1. 使用两种命令创建一个文件?

a. touch a.txt

b. vi a.txt

c. mkdir abc

d. cat > a.txt 建立一文件，然后把接下来的键盘输入写入文件，直到按Ctrl+D为止.

2. 硬链接和软连接的区别?

a. 硬链接：

1、文件有相同的 inode 及 data block；

2、只能对已存在的文件进行创建；

3、不能交叉文件系统进行硬链接的创建；

4、不能对目录进行创建，只可对文件创建；

5、删除一个硬链接文件并不影响其他有相同 inode 号的文件。

b. 软链接：

1、软链接有自己的文件属性及权限等；

2、可对不存在的文件或目录创建软链接；

3、软链接可交叉文件系统；

4、软链接可对文件或目录创建；

5、创建软链接时，链接计数 i_nlink 不会增加；

6、删除软链接并不影响被指向的文件，但若被指向的原文件被删除，则相关软连接被称为死链接（即 dangling link，若被指向路径文件被重新创建，死链接可恢复为正常的软链接）。

3. linux常用命令有哪些?

```
查找关闭端口进程 netstat -nlp | grep :3306 kill pid
删除文件 rm -rf
查找日志 cat xx.log | grep 'xxx' | more
解压tar.gz tar -xzvf file.tar.gz
创建文件 touch filename cat > filename
修改文件 vi
```

4. 怎么查看一个java线程的资源耗用?

> linux下，所有的java内部线程，其实都对应了一个进程id，也就是说，linux上的jvm将java程序中的线程映射为操作系统进程。

```
1、jps -lvm或者ps -ef | grep java查看当前机器上运行的Java应用进程
2、top -Hp pid可以查看Java所有线程的资源耗用
4、printf "%x\n" pid等到线程ID的16进制
5、jstack Java应用进程ID | grep 线程ID的16进制
```

5. Load过高的可能性有哪些?

cpu load的飙升，一方面可能和full gc的次数增大有关，一方面可能和死循环有关系

6. /etc/hosts文件什么作用?

在当前主机给ip设置别名，通过该别名可以访问到该ip地址，通过别名、ip访问的效果是一样的

7. 如何快速的将一个文本中的"abc"转换成"xyz"?

```
vi filename编辑文本，按Esc键，输入:%s/abc/xyz/g
```

8. 如何在log文件中搜索找出error的日志?

```
cat xx.log | grep 'error'
```

9. 发现硬盘空间不够，如何快速找出占用空间最大的文件?

```
find . -type f -size +100M | xargs du -h | sort -nr
```

10. Java服务端问题排查（OOM，CPU高，Load高，类冲突）?

a. https://blog.csdn.net/and1kaney/article/details/51214219

b. 业务日志相关：

i. less或者more

ii. grep

iii. tail -f filename

> ps:切忌vim直接打开大日志文件，因为会直接加载到内存的

c. 数据库相关：

i. 登录线上库，show processlist查看数据库连接情况

d. jvm相关：

i. jps显示java进程

ii. jinfo实时查看和调整jvm参数

iii. jstat监控jvm各种运行状态信息；

iv. jstack(Stack Trace for Java)命令用于生成JVM进程当前时刻的线程的调用堆栈，可以用来定位线程间死锁、锁等待、等待外部资源等

v. jmap(Memory Map for Java) 命令用于生成堆转储快照dump文件，除了这种方式还可以通过-XX:HeapDumpOnOutOfMemoryError参数，可以在虚拟机发生OOM的时候自动生成堆的dump文件，或者kill -3命令发出进程退出信号"吓唬"一下虚拟机，也能拿到dump文件。

e. oom问题：

i. 配置了-XX:+HeapDumpOnOutOfMemoryError, 在发生OOM的时候会在-XX:HeapDumpPath生成堆的dump文件，结合MAT，可以对dump文件进行分析，查找出发生OOM的原因。

ii. 另外手动dump堆快照，可以使用命令jmap -dump:format=b,file=file_name pid 或者kill -3 pid

f. 死锁：

i. jps -v

ii. jstack -l pid

g. 线程block、线程数暴涨：

i. jstack -l pid |wc -l

ii. jstack -l pid |grep "BLOCKED"|wc -l

iii. jstack -l pid |grep "Waiting on condition"|wc -l

> 线程block问题一般是等待io、等待网络、等待监视器锁等造成，可能会导致请求超时、造成造成线程数暴涨导致系统502等。

h. 服务器问题：

i. cpu：top

ii. 内存：

1. free -m -c10 -s1：

a. -m：以MB为单位显示，其他的有-k -g -b

b. -s: 间隔多少秒持续观察内存使用状况

c. -c:观察多少次

```
[sankuai@dx-cos-bcrm01 logs]$ free -m
             total       used       free     shared    buffers     cached
Mem:          7870       7592        277          0        236       3500
-/+ buffers/cache:       3855       4014
Swap:         2047          0       2047
```


2. vmstat 1 10：1表示每隔1s输出一次,10 表示输出10次

a. r: 运行队列中进程数量，这个值也可以判断是否需要增加CPU。（长期大于1）

b. b: 等待IO的进程数量。

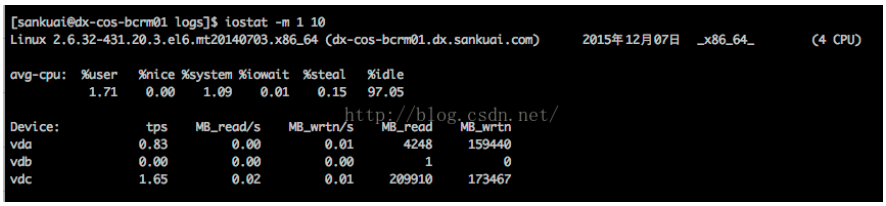
```
[sankuai@dx-cos-bcrm01 logs]$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 282840 242300 3586556    0    0     4     7    0    1  2  1 97  0  0
 0  0      0 282824 242300 3586560    0    0     0     0 6739 15977  3  2 96  0  0
 0  0      0 282832 242300 3586560    0    0     0     0 6766 16011  2  2 96  0  0
```


i. io：

i. iostat -m 1 10：

1. -m：某些使用block为单位的列强制使用MB为单位

2. 1 10：数据显示每隔1秒刷新一次，共显示10次



j. 网络：

i. netstat -antp：

1. -a (all)显示所有选项，默认不显示LISTEN相关
2. -t (tcp)仅显示tcp相关选项
3. -u (udp)仅显示udp相关选项
4. -n 拒绝显示别名，能显示数字的全部转化成数字。
5. -l 仅列出有在 Listen (监听) 的服服务状态
6. -p 显示建立相关链接的程序名

11. Java常用问题排查工具及用法（top,iostat,vmstat,sar,tcpdump,jvisualvm,jmap,jconsole）

https://blog.csdn.net/xad707348125/article/details/51985854

12. Thread dump文件如何分析（Runnable，锁，代码栈，操作系统线程id关联）

a. Thread Dump 能诊断的问题

i. 查找内存泄露，常见的是程序里load大量的数据到缓存；

ii. 发现死锁线程；

b. 如何抓取Thread Dump信息：

i. 一般当服务器挂起,崩溃或者性能底下时,就需要抓取服务器的线程堆栈(Thread Dump)用于后续的分析. 在实际运行中，往往一次 dump的信息，还不足以确认问题。为了反映线程状态的动态变化，需要接连多次做threaddump，每次间隔10-20s，建议至少产生三次 dump信息，如果每次 dump都指向同一个问题，我们才确定问题的典型性。

ii. linux命令获取：

```
ps -ef  | grep java
kill -3 <pid>
```

iii. jdk自带工具获取：

```
jps 或 ps -ef|grepjava (获取PID)
jstack [-l ]<pid> | tee -a jstack.log  (获取ThreadDump)
```

c. 分析：https://blog.csdn.net/rachel_luo/article/details/8920596

13. 如何查看Java应用的线程信息?

通过top命令拿到线程的pid后使用jstack命令

14. 计数?

```
wc -l
```