2021技术随手记

2021年2月18日　14:59

1. **Do while while foreach for 区别**
   Do while 和while类似，do while 会不管条件真假先执行一次，while 条件为真才执行，foreach 循环为先读取整块数据，然后再循环，而 for 主要用于限制循环次数
   例如循环数组，while 是移动内部指针，foreach 是对数组副本进行操作，而 foreach 在读操作比较快，在写操作比较慢，因为 php 的 **引用计数写时复制** 的特性
2. **进程，线程及通信方式**
   https://www.php.cn/php-ask-453612.html 进程和线程有点 主从关系一样的，线程共享进程的资源
   进程间通信方式：1. 管道 2. 信号 3. 消息队列 Posix 消息队列 4. 共享内存 5. 套接口
3. **Php 查看扩展路径**
   php -i | grep -i extension_dir　或者 phpinfo 来看一下　或者　echo ini_get('extension_dir');
4. **Php 底层数组实现方式** linked list + hashtable 双向链表是中间映射表，用来存放索引和具体存储位置，然后具体存储位置对应的具体值在 hashtable 中
5. **写时复制：**如果两个变量是相同的值，则共享同一块内存，而那块内存的 is_ref =1 refcount = 1 后者被引用一次 +1，为 0 的时候被销毁，相当于资源延迟分配。
   垃圾回收：不会立即回收，会放入缓冲区（一个双向链表），然后默认到了 10000 个开始回收，先将 refcount -1 ， =0 则进行回收
6. **解决内存溢出：**1、要增加PHP可用内存大小；2、对数组进行分批处理，将用过的变量及时销毁；3、尽可能减少静态变量的使用；4、数据库操作完成后，要马上关闭连接。5、可以使用 memory_get_usage () 函数，获取当前占用内存 根据当前使用的内存来调整程序
   引申：① unset () 函数只能在变量值占用内存空间超过 256 字节时才会释放内存空间 ② 有当指向该变量的所有变量（如引用变量）都被销毁后，才会释放内存 ③ unset 被引用的变量只会接触引用关系，不会销毁该变量
7. **Php7 新特性** https://www.php.net/manual/zh/migration70.new-features.php 简要总结就是：
   标量类型声明、返回值类型声明、通过 define() 定义常量数组、匿名类、相同命名空间类一次性导入
   Php7 底层优化：① ZVAL 结构体优化，占用由24字节降低为16字节 ② 内部类型 zend_string，结构体成员变量采用 char 数组，不是用 char* ③ PHP 数组实现由 hashtable 变为 zend array ④ 函数调用机制，改进函数调用机制，通过优化参数传递环节，减少了一些指令
8. **Php 排序二维数组**　array_multisort + array_column 就行

```php
<?php
$user_list = [
    ['name' => '张三', 'age' => 28],
    ['name' => '赵六', 'age' => 21],
    ['name' => '王五', 'age' => 20],
    ['name' => '李四', 'age' => 21]
];

```

```
array_multisort(array_column($user_list, 'age'), SORT_ASC, $user_list);

var_dump($user_list);
```

9. **缓存的应用场景：** ① 数据不需要强一致性 ② 读多写少，并且读取得数据重复性较高
10. **Php 异步执行脚本：** ① popen 调用脚本，缺点是无法跨越，不能传参，会产生进程，高并发了会创建大量进程 ② curl 方式，最小响应超时时间是 1s，也受限 ③ fsockopen 打开一个网络连接或者 unix 套接字连接，原理和 http 一致，支持毫秒级超时处理 ④ 引入 swoole
    https://www.php.cn/php-weizijiaocheng-469392.html
11. **Isset empty 区别：** isset 检测一个变量是否有值，包含 false 0 空字符串，不可以为 null Empty 检查是否有空值 空字符串 0 null false，有则返回 true 非空或者非零值则返回 false
12. **浅析OOP思想：** 面向对象程序设计（英语：Object-oriented programming，缩写：OOP）是种具有对象概念的编程典范，同时也是一种程序开发的抽象方针。它可能包含数据、属性、代码与方法。对象则指的是类（class）的实例。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性，对象里的程序可以访问及经常修改对象相关连的数据。在面向对象程序编程里，计算机程序会被设计成彼此相关的对象。
    向对象程序设计可以看作一种在程序中包含各种独立而又互相调用的对象的思想。
    对象的产生：① 以原型为基础 ② 以类为基础
13. **Cookie 和 session 区别：** ① cookie在浏览器，session在服务器上 ② session的生存周期根据浏览器进程存在，cookie可以设置和调整 ③ session 必须借助 cookie
    301 状态码是永久移动　302 是临时移动　500 代码 文件权限 资源有问题 501 请求方法服务器不支持
    502 网关错误，例如得到了一个无效响应一类的就会出现这种错误 503 超载或者维护模式 504 网关超时，即在指定时间内没有正确的响应
14. **Iptolong long2ip 注意转换成整形的时候负数问题**

```
function IP2Long($ip) {

    $ips = explode('.', $ip);

    if(count($ips) != 4) {

        return false;

    }

    return ($ips[0] << 24) + ($ips[1] << 16) + ($ips[2] << 8) + $ips[3];

}



function Long2IP($int) {

    $ip1 = $ipint >> 24;

    $ip2 = ($ipint >> 16) & 255;

    $ip3 = ($ipint >> 8) & 255;

```

```
    $ip4 = $ipint & 255;

    return $ip1.'.'.$ip2.'.'.$ip3.'.'.$ip4;

}
```

15. **验证邮箱有效性：**filter_var($email, FILTER_VALIDATE_EMAIL); 验证IP也是有这样的情况
16. **php链式调用：** ① 使用魔法函数__call结合call_user_func来实现 ② 使用魔法函数__call结合call_user_func_array来实现 ③ 不使用魔法函数__call来实现，修改 __call() 为 trim **重点在于，返回$this指针，方便调用后者函数。**
17. **不使用第三个变量来交换两个变量的值** ① 两个为 数字时

```php
<?php
/**
 * 双方变量为数字时，可用交换方法五
 * 使用加减运算符，相当于数学运算了
*/
$a = 1; // a变量原始值
$b = 2; // b变量原始值
echo '交换之前 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出原始值
$a=$a+$b; // $a $b和值
$b=$a-$b; // 不解释..
$a=$a-$b; // 不解释..
echo '交换之后 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出结果值
```

② 两个为字符串时

```php
<?php
/**
 * 双方变量为字符串或者数字时，可用交换方法四
 * 使用异或运算
 */
$a = "This is A"; // a变量原始值
$b = "This is B"; // b变量原始值
echo '交换之前 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出原始值

/**
 * 原始二进制:
 *
$a:01010100011010000110100101110011001000000110100101110011001000000100000001
 *
$b:01010100011010000110100101110011001000000110100101110011001000000100000010
 *
 * 下面主要使用按位异或交换，具体请参照下列给出的二进制过程，
 */

$a=$a^$b; // 此刻
$a:00000000000000000000000000000000000000000000000000000000000000000000000011
$b=$b^$a; // 此刻
$b:01010100011010000110100101110011001000000110100101110011001000000100000001
$a=$a^$b; // 此刻
$a:01010100011010000110100101110011001000000110100101110011001000000100000010
echo '交换之后 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出结果值
```

或者 str_replace 处理

```php
<?php
$a = "This is A"; // a变量原始值
$b = "This is B"; // b变量原始值
echo '交换之前 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出原始值
$a .= $b; // 将$b的值追加到$a中
$b = str_replace($b, "", $a); // 在$a(原始$a+$b)中，将$b替换为空，则余下的返回值为$a
$a = str_replace($b, "", $a); // 此时，$b为原始$a值，则在$a(原始$a+$b)中将$b(原始$a)替换为空，则余下的返回值则为原始$b,交换成功
echo '交换之后 $a 的值: '.$a.', $b 的值: '.$b,'<br>'; // 输出结果值
```

18. **Strtoupper/strtolower 遇到中文会乱码** ① 需要手动分割字符串，然后 ord 函数判断是否是单词，是则大小写转换，中文则不处理 ② mb_convert_case 函数中有可选参数，直接能处理这种情况
19. **Php-fpm 和 NGINX 通信机制**

    CGI：是 Web Server 与 Web Application 之间数据交换的一种协议。

    FastCGI：同 CGI，是一种通信协议，但比 CGI 在效率上做了一些优化。

    PHP-CGI：是 PHP （Web Application）对 Web Server 提供的 CGI 协议的接口程序。

    PHP-FPM：是 PHP（Web Application）对 Web Server 提供的 FastCGI 协议的接口程序，额外还提供了相对智能一些任务管理。

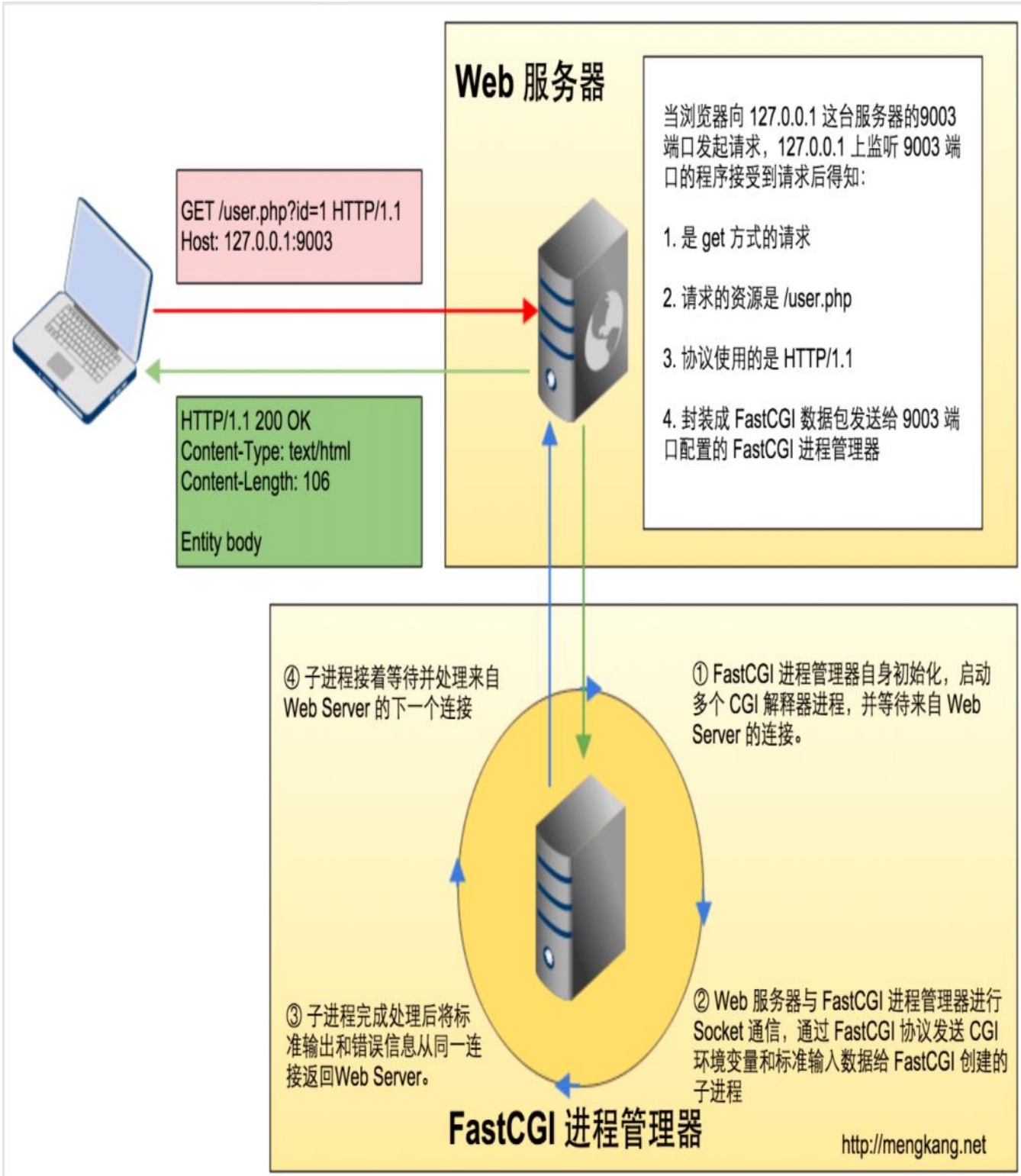
    CGI就是规定要传哪些数据，以什么样的格式传递给后方处理这个请求的协议，例如URL、查询字符串、POST数据、HTTP header，缺点是每次请求都有启动和退出操作，不适合并发场景

    Fastcgi 是常驻类型的，不需要每次去激活了

    FastCGI程序会 先启动一个master，解析配置环境，初始化执行环境，然后再启动多个worker。当请求过来时，master会传递给一个worker，然后立即可以接受下一个请求。

    首先要说的是：**fastcgi是一个协议，php-fpm实现了这个协议。**

    大家都知道，PHP的解释器是php-cgi。php-cgi只是个CGI程序，他自己本身只能解析请求，返回结果，不会进程管理，所以就出现了一些能够调度php-cgi进程的程序，php-fpm就是这样的一个东西。它克服了php-cgi变更php.ini配置后，需重启php-cgi才能让新的php-ini生效，不可以平滑重启，直接杀死php-cgi进程，php就不能运行了的问题。修改php.ini之后，php-cgi进程的确没办法平滑重启的。php-fpm对此的处理机制是新的worker用新的配置，已经存在的worker处理完手上的活就可以歇着了，通过这种机制来平滑过度。



20. **Include 和 require 区别**
    这两者是语言结构，不是函数，他们都可以直接引用参数，而不是括号内引用参数
    include在用时加载，一般放在代码段中，出错时继续执行下面的代码
    require一般放在脚本最前面，会一开始就读取，出错时停止运行代码
    _once 是已加载的不加载
21. **Http 和 HTTPS区别**
    Https 需要证书 http 是明文传输 使用 80 端口 https 是具有安全性的ssl加密传输协议，使用443端口 http的链接是无状态的，https 是ssl + http 协议构建的可进行加密传输，身份认证的协议
22. **Explain 后需要关注的信息：**

列名	备注
type	本次查询表联接类型，从这里可以看到本次查询大概的效率
key	最终选择的索引，如果没有索引的话，本次查询效率通常很差
key_len	本次查询用于结果过滤的索引实际长度，参见另一篇分享（FAQ系列-解读EXPLAIN执行计划中的key_len）
rows	预计需要扫描的记录数，预计需要扫描的记录数**越小越好**
Extra	额外附加信息，主要确认是否出现 **Using filesort、Using temporary** 这两种情况

首先看下 **type** 有几种结果，分别表示什么意思：

类型	备注
ALL	执行**full table scan**，这是**最差**的一种方式
index	执行**full index scan**，并且可以通过索引完成结果扫描并且直接从索引中取的想要的结果数据，也就是可以避免**回表**，比ALL略好，因为索引文件通常比全部数据要来的小
range	利用索引进行范围查询，比index略好
index_subquery	子查询中可以用到索引
unique_subquery	子查询中可以用到唯一索引，效率比 index_subquery 更高些
index_merge	可以利用**index merge**特性用到多个索引，提高查询效率
ref_or_null	表连接类型是ref，但进行扫描的索引列中可能包含NULL值
fulltext	全文检索
ref	基于索引的等值查询，或者表间等值连接
eq_ref	表连接时基于主键或非NULL的唯一索引完成扫描，比ref略好
const	基于主键或唯一索引唯一值查询，最多返回一条结果，比eq_ref略好
system	查询对象表只有一行数据，这是最好的情况

上面几种情况，从上到下一次是**最差到最好。**

再来看下Extra列中需要注意出现的几种情况：

关键字	备注
Using filesort	将用外部排序而不是按照索引顺序排列结果，数据较少时从内存排序，否则需要在磁盘完成排序，代价非常高，**需要添加合适的索引**
Using temporary	需要创建一个临时表来存储结果，这通常发生在对没有索引的列进行GROUP BY时，或者ORDER BY里的列不都在索引里，**需要添加合适的索引**
Using index	表示MySQL使用覆盖索引避免全表扫描，不需要再到表中进行二次查找数据，这是比较好的结果之一。注意不要和type中的index类型混淆

Using where	通常是进行了全表引扫描后再用WHERE子句完成结果过滤，**需要添加合适的索引**
Impossible WHERE	对Where子句判断的结果总是false而不能选择任何数据，例如where 1=0，无需过多关注
Select tables optimized away	使用某些聚合函数来访问存在索引的某个字段时，优化器会通过索引直接一次定位到所需要的数据行完成整个查询，例如MIN()\MAX()，这种也是比较好的结果之一

23. **Php-fpm 与 php 交互**

    Php-fpm 运行的三种模式：

    Static 模式最简单，直接启动配置的固定数量的进程，但是灵活性不够高

    Ondemand 模式相对 static 模式比较复杂，会根据请求量的增加动态增加，但是处理完请求后不会立即释放，而是由定时事件定时的检测空闲到一定时间的进程才会释放

    Dynamic 模式类似于 ondemand 模式，但进程的回收机制不同于 ondemand 模式，会根据 idle 数量进行增加和减少worker数量

    Php-fpm 运行的逻辑：

    Fpm 的实现就是创建一个 master 进程，在 master 进程中创建 worker pool 并监听 socket，然后 fork出多个子进程（work），这些 worker 在启动后阻塞在 fcgi_accept_request() 上，各自 accept 请求，有请求到达后 worker 开始读取请求数据，读取完成后开始处理然后再返回，在这期间是不会接收其它请求的，也就是说 fpm 的子进程同时只能响应一个请求，只有把这个请求处理完成后才会 accept 下一个请求。

    Nginx 与 php-fpm 有两种通信方式：tcp socket 和 unix socket，unix 不需要经过网络协议栈，不需要打包拆包，计算校验和，维护序号和应答，只是将应用层数据从一个进程拷贝到另一个进程，减少不必要的 tcp 开销，高并发时性能不稳定，tcp 模式可以保证通信的正确性和完整性，效率可以通过负载均衡等优化。

24. **数据库触发器 trigger**

    触发器是一种特殊的存储过程，它被分配给某个特定的表，触发器都是自动调用的。当一特定的表数据被插入，更新或删除时，数据库需要执行一定的动作，触发器是确保数据完整性和一致性的基本有效的方法。

```
use 数据库名
create/alter trigger 触发器名
on 表名
for insert / delete /update
as
触发器要执行的操作
go

# enable/disable/drop trigger 触发器名
```

    应用场景有：数据检查-例如周末禁止添加员工，安全性确认-例如年龄不能调低，数据备份

25. **数据库存储过程**

    存储过程是一个预编译的SQL语句，执行效率高；存储过程代码放在数据库中，直接调

用，无需网络通信；安全性高，需要有一定权限的用户才行；可以重复使用
缺点是：物理迁移困难

26. **数据库连接池实现原理**
    连接池的作用就是为了提高性能，将已经创建好的连接保存在池中，当有请求来时，直接使用已经创建好的连接对 Server 端进行访问。这样 省略了创建连接和销毁连接的过程，从而提高性能。
27. **Redis 常见应用场景**
    首页热点新闻/商品，避免频繁读取数据库　bitmap 用来记录连续签到/登录情况　新闻阅读量的计数器
    最新新闻列表 lpush 就行，然后读取　简单的消息发布系统 pubsub　sortedset 来做排行榜
28. **Linux 查看系统信息的基础命令**

```
系统
# uname -a         # 查看内核/操作系统/CPU信息
# head -n 1 /etc/issue   # 查看操作系统版本
# cat /proc/cpuinfo    # 查看CPU信息
# hostname         # 查看计算机名
# free -m          # 查看内存使用量和交换区使用量

资源
# df -h          # 查看各分区使用情况
# du -sh <目录名>     # 查看指定目录的大小
# grep MemTotal /proc/meminfo   # 查看内存总量
# grep MemFree /proc/meminfo   # 查看空闲内存量
# uptime          # 查看系统运行时间、用户数、负载
# cat /proc/loadavg    # 查看系统负载

网络
# ifconfig         # 查看所有网络接口的属性
# iptables -L       # 查看防火墙设置
# route -n         # 查看路由表
# netstat -lntp      # 查看所有监听端口
# netstat -antp      # 查看所有已经建立的连接
# netstat -s        # 查看网络统计信息

用户
# w            # 查看活动用户
# id <用户名>       # 查看指定用户信息
# last           # 查看用户登录日志
# cut -d: -f1 /etc/passwd   # 查看系统所有用户
# cut -d: -f1 /etc/group   # 查看系统所有组
# crontab -l        # 查看当前用户的计划任务

进程
# ps -ef          # 查看所有进程
# top           # 实时显示进程状态
```

29. **Find grep 命令区别**
    grep命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打

印出来。

Find 从指定的起始目录开始，递归地搜索其各个子目录，查找满足寻找条件的文件并对之采取相关的操作

简单是：**grep是查找匹配条件的行，find是搜索匹配条件的文件**

30. **Awk**

Awk 内置变量：

变量	用法
$0	当前记录（这个变量中存放着整个行的内容）
$1~$n	当前记录的第n个字段，字段间由FS分隔
FS	输入字段分隔符 默认是空格或\t
NF	当前记录中的字段个数，就是有多少列
NR	已经读出的记录数，就是行号，从1开始，如果有多个文件话，这个值也是不断累加中。
FNR	当前记录数，与NR不同的是，这个值会是各个文件自己的行号
RS	输入的记录分隔符， 默认为换行符
OFS	输出字段分隔符， 默认也是空格
ORS	输出的记录分隔符，默认为换行符
FILENAME	当前输入文件的名字

常用命令：

```
# 打印每一行的第二和第三个字段
awk '{print $2, $3}' file

# 统计文件的行数
awk ' END {print NR}' file

# 对 awk 处理的行做过滤
awk 'NR < 5' #行号小于5
awk 'NR==1,NR==4 {print}' file #行号等于1和4的打印出来
awk '/linux/' #包含linux文本的行（可以用正则表达式来指定，超级强大）
awk '!/linux/' #不包含linux文本的行

# 使用 -F 来设置定界符（默认为空格）
awk -F: '{print $NF}' /etc/passwd

# awk 实现head 命令
awk 'NR<=10{print}' filename

# 实现tail命令
awk '{buffer[NR%10] = $0;} END{for(i=0;i<11;i++){ \
print buffer[i %10]} } ' filename

# 查询访问最频繁的100个请求，主要是各种参数都包含了
grep -v ".php"  access.log | awk '{print $7}' | sort |uniq -c | sort -rn | head -n 100

# 查询访问 100 次以上的 ip
awk '{print $1}' access.log | sort -n |uniq -c |awk '{if($1 >100) print $0}'|sort -rn
```

```
28
29 # 查询指定 ip 访问最多的 100 个页面
30 grep '112.97.250.255' access.log |awk '{print $7}'|sort |uniq -c
|sort -rn |head -n 100
31
32 # 查询最近 1000 条请求访问最多的地址
33 tail -1000 access.log |awk '{print $7}'|sort|uniq -c|sort -
nr|less
34
35 # 按每秒统计请求数，显示top 100 的时间点 cut是截取 14-21 位，分钟为
14-18 小时为 14-15
36 awk '{print $4}' access.log |cut -c 14-21|sort|uniq -c|sort -
nr|head -n 100
```

31. **查看php进程和cpu占用**
    Ps -rf | grep "php-fpm"　　top | grep "php-fpm"
32. **Which 和 whereis 区别**
    Which 是用来查找系统***PATH目录下***的可执行文件。说白了就是查找那些我们已经安装好的可以直接执行的命令，which 是基于 path 目录查找的。
    Whereis 这个命令可以用来查找二进制（命令）、源文件、man文件。Whereis 是基于索引数据库的，locate也是基于数据库的，find 是基于硬盘文件的
33. **负载均衡的几种实现方式及原理**
    ① ip负载均衡，相当于多一到N次重定向，过程
    ② DNS 负载均衡，DNS支持一个域名多个ip地址了
    ③ 反向代理负载均衡，NGINX 根据一定规则进行请求分发
    ④ F5硬件级别
    ⑥ CDN 对于静态文件的负载均衡
    负载均衡构建在原有网络结构之上，它提供了一种透明且廉价有效的方法扩展服务器和网络设备的带宽、加强网络数据处理能力、增加吞吐量、提高网络的可用性和灵活性。
34. **数据库主从复制的原理，会不会延迟，会该怎样解决**
    三个要点：网络延迟，master 负载 slave 负载 slave 对数据安全性的要求
    原理 ① master 将数据改变记录到 binlog 中 ② slave 启动一个io线程，从指定位置开始同步 binlog ③ 读取到 master 数据的更新，slave 写入到 replaylog 中，然后开始重放数据
    Tps 是事务数/秒 qps 是每秒查询率
    延迟原因：主库的 tps 并发较高时，产生的 ddl 超过 slave 的执行，或者网络延迟较大
    解决：减少网络延迟，关闭 slave 的 sync_binlog 设置成大点就行，累计多次事务之后刷盘 innodb_flush_log_at_trx_commit = 2 事务提交之后刷盘，slave 上也可以关闭这个，缺点是意外断电了会丢失数据
35. 如何保障数据的可用性，即使被删库了也能恢复到分钟级别。你会怎么做。
    数据库集群方案就行，删掉主库了会自动选举从库，业务保持稳定，然后就是精细化的备份
36. **数据库链接过多的原因和解决方案**
    原因：① 配置的 max_connections 数量太少，修改配置或者 set global max_connections=xxx 就行 ② sleep 的链接回收太慢，修改 wait_timeout 就行，调小点加速回收 ③ 使用连接池
37. **502 504错误的原因**
    502 是无效响应，① nginx 无法与 php-fpm 进行连接，检查 php-fpm 是否启动 ② 脚本执行超时，然后 php-fpm 终止了执行和worker进程，也可能是高并发情况下，超过

了最大子进程数量

max_execution_time request_terminate_timeout max_children 这三个配置相关

504 是 php 脚本的执行时间超过了 nginx 的等待时间，可能由 502 升级成为 504，和以下的 nginx 配置相关

fastcgi_connect_timeout 60;

fastcgi_read_timeout 300;

fastcgi_send_timeout 300;

38. **从输入 url 到页面展现经历了哪些**

DNS 解析:将域名解析成 IP 地址
TCP 连接：TCP 三次握手
发送 HTTP 请求
服务器处理请求并返回 HTTP 报文
浏览器解析渲染页面
断开连接：TCP 四次挥手

39. **Redis 在具体业务中的应用**

考试部分，redis 存储某本书的数据，存储一小时，提升在课堂测验部分的查询性能

验证码收取部分，存储用完即销毁的验证码数据 存储当天 ip 手机号 的请求次数

40. **Php7 新增的特性**

?? 运算符（NULL 合并运算符）

组合比较符

函数返回值类型声明 : type

define 可以定义常量数组

标量类型声明

use 批量声明

匿名类，支持用 new class 来实例化一个匿名类，『用后即焚』

闭包（Closure）（匿名函数）增加了一个 call 方法