# 概念题

## 解释什么是队列(queue)，什么栈(stack)，有何区别？ 队列：队列是一种特殊的[线性表](http://baike.baidu.com/view/178622.htm)，它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作。进行插入操作的端称为队尾，进行删除操作的端称为队头。队列中没有元素时，称为空队列。

栈（stack）：是一种数据结构，是只能在某一端插入和删除的特殊线性表。它按照后进先出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据。

堆栈：都是一种数据项按序排列的数据结构，只能在一端(称为栈顶(top))对数据项进行插入和删除。

## 说明下列程序中变量的输出结果，并说明为何得到这个结果。

1. $str = “here is string”;

$str0 = $str;$str1 = $str0;unset($str0);  
echo $str;echo $str0;echo $str1;

1. $str = “here is string”;

$str0 = &$str;$str1 = $str0;unset($str0);

echo $str;echo $str0;echo $str1;

1. $str = “here is string”;

$str0 = &$str;$str1 = $str0;$str0 = null;

echo $str;echo $str0;echo $str1;

1. $str = “here is string”;

$str0 = &$str;$str1 = &$str0;unset($str1);

echo $str;echo $str0;echo $str1;

1. $str = “here is string”;

$str0 = &$str;$str1 = &$str0; $str1 = null;  
echo $str;echo $str0;echo $str1;

1. $str = “here is string”;

$str0 = &$str;$str1 = &$str0;$str0 = null;  
echo $str;echo $str0;echo $str1;

a)”here is string” null ”here is string” b) ”here is string” null ”here is string”

c)null null ”here is string” d)”here is string” ”here is string” null

e)null null null f)null null null

## echo、print、print\_r 的区别

print 非函数，一个语言结构。不能用于参数传递 Returns *1*, always

echo 非函数，一个语言结构。不能用于参数传递。无返回值。

print\_r 函数。返回值 Boolean。

## [var\_export](http://www.laruence.com/2008/04/03/15.html) 、debug\_zval\_dump 与 var\_dump的不同

var\_export必须返回合法的php代码， 也就是说，var\_export返回的代码，可以直接当作php代码赋值个一个变量。 而这个变量就会取得和被var\_export一样的类型。变量类型为resource的时候， 是无法简单copy复制的，所以， 当var\_export的变量是resource类型时， var\_export会返回NULL

var\_dump 输出变量信息，包含变量的类型。无返回。

debug\_zval\_dump 与var\_dump 类似多一个 refcount 信息。

## php中 如何获得 客户端ip 和服务器的ip

$\_SERVER["SERVER\_ADDR"] ["REMOTE\_ADDR"]

## 有一个IP地址（192.168.0.1），请写出其32位无符号整数形式。

printf("%u\n", ip2long($ip));

## 简述heredoc 与nowdoc异同，并各写一示例。

heredoc：

<<<EOD  
Example of string  
spanning multiple lines  
using heredoc syntax.  
EOD;  
nowdoc：  
<<<'EOD'  
Example of string  
spanning multiple lines  
using nowdoc syntax.  
EOD;

## 尽量写出你所知道的魔术变量(Magic Constants)

**\_\_LINE\_\_、\_\_FILE\_\_、 \_\_DIR\_\_、\_\_FUNCTION\_\_、\_\_CLASS\_\_、\_\_METHOD\_\_、\_\_NAMESPACE\_\_**

## 9 .魔术方法

\_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_toString(), \_\_invoke(), \_\_set\_state() 和 \_\_clone()

## $argc 与 $argv 的用途。

$argc — The number of arguments passed to script  
$argv — Array of arguments passed to script

## echo 9%(-2)的结果

**1**

## 在HTTP 1.0中，状态码 401 的含义是（4）；如果返回“找不到文件”的提示，则可用 header 函数，其语句为

无授权。header("HTTP/1.0 404 Not Found").

## XML的XSLT的用法

XSLT是把一种XML文档转换成另一种文档的语言。XPath是在XML文档中定位信息的语言。在转换过程中，XSLT使用XPath来定位源文档中匹配一个或多个预定义模板的部分。当一个匹配被找到后，XSLT将转换这一匹配的部分到结果文档中。源文档中无法匹配模板的部分将在结果文档中不做任何修改。

1. **打开php.ini中的Safe\_mode，会影响哪些函数？至少说出6个**。

system() exec() fopen() mkdir() rmdir() rename() unlink() copy() chgrp() chown() parse\_ini\_file()

mysql服务器所用的用户名必须与调用mysql\_connect()的文件的拥有者用户名相同

1. **PHP的垃圾收集机制是怎样的**

引用计数基本知识

回收周期(Collecting Cycles)

性能方面考虑的因素

gc\_enable(); // Enable Garbage Collector  
var\_dump(gc\_enabled()); // true  
var\_dump(gc\_collect\_cycles()); // # of elements cleaned up  
gc\_disable(); // Disable Garbage Collector

1. **define() 和 const 的比较**

define() 是在运行时定义常量，而 const 是在编译时定义常量。这给了const一个很轻微的速度优势，但达不到值得担心的程度，除非你在建立大型软件。

define() 把常量放在全局范围，虽然你可以在你的常量名称中包含命名空间。这意味着你不能用define()来定义类常量。

define() 允许你在常量名称和常量值中都使用表达式，而 const 则都不允许。这使得define() 灵活得多。

define() 可以在一个 if() 块中被使用，而 const 不能.

# 前端页面

1. **请使用JavaScript写出三种产生一个Image 标签的方法（提示：从方法、对象、HTML角度考虑）**

(1)var img = new Image();  
(2)var img = document.createElementById(“image”)  
(3)img.innerHTML = “<img src=”xxx.jpg” />”

1. **请描述出两点以上XHTML和HTML最显著的区别**

(1)XHTML必须强制指定文档类型DocType，HTML不需要

(2)XHTML所有标签必须闭合，HTML比较随意

1. **在web页面中不论显示器的分辨率的大小是800×600，还是1024×768使得页面都显示在屏幕中间。**
2. **请利用正则表达式验证一个字符串是否为数字串（请用javascript或csharp）。**

function RegExp(nstr){

var REstr = /^\d+$/g;

return REstr.test(nstr);

}

System.Text.RegularExpressions.Regex

reg=new ystem.Text.RegularExpressions.Regex(@"/^\d+$/g ");

System.Text.RegularExpressions.MatchCollection item=reg.Matches(str);

1. **请用HTML在客户端画出一个5行4列的表格，并用Javascript访问表格第1行的2列的内容**

<table id = tableNode>

<tr><td></td><td></td><td></td><td></td></tr>

<tr><td></td><td></td><td></td><td></td></tr>

<tr><td></td><td></td><td></td><td></td></tr>

<tr><td></td><td></td><td></td><td></td></tr>

<tr><td></td><td></td><td></td><td></td></tr>

</table>

tableNode.firstchild.firstchild.childNodes[1].firstchild

1. **请用javascript脚本实现弹出无边框窗口显示a.htm页**

<script laguage="JavaScript">

var win=window.open("PPA.htm","","fullscreen=1")

</script>

1. **请用javascript或csharp设计一个算法，实现将数字转化为字符。 如（0**
2. **A，1**
3. **B，26**
4. **AA）。**

function getKey(ikey){

var quotient = ikey/26;

var residue = ikey%26;

var s = “”;

if(quotient>25)

s = “0”;

else

{

if(quotient>0)

s = String.formCharCode(quotient+65-1);

s = s+String.formCharCode(residue+65);

var st = s.split(‘@’)

if(st[0] != ‘’)

result = st[0];

else

result = st[1];

}

return result;

}

1. **编写代码防止内容被拷贝（javascript）**

<body oncontextmenu="return false" onselectstart="return false">

1. **请介绍css**

CSS 是 Cascading Style Sheet 的缩写。译作「层叠样式表单」。是用于(增强)控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。

1. **请说明Css的调用(或 运用)2种以上的方式，能写出代码更佳**

外部样式表文件调用方式：

<link rel=stylesheet href="http://www.dhtmlet.com/dhtmlet.css" type="text/css">

定义内部样式块对象 (Embedding a Style Block)

<style type="text/css">  
<!--  
body {font: 10pt "Arial"}  
h1 {font: 15pt/17pt "Arial"; font-weight: bold; color: maroon}  
h2 {font: 13pt/15pt "Arial"; font-weight: bold; color: blue}  
p {font: 10pt/12pt "Arial"; color: black}  
-->  
</style>

内联内嵌定义 (Inline Styles)

<p style="margin-left: 0.5in; margin-right:0.5in">这一行被增加了左右的外补丁<p>

1. **说明css的4种以上选择方式，如 \*{ } 全选 。**

通配符选择 \*{}

类型选择（Type Selectors） td { font-size:14px; width:120px; }

属性选择（Attribute Selectors）

E [ attr ] { sRules }

E [ attr = value ] { sRules }

E [ attr ~= value ] { sRules }

E [ attr |= value ] { sRules }

继承选择（Descendant Selectors）

table td { font-size:14px; }

子对象选择（Child Selectors）

body > p { font-size:14px; } /\* 所有作为body的子对象的p对象字体尺寸为14px \*/

ID选择 #ID { sRules }

类选择 E.className { sRules }

分组选择 E1 , E2 , E3 { sRules }

1. **下面css 表示那些选择对象 : link : hover Selector : visited**

a : link设置 a 对象在未被访问前的样式。

: hover设置对象在其鼠标悬停时的样式。

Selector : visited设置 a 对象在其链接地址已被访问过时的样式

# 网络

1. **请描述出OSI七层网络模型的名称，由下到上（可以使用中文描述）**物理层、数据链路层、网络层、传输层、会话层、表示层、应用层
2. **Internet物理地址和IP地址转换采用什么协议？**ARP RARP
3. **Internet采用哪种网络协议？该协议的主要层次结构？**  
   TCP/IP 应用层，传输层，网络层，网络接口层．
4. **说说下面这些这些协议的全称和中文解释（提示：都是工作在应用层）SMTP、POP3、HTTP、FTP、DNS**

SMTP (Simple Mail Transfer Protocol) 简单邮件传输协议  
POP3 (Post Office Protocol 3) 邮局协议第3版  
HTTP (Hypertext Transfer Protocol) 超文本传输协议  
FTP   (File Transfer Protocol）文件传输协议  
DNS   (Domain Name System and Domain Name Service protocol) 域名系统（服务）协议

1. **写出、你能想到的所有HTTP返回状态值，并说明用途**

**HTTP 1\*\*(临时响应)**  
HTTP 100 Continue 初始的请求已经接受，客户应当继续发送请求的其余部分。（HTTP 1.1新）  
HTTP 101 Switching Protocols 服务器将遵从客户的请求转换到另外一种协议  
  
**HTTP 2\*\*(成功响应)**  
200(成功)服务器已成功处理了请求。通常，这表示服务器提供了请求的网页。如果是对您的 robots.txt 文件显示此状态码，则表示 Googlebot 已成功检索到该文件。  
201(已创建)请求成功并且服务器创建了新的资源。  
202(已接受)服务器已接受请求，但尚未处理。  
203(非授权信息)服务器已成功处理了请求，但返回的信息可能来自另一来源。  
204(无内容)服务器成功处理了请求，但没有返回任何内容。  
205(重置内容)服务器成功处理了请求，但没有返回任何内容。与 204 响应不同，此响应要求请求者重置文档视图(例如，清除表单内容以输入新内容)。  
206(部分内容)服务器成功处理了部分 GET 请求。  
  
**HTTP 3\*\*(重定向)**300(多种选择)针对请求，服务器可执行多种操作。服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。  
301(永久移动)请求的网页已永久移动到新位置。服务器返回此响应(对 GET 或 HEAD 请求的响应)时，会自动将请求者转到新位置。您应使用此代码告诉 Googlebot 某个网页或网站已永久移动到新位置。  
302(临时移动)服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来响应以后的请求。此代码与响应 GET 和 HEAD 请求的 301 代码类似，会自动将请求者转到不同的位置，但您不应使用此代码来告诉 Googlebot 某个网页或网站已经移动，因为 Googlebot 会继续抓取原有位置并编制索引。  
303(查看其他位置)请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。对于除 HEAD 之外的所有请求，服务器会自动转到其他位置。  
304(未修改)自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容。  
　　如果网页自请求者上次请求后再也没有更改过，您应将服务器配置为返回此响应(称为 If-Modified-Since HTTP 标头)。服务器可以告诉 Googlebot 自从上次抓取后网页没有变更，进而节省带宽和开销。  
305(使用代理)请求者只能使用代理访问请求的网页。如果服务器返回此响应，还表示请求者应使用代理。  
307(临时重定向)服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来响应以后的请求。此代码与响应 GET 和 HEAD 请求的 301 代码类似，会自动将请求者转到不同的位置，但您不应使用此代码来告诉 Googlebot 某个页面或网站已经移动，因为 Googlebot 会继续抓取原有位置并编制索引。

**HTTP** **(请求错误)**  
HTTP 400 - 请求无效  
HTTP 401.1 - 未授权：登录失败  
HTTP 401.2 - 未授权：服务器配置问题导致登录失败  
HTTP 401.3 - ACL 禁止访问资源  
HTTP 401.4 - 未授权：授权被筛选器拒绝  
HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败  
HTTP 403 - 禁止访问  
HTTP 403 - 对 Internet 服务管理器 (HTML) 的访问仅限于 Localhost  
HTTP 403.1 禁止访问：禁止可执行访问  
HTTP 403.2 - 禁止访问：禁止读访问  
HTTP 403.3 - 禁止访问：禁止写访问  
HTTP 403.4 - 禁止访问：要求 SSL  
HTTP 403.5 - 禁止访问：要求 SSL 128  
HTTP 403.6 - 禁止访问：IP 地址被拒绝  
HTTP 403.7 - 禁止访问：要求客户证书  
HTTP 403.8 - 禁止访问：禁止站点访问  
HTTP 403.9 - 禁止访问：连接的用户过多  
HTTP 403.10 - 禁止访问：配置无效  
HTTP 403.11 - 禁止访问：密码更改  
HTTP 403.12 - 禁止访问：映射器拒绝访问  
HTTP 403.13 - 禁止访问：客户证书已被吊销  
HTTP 403.15 - 禁止访问：客户访问许可过多  
HTTP 403.16 - 禁止访问：客户证书不可信或者无效  
HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效  
HTTP 404.1 - 无法找到 Web 站点  
HTTP 404 - 无法找到文件  
HTTP 405 - 资源被禁止  
HTTP 406 - 无法接受  
HTTP 407 - 要求代理身份验证  
HTTP 410 - 永远

可用  
HTTP 412 - 先决条件失败  
HTTP 414 - 请求 - URI 太长

**HTTP（服务器错误）**  
HTTP 500 - 内部服务器错误  
HTTP 500.100 - 内部服务器错误 - ASP 错误  
HTTP 500-11 服务器关闭  
HTTP 500-12 应用程序重新启动  
HTTP 500-13 - 服务器太忙  
HTTP 500-14 - 应用程序无效  
HTTP 500-15 - 不允许请求 global.asa  
Error 501 - 未实现  
HTTP 502 - 网关错误

1. **请写出HTTP头，并符合以下要求：**　　１）这是一个post请求  
   　　２）目标：<http://www.example.com:8080/test>  
   　　３）POST变量：  
   　　　　username: test  
   　　　　pwd: test2  
   　　　　intro: Hello world!  
   　　４）包含以下COOKIE信息：  
   　　　　cur\_query: you&me

POST /iMito-iM7---MID---Tablet-PC-7--TFT-Touch-Screen---Telechips-TCC8901-800MHZ-256-DDR2-2G--SMQ5057-\_p108909.html HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-silverlight, \*/\*

Referer: http://newsvn.lightinthebox.com/iMito-iM7---MID---Tablet-PC-7--TFT-Touch-Screen---Telechips-TCC8901-800MHZ-256-DDR2-2G--SMQ5057-\_p108909.html

Accept-Language: zh-cn

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)

Host: newsvn.lightinthebox.com

Content-Length: 216

Connection: Keep-Alive

Cache-Control: no-cache

Cookie: \_\_utma=1.1424167058.1277693732.1278038687.1278038736.15; \_\_utmz=1.1277693732.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); \_csoot=1268789507000; \_csuid=Xa34f33f6a1128c; \_\_cust=192.168.12.51.1258451142653861; \_\_gads=ID=4136e7330095a7d3:T=1253671430:S=ALNI\_MZhd-VC9uUntyM57p3LP74bzeQJ4Q; cookie\_test=please\_accept\_for\_session; \_\_utma=145114283.670235613.1277710853.1277710853.1277710853.1; \_\_utmz=145114283.1277710853.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); my\_cust=AAAAAEwGbsco0CCNAytKAg==; customer\_first\_name=YWZh; customer\_email=d3l3aW50ZXIuMjEzQGdtYWlsLmNvbQ%3D%3D; customer\_id=VOENOMUtTMkJIM0xYM1RGNUxKP; zenid=7915d30692805a5ff8193f950381971c; recently\_view=108909\_115325\_56319\_50680\_52854\_100365

product\_score=4&action=review&session\_key=MKG64wazd5NPOVGfvbQ1kLsD1XJIATsp&review\_image=&customer\_name=dfdfd&customer\_email=fffd@dfsdf.com&review\_title=2323&review\_content=1212&privacy\_conditions=1&check\_number=G383c

0. GET /danssion?topnav=1&wvr=5 HTTP/1.1

1. **Accept:**text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
2. **Accept-Charset:**GBK,utf-8;q=0.7,\*;q=0.3
3. **Accept-Encoding:**gzip,deflate,sdch
4. **Accept-Language:**zh-CN,zh;q=0.8
5. **Cache-Control:**max-age=0
6. **Connection:**keep-alive
7. **Cookie:**\_s\_tentry=login.sina.com.cn; Apache=4194834385998.547.1337232955628; ULV=1337232957690:180:12:5:4194834385998.547.1337232955628:1337132204438; NSC\_wjq\_xfjcp.dpn\_w3.6\_w4=ffffffff0941137a45525d5f4f58455e445a4a423660; NSC\_wjq\_xfjcp.dpn\_ipnfqbhf=ffffffff094113a745525d5f4f58455e445a4a423660; \_\_utma=182865017.1676840087.1319894782.1323009573.1342966689.4; \_\_utmc=182865017; ULOGIN\_IMG=13484897605924;
8. **Host:**weibo.com
9. **User-Agent:**Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.57 Safari/537.17

0. HTTP/1.1 200 OK

1. **Cache-Control:**no-cache, must-revalidate
2. **Connection:**close
3. **Content-Encoding:**gzip
4. **Content-Type:**text/html; charset=utf-8
5. **Date:**Tue, 12 Mar 2013 14:30:56 GMT
6. **DPOOL\_HEADER:**alice107
7. **Expires:**Sat, 26 Jul 1997 05:00:00 GMT
8. **P3P:**CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR"
9. **Pragma:**no-cache
10. **Server:**Apache
11. **Set-Cookie:**U\_TRS1=0000007c.7c891de9.513f3c20.a33595c3; path=/; expires=Fri, 10-Mar-23 14:30:56 GMT; domain=.sina.com.cn
12. **Set-Cookie:**U\_TRS2=0000007c.7c9a1de9.513f3c20.76b74cae; path=/; domain=.sina.com.cn
13. **Transfer-Encoding:**chunked
14. **Vary:**Accept-Encoding
15. **php中，保存http返回信息的变量是？请写出返回信息一般包含的内容。**

file\_get\_contents("http://example.com");  
var\_dump($http\_response\_header);

array(9) {

[0]=> string(15) "HTTP/1.1 200 OK"

[1]=> string(35) "Date: Sat, 12 Apr 2008 17:30:38 GMT"

[2]=> string(29) "Server: Apache/2.2.3 (CentOS)"

[3]=> string(44) "Last-Modified: Tue, 15 Nov 2005 13:24:10 GMT"

[4]=> string(27) "ETag: "280100-1b6-80bfd280""

[5]=> string(20) "Accept-Ranges: bytes"

[6]=> string(19) "Content-Length: 438"

[7]=> string(17) "Connection: close"

[8]=> string(38) "Content-Type: text/html; charset=UTF-8"

}

1. **什么是SOAP**

SOAP（Simple Object Access Protocol ）简单对象访问协议是在分散或分布式的环境中交换信息并执行远程过程调用的协议，是一个基于XML的协议。使用SOAP，不用考虑任何特定的传输协议（最常用的还是HTTP协议），可以允许任何类型的对象或代码，在任何平台上，以任何一直语言相互通信。

SOAP 是一种轻量级协议，用于在分散型、分布式环境中交换结构化信息。 SOAP 利用 XML 技术定义一种可扩展的消息处理框架，它提供了一种可通过多种底层协议进行交换的消息结构。 这种框架的设计思想是要独立于任何一种特定的编程模型和其他特定实现的语义。

SOAP 定义了一种方法以便将 XML 消息从 A 点传送到 B 点。 为此，它提供了一种基于 XML 且具有以下特性的消息处理框架：1) 可扩展，2) 可通过多种底层网络协议使用，3) 独立于编程模型。

1. **OSI网络结构的七层模型及其核心思想是什么**

第七层：应用层

* 定义了用于在网络中进行通信和数据传输的接口 - 用户程式；
* 提供标准服务，比如虚拟终端、文件以及任务的传输和处理；

　　第六层：表示层

* 掩盖不同系统间的数据格式的
* 同性；
* 指定独立结构的数据传输格式；
* 数据的编码和解码；加密和解密；压缩和解压缩

　　第五层：会话层

* 管理用户会话和对话；
* 控制用户间逻辑连接的建立和挂断；
* 报告上一层发生的错误

　　第四层：传输层

* 管理网络中端到端的信息传送；
* 通过错误纠正和流控制机制提供可靠且有序的数据包传送；
* 提供面向无连接的数据包的传送；

　　第三层：网络层

* 定义网络设备间如何传输数据；
* 根据唯一的网络设备地址路由数据包；
* 提供流和拥塞控制以防止网络资源的损耗

　　第二层：数据链路层

* 定义操作通信连接的程序；
* 封装数据包为数据帧；
* 监测和纠正数据包传输错误

　　第一层：物理层

* 定义通过网络设备发送数据的物理方式；
* 作为网络媒介和设备间的接口；

定义光学、电气以及机械特性。

# Php7、面向对象与设计模式

## php5中可以自动加载类声明的函数?

function \_\_autoload

## 写出构造函数和析构函数并说明作用。

\_\_construct \_\_destruct

## 抽象类与接口的区别？

## 在B/S模式中，MVN层架构实现的，说说各层之间的关系和这样的好处？

MVC（模型－视图－控制器）

视图(View)代表用户交互界面.

模型(Model)：就是业务流程/状态的处理以及业务规则的制定,业务模型还有一个很重要的模型那就是数据模型.

控制(Controller)可以理解为从用户接收请求, 将模型与视图匹配在一起，共同完成用户的请求.

## 您了解设计模式么？请列出您所知道的设计模式的名称。

GoF（“[四人帮](http://baike.baidu.com/view/22184.htm)”）：

FACTORY、BUILDER、SINGLETON、ADAPTER、BRIDGE、DECORATOR、OBSERVER、STRATEGY

GRASP patterns ： Creator、Export、Controller、Low Coupling、High Cohesion、Polymorphism、Pure Fabrication、Protected Variations

## php5中对象如何自动创建members 和 methods?写出相关magic methods

*\_\_set()* is run when writing data to inaccessible members.

*\_\_get()* is utilized for reading data from inaccessible members.

*\_\_isset()* is triggered by calling [isset()](mk:@MSITStore:E:\environment\php_manual_en.chm::/function.isset.html) or [empty()](mk:@MSITStore:E:\environment\php_manual_en.chm::/function.empty.html) on inaccessible members.

*\_\_unset()* is invoked when [unset()](mk:@MSITStore:E:\environment\php_manual_en.chm::/function.unset.html) is used on inaccessible members.

## 解释以下Magic Methods \_\_sleep \_\_wakeup \_\_toString \_\_invoke \_\_set\_state \_\_clone()\_\_call()\_\_callStatic()

*\_\_sleep* class serialize() 前调用的函数  
*\_\_wakeup* 反之。  
*\_\_toString* allows a class to decide how it will react when it is converted to a string。  
*\_\_invoke* is called when a script tries to call an object as a function.  
*\_\_set\_state* This [static](mk:@MSITStore:E:\environment\php_manual_en.chm::/language.oop5.static.html) method is called for classes exported by [var\_export()](mk:@MSITStore:E:\environment\php_manual_en.chm::/function.var-export.html) since PHP 5.1.0.  
\_\_clone() An object copy is created by using the clone keyword，$copy\_of\_object = clone $object。Once the cloning is complete, if a \_\_clone() method is defined, then the newly created object's \_\_clone() method will be called, to allow any necessary properties that need to be changed.  
*\_\_call()* triggered when invoking inaccessible methods in an object context.

*\_\_callStatic()*is triggered when invoking inaccessible methods in a static context.

## php5中解释一下的关键词：public / protected / private / namespace / final / const / static / abstract / clone / parent / self /

## 写出下面代码的输出：

<?php  
class A {  
    public static function foo() {  
        static::who();  
    }  
  
    public static function who() {  
        echo \_\_CLASS\_\_."\n";  
    }  
}  
  
class B extends A {  
    public static function test() {  
        A::foo();  
        parent::foo();  
        self::foo();  
    }  
  
    public static function who() {  
        echo \_\_CLASS\_\_."\n";  
    }  
}  
class C extends B {  
    public static function who() {  
        echo \_\_CLASS\_\_."\n";  
    }  
}  
  
C::test();  
?>  
A

C

C

## 面向对象语言中，方法重载Overload和方法的重写Overriding的区别。

方法的重写Overriding和重载Overloading是Java多态性的不同表现。重写Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写（Overriding）。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载（Overloading）。 Overloaded的方法是可以改变返回值的类型。

## 什么是反射？

反射。公共语言运行库加载器管理应用程序域。这种管理包括将每个程序集加载到相应的应用程序域以及控制每个程序集中类型层次结构的内存布局。

## 类型声明 与 返回值类型声明

作为参数声明的类型

1.class/interface 类或者接口

2.self 自身引用

3.array 数组

4.callable 可调用的函数或者闭包

5.bool 布尔

6.float 浮点型

7.int 整形

8.string 字符串

//PHP7之前

function sum($a, $b)

{

return $a + $b;

}

//PHP7

function sum(int $a, int $b): int

{

return $a + $b;

}

## null合并运算符

/PHP7之前

$param = isset($\_GET['param']) && !empty($\_GET['param']) ? $\_GET['param'] : '';

//或者

$param = '';

if (isset($\_GET['param']) && !empty($\_GET['param'])) {

$param = $\_GET['param'];

}

//PHP7

$param = $\_GET['param'] ?? '';

相比以前的写法是不是非常的简单，??运算符会根据一个值来判断它是否是存在且不为NULL，如果是则返回第一个数，否则返回第二个数。

## 组合比较符

组合表达式返回的有三个值，大于1、等于0和小于1。

///-1

echo 1 <=> 2;

//0

echo 2 <=> 2;

//1

echo 3.2 <=> 1.4

//-1

echo 'a' <=> 'c';

//0

echo 'b' <=> 'b';

//1

echo 'c' <=> 'b';

## 闭包的绑定 Closure::call()

//PHP7之前

class App {

public function execute() {

echo 'App execute';

}

}

$closure = function() {

echo $this->execute();

};

$newClosure = Closure::bind($closure, new App());

//或者$newClosure = $closure->bindTo(new App());

$newClosure(); //输出App execute

//PHP7

class App {

public function execute() {

echo 'App execute Call';

}

}

$closure = function() {

echo $this->execute();

};

$closure->call(new App());

//输出App execute Call

# 操作系统

1. **如何挂载samba 共享服务器目录？**

sudo mount //192.168.12.5/litb-writable ./share-server/ -t smbfs -o auto,iocharset=utf8

1. **写出四个以上查看linux下文件内容命令**tail more less cat tac head
2. **linux文件名长度不超过：**256
3. **linux 命令**

sudo su 转换为root 工作环境保持原登录用户

sudo su – 转换为root 包含工作环境

1. **用ftp进行文件传输时，有两种模式：**ASCII和binary
2. **某/etc/fstab文件中的某行如下：/dev/had5 /mnt/dosdata msdos defaults,usrquota 1 2请解释其含义。**参考答案:

（1）第一列：将被加载的文件系统名；

（2）第二项：就是您想要将其加入至哪个目录位置，如/home或/，这其实就是在安装时提示的挂入点

（3） 第三项：就是所谓的local filesystem，其包含了以下格式：如ext、ext2、msdos、iso9660、nfs、swap等，或如ext2，可以参见/prco/filesystems说明；  
（4）第四列：您mount时，所要设定的状态，如ro（只读）或defaults（包括了其它参数如rw、suid、exec、auto、nouser、async）。

（5）第五列：提供DUMP功能，在系统DUMP时是否需要BACKUP的标志位，其内定值是0.

（6）第六列：设定此filesystem是否要在开机时做check的动作，除了root的filesystem其必要的check为1之外，其它皆可视需要设定，内定值是0。

1. **如何查看当前Linux系统的状态，如CPU使用，内存使用，负责情况等**

proc文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。用户和应用程序可  
以通过proc得到系统的信息，并可以改变内核的某些参数。由于系统的信息，如进程，是动态改变的，所以用户或应用程序读取proc文件时，proc文件  
系统是动态从系统内核读出所需信息并提交的。它的目录结构如下：  
apm 高级电源管理信息  
cmdline 内核命令行  
Cpuinfo 关于Cpu信息  
Devices 可以用到的设备（块设备/字符设备）  
Dma 使用的DMA通道  
Filesystems 支持的文件系统  
Interrupts 中断的使用  
Ioports I/O端口的使用  
Kcore 内核核心印象  
Kmsg 内核消息  
Ksyms 内核符号表  
Loadavg 负载均衡  
Locks 内核锁  
Meminfo 内存信息  
Misc 杂项  
Modules 加载模块列表  
Mounts 加载的文件系统  
Partitions 系统识别的分区表  
Rtc 实时时钟  
Slabinfo Slab池信息  
Stat 全面统计状态表  
Swaps 对换空间的利用情况  
Version 内核版本  
Uptime 系统正常运行时间

1. **如何查看apache的进程数**ps –elf | grep apache | wc –l
2. **如何统计apache每秒访问数**tail access\_log | awk ‘{print $1,$4}’  
   grep –c `date –d ‘ 3 second ago ‘ +%T ` access\_log
3. **获得访问前10位的ip地址**
4. cat access.log|awk '{print $1}'|sort|uniq -c|sort -nr|head -10
5. cat access.log|awk '{counts[$(11)]+=1}; END {for(url in counts) print counts[url], url}'
6. **iptables 添加一个规则允许192.168.12.180 访问本机3306端口**iptables –I [name] 1 –p tcp –m tcp –dport 3306 –s 192.168.12.180 –j ACCEPT
7. **如何将本地80端口的请求转发到8080端口，本地ip 192.168.16.1 ， 其中网卡eth0**iptables –t nat –A PREROUTING –d 192.168.16.1 –p tcp --dport 80 –j DNAT --to 192.168.16.1:8080  
   iptables –t nat –A PREROUTING –i eth0 –d 192.168.16.1 –p –tcp –m tcp --dport 80 –j REDIRECT --to-ports 8080
8. **Apache服务器的配置文件httpd.conf中有很多内容，请解释如下配置项：  
   （1）MaxKeepAliveRequests 200   
   （2）UserDir public\_html  
   （3）DefaultType text/plain  
   （4）AddLanguare en.en  
   （5）DocumentRoot“/usr/local/httpd/htdocs”  
   （6）AddType application/x-httpd-php.php.php.php4**（1）允许每次连接的最大请求数目，此为200；  
   （2）设定用户放置网页的目录；  
   （3）设置服务器对于不认识的文件类型的预设格式；  
   （4）设置可传送语言的文件给浏览器；  
   （5）该目录为Apache放置网页的地方；  
   （6）服务器选择使用php4。
9. **Linux 下文件有隐藏属性。如何设置文件的隐藏属性 a 和 i 属性 并说明该属性有何含义。如何显示文件隐藏属性。**

chattr +ia \*\*\*

a 该文件只能增加，不能删除。

i 该文件不能被删除、改名、设置连接、无法写入和新增

lsattr \*\*\*

1. **请列举你能想到的Linux / UNIX信号，并说明信号用途**

1 ~ 31的信号为传统UNIX支持的信号，是不可靠信号(非实时的)，编号为32 ~ 63的信号是后来扩充的，称做可靠信号(实时信号)。不可靠信号和可靠信号的区别在于前者不支持排队，可能会造成信号丢失，而后者不会。

1) SIGHUP  
本信号在用户终端连接(正常或非正常)结束时发出, 通常是在终端的控制进程结束时, 通知同一session内的各个作业, 这时它们与控制终端不再关联。  
  
登录Linux时，系统会分配给登录用户一个终端(Session)。在这个终端运行的所有程序，包括前台进程组和后台进程组，一般都属于这个Session。当用户退出Linux登录时，前台进程组和后台有对终端输出的进程将会收到SIGHUP信号。这个信号的默认操作为终止进程，因此前台进程组和后台有终端输出的进程就会中止。不过可以捕获这个信号，比如wget能捕获SIGHUP信号，并忽略它，这样就算退出了Linux登录，wget也能继续下载。  
  
此外，对于与终端脱离关系的守护进程，这个信号用于通知它重新读取配置文件。  
  
2) SIGINT  
程序终止(interrupt)信号, 在用户键入INTR字符(通常是Ctrl-C)时发出，用于通知前台进程组终止进程。  
  
3) SIGQUIT  
和SIGINT类似, 但由QUIT字符(通常是Ctrl-\)来控制. 进程在因收到SIGQUIT退出时会产生core文件, 在这个意义上类似于一个程序错误信号。  
  
4) SIGILL  
执行了非法指令. 通常是因为可执行文件本身出现错误, 或者试图执行数据段. 堆栈溢出时也有可能产生这个信号。  
  
5) SIGTRAP  
由断点指令或其它trap指令产生. 由debugger使用。  
  
6) SIGABRT  
调用abort函数生成的信号。  
  
7) SIGBUS  
非法地址, 包括内存地址对齐(alignment)出错。比如访问一个四个字长的整数, 但其地址不是4的倍数。它与SIGSEGV的区别在于后者是由于对合法存储地址的非法访问触发的(如访问不属于自己存储空间或只读存储空间)。  
  
8）SIGFPE  
在发生致命的算术运算错误时发出. 不仅包括浮点运算错误, 还包括溢出及除数为0等其它所有的算术的错误。  
  
9) SIGKILL  
用来立即结束程序的运行. 本信号不能被阻塞、处理和忽略。如果管理员发现某个进程终止不了，可尝试发送这个信号。  
  
10) SIGUSR1  
留给用户使用  
  
11) SIGSEGV  
试图访问未分配给自己的内存, 或试图往没有写权限的内存地址写数据.  
  
12) SIGUSR2  
留给用户使用  
  
13) SIGPIPE  
管道破裂。这个信号通常在进程间通信产生，比如采用FIFO(管道)通信的两个进程，读管道没打开或者意外终止就往管道写，写进程会收到SIGPIPE信号。此外用Socket通信的两个进程，写进程在写Socket的时候，读进程已经终止。  
  
14) SIGALRM  
时钟定时信号, 计算的是实际的时间或时钟时间. alarm函数使用该信号.  
  
15) SIGTERM  
程序结束(terminate)信号, 与SIGKILL不同的是该信号可以被阻塞和处理。通常用来要求程序自己正常退出，shell命令kill缺省产生这个信号。如果进程终止不了，我们才会尝试SIGKILL。  
  
17) SIGCHLD  
子进程结束时, 父进程会收到这个信号。  
  
如果父进程没有处理这个信号，也没有等待(wait)子进程，子进程虽然终止，但是还会在内核进程表中占有表项，这时的子进程称为僵尸进程。这种情况我们应该避免(父进程或者忽略SIGCHILD信号，或者捕捉它，或者wait它派生的子进程，或者父进程先终止，这时子进程的终止自动由init进程来接管)。  
  
18） SIGCONT  
让一个停止(stopped)的进程继续执行. 本信号不能被阻塞. 可以用一个handler来让程序在由stopped状态变为继续执行时完成特定的工作. 例如, 重新显示提示符  
  
19) SIGSTOP  
停止(stopped)进程的执行. 注意它和terminate以及interrupt的区别:该进程还未结束, 只是暂停执行. 本信号不能被阻塞, 处理或忽略.  
  
20) SIGTSTP  
停止进程的运行, 但该信号可以被处理和忽略. 用户键入SUSP字符时(通常是Ctrl-Z)发出这个信号  
  
21) SIGTTIN  
当后台作业要从用户终端读数据时, 该作业中的所有进程会收到SIGTTIN信号. 缺省时这些进程会停止执行.  
  
22) SIGTTOU  
类似于SIGTTIN, 但在写终端(或修改终端模式)时收到.  
  
23) SIGURG  
有"紧急"数据或out-of-band数据到达socket时产生.  
  
24) SIGXCPU  
超过CPU时间资源限制. 这个限制可以由getrlimit/setrlimit来读取/改变。  
  
25) SIGXFSZ  
当进程企图扩大文件以至于超过文件大小资源限制。  
  
26) SIGVTALRM  
虚拟时钟信号. 类似于SIGALRM, 但是计算的是该进程占用的CPU时间.  
  
27) SIGPROF  
类似于SIGALRM/SIGVTALRM, 但包括该进程用的CPU时间以及系统调用的时间.  
  
28） SIGWINCH  
窗口大小改变时发出.  
  
29) SIGIO  
文件描述符准备就绪, 可以开始进行输入/输出操作.  
  
30) SIGPWR  
Power failure  
  
31) SIGSYS  
非法的系统调用。  
  
在以上列出的信号中，程序不可捕获、阻塞或忽略的信号有：SIGKILL,SIGSTOP  
不能恢复至默认动作的信号有：SIGILL,SIGTRAP  
默认会导致进程流产的信号有：SIGABRT,SIGBUS,SIGFPE,SIGILL,SIGIOT,SIGQUIT,SIGSEGV,SIGTRAP,SIGXCPU,SIGXFSZ  
默认会导致进程退出的信号有：SIGALRM,SIGHUP,SIGINT,SIGKILL,SIGPIPE,SIGPOLL,SIGPROF,SIGSYS,SIGTERM,SIGUSR1,SIGUSR2,SIGVTALRM  
默认会导致进程停止的信号有：SIGSTOP,SIGTSTP,SIGTTIN,SIGTTOU  
默认进程忽略的信号有：SIGCHLD,SIGPWR,SIGURG,SIGWINCH  
  
此外，SIGIO在SVR4是退出，在4.3BSD中是忽略；SIGCONT在进程挂起时是继续，否则是忽略，不能被阻塞。

# 数据库相关题

1. **写出知道几种关系型数据库和关系运算。说出你所知道的三种以上开源数据库的名称。（提示：想想目前国外流行的开源数据库）**MySQL、SQLite、BDB（Berkeley DB）、PostgreSQL、Firebird
2. **数据查询语言DQL**SELECT
3. **数据操纵语言DML**INSERT UPDATE DELETE
4. **数据定义语言DDL**CREATE、DROP、ALTER
5. **数据控制DCL**  
   GRANT ROLLBACK
6. **解释一下事务的含义和特点。**

数据库事务是指作为单个逻辑工作单元执行的一系列操作。数据库事务的ACID（Atomicity、Consistency、Isolation、Durability）属性。(原子性、一致性、隔离性和持久性)

1. **若要在基本表S中增加一列CN（课程名），可用**ALTER TABLE S ADD（CN CHAR（８））
2. **学生关系模式 S（ S＃，Sname，Sex，Age），S的属性分别表示学生的学号、姓名、性别、年龄。要在表S中删除一个属性“年龄”，可选用的SQL语句是**ALTER TABLE S DROP Age
3. **add from 各大IT 公司面试题 93 页面**
4. **索引的原理？索引的代价？索引的类型（即索引的数据结构）？**

根据建立索引的字段建立索引表，存放字段值以及对应记录的物理地址，从而在搜索的时候，根据字段值搜索索引表得到物理地址直接访问记录  
消耗存储空间，和系统处理时间。

类型：B树，位图，函数索引

1. **char / varchar 的区别，及谁的存储速度快？为何？**
2. **写出三种以上MySQL数据库存储引擎的名称**

MyISAM、InnoDB、BDB（Berkeley DB）、Merge、Memory（Heap）、Example、Federated、Archive、CSV、Blackhole、MaxDB 等等十几个引擎

1. **简述MyISAM、InnoDB 的不同**

InnoDB features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Storage limits | 64TB | Transactions | Yes | Locking granularity | Row |
| MVCC | Yes | Geospatial datatype support | Yes | Geospatial indexing support | No |
| B-tree indexes | Yes | Hash indexes | No | Full-text search indexes | No |
| Clustered indexes | Yes | Data caches | Yes | Index caches | Yes |
| Compressed data | Yes[[a](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] | Encrypted data[[**b**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] | Yes | Cluster database support | No |
| Replication support[[**c**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] | Yes | Foreign key support | Yes | Backup / point-in-time recovery[[**d**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] | Yes |
| Query cache support | Yes | Update statistics for data dictionary | Yes |  |  |
| [[a](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] Compressed InnoDB tables are supported only by InnoDB Plugin.  [[b](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] Implemented in the server (via encryption functions), rather than in the storage engine.  [[c](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] Implemented in the server, rather than in the storage engine  [[d](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s06.html)] Implemented in the server, rather than in the storage engine | | | | | |

MyISAM features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Storage limits | 256TB | Transactions | No | Locking granularity | Table |
| MVCC | No | Geospatial datatype support | Yes | Geospatial indexing support | Yes |
| B-tree indexes | Yes | Hash indexes | No | Full-text search indexes | Yes |
| Clustered indexes | No | Data caches | No | Index caches | Yes |
| Compressed data | Yes[[a](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] | Encrypted data[[**b**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] | Yes | Cluster database support | No |
| Replication support[[**c**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] | Yes | Foreign key support | No | Backup / point-in-time recovery[[**d**](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] | Yes |
| Query cache support | Yes | Update statistics for data dictionary | Yes |  |  |
| [[a](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.  [[b](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] Implemented in the server (via encryption functions), rather than in the storage engine.  [[c](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] Implemented in the server, rather than in the storage engine  [[d](mk:@MSITStore:E:\\environment\\MySQL-5%5B1%5D.4-en.chm::/ch13s05.html)] Implemented in the server, rather than in the storage engine | | | | | |

1. **如何优化数据库？**

数据库设计：

合理确定字段类型和长度

选取高效的主键和索引

精简表结构 ,将变长字段的内容拆分到子表中,将记录的状态位集中到一个tinyint字段中

适当采用冗余字段 ,减少连接查询

explain分析SQL语句的效率 ，拆分负责SQL语句

数据库架构：

数据库的配置更改，如增加MySQL的key\_buffer、cache\_buffer、query\_cache

数据库的缓慢sql语句的记录

数据库表的拆分：水平拆分，垂直拆分（静态，代理）

数据库的读写分离

1. [**MyISAM**](http://xiaoyi829.javaeye.com/blog/783654)**和InnoDB的设置**

key\_buffer\_size - 这对MyISAM表来说非常重要。如果只是使用MyISAM表，可以把它设置为可用内存的 30-40%。合理的值取决于索引大小、数据量以及负载。

记住，MyISAM表会使用操作系统的缓存来缓存数据，因此需要留出部分内存给它们，很多情况下数据比索引大多了。尽管如此，需要总是检查是否所有的 key\_buffer 都被利用了。

.MYI 文件只有 1GB，而 key\_buffer 却设置为 4GB 的情况是非常少的。这么做太浪费了。如果你很少使用MyISAM表，那么也保留低于 16-32MB 的 key\_buffer\_size 以适应给予磁盘的临时表索引所需。

innodb\_buffer\_pool\_size - 这对Innodb表来说非常重要。Innodb相比MyISAM表对缓冲更为敏感。MyISAM可以在默认的 key\_buffer\_size 设置下运行的可以，然而Innodb在默认的 innodb\_buffer\_pool\_size 设置下却跟蜗牛似的。由于Innodb把数据和索引都缓存起来，无需留给操作系统太多的内存，因此如果只需要用Innodb的话则可以设置它高达 70-80% 的可用内存。如果你的数据量不大，并且不会暴增，那么无需把 innodb\_buffer\_pool\_size 设置的太大了。

innodb\_additional\_pool\_size - 这个选项对性能影响并不太多，至少在有差不多足够内存可分配的操作系统上是这样。不过如果你仍然想设置为 20MB(或者更大)，因此就需要看一下Innodb其他需要分配的内存有多少。

innodb\_log\_file\_size 在高写入负载尤其是大数据集的情况下很重要。这个值越大则性能相对越高，但是要注意到可能会增加恢复时间。我经常设置为 64-512MB，跟据服务器大小而异。innodb\_log\_buffer\_size 默认的设置在中等强度写入负载以及较短事务的情况下，服务器性能还可以。如果存在更新操作峰值或者负载较大，就应该考虑加大它的值了。如果它的值设置太高了，可能会浪费内存。它每秒都会刷新一次，因此无需设置超过1秒所需的内存空间。通常 8-16MB 就足够了。越小的系统它的值越小。

innodb\_flush\_logs\_at\_trx\_commit 是否为Innodb比MyISAM慢1000倍而头大？看来也许你忘了修改这个参数了。默认值是 1，这意味着每次提交的更新事务（或者每个事务之外的语句）都会刷新到磁盘中，而这相当耗费资源，尤其是没有电池备用缓存时。很多应用程序，尤其是从 MyISAM转变过来的那些，把它的值设置为 2 就可以了，也就是不把日志刷新到磁盘上，而只刷新到操作系统的缓存上。日志仍然会每秒刷新到磁盘中去，因此通常不会丢失每秒1-2次更新的消耗。如果设置为 0 就快很多了，不过也相对不安全了。MySQL服务器崩溃时就会丢失一些事务。设置为 2 指挥丢失刷新到操作系统缓存的那部分事务。

table\_cache - 打开一个表的开销可能很大。例如MyISAM把MYI文件头标志该表正在使用中。你肯定不希望这种操作太频繁，所以通常要加大缓存数量，使得足以最大限度地缓存打开的表。它需要用到操作系统的资源以及内存，对当前的硬件配置来说当然不是什么问题了。如果你有200多个表的话，那么设置为 1024 也许比较合适（每个线程都需要打开表），如果连接数比较大那么就加大它的值。我曾经见过设置为 100,000 的情况。

thread\_cache - 线程的创建和销毁的开销可能很大，因为每个线程的连接/断开都需要。我通常至少设置为 16。如果应用程序中有大量的跳跃并发连接并且 Threads\_Created 的值也比较大，那么我就会加大它的值。它的目的是在通常的操作中无需创建新线程。

query\_cache - 如果你的应用程序有大量读，而且没有应用程序级别的缓存，那么这很有用。不要把它设置太大了，因为想要维护它也需要不少开销，这会导致MySQL变慢。通常设置为 32-512Mb。设置完之后最好是跟踪一段时间，查看是否运行良好。在一定的负载压力下，如果缓存命中率太低了，就启用它。

1. **MySQL数据库基本的三个优化法则是什么，除了增加硬件和带宽？（提示：从服务配置、应用、开发角度考虑）**

（1）系统服务优化，把MySQL的key\_buffer、cache\_buffer、query\_cache等增加容量

（2）给所有经常查询的字段增加适当的索引

（3）优化SQL语句，减少Ditinct、Group、Join等等语句的操作

1. **比较truncate和delete命令？**

两者都可以用来删除表中所有的记录。区别在于：truncate是DDL（data defining language数据定义语言），它移动HWK，不需要rollback segment（处理事务回滚操作）而Delete是DML（data manufacturing language数据操作语言）操作，需要rollback segment（处理事务回滚操作）且花费较长时间。

1. **给出数据的相关约束类型？**

主键约束，外键约束，非空约束，唯一约束，检查约束。

1. **存储过程和函数的区别**

存储过程是编译好的存储在数据库的操作，函数中不能使用临时表,和动态SQL

1. **为管理业务培训信息，建立3个表：**

     S(S#,SN,SD,SA)S#,SN,SD,SA分别代表学号，学员姓名，所属单位，学员年龄

     C(C#,CN)C#,CN分别代表课程编号，课程名称

      SC(S#,C#,G) S#,C#,G分别代表学号，所选的课程编号，学习成绩

    （1）使用标准SQL嵌套语句查询选修课程名称为’税收基础’的学员学号和姓名?

          答案：select s# ,sn from s where S# in(select S# from c,sc where c.c#=sc.c# and cn=’税收基础’)

      (2) 使用标准SQL嵌套语句查询选修课程编号为’C2’的学员姓名和所属单位?

答：select sn,sd from s,sc where s.s#=sc.s# and sc.c#=’c2’

      (3) 使用标准SQL嵌套语句查询不选修课程编号为’C5’的学员姓名和所属单位?

答：select sn,sd from s where s# not in(select s# from sc where c#=’c5’)

       (4)查询选修了课程的学员人数

答：select 学员人数=count(distinct s#) from sc

       (5) 查询选修课程超过5门的学员学号和所属单位?

答：select sn,sd from s where s# in(select s# from sc group by s# having count(distinct c#)>5)

# Mysql

## ****一、MySQL中索引的语法****

**创建索引**

在创建表的时候添加索引

CREATE TABLE mytable(

ID INT NOT NULL,

username VARCHAR(16) NOT NULL,

INDEX [indexName] (username(length))

);

在创建表以后添加索引

ALTER TABLE my\_table ADD [UNIQUE] INDEX index\_name(column\_name);

或者

CREATE INDEX index\_name ON my\_table(column\_name);

**注意：**

1、索引需要占用**磁盘空间**，因此在创建索引时要考虑到磁盘空间是否足够

2、创建索引时需要**对表加锁**，因此实际操作中需要在业务空闲期间进行

**根据索引查询**

具体查询：

SELECT \* FROM table\_name WHERE column\_1=column\_2;(为column\_1建立了索引)

或者模糊查询

SELECT \* FROM table\_name WHERE column\_1 LIKE '%三'

SELECT \* FROM table\_name WHERE column\_1 LIKE '三%'

SELECT \* FROM table\_name WHERE column\_1 LIKE '%三%'

SELECT \* FROM table\_name WHERE column\_1 LIKE '\_好\_'

如果要表示在字符串中既有A又有B，那么查询语句为：

SELECT \* FROM table\_name WHERE column\_1 LIKE '%A%' AND column\_1 LIKE '%B%';

SELECT \* FROM table\_name WHERE column\_1 LIKE '[张李王]三'; //表示column\_1中有匹配张三、李三、王三的都可以

SELECT \* FROM table\_name WHERE column\_1 LIKE '[^张李王]三'; //表示column\_1中有匹配除了张三、李三、王三的其他三都可以

//在模糊查询中，%表示任意0个或多个字符；\_表示任意单个字符（有且仅有），通常用来限制字符串长度;[]表示其中的某一个字符；[^]表示除了其中的字符的所有字符

或者在全文索引中模糊查询

SELECT \* FROM table\_name WHERE MATCH(content) AGAINST('word1','word2',...);

**删除索引**

DROP INDEX my\_index ON tablename；

或者

ALTER TABLE table\_name DROP INDEX index\_name;

**查看表中的索引**

SHOW INDEX FROM tablename

**查看查询语句使用索引的情况**

//explain 加查询语句

explain SELECT \* FROM table\_name WHERE column\_1='123';

## ****二、索引的优缺点****

优势：可以快速检索，减少I/O次数，加快检索速度；根据索引分组和排序，可以加快分组和排序；

劣势：索引本身也是表，因此会占用存储空间，一般来说，索引表占用的空间的数据表的1.5倍；索引表的维护和创建需要时间成本，这个成本随着数据量增大而增大；构建索引会降低数据表的修改操作（删除，添加，修改）的效率，因为在修改数据表的同时还需要修改索引表；

## ****三、索引的分类****

**常见的索引类型有：主键索引、唯一索引、普通索引、全文索引、组合索引**

1、主键索引：即主索引，根据主键pk\_clolum（length）建立索引，**不允许重复，不允许空值**；

ALTER TABLE 'table\_name' ADD PRIMARY KEY pk\_index('col')；

2、唯一索引：用来建立索引的列的值必须是**唯一的，允许空值**

ALTER TABLE 'table\_name' ADD UNIQUE index\_name('col')；

3、普通索引：用表中的普通列构建的索引，没有任何限制

ALTER TABLE 'table\_name' ADD INDEX index\_name('col')；

4、全文索引：用大文本对象的列构建的索引（下一部分会讲解）

ALTER TABLE 'table\_name' ADD FULLTEXT INDEX ft\_index('col')；

5、组合索引：用多个列组合构建的索引，这多个列中的值不允许有空值

ALTER TABLE 'table\_name' ADD INDEX index\_name('col1','col2','col3')；

\*遵循“最左前缀”原则，把最常用作为检索或排序的列放在最左，依次递减，组合索引相当于建立了col1,col1col2,col1col2col3三个索引，而col2或者col3是不能使用索引的。

\*在使用组合索引的时候可能因为列名长度过长而导致索引的key太大，导致效率降低，在允许的情况下，可以只取col1和col2的前几个字符作为索引

ALTER TABLE 'table\_name' ADD INDEX index\_name(col1(4),col2（3))；

表示使用col1的前4个字符和col2的前3个字符作为索引

## ****四、索引的实现原理****

MySQL支持诸多存储引擎，而各种存储引擎对索引的支持也各不相同，因此MySQL数据库支持多种索引类型，如**BTree索引，B+Tree索引，哈希索引，全文索引**等等**，**

**1、哈希索引：**

只有memory（内存）存储引擎支持哈希索引，哈希索引用索引列的值计算该值的hashCode，然后在hashCode相应的位置存执该值所在行数据的物理位置，因为使用散列算法，因此访问速度非常快，但是一个值只能对应一个hashCode，而且是散列的分布方式，因此哈希索引不支持范围查找和排序的功能。

**2、全文索引：**

FULLTEXT（全文）索引，仅可用于MyISAM和InnoDB，针对较大的数据，生成全文索引非常的消耗时间和空间。对于文本的大对象，或者较大的CHAR类型的数据，如果使用普通索引，那么匹配文本前几个字符还是可行的，但是想要匹配文本中间的几个单词，那么就要使用LIKE %word%来匹配，这样需要很长的时间来处理，响应时间会大大增加，这种情况，就可使用时FULLTEXT索引了，在生成FULLTEXT索引时，会为文本生成一份单词的清单，在索引时及根据这个单词的清单来索引。FULLTEXT可以在创建表的时候创建，也可以在需要的时候用ALTER或者CREATE INDEX来添加：

//创建表的时候添加FULLTEXT索引

CTREATE TABLE my\_table(

id INT(10) PRIMARY KEY,

name VARCHAR(10) NOT NULL,

my\_text TEXT,

FULLTEXT(my\_text)

)ENGINE=MyISAM DEFAULT CHARSET=utf8;

//创建表以后，在需要的时候添加FULLTEXT索引

ALTER TABLE my\_table ADD FULLTEXT INDEX ft\_index(column\_name);

全文索引的查询也有自己特殊的语法，而不能使用LIKE %查询字符串%的模糊查询语法

SELECT \* FROM table\_name MATCH(ft\_index) AGAINST('查询字符串');

注意：

\*对于较大的数据集，把数据添加到一个没有FULLTEXT索引的表，然后添加FULLTEXT索引的速度比把数据添加到一个已经有FULLTEXT索引的表快。

\*5.6版本前的MySQL自带的全文索引只能用于MyISAM存储引擎，如果是其它数据引擎，那么全文索引不会生效。5.6版本之后InnoDB存储引擎开始支持全文索引

\*在MySQL中，全文索引支队英文有用，目前对中文还不支持。5.7版本之后通过使用ngram插件开始支持中文。

\*在MySQL中，如果检索的字符串太短则无法检索得到预期的结果，检索的字符串长度至少为4字节，此外，如果检索的字符包括停止词，那么停止词会被忽略。

**3、BTree索引和B+Tree索引**

* BTree索引

BTree是平衡搜索多叉树，设树的度为2d（d>1），高度为h，那么BTree要满足以一下条件：

* 每个叶子结点的高度一样，等于h；
* 每个非叶子结点由n-1个key和n个指针point组成，其中d<=n<=2d,key和point相互间隔，结点两端一定是key；
* 叶子结点指针都为null；
* 非叶子结点的key都是[key,data]二元组，其中key表示作为索引的键，data为键值所在行的数据；

BTree的结构如下：

图片包含 游戏机, 截图, 钟表

描述已自动生成

image

在BTree的机构下，就可以使用二分查找的查找方式，查找复杂度为h\*log(n)，一般来说树的高度是很小的，一般为3左右，因此BTree是一个非常高效的查找结构。

* **B+Tree索引**

B+Tree是BTree的一个变种，设d为树的度数，h为树的高度，B+Tree和BTree的不同主要在于：

* B+Tree中的非叶子结点不存储数据，只存储键值；
* B+Tree的叶子结点没有指针，所有键值都会出现在叶子结点上，且key存储的键值对应data数据的物理地址；
* B+Tree的每个非叶子节点由**n个键值key**和**n个指针point**组成；

B+Tree的结构如下：

图片包含 游戏机, 钟表

描述已自动生成

B+Tree对比BTree的优点：

1、磁盘读写代价更低

一般来说B+Tree比BTree更适合实现外存的索引结构，因为存储引擎的设计专家巧妙的利用了外存（磁盘）的存储结构，即磁盘的最小存储单位是扇区（sector），而操作系统的块（block）通常是整数倍的sector，操作系统以页（page）为单位管理内存，一页（page）通常默认为4K，数据库的页通常设置为操作系统页的整数倍，因此索引结构的节点被设计为一个页的大小，然后利用外存的“预读取”原则，每次读取的时候，把整个节点的数据读取到内存中，然后在内存中查找，已知内存的读取速度是外存读取I/O速度的几百倍，那么提升查找速度的关键就在于尽可能少的磁盘I/O，那么可以知道，每个节点中的key个数越多，那么树的高度越小，需要I/O的次数越少，因此一般来说B+Tree比BTree更快，因为B+Tree的非叶节点中不存储data，就可以存储更多的key。

2、查询速度更稳定

由于B+Tree非叶子节点不存储数据（data），因此所有的数据都要查询至叶子节点，而叶子节点的高度都是相同的，因此所有数据的查询速度都是一样的。

更多操作系统内容参考：

硬盘结构

扇区、块、簇、页的区别

操作系统层优化（进阶，初学不用看）

* 带顺序索引的B+TREE

很多存储引擎在B+Tree的基础上进行了优化，添加了指向相邻叶节点的指针，形成了带有顺序访问指针的B+Tree，这样做是为了提高区间查找的效率，只要找到第一个值那么就可以顺序的查找后面的值。

B+Tree的结构如下：

图片包含 游戏机, 物体, 钟表

描述已自动生成

**聚簇索引和非聚簇索引**

分析了MySQL的索引结构的实现原理，然后我们来看看具体的存储引擎怎么实现索引结构的，MySQL中最常见的两种存储引擎分别是MyISAM和InnoDB，分别实现了非聚簇索引和聚簇索引。

聚簇索引的解释是:聚簇索引的顺序就是数据的物理存储顺序

非聚簇索引的解释是:索引顺序与数据物理排列顺序无关

（这样说起来并不好理解，让人摸不着头脑，清继续看下文，并在插图下方对上述两句话有解释）

首先要介绍几个概念，在索引的分类中，我们可以按照索引的键是否为主键来分为“主索引”和“辅助索引”，使用主键键值建立的索引称为“主索引”，其它的称为“辅助索引”。因此主索引只能有一个，辅助索引可以有很多个。

**MyISAM——非聚簇索引**

MyISAM存储引擎采用的是非聚簇索引，非聚簇索引的主索引和辅助索引几乎是一样的，只是主索引不允许重复，不允许空值，他们的叶子结点的key都存储指向键值对应的数据的物理地址。

非聚簇索引的数据表和索引表是分开存储的。

非聚簇索引中的数据是根据数据的插入顺序保存。因此非聚簇索引更适合单个数据的查询。插入顺序不受键值影响。

只有在MyISAM中才能使用FULLTEXT索引。(mysql5.6以后innoDB也支持全文索引)

最开始我一直不懂既然非聚簇索引的主索引和辅助索引指向相同的内容，为什么还要辅助索引这个东西呢，后来才明白索引不就是用来查询的吗，用在那些地方呢，不就是WHERE和ORDER BY 语句后面吗，那么如果查询的条件不是主键怎么办呢，这个时候就需要辅助索引了。

**InnoDB——聚簇索引**

聚簇索引的主索引的叶子结点存储的是键值对应的数据本身，辅助索引的叶子结点存储的是键值对应的数据的主键键值。因此主键的值长度越小越好，类型越简单越好。

聚簇索引的数据和主键索引存储在一起。

聚簇索引的数据是根据主键的顺序保存。因此适合按主键索引的区间查找，可以有更少的磁盘I/O，加快查询速度。但是也是因为这个原因，聚簇索引的插入顺序最好按照主键单调的顺序插入，否则会频繁的引起页分裂，严重影响性能。

在InnoDB中，如果只需要查找索引的列，就尽量不要加入其它的列，这样会提高查询效率。

使用主索引的时候，更适合使用聚簇索引，因为聚簇索引只需要查找一次，而非聚簇索引在查到数据的地址后，还要进行一次I/O查找数据。

\*因为聚簇辅助索引存储的是主键的键值，因此可以在数据行移动或者页分裂的时候降低成本，因为这时不用维护辅助索引。但是由于主索引存储的是数据本身，因此聚簇索引会占用更多的空间。

\*聚簇索引在插入新数据的时候比非聚簇索引慢很多，因为插入新数据时需要检测主键是否重复，这需要遍历主索引的所有叶节点，而非聚簇索引的叶节点保存的是数据地址，占用空间少，因此分布集中，查询的时候I/O更少，但聚簇索引的主索引中存储的是数据本身，数据占用空间大，分布范围更大，可能占用好多的扇区，因此需要更多次I/O才能遍历完毕。

下图可以形象的说明聚簇索引和非聚簇索引的区别

图片包含 游戏机, 文字

描述已自动生成

从上图中可以看到聚簇索引的辅助索引的叶子节点的data存储的是主键的值，主索引的叶子节点的data存储的是数据本身，也就是说数据和索引存储在一起，并且索引查询到的地方就是数据（data）本身，那么索引的顺序和数据本身的顺序就是相同的；

而非聚簇索引的主索引和辅助索引的叶子节点的data都是存储的数据的物理地址，也就是说索引和数据并不是存储在一起的，数据的顺序和索引的顺序并没有任何关系，也就是索引顺序与数据物理排列顺序无关。

此外MyISAM和innoDB的区别总结如下：

image

总结如下：

* InnoDB 支持事务，支持行级别锁定，支持 B-tree、Full-text 等索引，不支持 Hash 索引；
* MyISAM 不支持事务，支持表级别锁定，支持 B-tree、Full-text 等索引，不支持 Hash 索引；

此外，Memory 不支持事务，支持表级别锁定，支持 B-tree、Hash 等索引，不支持 Full-text 索引；

## ****五、索引的使用策略****

**什么时候要使用索引？**

* 主键自动建立唯一索引；
* 经常作为查询条件在WHERE或者ORDER BY 语句中出现的列要建立索引；
* 作为排序的列要建立索引；
* 查询中与其他表关联的字段，外键关系建立索引
* 高并发条件下倾向组合索引；
* 用于聚合函数的列可以建立索引，例如使用了max(column\_1)或者count(column\_1)时的column\_1就需要建立索引

**什么时候不要使用索引？**

* 经常增删改的列不要建立索引；
* 有大量重复的列不建立索引；
* 表记录太少不要建立索引。只有当数据库里已经有了足够多的测试数据时，它的性能测试结果才有实际参考价值。如果在测试数据库里只有几百条数据记录，它们往往在执行完第一条查询命令之后就被全部加载到内存里，这将使后续的查询命令都执行得非常快--不管有没有使用索引。只有当数据库里的记录超过了1000条、数据总量也超过了MySQL服务器上的内存总量时，数据库的性能测试结果才有意义。

**索引失效的情况：**

* 在组合索引中不能有列的值为NULL，如果有，那么这一列对组合索引就是无效的。
* 在一个SELECT语句中，索引只能使用一次，如果在WHERE中使用了，那么在ORDER BY中就不要用了。
* LIKE操作中，'%aaa%'不会使用索引，也就是索引会失效，但是‘aaa%’可以使用索引。
* 在索引的列上使用表达式或者函数会使索引失效，例如：select \* from users where YEAR(adddate)<2007，将在每个行上进行运算，这将导致索引失效而进行全表扫描，因此我们可以改成：select \* from users where adddate<’2007-01-01′。其它通配符同样，也就是说，在查询条件中使用正则表达式时，只有在搜索模板的第一个字符不是通配符的情况下才能使用索引。
* 在查询条件中使用不等于，包括<符号、>符号和！=会导致索引失效。特别的是如果对主键索引使用！=则不会使索引失效，如果对主键索引或者整数类型的索引使用<符号或者>符号不会使索引失效。（经erwkjrfhjwkdb同学提醒，不等于，包括<符号、>符号和！，如果占总记录的比例很小的话，也不会失效）
* 在查询条件中使用IS NULL或者IS NOT NULL会导致索引失效。
* 字符串不加单引号会导致索引失效。更准确的说是类型不一致会导致失效，比如字段email是字符串类型的，使用WHERE email=99999 则会导致失败，应该改为WHERE email='99999'。
* 在查询条件中使用OR连接多个条件会导致索引失效，除非OR链接的每个条件都加上索引，这时应该改为两次查询，然后用UNION ALL连接起来。
* 如果排序的字段使用了索引，那么select的字段也要是索引字段，否则索引失效。特别的是如果排序的是主键索引则select \* 也不会导致索引失效。
* 尽量不要包括多列排序，如果一定要，最好为这队列构建组合索引；

## ****六、索引的优化****

1、最左前缀

索引的最左前缀和和B+Tree中的“最左前缀原理”有关，举例来说就是如果设置了组合索引<col1,col2,col3>那么以下3中情况可以使用索引：col1，<col1,col2>，<col1,col2,col3>，其它的列，比如<col2,col3>，<col1,col3>，col2，col3等等都是不能使用索引的。

根据最左前缀原则，我们一般把排序分组频率最高的列放在最左边，以此类推。

2、带索引的模糊查询优化

在上面已经提到，使用LIKE进行模糊查询的时候，'%aaa%'不会使用索引，也就是索引会失效。如果是这种情况，只能使用全文索引来进行优化（上文有讲到）。

3、为检索的条件构建全文索引，然后使用

SELECT \* FROM tablename MATCH(index\_colum) ANGAINST(‘word’);

4、使用短索引

对串列进行索引，如果可能应该指定一个前缀长度。例如，如果有一个CHAR(255)的 列，如果在前10 个或20 个字符内，多数值是惟一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作

作者：Java程序猿阿谷  
链接：https://www.jianshu.com/p/c82148473235  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# 数据结构和算法

1. **假设执行语句S的时间为O(1)，则执行下列程序短的时间为（B）**

for(i=1;i<=n;i++)

for(j=i;j<=n;j++)

S;

O(n2)

1. **设有98个已排序列元素，采用二分法查找时，最大比较次数是（）**

7

1. **n个数冒泡排序算法的时间复杂度是**

n\*n

1. **写出下列算法的时间复杂度。**

冒泡排序；

选择排序；

插入排序；

快速排序；

堆排序；

归并排序；

1. **每一棵树都能唯一地转换为它所对应的二叉树，树的这种二叉树表示对树的运算带来很大的好处。遍历（周游）是树形结构的一种重要运算，二叉树的基本组成部分是：根（N）、左子树（L）和右子树（R）。因而二叉树的遍历次序有六种。最常用的是三种：前序法（即按（A）次序），后序法（即按（B）次序）和中序法（也称对称序法，即按（C）次序）。这三种方法相互之间的关联。若已知一棵二叉树的前序序列是BEFCGDH，中序序列是FEBGCHD，则它的后序序列必是（D），而且可得该二叉树所表示的树的先根次序序列是（B）。**

**A、B、C：1、RLN 2、RNL 3、LRN 4、LNR 5、NLR 6、NRL**

**D、E：1、EFGHBCD 2、FEGHDCB 3、BCDEFGH 4、EFBGCHD 5、BEFCGDH 6、FEGBHDC**

# 编程题

1. **实现一个字符串截取的函数，类似于substr，必须能够截取中文这种多字节编码。假设每个中文也是一个字符，普通的数字、符号、字母也是一个字符。（提示：GB编码的中文字符高位范围是 0x81-0xFE ）**

/\*\*  
\* 截取字符串子串 （1）(GBK)  
\*  
\* @param string $str 原始字符串  
\* @param int $len 需要截取字符串的长度  
\* @return string 返回截取到的字符串  
\*/  
function GBSubstr($str, $len){  
$count = 0;  
for($i=0; $i<strlen($str); $i++){  
if($count == $len) break;  
if(preg\_match("/[\x80-\xff]/", substr($str, $i, 1))) ++$i;  
++$count;         
}  
return substr($str, 0, $i);  
}

/\*\*  
\* 截取字符串子串函数（2）（GB)  
\*   
\* @param string $src 源字符串  
\* @param int $start 开始截取的位置  
\* @param int $length 需要截取字符串的长度  
\* @return string 返回截取的字符串  
\*/

function GBSubstr2($src, $start=0, $length=0){  
$suffix="";  
$len = strlen($src);  
if ( $len <= $length ) return $src;

$cut\_length = 0;  
for( $idx = 0; $idx<$length; $idx++){   
$char\_value = ord($src[$idx]);   
if ( $char\_value < 0x80 || ( $char\_value & 0x40 ) )  
$cut\_length++;  
else  
$cut\_length = $cut\_length + 3;   
}   
$curstr = substr($src, 0, $cut\_length) ;  
preg\_match('/^([\x00-\x7f]|.{3})\*/', $curstr, $result);  
return   $result[0];  
}

/\*\*  
\* 中文字符串截取（3）（支持GB2312/GBK/UTF-8/BIG5）  
\*  
\* @param string $str 要截取的字串  
\* @param int $start 截取起始位置  
\* @param int $length 截取长度  
\* @param string $charset 字符串的字符集，包括有 utf-8|gb2312|gbk|big5 编码  
\* @param bool $suffix 是否加尾缀  
\* @return string 返回接续字符串的结果  
\*/  
function CSubstr($str, $start=0, $length, $charset="gbk", $suffix=false){  
if(function\_exists("mb\_substr")){  
return mb\_substr($str, $start, $length, $charset);  
}  
$re['utf-8'] = "/[\x01-\x7f]|[\xc2-\xdf][\x80-\xbf]|[\xe0-\xef][\x80-\xbf]{2}|[\xf0-\xff][\x80-\xbf]{3}/";  
$re['gb2312'] = "/[\x01-\x7f]|[\xb0-\xf7][\xa0-\xfe]/";  
$re['gbk'] = "/[\x01-\x7f]|[\x81-\xfe][\x40-\xfe]/";  
$re['big5'] = "/[\x01-\x7f]|[\x81-\xfe]([\x40-\x7e]|\xa1-\xfe])/";

preg\_match\_all($re[$charset], $str, $match);  
$slice = join("", array\_slice($match[0], $start, $length));

if($suffix) {  
return $slice ."…";  
}  
return $slice;  
}

1. **有n个人围成一圈，顺序排号。从第一个人开始报数（从1到3报数），凡报到3的人退出圈子，问最后留下的是原来第几号的那位。用你熟悉的语言写出代码。**

#include "stdio.h"   
#include "conio.h"   
#define nmax 50   
main()   
{   
  int i,k,m,n,num[nmax],\*p;   
  printf("please input the total of numbers:");   
  scanf("%d",&n);   
  p=num;   
  for(i=0;i<n;i++)   
    \*(p+i)=i+1;   
  i=0;   
  k=0;   
  m=0;   
  while(m<n-1)   
  {   
    if(\*(p+i)!=0) k++;   
    if(k==3)   
    {   
      \*(p+i)=0;   
      k=0;   
      m++;   
    }   
    i++;   
    if(i==n) i=0;   
  }   
  while(\*p==0) p++;   
  printf("%d is left\n",\*p);   
  getch();   
}

#include <iostream>   
using namespace std;   
int main()   
{   
int i,flag=0,n=17,m=0;//flag为报数，m为退出的人数，n为每轮报数的人数   
int a[17];   
for(i=0;i<17;i++)   
a[i]=i+1;   
 i=0;   
while (n>1)   
{   
  
if(a[i]==0) flag+=0;//当a[i]为0 时，不报数   
if(a[i]!=0) flag+=1;   
if(flag%3==0)   
{   
a[i]=0;   
 n--;   
m++;   
}   
i++;   
if(i==m+n) i=0; //每一轮报完数后，i从零开始，继续报数   
}   
while(a[i]==0) i++;   
cout<<"最后剩下的那个人的编号为："<<i+1<<endl;   
return 0;   
}

1. **出匹配邮箱地址和URL的两个正则表达式。类似下面的：**

邮箱地址：user\_name.first@example.com.cn  
URL地址：http://www.example.com.cn/user\_profile.php?uid=100   
（提示：使用标准的正则表达式，就是PHP中preg\_\* 类的正则处理函数能够解析的正则）  
邮箱：/\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*/  
URL：/^http:\/\/[\w]+\.[\w]+[\S]\*/

1. **写一个遍历指定目录下所有子目录和子文件的函数（提示：可以使用递归的方法）**

/\*\*  
\* 递归的遍历一个指定目录  
\*/  
function dir\_all ( $path ) {   
$handler = opendir($path);   
while (false!==($tmp = readdir($handler))) {   
if(is\_dir( "$path/$tmp" )) {   
if ($tmp=="." | $tmp=="..") continue;   
echo $tmp."<br>\n";   
dir\_all ("$path/$tmp");   
} else {  
echo $tmp ."<br>\n";   
}   
}   
}

1. **用PHP写一段代码，实现不使用第3个变量，交换$a、$b的值，$a、$b的初始值自己定。**

$a = "abcd";  
$b = "1234";  
echo "初始化时 a=$a,b=$b<br>";  
$a = $a . $b;  
$b = strlen( $b );  
$b = substr( $a, 0, (strlen($a) - $b ) );  
$a = substr( $a, strlen($b) );  
echo "交换后 a=$a,b=$b<br>";

1. **用最有效率的方法算出2乘以8等於几?**

2<<3

1. **编写类String的构造函数、析构函数和赋值函数**

**已知类String的原型为：**

**class String**

**{**

**public:**

**String(const char \*str = NULL); // 普通构造函数**

**String(const String &other); // 拷贝构造函数**

**~ String(void); // 析构函数**

**String & operate =(const String &other); // 赋值函数**

**private:**

**char \*m\_data; // 用于保存字符串**

**};**

**请编写String的上述4个函数。**

1. **打印一个N\*N的方阵，N为每边字符的个数（ 3〈N〈20 ），要求最外层为"X"，第二层为"Y"，从第三层起每层依次打印数字0，1，2，3，...**

例子：当N =5，打印出下面的图形：

X X X X X

X Y Y Y X

X Y 0 Y X

X Y Y Y X

X X X X X

## 请使用一句话将类似“1910-09-09”的日期各式转换为“一九一零年九月九日”这样的各式。(需要有通用性，是不是一句话不是关键，但不允许使用if\switch\while\for等语句)

<?php

function convertNumberToChinese($str){

//先将-替换成年月日

$pattern='/(\d+)-(0\*)([1-9]+)-(0\*)([1-9]+)/';

$replace='${1}年${3}月${5}日';

$str= preg\_replace($pattern,$replace,$str);

//再将数字替换成汉字

$patterns=array("/0/","/1/","/2/","/3/","/4/","/5/","/6/","/7/","/8/","/9/");

$replaces=array("零","一","二","三","四","五","六","七","八","九");

$re = preg\_replace($patterns,$replaces,$str);

return $re;

}

$str = '1910-09-09';

echo convertNumberToChinese($str)

?>

# 综合题

1. **使用过Vim编辑器吗？如果使用过，如何在vim里切分多个可视化窗口，包括横排和纵排。（本题选作）**

横：split（sp）  
纵：vsplit（vsp）

1. **了解MVC模式吗？请说明MVC模式并写出三种以上目前PHP流行的MVC框架名称（不区分大小写）**

模型(Model),视图(View)和控制Controller)。 MVC模式的目的就是实现Web系统的职能分工。 Model层实现系统中的业务逻辑，通常可以用JavaBean或EJB来实现。 View层用于与用户的交互，通常用JSP来实现。 Controller层是Model与View之间沟通的桥梁，它可以分派用户的请求并选择恰当的视图以用于显示，同时它也可以解释用户的输入并将它们映射为模型层可执行的操作。  
FleaPHP、Zend Framework、CakePHP、Symfony、Plite、php.MVC、PHP on Trax、Smutty 等等

1. **根粗细不一的绳子有A,B两端,从A端烧到B端的时间是一个小时,从B端烧到A端也是一个小时 .但从中间烧到A端或是B端不是半个小时. 给你两跟绳子和一个打火机 怎么样得到45分钟?15分**

两根绳子同时点，只不过，一根两头一起点，一根只点一个头，当两头点的烧玩时，花掉30分钟，再把另一根绳子的另一头也点燃，这样它烧玩花掉的时间就是15分钟，两个时间加起来就是45分钟

1. **五个人来自不同地方，住不同房子，养不同动物，吸不同牌子香烟，喝不同饮料，喜欢不同食物。根据以下线索确定谁是养猫的人。**

（1）红房子在蓝房子的右边，白房子的左边（不一定紧邻）

（2）黄房子的主人来自香港，而且他的房子不在最左边。

（3）爱吃比萨饼的人住在爱喝矿泉水的人的隔壁。

（4）来自北京的人爱喝茅台，住在来自上海的人的隔壁。

（5）吸希尔顿香烟的人住在养马的人右边隔壁。

（6）爱喝啤酒的人也爱吃鸡。

（7）绿房子的人养狗。

（8）爱吃面条的人住在养蛇的人的隔壁。

（9）来自天津的人的邻居（紧邻）一个爱吃牛肉，另一个来自 成都。

（10）养鱼的人住在最右边的房子里。

（11）吸万宝路香烟的人住在吸希尔顿香烟的人和吸“555”香烟的人的中间（紧邻）

（12）红房子的人爱喝茶。

（13）爱喝葡萄酒的人住在爱吃豆腐的人的右边隔壁。

（14）吸红塔山香烟的人既不住在吸健牌香烟的人的隔壁，也不与来自上海的人相邻 。

（15）来自上海的人住在左数第二间房子里。

（16）爱喝矿泉水的人住在最中间的房子里。

（17）爱吃面条的人也爱喝葡萄酒。

（18）吸“555”香烟的人比吸希尔顿香烟的人住的\*右。

1. **如果六千，六百，六表示成6606，那么十一千，十一百，十一表示成什么？**
2. **-1,2,7,28,,126请问28和126中间那个数是什么？为什么？**

应该是4^3-1=63

规律是n^3-1(当n为偶数0，2，4)

n^3+1(当n为奇数1，3，5)

答案：63

1. **某一密码仅使用K、L、M、N、O共5个字母，密码中的单词从左向右排列，密码单词必须遵循如下规则：**

（1） 密码单词的最小长度是两个字母，可以相同，也可以不同

（2） K不可能是单词的第一个字母

（3） 如果L出现，则出现次数不止一次

（4） M不能使最后一个也不能是倒数第二个字母

（5） K出现，则N就一定出现

（6） O如果是最后一个字母，则L一定出现

问题一：下列哪一个字母可以放在LO中的O后面，形成一个3个字母的密码单词？B

A) K B)L C) M D) N

问题二：如果能得到的字母是K、L、M，那么能够形成的两个字母长的密码单词的总数是多少？A

A)1个 B)3个 C)6个 D）9个

问题三：下列哪一个是单词密码？C

A) KLLN B) LOML C) MLLO D)NMKO

问题四：3个相同的字母可以构成的不同密码单词的总数是多少？B

A) 1个 B)2个 C) 3个 D) 4个

问题五：单词密码MMLLOKN可以通过下列变成另一个密码单词，除了： C

A) 用N代替每个L B)用O代替每一个M C)用O代替N D)把O移到N的左边

**8.有 4 头猪，这 4 头猪的重量都是整千克数，把这 4 头猪两两合称体重，共称 5 次，分别 是 99、113、125、130、144，其中有两头猪没有一起称过。那么，这两头猪中重量较重那头 有多重?**

ab+cd=ac+bd=ad+bc(ab 指 a 与 b 的体重和)明显 99+144=113+130=125+x，可以看出， 少掉的那个数是:118。不失一般性，ab+ac(cd+bd)=2a2d=62 即 ad=31 或 bc=31 即某两头猪 的体重之差为 31，并且这两头猪要么和为 118，要么两头猪都不是和为 118 的那两头猪。而 两个数的和与差的奇偶性是相同的，所以可以看出，必定是 b 与 c 之外的两头猪的体重之差 为 31

**9.巧分遗产。**

**有一个人得了绝症，不久就离开了人世。这个人生前有 70000 元的遗产，他死前他的妻 子已经怀孕了。在遗嘱中这人说，如果他的妻子生下的是儿子的话，女人所得的遗产将是她 儿子的一半，如果是女儿的话她的遗产就是女儿的二倍。结果女人生下的是双胞胎，一儿一 女。这下子律师为难了。恰在这时一个高中生说了一个方法，便轻松的解决了这个难题。你 知道这个高中生是怎么分的吗**

女儿 10000，母亲 20000，儿子 40000。设母亲得到 X 元，则儿子得到 2X，女儿得到 X/2。2X+X+X/2=70000。最后求得女儿 10000，母亲 20000，儿子 40000

93.薪水难题。

有两个人在一家工地做工，由于一个是学徒，一个是技工，所以他们的薪水是不一样的。 技工的薪水比学徒的薪水多 20 美元，但两人的薪水之差是 21 美元。你觉得他俩的薪水各是 多少?

假设技工和学徒的比较标准是以 1 美元为准的。那么技工的薪水是 20 美元 50 美分， 学徒的薪水是 50 美分。与 1 美元相比，技工的薪水就是正值，学徒的就是负值，二者之差就 是 21 美元，而从实际来讲技工的薪水比学徒的高 20 美元