率。另外，可以说说在以前的项目测试是如何结束的。

9. *常见测试模型？*

答:常见的软件测试模型包括 V 模型、W 模型、H 模型、X 模型和前置模型。([注] :具体解释太长了，见附件 1 的前几页。)

Chap2 测试设计

1. *配置文件测试设计*

题目：一个程序需要根据配置文件，将本地的多个文件（model.0, model.1, model.2…）分发到不同机房的不同机器上去。其中，配置文件格式如下：

#机房数量
SITE_NUM : 5
#第 0 个机房机器数量
SITE_0_HOST_NUM : 10
#该机房第 n 个机器的 ip
SITE_0_HOST_0 : 192.168.0.1
SITE_0_HOST_1 : 192.168.0.2
。。。
SITE_0_HOST_9 : 192.168.0.10

SITE_1_HOST_NUM : 10
SITE_1_HOST_0 : 192.168.1.1
SITE_1_HOST_1 : 192.168.1.2
。。。

#文件数量
MODEL_NUM : 5
#第 n 个文件在第 m 个机房需要的备份数
MODEL_0_REP_NUM : 0 : 3, 1 : 3,2:3, 3:3:4:3
MODEL_1_REP_NUM : 0 : 3, 1 : 3,2:3, 3:3:4:3
MODEL_2_REP_NUM : 0 : 3, 1 : 3,2:3, 3:3:4:3
MODEL_3_REP_NUM : 0 : 3, 1 : 3,2:3, 3:3:4:3
MODEL_4_REP_NUM : 0 : 3, 1 : 3,2:3, 3:3:4:3

分发要求：一台机器上不能布置多份相同的文件
每台机器上要求分发的文件数量尽量均匀
问题：请设计测试用例。

答：各种边界值；不同机器的 IP 重复；在某机房的需要的备份数超过了机器数；

2. *杯子的测试（校招）*

答：冒烟测试：速度装一杯水，是否漏水
功能测试：漏水测试，透明度测试，卫生情况测试，杯口平滑测试，重量测试，均匀度测试
压力测试：抗摔测试，抗高温测试
欢迎添加

3. *描述 bs 这类模块的功能，设计测试用例【标记】*

4. strstr 测试（可以扩展到其他函数测试，主要考察边界，基本情况，鲁棒性，性能等方面是否考虑全面，实习生 2 面）

答：基本情况；边界值；鲁棒性；性能以及其算法优化；

5. 请使用等价类划分的测试方法完成用例设计。

题目：设有一个档案管理系统 ,要求用户输入以年月表示的日期。假设日期限定在 1990 年 1 月~2049 年 12 月，并规定日期由 6 位数字字符组成，前 4 位表示年，后 2 位表示月。

问题：现用等价类划分法设计测试用例，来测试程序的"日期检查功能"。

答：

- 划分等价类并编号,下表等价类划分的结果

输入等价类	有效等价类	无效等价类
日期的类型及长度	6 位数字字符	有非数字字符
		少于 6 位数字字符
		多于 6 位数字字符
年份范围	在 1990~2049 之间	小于 1990
		大于 2049
月份范围	在 01~12 之间	等于 00
		大于 12

- 设计测试用例，以便覆盖所有的有效等价类在表中列出了 3 个有效等价类，编号分别为①、⑤、⑧，设计的测试用例如下：

测试数据　期望结果　覆盖的有效等价类
200211　输入有效　①、⑤、⑧

为每一个无效等价类设计一个测试用例，设计结果如下：

测试数据　期望结果　覆盖的无效等价类
95June　无效输入　②
20036　无效输入　③
2001006　无效输入　④
198912　无效输入　⑥
200401　无效输入　⑦
200100　无效输入　⑨
200113　无效输入　⑩

6. CP 命令设计测试用例（5 分钟）

答：主要从异常、功能和性能三方面考虑：

- 异常

  参数异常：源和目标参数异常：包含特殊字符；参数超长；指定的位置实际不存在

  拷贝对象异常：非法的执行权限；存储介质有损坏；非法的文件格式和内容

  执行过程异常：拷贝到一半断电；拷贝过程中硬盘满；拷贝过程中源或目的被删除
- 功能
  - 文件

    不同文件大小：0，1k，10k。。。

    不同的文件类型：文本，二进制，设备文件。。。
  - 目录

包含各种文件类型
包含子目录，目录深度
目录文件数量很多
针对文件和目录分别验证拷贝的准确性，完整性。

- 性能
  - 场景：
    拷贝大文件
    拷贝目录中存在大量小文件
    跨文件系统间拷贝
    跨存储介质间拷贝（硬盘到 U 盘。。。）
    构造源的各种磁盘分布（磁盘扇区分布。。。）
    并发执行拷贝
  - 关注的性能点：拷贝时间, CPU，内存，磁盘 IO

7. 如何测试模板（10 分钟）

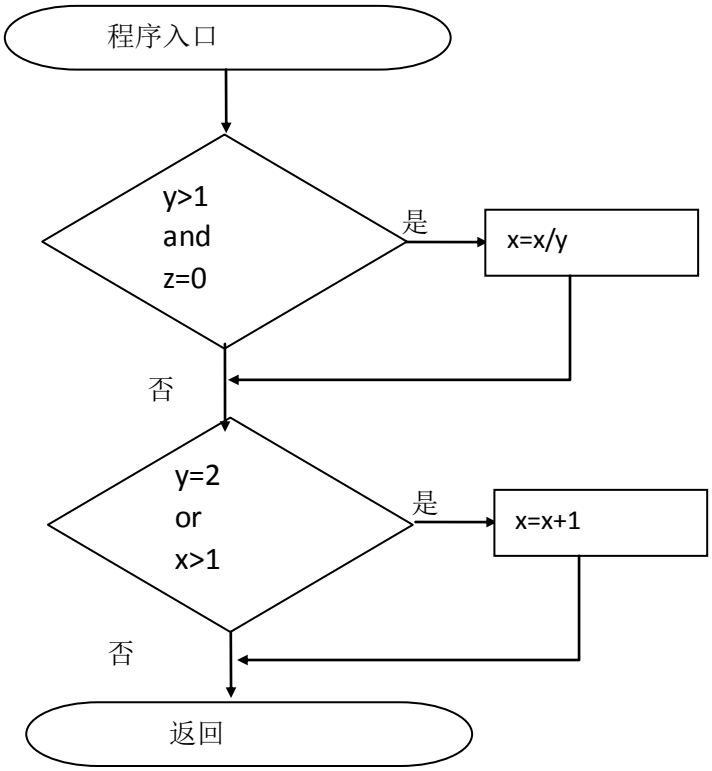
题目：百度首页是由模板展现，请问如何对它进行测试；
要求：不需要考虑性能相关因素。建议多从用户行为和使用环境角度进行测试
（考察点：测试能力+思维系统性+思维发散性）

答：

展现检查：文字图片内容，链接，一致性
数据传递：字符串长短与截断；特殊字符；中英文；空格；下拉条提示
兼容性检查：不同浏览器，不同操作系统，不同分辨率
用户行为：窗口拉大，放小；字号大小；编码格式；刷新；前进后退

8. 白盒测试对子程序进行测试（15 分钟）

题目：本流程图描述了某子程序的处理流程，现要求用白盒测试法对子程序进行测试。
要求：根据白盒测试常用的以下几种方式：语句覆盖、判定覆盖、条件覆盖、判定 / 条件覆盖、多重条件覆盖(条件组合覆盖)、路径覆盖六种覆盖标准，从供选择的答案中分别找出满足相应覆盖标准的最小的测试数据组并简述各种测试方法。



供选择的答案

x=3 y=3 z=0;x=1 y=2 z=1

x=1 y=2 z=0;x=2 y=1 z=1

x=4 y=2 z=0; x=3 y=3 z=0; x=2 y=1 z=0; x=1 y=1 z=1

x=4 y=2 z=0; x=1 y=2 z=1; x=2 y=1 z=0; x=1 y=1 z=1

x=4 y=2 z=0

x=4 y=2 z=0;x=1 y=1 z=1

参考答案：

- 语句覆盖 E：语句覆盖是指选择足够的测试用例，使得运行这些测试用例时，被测程序的每个语句至少被执行一次，语句覆盖是一种比较弱的覆盖标准
- 判定覆盖 A：也称分支覆盖，是指选择足够的测试用例，使得运行这些测试用例时，被测程序的每个判定的所有可能结果至少出现一次
- 条件覆盖 B：是指选择足够的测试用例，使得运行这些测试用例时，判定中的每个条件的所有可能结果至少出现一次
- 判定/条件覆盖 F：是指选择足够的测试用例，使得运行这些测试用例时，判定中每个条件的所有可能结果至出现一次，并且每个判定本身的所有可能结果也至少出现一次
- 多重条件覆盖 D：是指选择足够的测试用例，使得运行这些测试用例时，每个判定中条件结果的所有可能组合至少出现一次
- 路径覆盖 C：是指选择足够的测试用例，使得运行这些测试用例时，程序的每条可能执行到的路径都至少经过一次

9. *Baidu hi 聊天消息收发的测试思路（10 分钟）*

问题：请给出 BAIDU hi 聊天消息收发的测试思路？（10 分钟）

（考察点：基本测试思路）

参考答案：主要从以下几个方面来考察：正常测试、异常测试、不同的消息类型、组合测试、长度极值、是否延迟、是否丢失、是否被篡改、安全性

10. *登录界面测试（10 分钟）*

参考答案：希望可以对测试点做分类划分，如功能、UI、性能、安全

11. *测试自动贩卖机 （20 分钟）*

题目: 测试自动贩卖机，场景：贩卖机将用在露天的繁华的大街上

（考察点: 主要考察逻辑思维、思维的发散性）

参考答案:

大概可以从以下几个方面来考虑：

- 考虑到管理员的功能：如添加货物功能、定价等功能
- 考虑到界面外观、用户说明的
- 容错考虑比较多的

12. *三角形测试（20 分钟）*

题目：一个程序，从输入框中读取三个整数值，这三个数值代表了三角形三边的长度。程序显示提示信息，指出该三角形究竟是不规则三角形、等腰三角形还是等边三角形。（注：不规则三角形指三角形中任意两边不相等，等腰三角形指有两条边相等，等边三角形指三条边相等）

要求：假设你将作为一名测试工程师对该程序进行测试，你会采用什么方法进行测试用例设计？请写出你采用的方法、测试用例设计的过程以及最后的测试用例。（30 分钟）

(考查点：考察测试思维的严谨性，答全难)

参考答案：可以采用等价类划分的方法进行测试用例的设计。

- 等价类表：

输入条件	有效等价类	无效等价类
是否三角形的三条边	(1)A>0	(7)A<=0
	(2)B>0	(8)B<=0
	(3)C>0	(9)C<=0
	(4)A+B>C	(10)A+B<=C
	(5)B+C>A	(11)B+C<=A
	(6)A+C>B	(12)A+C<=B
是否等腰三角形	(13)A=B	(16) A!=B&&B!=C&&C!=A
	(14)B=C	
	(15)C=A	

是否等边三角形	⑰ A=B&&B=C&&C=A	⑱A!=B
		⑲B!=C
		⑳C!=A

- 测试用例：

序号	[A,B,C]	覆盖等价类	输出
1	[3,4,5]	⑴⑵⑶⑷⑸⑹	一般三角形
2	[0,1,2]	⑺	不能构成三角形
3	[1,0,2]	⑻	
4	[1,2,0]	⑼	
5	[1,2,3]	⑽	
6	[1,3,2]	⑾	
7	[3,1,2]	⑿	
8	[3,3,4]	⑴⑵⑶⑷⑸⑹⒀	等腰三角形
9	[3,4,4]	⑴⑵⑶⑷⑸⑹⒁	
10	[3,4,3]	⑴⑵⑶⑷⑸⑹⒂	
11	[3,4,5]	⑴⑵⑶⑷⑸⑹⒃	非等腰三角形
12	[3,3,3]	⑴⑵⑶⑷⑸⑹⒄	等边三角形
13	[3,4,4]	⑴⑵⑶⑷⑸⑹⑭⒅	非等边三角形
14	[3,4,3]	⑴⑵⑶⑷⑸⑹⑮⒆	
15	[3,3,4]	⑶⑷⑸⑹⒀⒇	

13. 较复杂功能程序设计用例（30 分钟）

程序从标准输入中读取，判断输入字符是固定电话号码或者手机号码

a) 手机号码：以 13 开头，长度为 11 的连续数字

b) 固定电话号码：固定电话号码包括区号和号码两部分，其中号码为长度为 7 或 8，并且不以 0 开头的连续数字。区号可有可不有。区号和号码间可有“-”，也可以没有。

c) 当用户输入完毕后，系统返回的答案包括： 手机号码 固定号码 无正确电话号码

d) 一次输入中如果有多个正确号码（空格为分割符），以最后一个正确号码的类型为准

对实现上述功能的程序设计测试用例。（40 分钟）

区号范围（x 表示任意数字）：

3 位区号	4 位区号
10	03××
20	04××
21	05××
22	06××
23	07××
24	08××
25	09××
27	
28	
29	

参考答案： 以下是测试设计的参考思路：

<table>
<tr><td rowspan="12">有空格</td><td rowspan="3">空格在两头</td><td>中间有正确电话号码</td><td></td><td></td><td></td></tr>
<tr><td>中间无正确电话号码</td><td></td><td></td><td></td></tr>
<tr><td colspan="2">中间无字符</td><td></td><td></td></tr>
<tr><td rowspan="5">空格在中间</td><td rowspan="3">一个空格</td><td>最后一个为错误手机号</td><td></td><td></td></tr>
<tr><td>全部为错误手机号</td><td></td><td></td></tr>
<tr><td>最后一个为正确手机号</td><td></td><td></td></tr>
<tr><td rowspan="3">多个空格</td><td>最后一个为正确手机号</td><td></td><td></td></tr>
<tr><td>第一个为正确手机号</td><td></td><td></td></tr>
<tr><td>全部为错误手机号</td><td></td><td></td></tr>
<tr><td colspan="3">超长字符含空格</td><td></td><td></td></tr>
<tr><td rowspan="11">无空格</td><td rowspan="4">只含数字</td><td rowspan="2">以 0 开头</td><td colspan="2">前三位是区号</td><td>区号识别</td></tr>
<tr><td rowspan="2">前三位非区号</td><td>前四位为区号</td><td>区号识别</td></tr>
<tr><td rowspan="2">非 0 开头</td><td>前四位非区号</td><td></td></tr>
<tr><td rowspan="3">13 开头</td><td colspan="2">长度为 11</td></tr>
<tr><td rowspan="2">长度非 11</td><td>座机号码</td></tr>
<tr><td colspan="2">非座机号码</td></tr>
<tr><td rowspan="3">含数字和 -</td><td colspan="2">非 13 开头</td><td>座机号码识别</td></tr>
<tr><td rowspan="3">一个 -</td><td rowspan="2">- 前是区号</td><td colspan="2">- 后为座机</td></tr>
<tr><td colspan="2">- 后非座机</td></tr>
<tr><td>- 前非区号</td><td colspan="2">- 后为座机</td></tr>
<tr><td colspan="4">多个 -</td></tr>
<tr><td colspan="5">含其他字符</td></tr>
<tr><td colspan="5">超长字符不含空格</td></tr>
<tr><td colspan="6">空输入</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>座机识别</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">首位为 0</td><td>长度为 6</td><td colspan="2">所有数字均能被识别？？</td><td></td><td></td></tr>
<tr><td>长度为 7</td><td></td><td></td><td></td><td></td></tr>
<tr><td>长度为 5</td><td></td><td></td><td></td><td></td></tr>
<tr><td>长度为 8</td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">首位非 0</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

区号识别					
三位长度	010 开头				
	02 开头	026 开头			
		非 026			
	非 010、02 开头	011 开头			
		030 开头			
四位长度	03 - 09 开头				
	01				
	02				

Chap3 算法

1. 请列举的常用排序算法，并说明其时间复杂度，并说明排序思想。

答：

- 冒泡排序：两两比较待排序数据元素的大小，发现两个数据元素的次序相反时即进行交换，直到没有反序的数据元素为止。算法时间复杂度是 O(n^2)。
- 选择排序：每一趟从待排序的数据元素中选出最小（或最大）的一个元素，顺序放在已排好序的数列的最后，直到全部待排序的数据元素排完，算法复杂度是 O(n^2)。
- 插入排序：每次将一个待排序的数据元素，插入到前面已经排好序的数列中的适当位置，使数列依然有序；直到待排序数据元素全部插入完为止。算法时间复杂度是 O(n^2)
- 快速排序：在当前无序区 R[1..H]中任取一个数据元素作为比较的"基准"(不妨记为 X)，用此基准将当前无序区划分为左右两个较小的无序区：R[1..I-1]和 R[I+1..H]，且左边的无序子区中数据元素均小于等于基准元素，右边的无序子区中数据元素均大于等于基准元素，而基准 X 则位于最终排序的位置上，即 R[1..I-1]≤X.Key≤R[I+1..H](1≤I≤H) ,当 R[1..I-1]和 R[I+1..H]均非空时 ,分别对它们进行上述的划分过程，直至所有无序子区中的数据元素均已排序为止。不稳定，最理想情况算法时间复杂度 O(nlog2n)，最坏 O(n^2)。
- 堆排序：堆排序是一树形选择排序，在排序过程中，将 R[1..N]看成是一颗完全二叉树的顺序存储结构，利用完全二叉树中双亲结点和孩子结点之间的内在关系来选择最小的元素。 算法时间复杂度 O(nlogn)。
- 希尔排序：其实就是用步长控制的插入排序，希尔排序通过加大插入排序中元素之间的间隔，并在这些有间隔的元素中进行插入排序，从而让数据项可以大幅度移动，这样的方式可以使每次移动之后的数据离他们在最终序列中的位置相差不大，保证数据的基本有序，大大提升了排序速度，运算时间复杂度 N*logN。
- 归并排序：

  Divide: 把长度为 n 的输入序列分成两个长度为 n/2 的子序列。

  Conquer: 对这两个子序列分别采用归并排序。

  Combine: 将两个排序好的子序列合并成一个最终的排序序列。

  时间复杂度是 O(nlog2n)。

2. 快速排序的平均时间复杂度是多少？最坏时间复杂度是多少？在哪些情况下会遇到最坏的时间复杂度。

答：

快速排序的平均时间复杂度 O(nlog2n)，最坏时间复杂度 O(n^2)。

最坏情况：当每次 pivot 选择恰好都把列表元素分成了(1，n-1)

采取措施：pivot 的选取是通过 random 来进行

3. 各个排序算法的稳定性，并给出理由。

答：

选择排序、快速排序、希尔排序、堆排序不是稳定的排序算法，而冒泡排序、插入排序、归并排序和基数排序是稳定的排序算法。

参考：http://hi.baidu.com/cuifenghui/blog/item/0587932b039557f9e7cd4051.html

4. *两个单项链表求交点。*

单向链表有交点意思是交点后的节点都是一样的；因此，如果两个单向链表相交，是成 Y 字形的。思路：求出第一个链表长 m，第二个长 n。假设 m>=n，那么就去掉第一个链表的前 m-n 个元素，使之等长，然后依次比较第一个、第二个、第三个元素，直到找到或者结束。

```
NODE* FindNode(NODE*  pHead1, NODE* pHead2)
{
      NODE* p1 = pHead1;
      NODE* p2 = pHead2;
      int i = 1, j = 1, k = 0, f = 0;
      if(pHead2 == NULL || pHead2 == NULL)
      {
            return NULL;
      }
      while(p1->next  != NULL)
      {
            p1 = p1->next;
            i++;
      }
      while(p2->next  != NULL)
      {
            p2 = p2->next;
            j++;
      }
      if(p1 != p2)
      {
            return NULL;            //如果尾节点不同，直接返回  NULL
      }
      else                          //否则寻找第一个相同的节点
      {
            p1 = pHead1;                        // 1
            p2 = pHead2;                        // 2
            f = fabs(i, j);     //计算两条链表长度的差
            if(i > j)               //如果第一个链表比第二个长，第一个链表先向前移动 f 步
            {
                  for(k=0; k<f; k++)
                  {
                      p1 = p1->next;
                  }
                  while(p1  != p2)
                  {
                        p1 = p1->next;
                        p2 = p2->next;
                  }
                  return p1;
```

```
            }
            else
            {
                for(k=0; k<f; k++)
                {
                    p2 = p2->next;
                }
                while(p1 != p2)
                {
                    p1 = p1->next;
                    p2 = p2->next;
                }
                return p1;
            }
        }
}
```

*5. 递增数列中每一项都可以表示为 3^i*5^j*7^k（0<=i,j,k），即 1,3,5,7,9,15,21,25,27…，实现算法，求出该数列中的第 n 项*

答：

先放置几个队列：

L1: 3^i (i >=0)

L2: 3^i * 5^j (j >= 1)

L3: 3^i * 5^j * 7^k (k >= 1)

Step1: 清空三个队列、分别把 3,5,7 放入三个队列的首位。准备一个新队列 L（目前为空）。

Step2: 比较三个队列的头，找出最小的那个。把这个元素从队列出队，并加到 L 的尾部。

Step3:

如果 Step2 中的元素是从 L1 中获取的，假设是 3^m，则在 L1 的尾部加入 3^(m+1)，L2 的尾部加入 3^m*5，L3 的尾部加入 3^m*7。

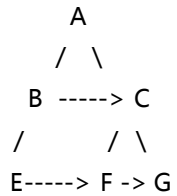
如果 Step2 中的元素是从 L2 中获取的 ,假设是 3^m * 5^n ,则在 L2 的尾部加入 3^m * 5^(n+1)，L3 的尾部加入 3^m * 5^n *7。

如果 Step3 中的元素是从 L3 中获取的 ,假设是 3^m * 5^n * 7^p ,则在 L3 的尾部加入 3^m * 5^n * 7^(p+1)。

Step4: L 的长度到达 N 了吗？如果没到达，重复 L2-L4。如果到达，则进行 L5。

Step5: 取得 L 的末尾元素，即为所求。

6. 为二叉树中每一个节点建立他的右邻居节点（可以不属于同一个父节点），要求递归与非递归都实现，如



答：

- 递归：

```
buildRightSibling(Linkedlist  list)
{
   Linkedlist  nextlevel = new Linkedlist();
   把 list 中的每个元素的子节点，放入 nextlevel 中。
   //对 list 中每个元素，设置右邻结点，指向 list 中的下一个元素
   //这一步可以在链表中就实现掉。
   If(nextlevel == null || nextlevel.length() ==0)
      Return;
   Else
      buildRightSibling(nextlevel);
}
//启动函数
buildRightSibling(new  Linkedlist(root));
```

- 非递归

层序遍历每一层，对每一个元素都打上标记。比如(1,1)代表第 1 层第 1 个元素。
所有元素都打上标记以后，对每个元素(m,n)都寻找(m,n+1)元素

7. 走台阶问题，一次可以走 1，2，3 级，都 N 级台阶的方法数。

答：

初始：f(0) = 0; f(1) =1; f(2) = 1 + 1 = 2;
递推公式：f(n) = f(n - 1) + f(n-2) + f(n - 3)

8. 10 进制数转 2 进制数，

题目：2 进制除了 0，1，还可以用 2 表示。例如：

1-> 1
2-> 10 or 02
3->11
4 ->100 or 020 or 012

问题：这样一个十进制数转为二进制数，就不是唯一的了。现求十进制数 N 转换为这种二进制数的所有表示方法数。

答：

f(0)=1, f(1)=1, f(2)=2,
f(n) = f((n-1)/2) 当 n 为奇数
= f(n/2)+f((n-2)/2)当 n 为偶数

9. 一个环状链表（收尾相连），两个指针 head1 和 head2 从同一个节点出发，head1 每次走一步，head2 每次走两步，请证明，两个指针第一次相遇于出发的节点

答：

设两个指针走的次数为 x，使用简单的数学公式即可证明。难度 1 面。考察基本的数学　　知识。
设链表有 m 个元素，head1 在第一次相遇时走了 n 步，c 为 head1 和 head2 第一次相遇的节点距离出发节点的距离。

则： head1 走过的路程为 c = n;

head2 走过的路程为 c + k *m = 2n; （k 为整数）

因此，c = k*m，即 c 恰好是链表长度的整数倍，即两个指针第一次相遇一定是在出发的节点。

10. 一个链表中含有环。请找出环的起始节点

答：

方法 1： 使用标记法，走过的节点标记 1。使用这种方法，需要时间/空间复杂度 O(n)

方法 2： 使用第 6 题的思路，设两个指针走的次数为 x 即可得到，时间复杂度不变，但空间 复杂度为 O(1)。难度 1 面，考察能否使用已经掌握的知识（第一题）来求解。

让两个指针 head1 和 head2 从同一个节点出发，head1 每次走一步，head2 每次走两步，当二者重合时，让 head2 回到链表的头部，以每次一步的步长走，当 head1 和 head2 再次相遇时，就是环路的起始节点了。

11. 给定 N 个数，其中有一个数的出现次数超过 N/2，请找出这个数，O（n）算法

（考察点：该数出现次数超过其他所有数出现次数总和，使用计数方式解答。难度：2 面）

答：

解法 1： 开一个新的列

比较第 1,2 个数字。如果相同，则放入新的列中 如果不同，则两个都丢弃

然后比较第 3,4 个数字，5,6 个数字

这个时候 新的列 最长为 n/2 （实际上会远远更短）

然后对新的列 如法炮制 再次缩短一半

当某个时刻 列的长度是 1 或者列突然消失时候 结束

长度为 1 说明这个就是的。消失，说明不存在大于 n/2 的个数的数

解法 2：

构造一个 hashtable，其中 key 是这个 N 个数的值，而 value 则是他们出现的次数；

最后遍历这个 hashtable，找出最大的即可。[ZhaiYao: 这个方法应该不好。不应该使用 hashtable]

12. 最长连续子序列之和（和最接近 0 的子序列），环形数组的最大子序列和。

答：

环形的拼成 2n 的数组后求解。

最长连续子序列之和：扫描数组，从左到右记录当前子序列的和 ThisSum，若这个和不断增加，那么最大子序列的和 MaxSum 也不断增加(不断更新 MaxSum)。如果往前扫描中遇到负数，那么当前子序列的和将会减小。此时 ThisSum 将会小于 MaxSum ,当然 MaxSum 也就不更新。如果 ThisSum 降到 0 时，说明前面已经扫描的那一段就可以抛弃了，这时将 ThisSum 置为 0。然后，ThisSum 将从后面开始将这个子段进行分析，若有比当前 MaxSum 大的子段，继续更新 MaxSum。这样一趟扫描结果也就出来了。

和最接近 0 的子序列：【标记】

环形数组的最大子序列之和：将环形数组首尾拼接成一个 2n 的线型数组（如环形数组 0 1 2， 可以拼接成 0 1 2 0 1 2），按 1)的方法可以找到最大连续子序列之和

13. 字符串按字母 a-z 排序

题目要求：

（1）不是用排序库函数；

（2）代码实现；

（3）算法复杂度估算；

（4）测试自己的代码；

（5）能否用另一种排序方式，比较优缺点；（plus）

答：

```
1.  char *sort(char *a){
        int i, j;
        char tmp;
        for( i = 0; a[i] != '\0'; i++){
            for( j = i + 1; a[j] != '\0'; j++){
                if(a[i] > a[j]){
                    tmp = a[i];
                    a[i] = a[j];
                    a[j] = tmp;
                }
            }
        }
```

2. 算法复杂度：输入为一个 n 个字符的字符串，则复杂度为 $O(n^2)$

3. 另一种排序算法：快排，复杂度 O(nlgn)

[ZHAIYAO:可不可以用另一种方式？开一个 26 长的数组，记录每个字母出现的次数。然后根据这个记录，重新打印（构造）出来排序后的字符串]

14. 已知一个乱序的整数数组，求该数组排序相邻两数的最大间隔，要求时间复杂度为 O(n)。

例如：给定数组为 10 23 7 1 35 27 50 41

排序后的结果为：1 7 10 23 27 35 41 50

相邻两数的最大间隔为 13（10-23）。

遍历找到数列找到最大最小值(max,min)，则所求的 gap>=(max-min)/n，以(max-min)/n 为步长建立 k 个桶，每个桶记录落入桶内的最大最小值，顺序比较各个桶之间的最大 gap。

答：

用基于桶排序的方式，具体如下：

先找到最小和最大，分别记为 min 和 max，并设 avg=（max-min）/n

按照 avg 的大小将[min，max]分配(N-1)个桶

将数组中的数存入桶中

然后按顺序对每个相邻桶（跳过没有数的桶）进行比较，如相邻桶（a, b）（c, d）的距离 D=c-b

最终比较 D 的最大值，则为所求

15. 求两个相同大小已排序数组的中位数

题目：设 a[0..n-1]和 b[0..n-1]是两个已经排好序的数组，设计一个算法，找出 a 和 b 的 2n 个数的中位数。要求给出算法复杂度（O(lgn)）。

答：

设 a[0..n-1]的中位数是 m1，b[0..n-1]的中位数为 m2

如果 m1=m2，则 m1 则为 2n 个数的中位数

如果 m1>m2，则 2n 个数的中位数一定在 a[0..2/n]和 b[n/2..n]，在求这两个子数组的中位数

如果 m1<m2，则 2n 个数的中位数一定在 a[n/2..n]和 b[0..2/n]，在求这两个子数组的中位数

16. 已知一个数组 a1, a2, ..., an, b1, b2, ..., bn, 设计一个算法把数组变成 a1, b1, a2, b2, ..., an, bn。

答：

如果 n = 2k,　　则 C(k) = b_k

如果 n = 2k+1,　则 C(k) = a_k

ZhaiYao: 综合考虑时间和空间。时间 O(n)应该没办法降低了，觉得空间可以达到是 O(1)。但是没想出来算法，【标记】

17. 全排序算法

全排序算法就是列举一些字符的所有排列顺序。

```
void Perm(char list[], int k, int m)
{ //生成 list [k：m ]的所有排列方式
      int i;
      if (k == m) {//输出一个排列方式
            for (i = 0; i <= m; i++)
                  putchar(list[i]);
            putchar('\n');
      }
      else // list[k：m ]有多个排列方式
            //  递归地产生这些排列方式
            for (i=k; i <= m; i++) {
                  Swap (&list[k], &list[i]);
                  Perm (list, k+1, m);
                  Swap (&list [k], &list [i]);
            }
}
```

Chap4 设计题[标记]

1. *多线程锁机制设计*

在多线程编程中 ,对临界资源 ,经常需要 lock(),unlock()这样的操作 ,但是经常在 lock 之后忘记 unlock，造成多线程问题。现在用 C++类的思想实现一个 scopelock 类（校招）

例如：

```
{
   lock()
   ……
   unlock()
}
```

这种使用模式变成

```
{
   Scopelock()
   …
}
```

(在构造函数和虚构函数中实现 lock 和 unlock)

2. *设计 Spider*

题目：Spider 抓取能力一天 10w 个 url，互联网每天新增 1000w 个 url （社招）

这 1000w 个都是新增的 url，spider 怎么选取 10w 个进行抓取，选优的准则

这 1000w 个有些是抓取过的，存在历史抓取信息，spider 怎么选取 10w 个进行抓取，选优的准则。

3. *海量 url 除重（spilt&merge 的思想，主要看候选人的思路，2 面）*

答：

在获取 URL 的时候按照一定数量分组，然后分别在每组用 URL 的 sign 作为 key，用 awk 以 key 为标来构造数组；最后把每组过滤过的结果在 merge 起来过滤。

4. *给定一个 file，包含各类 url，统计出现次数 topN 的 url*

答： sort|uniq -c|sort -nr|head

5. *多路数组交集*

题目：有十路数组，数组元素组成为：{int id, unsigned int weight}，每路数组内 id、weight 均为无序

要求：如何求出这十路数组的交集（交集：id 相同），并将交集内的元素按照 weight 排序？给出思路和复杂度；如果原始的十路数组中的 id 均为有序的，又该如何做？给出思路和复杂度；

6. *兄弟单词索引*

题目 :dog 和 god 这类字母出现的次数和种类都完全一样的字符串称为兄弟单词 ,现在有海量的字符串，设计检索系统，支持查找用户输入字符串的所有兄弟单词。

7. *死链监测设计*

题目：互联网上每天有大量的网页成为死链，如何用最小的代价降低搜索引擎的死链率。
（考察其工程思维能力，包括相关死链反馈、用户点击检查、利用插件数据、站点稳定性等等）

8. *用户经常会输错 query，如何纠错*

参考答案：发散问题，常见输入法错误、缺字多字编辑距离判断，用户 session 挖掘等等

Chap5 逻辑题

1. 一天，harlan 的店里来了一位顾客，挑了 25 元的货，顾客拿出 100 元，harlan 没零钱找不开，就到隔壁飞白的店里把这 100 元换成零钱，回来给 顾客找了 75 元零钱。过一会，飞白来找 harlan，说刚才的是假钱，harlan 马上给飞白换了张真钱，问 harlan 赔了多少钱（低）？

答：

方法 A：

- 列出每笔交易的收入和支出，给分
- 答出结果，陪 100 元，给分

方法 B：

- 直接从结果出发，如果没有假钱，不赚不亏，给分
- 由于收入 100 假钱，赔 100，给分

2. 以 5 只猫 5 分钟捉 5 只老鼠的速度计算，要在 100 分钟内捉 100 只老鼠，需要多少只猫（低）

答：5 只猫

方法 A：

- 算出每只猫每分钟的捉鼠能力，给分
- 算出答案，给分

方法 B：

- 把 5 只猫看为整体，整体每分钟的捉鼠能力，给分
- 算出答案，给分

3. 一副扑克牌 54 张，红黑各一半，从里面任意翻两张，一红一黑的可能性是多少（低）

答：27/53

4. 一只手表 100 元买进，110 元卖出；120 元又买进，130 元再卖出，问共赚了多少钱（低）

答：赚 20 元

5. 有一牧场，已知养牛 27 头，6 天把草吃尽；养牛 23 头，9 天把草吃尽。如果养牛 21 头，那么几天能把牧场上的草吃尽呢？并且牧场上的草是不断生长的（中）

答：12 天

1. 列出 2 个已知方程和一个求解方程，给分
2. 算出答案，给分

假设牧场的草为 1 单位，每天生长 x 单位，每个牛每天吃 y 单位

27y*6-6x=1

23y*9-9x=1

解得：y=1/72, x=15/72　　[得一半分]

21 头牛，就是 1/(21y-x)=12 天　　[得满分]

6. 4 个人进入餐厅前都把自己的帽子交给寄存部的小姐保存。当他们一起离开时粗心的小

姐把那 4 顶帽子随便的戴在每个人的头上。发完帽子以后大家发现没有一个人 戴的是自己的帽子。请问这种情况发生的概率是多少？如果是 n 个人呢(n>1) （中）

答：

【4 人时候，答案是 9/24。9 种可能分别是 2143,4123,3142,3412,4312,2413,4321,3421,2341）】

【n 人时候，解法如下】

解法 1：

n 人时候，运用容斥原理的高级形式求解。【想到容斥原理就给出一半分数】

lixin：这是全错位排列问题 http://baike.baidu.com/view/1926671.htm?fr=ala0_1

zhaiyao:本科的离散数学课程，本帽子题目是一个例题。我做助教的时候背熟了。。。

设人站成一排，第 1 个的帽子设为帽子 1,第 i 的人的帽子设为帽子 i。

设帽子 i 戴在人 i 的头上（其他人无所谓）的情况数为 Ai

那么很明显有|Ai| =(n-1)!；

i 和 j 同时带对（不考虑其他人对不对）情况是 |Ai ∩Aj| = (n-2)!；

i 和 j，k 同时带对（不考虑其他人对不对）|Ai ∩Aj ∩Ak | = (n-3)!

问题问没有一个人带对帽子，问题的反面就是问“存在至少一个人戴对帽子”

“存在至少一个人戴对帽子”的情况数为：|A1∪A2∪…∪An|

根据容斥原理，

|A1∪A2∪…∪An| =(|A1|+|A2|+…+|An|) - (| A1 ∩A2| + | A1 ∩A3|) + (| A1 ∩A2 ∩A3| +) - …. (参见容斥原理)

= n*(n-1)! - C(n,2)*(n-2)! + C(n,3)*(n-3)! - … + (-1)^(n-1)*C(n,n)*0!

=n!(1/1!-1/2!+…+(-1)^(n-1)*1/n!)

所以都带错的情况是：

n! - n!(1/1!-1/2! + …+(-1)^(n-1)*1/n!)

= n! *(1 - 1/1!+1/2!+…+(-1)^n* 1/n!)

概率是 1 - 1/1!+1/2!-…+(-1)^(n-1)*1/n!

= 1/2!-…+(-1)^(n-1)*1/n! 【得到答案得到全部分数】

形象点就是：

2 个人: 1/2!

3 个人:1/2! – 1/3!

4 个人: 1/2! – 1/3! + 1/4! （不是 4/24,而是 9/24。）

5 个人: 1/2! – 1/3! + 1/4! – 1/5!

6 个人: 1/2! – 1/3! + 1/4! – 1/5! + 1/6!

。。。

解法 2：

人肉写出 n=2,3,4 的情况。然后进行数列规律解析，猜测出来最终表达式 1/2!-...+(-1)^(n-1)*1/n!。

【如果只能列出 n=2,3,4 等情况，建议不给分，因为这个思路很容易想。

但是能够找到规律并猜测出表达式，建议给三分之二或者四分之三的分数。因为这个数列规律很不好找。但是这种思路一定不给满分，因为没有严格的推理过程】

7. 甲、乙、丙每人出两元共六元买了一个收音机，然后就一起离开了商店，和他们一起去的丁因有事没有离开，这时售货员发现收音机的售价应该是五元，就把那一元 钱退给了丁，丁回去时坐车花了 4 角，然后把剩下的 6 角分给了甲、乙、丙三人各 2 角，这时算帐甲、乙、丙每人花了 1.8 元，加上丁坐车花的 4 角，是 5.8 元，那剩下的两角钱哪儿去了（中）

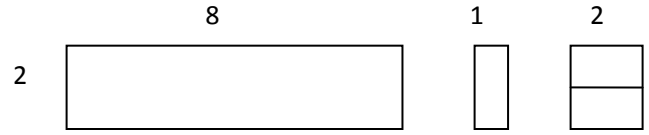
答：

甲乙丙每人花了 1.8 元，但丁坐车没有花 4 角（丁自己没有钱），甲乙丙花的钱里面，已经包含了丁的钱

这题无法根据中间过程给分，不推荐

【ZhaiYao:这题有点混淆概念，可以看出来一个人在混乱时候脑子能不能反应过来，分析清楚各种关系。解释清楚就得满分，否则不得分】

8. 长 8 高 2 的矩形中，填入后两种图形，一共有几种填法（中）



答：34

1. 根据第 2 块图形填入的个数分类讨论（或其他方式分类讨论），给分
2. 回答对答案，给分

[追问：对 2*n 的情况，有多少种？递推关系：n=1 或者 2,有 1 种；否则，f(n)=f(n-1)+ f(n-2)，斐波拉契数列]

9. 村子里有 50 个人，每人有一条狗。在这 50 条狗中有病狗(这种病不会传染)。于是人们就要找出病狗。每个人可以观察其他的 49 条狗，以判断它们是否生病， 只有自己的狗不能看。观察后得到的结果不得交流，也不能通知病狗的主人。主人一旦推算出自己家的是病狗就要枪毙自己的狗，而且每个人只有权利枪毙自己的 狗，没有权利打死其他人的狗。第一天，第二天都没有枪响。到了第三天传来一阵枪声，问有几条病狗，如何推算得出（中）

答：3 条

1. 从 1 条病狗的情况开始分析，给三分之一的分
2. 推广到 3 条病狗，给满分（需要解释，直接说 3 条，却解释不清楚的，不得分）

[追问：如果第 x 天传来一阵枪声，那么问一共有几条病狗？这天响了多少枪？]

由于题目的知名度较高，如果面试者很快答对，考虑不予计分

10. 有 5 只猴子在海边发现一堆桃子，决定第二天来平分。第二天清晨，第一只猴子最早来到，它左分右分分不开，就朝海里扔了一只，恰好可以分成 5 份，它拿上自己 的一份走了。第 2，3，4，5 只猴子也遇到同样的问题，采用了同样的方法，都是扔掉一只后，恰好可以

分成 5 份。问这堆桃子至少有多少只

答：3121

1. 从最后一只猴子开始考虑，给分
2. 设最后一只猴子来的时候，桃子的数量为 5x+1；第 4 只就为(5x+1) * (5/4) + 1，给分
3. 依次类推，得到第一只猴子来的时候，桃子的数量，给分
4. 算出 x 的值，给分

[张一]: 用递归的方式来做，很巧妙！设 f(n)为第 n 次每只猴子的份数。则 4f(n) = 5f(n+1) + 1

1. 给出递归式，给分
2. 化为等比数列，给分

f(n)=a*(4/5)^n-1。

2. 算出答案，

F(5)有意义，a 最少为 5^5，所以 f(1)=5^4 * 4 – 1 =624,总数为 624*5+1=3121

给分

11. 有三个人都不能说话，但都很聪明。他们每人戴一顶帽子，帽子不是黑色就是红色。这三人都苦思苦想，希望知道自己帽子的颜色，但终始终无法得知。有一天，一个外地人见到这三人，随口说了一句话："你们三人至少有一个是戴着红帽子。"说完就走了。当天三人听完这句话，都纷纷回家苦思。第二 天中午，三人依旧一起在广场见面，有两人当即宣布了自己帽子的颜色。随后，第三个人也知道了自己帽子的颜色。请问：这三人的帽子分别为什么颜色

答：

1. 如果只有一顶红帽子，则红帽子的人当场就知道了，排除。给分
2. 如果三顶都是红帽子，则无人能知道自己的颜色，因为他看到的另两顶都是红帽子。可以排除。给分
3. 只能是 2 顶红帽子，1 顶黑帽子。思路如下：A 黑，BC 红。3 人都无法确定，回家苦思。B 想：如果自己头上黑帽，则 C 立即知道头上是红帽，既然 C 不知道，则说明自己头上也是红帽。第二天猜出自己的帽子。C 也是这样。A 根据 BC 的表现，知道只有 2 顶红帽，自己头上是黑帽。给分。

面试者解释比较繁琐，不推荐。

【ZhaiYao: 该题目面试者有自信的说出答案就可以给分，不需要面试者详细解释】

12. A,B 两人射箭，命中率都是 50%。两人比赛射箭，各射 50 箭，射中靶子多的人获胜。比赛中 A 耍赖，多射了 1 箭，问 A 有多大的可能性获胜。

答：

1. 将比赛结果分为胜、平、负三种。给分
2. 各射 50 箭时，三种结果的可能性为 x, y, x；且 x+y+x=100%。给分
3. A 多射 1 箭，则原来获胜时，现在还是获胜；原来打平时，现在 50%可能获胜；原来输时，现在不可能获胜。因此获胜的可能性为 x+0.5y，正好等于 50%。给分

注：如果面试者能以各射 0 箭，各射 1 箭等简单情况，推算出结果和原本的箭数无关，固定为 50%，可以考虑给分。

13. 假设一个国家没有离婚、死亡等情况出现。该国国王规定：一对夫妻的第一胎是男孩时候，不允许再生育；第一胎是女孩时候，夫妻可以选择生第二胎，也可以不生；第二胎是女孩子的时候，还可以继续生。但是一旦生育了一个男孩，那么就严禁再生育。当这个政策运行足够久之后，该国的男女孩子的比例是多少？

答：

解法 1：假设一共生育 x 孩子。那么每次生育并不被其他事件所影响，所以男女孩子各半。1:1

解法 2：数学建模。

解法 3：设想极端情况：(1)大家都坚持只生一胎；(2)大家都一直生育，直到得到男孩为止；结论均为 50%。所以推出一般情况下，也为 50%

注：如果面试者能以各射 0 箭，各射 1 箭等简单情况，推算出结果和原本的箭数无关，固定为 50%，可以考虑给分。

Chap6 编程及代码题

1. *Strcpy 函数的实现。（校招）*

答：

```
1.  char * strcpy(char * strDest,const char * strSrc){
        if ((strDest==NULL)||(strSrc==NULL)) //[1]
            throw "Invalid argument(s)"; //[2]
        char * strDestCopy=strDest;    //[3]
        while ((*strDest++=*strSrc++)!='\0'); //[4]
        return strDestCopy;
    }
2.  为了实现链式返回
```

2. *Bigint 乘法实现（bigint 用数组存储每一个位） 社招*

题目：void multiple(int a[], int b[], int c[]); //a，b 为两个乘数，c 为输出结果

考察点：就是编程实现一下小学的乘法运算，关注 a，b 各位相乘结果写入 c 时，对应的下标运算，以及进位的处理。

如果写的比较快，再考察一下怎么测自己的代码

答：

```
void multiply(int a[], int b[], int c[]) {
    int i, j, k;
    int tmp;
    for (i = 0; i < a.size(); ++i) {
        k = i;
        for (j = 0; j < b.size(); ++j) {
            result[k++] += a[i] * b[j];
        }
    }
    for (k = c.size() - 1;  k >= 0; --k) {
        if (result[k] > 9) {
            if (k != 0){
                c[k-1] += c[k] / 10;
                c[k] %= 10;
            }else{
                tmp = result[k] / 10;
                result[k] %=10;
                result.insert(result.begin(), tmp);
            }
        }
    }
}
```

*3. 实现 char *strtok(char *s, const char *delim);*

题目：The strtok() function can be used to parse the string s into tokens. The first call to strtok() should have s as its first argument. Subsequent calls should have the first argument set to NULL. Each call returns a pointer to the next token, or NULL when no more tokens are found.

答：

```
char *mystrtok(char *s,const char *delim)
{
    static char *last;
    char *tok;
    char *ucdelim;
    char *spanp;
    int c,sc;
    /**//*s 为空，并且上次剩余值也为空，则直接返回 NULL，否则 s 为 last 或当前值中有值的一方*/
    if (s == NULL && (s = last) == NULL)
        return NULL;

    int found = 0;//是否找到与 delim 匹配的字符

    //处理连续的待匹配的字符
    cont:
    c=*s++;
    for (spanp = (char *)delim;(sc = *spanp++) != 0;)
    {
        if (c == sc)
            goto cont;
    }
    if (c == 0)
    {
        last = NULL;
        return NULL;
    }
    tok = s-1;
    while (!found && *s != '\0')
    {
        ucdelim = (char *) delim;
        while (*ucdelim)
        {
            if (*s == *ucdelim)
            {
                found = 1;
                *s = '\0';
                last = s + 1;
```

```
                    break;
                }
                ucdelim++;
            }
            if (!found)
            {
                s++;
                if(*s=='\0')
                    last = NULL;
            }
        }
        return tok;
}
```

4. struct { int a, char b, long c} 的 sizeof 结果是多少，32 位机器和 64 位机器下

答：32bit 下 12; 64bit 下 16

5. int a = (int)(char)(byte)-1; a 的值为？ （位数补齐）

答：byte ⇔ unsigned char => a = -1;

6. aoti()的实现：

要求：

- 不使用库函数，实现 int atoi(char *s)函数；
- 在此基础上，实现自定义的 atoi()升级版，要求：
- int atoi(char *s, unsigned int base)中，2<=base<=16，s 是以进制为 base 表示形式的字符串，比如：base=8 时，”123”表示数字 83。
- 对以上代码进行测试。

考察点：

- 写代码能力，异常处理是否全面。
- 主要考虑点：
- NULL 指针输入；
- atoi()在检测到错误时，无法 return 合适的值来表示这个错误；
- 输入字符串前缀的合法性检查，
- 字符串中的字符是否在当前进制能表示的范围内，例如：字符’F’不在进制为 15 的数字表示范围内。

答：

```
1.  int atoi(char *s){
        int c; // current char
        int result;
        int sign; // if '-', negative; otherwise positive;
        /*skip the whitespace*/
        while( isspace(*s) )
            s++;
        sign = ( *s == '-' ) ? -1 : 1;
```

```
        if( *s == '+' || *s == '-' ) //skip the sign
            s++;
        result = 0;
        while(isdigit(*s)){
            result = 10 * result + ( *s - '0' );
            s++;
        }
        return sign * result;
    }
```

2. result = 10 * result + (*s - '0'); 替换为：result = base * result + (*s – ‘0’)即可

7. *strstr()的实现；*

```
char *strstr(const char *s, const char *find){
        char *cp = s;
        char *s1;
        char *s2;
        /*find="\0"*/
        if(!*find)
            return s;
        while(*cp){
            s1 = cp;
            s2 = find;
            while(*s1 && *s2 && !(*s1 - *s2)){
                s1++;
                s2++;
            }
            if(!*s2)
                return cp;
            cp++;
        }
        return NULL;
}
```

8. *给出单向链表的定义，并实现如下操作：*

- add()：在 head 增加一个节点；
- delete()：删除指定的节点
- retrieve()：对所有节点进行“指定的函数操作”，“指定的函数操作”由用户输入；

考察点：

- 单向链表的定义；
- 功能代码的实现；函数指针的运用（retrieve()中）。

答：

单向链表（单链表）是链表的一种，其特点是链表的链接方向是单向的，对链表的访问要通过顺序读取从头部开始。

1. Node 定义：

```
typedef struct Node{
   DataType data;
   struct node *next;
}node;
```

2. add()：在 head 增加一个节点

```
int add(Node *head, DataType DataX){
    //返回参数：0 分配空间失败,1 成功
    Node NodeAdd=new LinkList;
    if(!NodeAdd)
       return (0);
    NodeAdd->data=DataX;
    NodeAdd->Next=head->Next;
    head->Next=NodeAdd;
    return (1);
}
```

3. delete()：删除指定节点

```
int delete(Node *head, DataType DataX){
    Node *p = head;
    Node *s = p->next;
    while(s != NULL){
        if(s->data != Datax){
                p = p->next;
                s = s->next;
        }
        else{
              p -> next = s -> next;
              free(s);
              return 1;
        }
    }
   return 0;
}
```

4. retrieve()：对所有节点进行“指定的函数操作”，“指定的函数操作”由用户输入；

```
void retrieve(Node *head, void (*visit)(data)){
    Node *p = head;
    while(p != NULL){
        (*visit)(p->data);
        p = p->next;
    }
}
```

9. 请找出下面代码中的所有错误

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);  // 分配 len+1
8、 char* d=dest;
9、 char* s=src[len];    //len-1
10、 while(len--!=0)
11、 d++=s--;
12、 printf("%s",dest);  //尾部要加\0
13、 return 0;  //返回前要释放 malloc 的内存
14、}
```

10. 两路归并

```
void Merge(int *p1, unsigned uCount1, int *p2, unsigned uCount2)
{
	if (p1 == NULL || p2 == NULL || uCount1 == 0 || uCount2 == 0)
	{
		return;
	}
	int i = 0;
	int j = 0;
	bool fSign = false;

	while ((i < uCount1) && (j < uCount2))
	{
		if (p1[i] == p2[j])
		{
			if (!fSign)
			{
				printf("%d\n", p1[i]);
				fSign = true;
			}
			++i;
		}
		else if (p1[i] < p2[j])
		{
			fSign = false;
			++i;
		}
		else
		{
```

```
                fSign = false;
                ++j;
            }
        }
}
```

11. 以下代码会是什么效果？

```
char str[]=”hello”;
int *p=(int *)str;
*p=0x00313200;
printf("%s",str);
```

//提示 0x31 对应字符'1',0x32 对应字符'2'。

答：返回空。” \0”

12. 下列三个函数有没有问题，如果有请指出：

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

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

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

答：

试题 1 字符串 str1 需要 11 个字节才能存放下（包括末尾的’\0’），而 string 只有 10 个字节的空间，strcpy 会导致数组越界；

试题 2 ,如果面试者指出字符数组 str1 不能在数组内结束可以给 3 分 ;如果面试者指出 strcpy(string, str1)调用使得从 str1 内存起复制到 string 内存起所复制的字节数具有不确定性可以给 7 分，在此基础上指出库函数 strcpy 工作方式的给 10 分；

对试题 3，if(strlen(str1) <= 10)应改为 if(strlen(str1) < 10)，因为 strlen 的结果未统计’\0’所占用的 1 个字节。

13. 说明以下函数所实现的功能以及可能存在的问题：

```
int function(char *s1, char *s2)
{
        int i=0;
        while(s1[i]==s2[i]  && s2[i]!=0 ) i++;
        return (s1[i]==0 && s2[i]==0);
 }
```

答：

功能——判断 s2 与 s1 是否内容一致

问题——没有判断参数为 NULL 的情况；当 s1==s2 的时候，函数应该直接返回 true，不需要一个个去比较下去；

14. 以下程序的输出是怎样的？请给出分析过程。

```
class myclass
{
  public:
    //Constructor
    myclass( string s ):str(s)
    {
      cout << "Constructing " << endl ;
    }

    //Destructor
    ~myclass()
    {
      cout << "Destructing " << endl ;
    }

    string getValue()
    {
      return str ;
    }

  private:
    string str ;
};
```

```
void display( myclass t )
{
  cout << t.getValue()<< endl ;
}

myclass getTest()
{
  return myclass("Jun") ;
}

int main()
{
  myclass te = myclass("chenjq") ;
  display(te) ;
  getTest() ;
  return 0;
}
```

答：

输出：

Constructing
chenjq
Destructing
Constructing
Destructing
Destructing

重点：需要了解在按值传向函数，以及按值从函数返回的时候对象的构造、析构过程：

（1） 将对象按值传给函数的时候，首先会创建该对象的一个副本，将这个副本传给函数，但是在创建这个副本的时候，不会调用构造函数，而在这个函数返回的时候会调用析构函数。

（2） 函数按值返回对象的时候，也会创建返回对象的一个副本，同样的也不会调用构造函数，在返回后调用析构函数销毁该对象。

15. 解释以下代码中函数 function()的执行过程，指出可能存在的问题：

```
class myclass{
      public:
        myclass(unsigned  int k):length(k){
            content=(int *)malloc(sizeof(int)*k);
        }

        ~myclass(){
            if(content)free(content);
        }
      private:
        unsigned int length;
```

```
        int *content;
    }

void function()
    {
     myclass c1=100;
     myclass c2=c1;
     myclass c3;

    myclass c4=200;
     c4=c1;

}
```

答：

- myclass c1=100; 调用 constructor myclass(unsigned int k)，其中伴随 automatic type conversion；为 c1.content 指针分配了内存。
- myclass c2=c1; 调用默认的 copy constructor，采用 member-wize 的 copy，导致 c2 的 content 指针指向 c1. Content 指向的区域；
- myclass c3; 这里需要调用 default constructor ,但是由于自己定义了其他类型的 constructor，编译器不会在自动帮助生成 default constructor，编译不通过；
- c4=c1; 调用 copy assignment operator，实施 member-wize 的 copy，导致 c4 的 content 指针也指向 c1. Content 指向的区域 ,构造函数为 c4 分配的内存成为孤立区域 ,导致内存泄露。
- 函数退出 ,噩梦开始 :不考虑 c3 的话 ,这里共调用 constructor myclass(unsigned int k)两次，但是却要调用 destructor 三次。由于指针浅 copy ,三次 destructor 会对同一块内存 free 三次，这是个问题。

16. *解释以下代码中函数 function()的执行过程，指出可能存在的问题：*

```
class myclass{
public:
    myclass(unsigned  int k):length(k){
        content=(int *)malloc(sizeof(int)*k);

    }

    ~myclass(){
        if(content)free(content);
    }
private:
    unsigned int length;
    int *content;

}

void function()
```

```
{
    void *p=new myclass(100);
    delete p;
}
```

答：

Delete 操作，由于 p 是 void *类型，在 free 对象本身所占的内存空间之前，不会调用 destructor，带造成 content 指针指向的内存区域的泄露。

17. 请写出以下程序的运行结果，并解释导致这样运行结果的关键性原因。

```
#include <iostream>
using std::cout;
class P
{
public:
virtual void print()
{
cout << "P";
}
};
class Q: public P
{
public:
virtual void print()
{
cout << "Q";
}
};
int main()
{
P * p = new P;
Q * q = static_cast <Q *> (p);

q->print();
    delete p;
cout << endl;

    q = new Q;
    p = q;
    q->print();
    p->print();
    cout << endl;

    p = new (q) P;
    q->print();
```

```
        p->print();
        cout << endl;

        p->~P();
        delete q;
        return 0;
    }
```

答：输出为 —— P QQ PP

Chap7 计算机基础

1. C/C++[标记]

- Inline 函数、虚函数的概念；虚函数是否可以实现成 inline

  inline 函数：

  定义：

  关键字用来定义一个类的内联函数，函数的代码被放入符号表中，在使用时直接进行替换，（像宏一样展开），没有了调用的开销，效率也很高。

  用途：

  引入它的主要原因是用它替代 C 中表达式形式的宏定义。

  虚函数：

  定义：

  在某基类中声明为 virtual 并在一个或多个派生类中被重新定义的成员函数。

  用途：

  实现多态性，通过指向派生类的基类指针，访问派生类中同名覆盖成员函数。

  虚函数是否可以实现成 inline？

  从狭义的角度来讲是不能的，因为虚函数是在运行期间决定如何调用，而 inline 函数实在编译期间决定是否 inline。从广义的角度讲是可以的。参见《More Exceptional C++》第 8 条款，《Exceptional C++ Style》第 25 条款。

- 函数调用过程中，函数参数的入栈顺序，why？

  函数调用过程中，第一个进栈的是（主函数中的）**调用处的下一条指令**（即函数调用语句的下一条可执行语句）**的地址**；然后是函数的各个**参数**，而在大多数 C/C++编译器中，在函数调用的过程中，函数的参数是**由右向左**入栈的；然后是函数内部的**局部变量**（注意 static 变量是不入栈的）；在函数调用结束（函数运行结束）后，局部变量最先出栈，然后是参数，最后栈顶指针指向最开始存的指令地址，程序由该点继续运行。

  函数调用方式决定了函数参数入栈的顺序，是由调用者函数还是被调用函数负责清除栈中的参数等问题，而函数名修饰规则决定了编译器使用何种名字修饰方式来区分不同的函数，如果函数之间的调用约定不匹配或者名字修饰不匹配就会产生以上的问题。

  参考：http://caifuchang.blog.163.com/blog/static/339123312010416112601 51/

- extern 的使用场合以及工作原理

  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，假定你将那个编写错了，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。

C++语言支持函数重载，C 语言不支持函数重载。函数被 C++编译后在库中的名字与 C 语言的不同。假设某个函数的原型为： void foo(int x, int y);该函数被 C 编译器编译后在库中的名字为_foo ， 而 C++ 编译器则会产生像_foo_int_int 之类的名字。C++提供了 C 连接交换指定符号 extern“C”来解决名字匹配问题。

- C++中 struct 和 class 的区别

C++的 class 具有数据封装功能，其包含属性访问级别可以为 private,public 和 protect,还具有实现类接口功能和辅助功能的操作函数，而 struct 属性访问权限只有 public，没有数据封装功能，也就没有实现信息隐藏这一面向对象的思想的机制,struct 本身不含有操作函数，只有数据。

- OO 语言中动态绑定的概念。参考答案网上都有考察精通 c++/java 的面试者对基本概念的了解

静态绑定：编译时绑定，通过对象调用

动态绑定：运行时绑定，通过地址实现

只有采用“指针->函数()”或“引用变量.函数()”的方式调用 C++类中的虚函数才会执行动态绑定。对于 C++中的非虚函数，因为其不具备动态绑定的特征，所以不管采用什么样的方式调用，都不会执行动态绑定。

- C++中多态的概念及实现机制（答题点：虚函数，虚表）

多态(Polymorphism)是面向对象的核心概念， C++中多态可以分为基于继承和虚函数的动态多态以及基于模板的静态多态，

1) 描述以下表示形式的意义：Char *q[]={ “xxx”， “yyy”， “zzz” };
Char (*p)[]=a;

答:

前者：指针数组，q[0],q[1],q[2]都为 char *类型的指针；
后者：数组指针，p 指向一个 char 数组。

2) 解释以下表示形式的意义：

A) "a" B) '\\' C) 'W' D) ’abc’

3) 解释以下表示形式的意义及其作用：typedef int (*PFUN)(int x,int y);

答案：函数指针；

4) Call by value、call by reference 和 call by pointer 的优缺点，举例说明各自的应用形式。

5) C++对象的 copy constructor 与 copy assignment 的区别与联系，谈谈使用时的注意事项，列举它们应用的场景及。

答：

copy constructor：从一个已有的对象来构造另一个对象；包括：
用已有对象来初始化新声明的对象；
将对象按值传递给函数作为参数；

函数按值返回对象。

copy assignment：将已有的对象赋值个另一个已有的对象；

实例：

```
Person A(B);          // copy constructor
Person C=B;           //copy constructor
Function1(D);         //copy constructor
B= Function2(…);      //copy constructor
Person D;
D=B;                          //copy assignment
```

注意事项 :编译器默认的 copy constructor 和 copy assignment 操作 ,是按照 member-wise copy 的方式逐个 copy 每个 member，这种浅拷贝操作在有些情况下可能造成资源泄漏/指向重叠。

如果的确需要 deep copy，需要自定义相应操作。这时需要清楚哪些地方用了 copy constructor，哪些地方用了 copy assignment，从而分别自定义 copy constructor 和 copy assignment。一般来说，自定义的 copy constructor、destructor 和 copy assignment 操作常常同时出现。

区别与联系：

copy constructor 不用检测是否是用一个对象来初始化它自己；

copy constructor 不用对被构造对象做资源清理操作，如 delete 操作；

6) 说明以下表达式的意义：TYPE *t = new(a) TYPETYPE;

答： Placement new 操作：在 a 指向的内存区域，调用 default constructor 构造一个 TYPE 类型的对象，并返回该对象的指针。

7) 解释以下表达式的区别：TYPE t 与 TYPE *PT=new TYPE;

答： TYPE t; 从 static area 或者 stack 上分配内存，并调用 default constructor;

TYPE *PT=new TYPE; 从堆上分配内存，并调用 default constructor;

2. JAVA

1) Java 多线程编程机制，线程类，互斥机制（synchronize）

Java 线程类：Thread，Runnable 接口

Java 互斥：synchronize 关键字

wait 和 notify 方法

答：

- 继承和实现的区别。当一个类已经继承了其他类 ,就不能继承 Thread ,只能实现 runnable。
- 每个 JAVA 对象都有一把所，当有多个线程同时访问共享资源的时候，需要 Synchronize 来控制安全性，synchronize 分 synchronize 方法和 synchronize 块，使用 synchronize 块时， 一定要显示的获得该对象的锁(如 synchronize(object))而方法则不需要
- 在 java 多线程编程中，最被经常用到的便是 wait 与 notfiy 方法，这两个方法可以用来更加精确地控制被同步的代码，从而使得被同步的代码最小化，提高并发效率。当某个类的某个方法被标记为 synchronized 时，这个方法在同一时间只能被一个线程访问。此时这个方法中的所有代码都是被同步的，只有当一个线程执行完所有的代码之后，下一个线程才能开始执行。当被同步的方法代码量比较小，而且每一步执行都非常快的时候仅仅使用 synchronized 关键字就够了。

但是，如果被同步的方法里面有一些代码是可以被共享的，而且这些能够被共享的代码里面存在比较耗时的操作时，仅仅使用 synchronized 关键字就无法达到最高的效率，这个时候可以使用 wait 与 notify 方法来对并发访问做更进一步的控制。

wait()方法表示，放弃当前对资源的占有权，等啊等啊，一直等到有人通知我，我才会运行后面的代码。 notify()方法表示，当前的线程已经放弃对资源的占有，通知等待的线程来获得对资源的占有权，但是只有一个线程能够从 wait 状态中恢复，然后继续运行 wait()后面的语句；notifyAll()方法表示，当前的线程已经放弃对资源的占有，通知所有的等待线程从 wait()方法后的语句开始运行。

2) 引用传递

Java 函数传递的是参数的引用

深度拷贝：重写 clone()

3) 接口和抽象类的作用有什么区别

答：

接口能够多重继承；抽象类只能继承一个；

接口只能定义常量；抽象类能够定义变量

接口的函数都是抽象的；抽象类的函数可以是非抽象的

接口 implement,抽象 extend

4) 重载和覆写

答：

重载时 function(int i), function(int i, int j)。相同的函数名，不同的签名；重写是函数签名相同。但是子类的函数覆盖了父类的同名（签名）函数。

5) 容器类型选择：

ArrayList：随机访问好

LinkedList：元素添加和移出（中心位置）

HashSet：普遍性能都比 TreeSet 好

TreeSet：保持内部元素的排序状态

HashMap：Hashtable 的替代品，一般性能都较好

TreeMap：保持内部元素的排序状态

6) 向上转型和向下转型

向上安全，向下执行期检查：ClassCastException

7) 多态

继承，接口与多态的关系

3. 多线程:

1) 是否写过多线程代码

2) 线程间通信的方法

答：同一进程的各线程可以直接读写进程数据段进行通信，同时需要同步和互斥手段的辅助。而不同进程间的线程通信可以参考不同进程间的通信。

3) 多线程同步方式（能答出几个就可以）

答：事件 Event，临界区域 Critical Section，互斥器 Mutex， 信号量 Semaphore

详细参考：http://baike.baidu.com/view/2808915.htm

4) 同步和异步的区别

答：同步是阻塞模式，即发送方发出数据后，等接收方发回响应以后才发下一个数据包的通讯方式；而异步是非阻塞方式，发送方发出数据后，不等接收方发回响应，接着发送下个数据包的通讯方式。

例如：普通 B/S 模式（同步）AJAX 技术（异步）

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事

异步: 请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕

5) 进程和线程的概念

答：进程是操作系统结构的基础；是一个正在执行的程序；计算机中正在运行的程序实例；可以分配给处理器并由处理器执行的一个实体；由单一顺序的执行显示，一个当前状态和一组相关的系统资源所描述的活动单元。

线程(thread),有时被称为轻量级进程(Lightweight Process，LWP)，是程序执行流的最小单元。一个标准的线程由线程 ID，当前指令指针(PC)，寄存器集合和堆栈组成。另外，线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有就绪、阻塞和运行三种基本状态。

二者区别：子进程和父进程有不同的代码和数据空间,而多个线程则共享数据空间,每个线程有自己的执行堆栈和程序计数器为其执行上下文.多线程主要是为了节约 CPU 时间,发挥利用,根据具体情况而定. 线程的运行中需要使用计算机的内存资源和 CPU。通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位。

6) 进程的地址空间是怎么回事？虚拟内存是如何实现的

答：虚拟内存的存在使得 CPU 上的指令访问的地址都是虚拟地址，而这些地址是需要在物理内存中真实存在的，这里就需要在虚拟地址和物理地址直接建立一个映射关系(应当是多对一的关系)，说白了就是一个整数集合到另一个整数集合的映射。然后在查找时，根据相应的算法（如先进先出，最多使用等）进行调度。

7) Linux 进程间通信有哪些方式，优缺点如何

答： Linux 下进程间通信的几种主要手段：

- 管道（Pipe）及有名管道（named pipe）：管道可用于具有亲缘关系进程间的通信，有名管道克服了管道没有名字的限制，因此，除具有管道所具有的功能外，它还允许无亲缘关系进程间的通信；

- 信号（Signal）：信号是比较复杂的通信方式，用于通知接受进程有某种事件发生，除了用于进程间通信外，进程还可以发送信号给进程本身；Linux 除了支持 Unix 早期信号语义函数 sigal 外，还支持语义符合 Posix.1 标准的信号函数 sigaction（实际上，该函数是基于 BSD 的，BSD 为了实现可靠信号机制，又能够统一对外接口，用 sigaction 函数重新实现了 signal 函数）；

- 报文（Message）队列（消息队列）：消息队列是消息的链接表，包括 Posix 消息队列 systemV 消息队列。有足够权限的进程可以向队列中添加消息，被赋予读权限的进程则可以读走队列中的消息。消息队列克服了信号承载信息量少，管道只能承载无格式字节流以及缓冲区大小受限等缺点。

- 共享内存：使得多个进程可以访问同一块内存空间，是最快的可用 IPC 形式。是针对其他通信机制运行效率较低而设计的。往往与其它通信机制，如信号量结合使用，来达到进程间的同步及互斥。

- 信号量（semaphore）：主要作为进程间以及同一进程不同线程之间的同步手段。

- 套接口（Socket）：更为一般的进程间通信机制，可用于不同机器之间的进程间通信。起初是由 Unix 系统的 BSD 分支开发出来的，但现在一般可以移植到其它类 Unix 系统上：Linux 和 System V 的变种都支持套接字。

4. 网络编程

1) Tcp/IP 连接模型

答：TCP/IP 通讯协议采用了四层的层级模型结构，每一层都调用它的下一层所提供的网络任务来完成自己的需求：

应用层 (Application)：应用层是个很广泛的概念，有一些基本相同的系统级 TCP/IP 应用以及应用协议，也有许多的企业商业应用和互联网应用。

传输层 (Transport)：传输层包括 UDP 和 TCP，UDP 几乎不对报文进行检查，而 TCP 提供传输保证。

网络层 (Network) :网络层协议由一系列协议组成，包括 ICMP、IGMP、RIP、OSPF、IP(v4,v6) 等。

链路层 (Link)：又称为物理数据网络接口层，负责报文传输。

2) Io 复用，select 和 poll 的区别

答：IO 复用模型：调用 select 或 poll，在这两个系统调用中的某一个上阻塞，而不是阻塞于真正 I/O 系统调用。阻塞于 select 调用 ,等待数据报套接口可读。当 select 返回套接口可读条件时 ,调用 recevfrom 将数据报拷贝到应用缓冲区中。

select 和 poll 的区别：select()和 poll()本质上来讲做的是同一件事，只是完成的方法不一样。两者都通过检验一组文件描述符来检测是否有特定的时间将在上面发生并在一定的时间内等待其发生。

select()函数的接口主要是建立在一种叫'fd_set'类型的基础上。它('fd_set') 是一组文件描述符 (fd) 的集合。

poll ()接受一个指向结构'struct pollfd'列表的指针，其中包括了你想测试的文件描述符和事件。事件由一个在结构中事件域的比特掩码确定。当前 的结构在调用后将被填写并在事件发生后返回。

3) 实现一个 client/server 的通信过程，写一段伪代码（重点写出需要调用的各 socket api）

答：

```
Server 端：
  socket addr_in svr_addr;
  socket addr_in client_addr;
  ServerSocket(addr_in_svr_addr);
  bind();
  listen();
  while(true){
        client = accept(addr_in client_addr);
        if(!fork()){
            read();
            send(client);
            close(client);
        }
  }
```

```
Client 端:
  socket(addr_in_client_addr);
  connect(server);
```

```
data = recv();
close();
```

4) TCP 和 UDP 的区别，拥塞窗口的概念，如何建立一个 TCP 连接

答：TCP---传输控制协议,提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个 TCP 连接，之后才能传输数据。TCP 提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。

UDP---用户数据报协议，是一个简单的面向数据报的运输层协议。UDP 不提供可靠性，它只是把应用程序传给 IP 层的数据报发送出去，但是并不能保证它们能到达目的地。由于 UDP 在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快

拥塞窗口： 在 TCP 传送中的拥塞控制中，发送端通过网络的拥塞程度所给出的一个大小值，而这个值就是拥塞窗口。

建立 TCP 连接：TCP 连接时通过三次握手建立的，详细见下面所示。

5) Tcp 连接建立的三次握手是指什么

答：第一步是请求端（客户端）发送一个包含 SYN 即同步（Synchronize）标志的 TCP 报文，SYN 同步报文会指明客户端使用的端口以及 TCP 连接的初始序号；

第二步，服务器在收到客户端的 SYN 报文后，将返回一个 SYN+ACK 的报文，表示客户端的请求被接受，同时 TCP 序号被加一，ACK 即确认（Acknowledgement）。

第三步，客户端也返回一个确认报文 ACK 给服务器端，同样 TCP 序列号被加一，到此一个 TCP 连接完成。然后才开始通信的第二步：数据处理。

TCP Client	Flags	TCP Server
1 Send SYN (seq=x)	----SYN--->	SYN Received
2 SYN/ACK Received	<---SYN/ACK----	Send SYN (seq=y)， ACK (x+1)
3 Send ACK (y+1)	----ACK--->	ACK Received， Connection Established

在 TCP/IP 协议中，TCP 协议提供可靠的连接服务，采用三次握手建立一个连接。

第一次握手：建立连接时，客户端发送 syn 包(syn=j)到服务器，并进入 SYN_SEND 状态，等待服务器确认；

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手：客户端收到服务器的 SYN＋ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据，

6) tcp 连接的 time_wait 是什么状态，描述其发生的场景，说明它存在的好处/坏处

答：TCP 包括一种机制，确认与连接相关的包在网络上延迟后不被同一对主机之间后来的连接所接受. 这种机制是由 TCP 协议的 T IM E_WA IT 状态实现的. 当一端点关闭一个 TCP 连接时，它保留与连接有关的状态—— 通常是 TCB(TCP protocol control block) 的复制——两倍于最大段存活时间(maximum segment life time，即 MSL). 处于此状态的连接即处于 TIME_WAIT 状态；

TIME_WAIT 的优点：可靠地实现 TCP 全双工连接的终止；允许老的重复分节在网络中消逝 。

TIME_WAIT 的缺点 :重负载服务器有可能保留有上千个 TIME_WAIT TCB，因而消耗了大量的内存并能减慢活动的连接。

5. Linux

1) Shell 命令，sed，awk 等。

参考问题及答案：

Shell:

用没用过 shell 命令。如果用过，说下常用的 shell 命令。

问下 cat chmod diff more/less, paste, head/tail, uniq, sort, who

Sed:

Sed 简介参看翟耀的分享： http://com.baidu.com/twiki/bin/view/Sdc/SedEditorShare

考察这几个命令的理解：

(1) sed '/line/ p' list.txt

(2) sed -n '/line/ p' list.txt

(3) sed '3q' new

(4)

$ sed -n -f holdtest list.txt

其中 holdtest 是个文本文件

$ cat holdtest

h

n

p

g

p

(***Only For 面试官***

h #拿到一行 line1 并且复制到 hold 区。目前 p 区[line1],h 区[line1]

n #取下一行 line2 放入 p 区。目前 p 区[line2],h 区[line1]

p #打印 p 区内容:line2

g #把 h 区内容复制到 p 区。目前 p 区[line1],h 区[line1]

p #打印 p 区内容:line1。这样就做到了每两行互相替换。)

Awk:

Awk 简介参看翟耀的分享： http://com.baidu.com/twiki/bin/view/Sdc/GawkShare

可以考察如下题目：

gawk '/^f/' cars

gawk '$1 ~ /l/' cars

用 gawk 把所有行加上行长，并按照长度大小排序

gawk '{print length, $0}' cars | sort –n

2) 是否熟悉 Linux 下的 C/C++开发，gcc，gdb，makefile 等一些编译，调试工具，是否熟悉。

3) 文件系统的组织结构，文件重定向是如何实现的

参考答案：ls -la /

下面我们把 Linux 文件系统的树形结构的主要目录列一下

/ Linux 文件系统的入口，也是处于最高一级的目录；

/bin 基础系统所需要的那些命令位于此目录，也是最小系统所需要的命令；比如 ls、cp、mkdir 等命令；功能和/usr/bin 类似，这个目录中的文件都是可执行的，普通用户都可以使用的命令。做为基础系统所需要的最基础的命令就是放在这里。

/boot Linux 文件系统的内核及引导系统程序所需要的文件，比如 vmlinuz initrd.img 文件都位于这个目录中。在一般情况下，GRUB 或 LILO 系统引导管理器也位于这个目录；

/dev 设备文件存储目录，比如声卡、磁盘... ...

/etc 系统配置文件的所在地，一些服务器的配置文件也在这里；比如用户帐号及密码配置文件；

/home 普通用户家目录默认存放目录；

/lib 库文件存放目录

/mnt 这个目录一般是用于存放挂载储存设备的挂载目录的，比如有 cdrom 等目录。

/root Linux 文件系统超级权限用户 root 的 Home 目录；

考察管道，重定向等概念和使用。

重定向的分享见： http://com.baidu.com/twiki/bin/view/Sdc/OutputRedirection

重定向用着简单，重要是思考重定向的实现方式。

可以考下 ./myprog 1>tmp 2>tmp 是否正确？应该怎么改？

答案：./myprog 1>tmp 2>&1

6. *cookie 和 session 的概念；两者的实现方式；两者的应用场景*

- cookie 机制和 session 机制的区别

具体来说 cookie 机制采用的是在客户端保持状态的方案，而 session 机制采用的是在服务器端保持状态的方案。

同时我们也看到，由于才服务器端保持状态的方案在客户端也需要保存一个标识，所以 session 机制可能需要借助于 cookie 机制来达到保存标识的目的。其实有其他办法。

session 机制是一种服务器端的机制，服务器使用一种类似于散列表的结构(也可能就是使用散列表)来保存信息。但程序需要为某个客户端的请求创建一个 session 的时候，服务器首先检查这个客户端的请求里是否包含了一个 session 标识 - 称为 session id,如果已经包含一个 session id 则说明以前已经为此客户创建过 session，服务器就按照 session id 把这个 session 检索出来使用(如果检索不到，可能会新建一个，这种情况可能出现在服务端已经删除了该用户对应的 session 对象，但用户人为地在请求的 URL 后面附加上一个 JSESSION 的参数)。

如果客户请求不包含 session id，则为此客户创建一个 session 并且生成一个与此 session 相关联的 session id，这个 session id 将在本次响应中返回给客户端保存。

cookie 将你的所有提供的这些信息和登陆信息,以文件的形式保存到你的计算机上,当访问其他文件的时候,首先把 cookies 提交给服务器,服务器判断这里的信息,得知你是什么样的一个用户,然后给出你相应的功能页面.可以自行设置 cookies 的存在周期，除非设置了临时 cookies，否则关闭浏览器后 cookies 信息仍旧保存在主机的硬盘上。

session:将你所提供的信息,以牺牲服务器资源的方式记录下来,当你访问其他页面的时候,先判断服务器上的关于你的信息,并提供给你相应的功能页面.变量保存在客户端主机的内存上 ,关闭浏览器或者 session 脚本过期后，即自动清除。

- Cookies 与 Session 的应用场景：

Cookies 的安全性能一直是倍受争议的。虽然 Cookies 是保存在本机上的，但是其信息的完全可见性且易于本地编辑性，往往可以引起很多的安全问题。所以 Cookies 到底该不该用，到底该怎样用，就有了一个需要给定的底线。

先来看看，网站的敏感数据有哪些。

登陆验证信息。一般采用 Session("Logon") = true or false 的形式。

用户的各种私人信息，比如姓名等，某种情况下，需要保存在 Session 里

需要在页面间传递的内容信息，比如调查工作需要分好几步。每一步的信息都保存在 Session 里，最后在统一更新到数据库。

当然还会有很多，这里列举一些比较典型的

假如，一个人孤僻到不想碰 Session，因为他认为，如果用户万一不小心关闭了浏览器，那么之前保存的数据就全部丢失了。所以，他出于好意，决定把这些用 Session 的地方，都改成用 Cookies 来存储，这完全是可行的，且基本操作和用 Session 一模一样。那么，下面就针对以上的 3 个典型例子，做一个分析

很显然，只要某个有意非法入侵者，知道该网站验证登陆信息的 Session 变量是什么，那么他就可以事先编辑好该 Cookies，放入到 Cookies 目录中，这样就可以顺利通过验证了。这是不是很可怕？

Cookies 完全是可见的，即使程序员设定了 Cookies 的生存周期（比如只在用户会话有效期内有效），它也是不安全的。假设，用户忘了关浏览器 或者一个恶意者硬性把用户给打晕，那用户的损失将是巨大的。

这点如上点一样，很容易被它人窃取重要的私人信息。但，其还有一个问题所在是，可能这些数据信息量太大，而使得 Cookies 的文件大小剧增。这可不是用户希望所看到的。

显然，Cookies 并不是那么一块好啃的小甜饼。但，Cookies 的存在，当然有其原因。它给予程序员更多发挥编程才能的空间。所以，使用 Cookies 改有个底线。这个底线一般来说，遵循以下原则。

不要保存私人信息。

任何重要数据，最好通过加密形式来保存数据（最简单的可以用 URLEncode，当然也可以用完善的可逆加密方式，遗憾的是，最好不要用 md5 来加密）。

是否保存登陆信息，需有用户自行选择。

长于 10K 的数据，不要用到 Cookies。

参考答案网上都有

考察项目多为前端开发/测试的面试者对基本概念的了解

Chap 8 项目和背景

结合项目询问对测试了解的情况，谈谈对测试流程的理解 什么阶段介入测试比较好 对 Code Review 的看法，RD 和 QA 各自的侧重点；测试完成的衡量标准，是否接触过测试覆盖率；自动化的实现方法，谈一下测 web 常见的一些自动化思路。。。

Chap9 外部工具

1. *Selenium*

1) 介绍一下 selenium？ Selenium 有哪些特点？

答： Selenium 是一个针对 Web 应用的开源测试框架，它的测试用例可以用 HTML table 和 HTML 代码或者一些非常流行的编程序言进行开发，而且它能在几乎所有现在的浏览器上执行。selenium 可以被部署到 Windows, Linux 和 Macintosh 平台上。它支持的语言有 Java, Python, Ruby, .Net, Perl 等等

主要的特点：

- 支持录制和回放
- 能够灵活的根据 ID, Name 或者 XPath 来进行页面元素的选取
- 能够进行 Debug 和设置断点
- 能够把测试脚本保存成 HTML, Ruby 或者其他语言
- 支持 user-extensions.js 形式的用户扩展
- 能够自动进行页面的断言

2) Selenium 分为哪几部分?

Selenium IDE: 一个 firefox 插件，可以录制、回放测试脚本。

Selenium RC：支持用程序语言编写测试用例，比如 Ruby、Java、C#等，这样做的好处是，可以将 Selenium 和其他测试框架集成，比如.NET 环境下，可以把 Selenium 和 NUnit 集成，用 Selenium 来编写测试用例，用 NUnit 来实现测试用例的自动化运行。

Selenium Core：Selenium Core 是 Selenium 的核心，是由 Javascript 和 Html 文件组成的，它是 Selenium IDE 和 Selenium RC 的核心引擎。

Selenium Grid：Selenium Grid 是 Selenium 的一个扩展，它可以将一批 selenium 脚本分配到不同的测试机上面同步运行，从而节省执行时间。

3) Selenium 中 verifyTextPresent 和 assertTextPresent 命令的区别？

答：verifyTestPresent 和 assertTextPresnt 命令都用于判断页面上是否存在指定的文本，区别是 verifyTestPresent 结果是 false 时，剩余的步骤会继续执行。但 assertTextPresent 结果是 false 时直接终结该脚本剩余步骤的运行。

4) Selenium 中 click 和 clickAndWait 的区别？

答：click 命令模拟用户点击的动作。命令执行完毕后立刻执行下一条命令。ClickAndWait 命令在点击后有一个等待的过程，会等页面重新加载完毕再执行下一条命令。

5) 如果有一个按钮，点击该按钮后会发出一个 ajax call，然后得到返回结果后将该内容显示到新弹出的一个 layer 中。在写脚本的时候，点击按钮这个动作是否可以用 clickAndWait 命令？如果不行，怎么解决？

答：不能使用 clickAndWait。因为 ajax call 不会刷新页面，clickandWait 命令会因为等待页面重新加载而出现 Time out. 对这种情况应该使用 click 命令结合 pause 命令。

6) 下面是某页面中的一段 html source，其中某 a 链接的地址中包含关键字 test。用 xpath 定位该 a 元素。

<html>

……

baidu

.....

</html>

答： //a[contains(@href, 'test')]

7) Selenium 内部运行机制是这样的？为什么 Selenium 脚本可以运行在几乎所有的主流浏览器上。

答：Selenium 的核心是 Javascript 写的，它通过 javascript 来 实现对 Html 页面的操作的。它提供了丰富的指定 Html 页 面元素和操作页面元素的方法。Selenium 打开浏览器时，把自己的 JavaScript 文件嵌入网页中。然后 Selenium 的网页通过 frame 嵌入目标网页。这样 ,就可以使用 Selenium 的 JavaScript 对象来控制目标网页。因为 selenium 是用 javascript 去操作页面元素 ,所以支持几乎所有主流的浏览器。

8) Selenium 用 javascript 去操作页面元素会碰到什么问题？ Selenium 是如何解决这个问题的？

答：javascript 受同源策略的限制。当浏览器要运行一个脚本时，便会进行同源检查，只有和被操控网页同源的脚本才能被执行。

Selenium 通过采用代理模式来解决这个问题。测试脚本向 Selenium Server 发送 Http 请求，要求和 Selenium Server 建立连接。Selenium Server 启动浏览器 ,把 Selenium Core 加载入浏览器页面当中，并把浏览器的代理设置为 Selenium Server 的 Http Proxy。测试脚本向 Selenium Server 发送 Http 请求，Selenium Server 对请求进行解析 ,然后通过 Http Proxy 发送 JS 命令通知 Selenium Core 执行操作浏览器的动作。Selenium Core 接收到指令后，执行操作。Selenium Server 接收到浏览器的发送的 Http 请求后，自己重组 Http 请求，获取对应的 Web 页面。Selenium Server 的 Http Proxy 把接收的 Web 页面返回给浏览器。因为浏览器存在同源策略，所以 Selenium RC 中的 Selenium Server 需要以这种代理模式运行

2. Ruby-Watir

1) 对于 Javascript 生成的元素，Watir 识别和操作该元素是如何做的？

答 实际上 ,Watir::IE 封装了一个当前页面的 DOM tree ,而不是 html source。比如如果用 javascrīpt 动态产生一个元素，在这里仍然可以访问。普通元素，按照 Watir 封装的类就可以实现访问和操作。Windows 对象支持并不好。

详细参考该篇文档：http://www.51testing.com/?uid-84226-action-viewspace-itemid-147237

2) 对于多浏览器的 web 测试，Ruby-watir 是如何支持的？

答：Watir，FireWatir，SafariWatir 各自的类库支持，然而对于多浏览器的 web 自动化 case 编写，不建议直接使用各个类库去编写 case。在支持各浏览器操作的类库之上进行一层封装。在自动化 case 中不需要去关注各个浏览器的操作之间的区别，只需要通过封装的类库直接初始化相应的浏览器实例即可。

3) 简述 ruby-watir 中如何实现数据驱动。

答 :将与逻辑无关的数据剥离开来 ,多组数据作为一组输入 ,循环调用重复的用例接口。在 ruby-watir 中，支持读取 Excel 等多种数据管理格式。

4) Ruby-watir 中对于正则的支持，正则表达式的考察。

答：在 Ruby 中，要建立一个正则表达式，只要把要匹配的模式放到两个斜线中就行了(/*pattern*/)，而且，在 Ruby 中，正则表达式也是对象，可以像对象一样被操作。

例如，匹配 Email 地址的正则表达式：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*等。

5) watir 不支持的 windows 控件，是如何解决的？

解答：采用第三方 autoit 技术，来模拟键盘或鼠标操作。参考

http://wiki.openqa.org/display/WTR/JavaScript+Pop+Ups

6) 简述 ruby 元编程的概念，应用场景，以及如何应用。

答：通常元编程被认为是通过程序来生成程序，在 Ruby 中，Ruby 元编程的使用变得相当的简单和容易实现，使用 Ruby 语言本身来产生 Ruby 代码，不需要借助外部的工具，著名的 RoR 框架就是建立在 Ruby 元编程的基础上的。

比如 ROR 框架中的

```
class Person
attr_reader :name
end
```

attr_reader 的实现如下：

```
# def attr_reader(*syms)
# syms.each do |sym|
# class_eval %{def #{sym}
# @#{sym}
# end
# end
# end
```

#注：class_eval 是为一个 class 增加 method 的。可以接 string 和 block 为参数。

在 ROR 框架中已经广泛应用了元编程，ActiveRecord 在 OR 映射模型中，将关系数据库中的关系表映射到对象模型时，将关系表的表名映射到类名，表中的每一个元组映射到对应于这个类的一个对象，元组的一个字段对应于对象的一个属性。ActiveRecord 在这里灵活应用了 ruby 元编程的动态特性，实现了优雅的解决方案。

3. QTP:

1) 简单介绍下 QTP

答 :QTP 是个用于录制和回放脚本的自动化测试工具。它可以用于 web 和客户端程序的自动化测试。

2) QTP 中 RO 与 TO 的区别？

答：TO 是 Test Object 的简称，RO 是 Runtime Object 简称，既用来区分仓库对象和实际对象，又用来区分对象的封装接口和自身接口。

从实际作用上来看，应该说 TO 就是是仓库文件里定义的仓库对象，RO 是被测试软件的实际对象。

为用户提供了两种操作对象的接口，一种就是对象的封装接口，另一种是对象的自身接口。对象的自身接口是对象控件本身的接口，对象的封装接口是 QTP 为对象封装的另一层接口，它是 QTP 通过调用对象的自身接口来实现的。两种接口的脚本书写格式的差别在于：自身接口需要在对象名后面加 object 再加属 性名或方法名，封装接口就不用在对象名后面加 object.

3) QTP 中 OBJECT SPY 的作用？

答：查看对象，在查看窗口里有列出这些接口，包括属性和方法。

窗口中间有选择栏让你选择 Run-time Object 或者 Test Object,当你选择 Runtime Object 时，它显示的就是对象的自身接口（自身的 属性和方法）.当你选择 Test Object 时，它显示的就是对象的封装接口（封装的属性和方法）

4) 如何激活一个窗口？

答：激活窗口使用的方法 Window("").Activate

5) 编写一个 QTP 脚本，实现向记事本中输入“baidu”。

答：SystemUtil.Run “C:\WINDOWS\system32\notepad.exe”

Window(" Notepad").Activate
Window(" Notepad").WinEditor(" Edit").Type "baidu"

4. LoadRunner

1) 什么是负载测试？什么是性能测试？

答：负载测试是通过改变系统负载方式、增加负载等来发现系统中所存在的性能问题。负载测试是一种测试方法，可以为性能测试、压力测试所采用。负载测试的加载方 式也有很多种，可以根据测试需要来选择。

性能测试是为获取或验证系统性能指标而进行测试。多数情况下，性能测试会在不同负载情况下进行。

压力测试通常是在高负载情况下来对系统的稳定性进行测试，更有效地发现系统稳定性的隐患和系统在负载峰值的条件下功能隐患等。

2) 性能测试包含了哪些测试（至少举出 3 种）

答：压力测试、负载测试、并发测试、可靠测试、失效恢复测试。

3) 简述使用 Loadrunner 的步骤

答：脚本录制设置—录制脚本—调试脚本—场景设置—结果分析

4) LoadRunner 由哪些部件组成？你使用 LoadRunner 的哪个部件来录制脚本，哪个部件可以模拟多用户并发下回放脚本？

答：virtual user generator

controller

analysis

virtual user generator 来录制脚本，controller 来回放脚本

5) 什么是集合点？设置集合点有什么意义?Loadrunner 中设置集合点的函数是哪个?

答：集合点：设置多个用户到达某个用户数量点集合，同时触发一个事务,以达到模拟真实环境下同时多个用户操作,同时模拟负载,实现性能测试的最终目的

LR_rendezvous（ "集合点名称" ）

6) 你在 LR 中如何编写自定义函数？请给出一些你在以前进行的项目中编写的函数。

答：在创建用户自定义函数前我们需要和创建 DLL（external libary）。把库放在 VuGen bin 目录下。一旦加了库，把自定义函数分配做一个参数。该函数应该具有一下格式：__declspec (dllexport) char* <function name>(char*, char*)。

7) 以线程方式运行的虚拟用户有哪些优点？

答：VuGen 提供了用多线程的便利。这使得在每个生成器上可以跑更多的虚拟用户。如果是以进程的方式跑虚拟用户，为每个用户加载相同的驱动程序到内存中，因 此占用了大量的内存。这就限制了在单个生成器上能跑的虚拟用户数。如果按进程运行，给定的所有虚拟用户数（比如 100）只是加载一个驱动程序实例到内存 里。每个进程共用父驱动程序的内存，因此在每个生成器上可以跑更多的虚拟用户。