**一．有文件file1**

**1.** **查询file1里面空行所在的行号；**

**2.** **查询file1中以abc结尾的行；**

**3.** **打印file1** **文件的第一到三行；**

**二．写一条192.168.10.0网段从网关192.168.9.1出去的路由**

**三．写一条放行80端口的防火墙规则。**

**四. 每天早上6点到12点，每隔2小时执行一次/usr/bin/httpd.sh怎么实现**

**五．编写个shell脚本判断根目录下有没有abc目录，如果没有就发邮件给admin@121.com**

**六．Raid0，raid1，raid5原理**

**七．备份mysql数据库test库**

**八. 如何查看占用端口8080的进程**

**九．Apache有几种工作模式，分别介绍下其特点，并说明什么情况下采用不同的工作模式？**

**说明：这个问题在前面已经有过介绍。相信后续的笔试题还会出现。所以我从网上总结一些比较好的答案写在这里，帮助大家理解。**

**答案：Web服务器Apache目前一共有三种稳定的MPM（Multi-Processing Module，多进程处理模块）模式。它们分别是prefork，worker和event，它们同时也代表这Apache的演变和发展。使用httpd -V 命令查看。在configure配置编译参数的时候，可以使用 --with-mpm=prefork|worker|event 来指定编译为那一种MPM，当然也可以用编译为三种都支持：--enable-mpms-shared=all，这样在编译的时候会在modules目录下自动编译出三个MPM文件的so，然后通过修改httpd.conf配置文件更改MPM。**

**1、Prefork MPM**

**Prefork MPM实现了一个非线程的、预派生的web服务器。它在Apache启动之初，就先预派生一些子进程，然后等待连接；可以减少频繁创建和销毁进程的开销，每个子进程只有一个线程，在一个时间点内，只能处理一个请求。这是一个成熟稳定，可以兼容新老模块，也不需要担心线程安全问题，但是一个进程相对占用资源，消耗大量内存，不擅长处理高并发的场景。**

**2、Worker MPM**

**和prefork模式相比，worker使用了多进程和多线程的混合模式，worker模式也同样会先预派生一些子进程，然后每个子进程创建一些线程，同时包括一个监听线程，每个请求过来会被分配到一个线程来服务。线程比起进程会更轻量，因为线程是通过共享父进程的内存空间，因此，内存的占用会减少一些，在高并发的场景下会比prefork有更多可用的线程，表现会更优秀一些；另外，如果一个线程出现了问题也会导致同一进程下的线程出现问题，如果是多个线程出现问题，也只是影响Apache的一部分，而不是全部。由于用到多进程多线程，需要考虑到线程的安全了，在使用keep-alive长连接的时候，某个线程会一直被占用，即使中间没有请求，需要等待到超时才会被释放（该问题在prefork模式下也存在）。**

**3、Event MPM**

**这是Apache最新的工作模式，它和worker模式很像，不同的是在于它解决了keep-alive长连接的时候占用线程资源被浪费的问题，在event工作模式中，会有一些专门的线程用来管理这些keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务器的线程，执行完毕后，又允许它释放。这增强了在高并发场景下的请求处理。**

**十．你使用过监控软件吗？说说期特点。**

**十一．你对运维工程师的理解以及对其工作的认识**

**十二．简述mysql主从复制过程**

**答：Mysql的 Replication 是一个异步的复制过程，从一个 Mysql instace(我们称之为 Master)复制到另一个 Mysql instance(我们称之 Slave)。在 Master 与 Slave 之间的实现整个复制过程主要由三个线程来完成，其中两个线程(Sql线程和IO线程)在 Slave 端，另外一个线程(IO线程)在 Master 端。**

**要实现** **MySQL 的 Replication ，首先必须打开 Master 端的Binary Log(mysql-bin.xxxxxx)功能，否则无法实现。因为整个复制过程实际上就是Slave从Master端获取该日志然后再在自己身上完全顺序的执行日志中所记录的各种操作。打开MySQL的 Binary Log可以通过在启动MySQL Server的过程中使用“log-bin”参数选项，或者在my.cnf配置文件中的mysqld参数组([mysqld]标识后的参数部分)增加 “log-bin”** **参数项。**

**MySQL复制的基本过程如下：**

**1. Slave上面的IO线程连接上Master，并请求从指定日志文件的指定位置(或者从最开始的日志)之后的日志内容;**

**2. Master接收到来自Slave的IO线程的请求后，通过负责复制的IO线程根据请求信息读取指定日志指定位置之后的日志信息，返回给Slave端的IO线程。返回信息中除了日志所包含的信息之外，还包括本次返回的信息在Master端的Binary Log文件的名称以及在Binary Log中的位置;**

**3. Slave的IO线程接收到信息后，将接收到的日志内容依次写入到 Slave 端的Relay Log文件(mysql-relay-bin.xxxxxx)的最末端，并将读取到的Master端的bin-log的文件名和位置记录到master- info文件中，以便在下一次读取的时候能够清楚的高速Master“我需要从某个bin-log的哪个位置开始往后的日志内容，请发给我”**

**4. Slave的SQL线程检测到Relay Log中新增加了内容后，会马上解析该Log文件中的内容成为在Master端真实执行时候的那些可执行的Query语句，并在自身执行这些Query。这样，实际上就是在Master端和Slave端执行了同样的Query，所以两端的数据是完全一样的。**