long pathconf( const char\* path,int name );

返回配置文件的限制值，是与文件或目录相关联的运行时限制。path参数是你想得到限制值的路径，name是想得到限制值的名称，name的取值主要有以下几个取值：

|  |  |  |
| --- | --- | --- |
| 限制名 | 说明 | name参数 |
| FILESIZEBITS | 在指定目录中允许的普通文件最大长度所需的最少位数 | \_PC\_FILESIZEBITS |
| LINK\_MAX | 文件链接数的最大值 | \_PC\_LINK\_MAX |
| MAX\_CANON | 终端规范输入队列的最大字节数 | \_PC\_MAX\_CANON |
| MAX\_INPUT | 终端输入队列可用空间的字节数 | \_PC\_MAX\_INPUT |
| NAME\_MAX | 文件名的最大字节数 | \_PC\_NAME\_MAX |
| PATH\_MAX | 相对路径名的最大字节数，包括null | \_PC\_PATH\_MAX |
| PIPE\_BUF | 能原子的写到管道的最大字节数 | \_PC\_PIPE\_BUF |
| SYMLINK\_MAX | 符号链接中的字节数 | \_PC\_SYMLINK\_MAX |

该函数在错误的时候返回-1，并且设置相应的errno值。

函数原型：char \*getcwd( char \*buffer, int maxlen );

功 能：获取当前工作目录

参数说明：getcwd()会将当前工作目录的[绝对路径](https://baike.baidu.com/item/%E7%BB%9D%E5%AF%B9%E8%B7%AF%E5%BE%84)复制到参数buffer所指的内存空间中,参数maxlen为buffer的空间大小。

返 回 值：成功则返回当前工作目录，失败返回 **FALSE**。

1汉字=2字节  
1字节（[Byte](https://www.baidu.com/s?wd=Byte&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v))＝8字位＝8个[二进制数](https://www.baidu.com/s?wd=%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)  
1字位([bit](https://www.baidu.com/s?wd=bit&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v))＝1个[二进制数](https://www.baidu.com/s?wd=%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)

1字节等于8个字位（8个二进制数）

1B=8b  
1KB 等于1024个字节

1[KB](https://www.baidu.com/s?wd=KB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)=1024B  
1[MB](https://www.baidu.com/s?wd=MB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)=1024[KB](https://www.baidu.com/s?wd=KB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)  
1[GB](https://www.baidu.com/s?wd=GB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)=1024[MB](https://www.baidu.com/s?wd=MB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)  
  
硬件商标准：   
1[GB](https://www.baidu.com/s?wd=GB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)=1000[MB](https://www.baidu.com/s?wd=MB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)   
1MB=1000[KB](https://www.baidu.com/s?wd=KB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)   
1KB=1000B   
  
通常情况下，把B称为字节、b称为字位、KB称为[千字节](https://www.baidu.com/s?wd=%E5%8D%83%E5%AD%97%E8%8A%82&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)、MB称为[兆字节](https://www.baidu.com/s?wd=%E5%85%86%E5%AD%97%E8%8A%82&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)、[GB](https://www.baidu.com/s?wd=GB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPvRznj-Bm1-9uhDvPWD10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmLPHfvPj6v)称为吉字节。

**函数名:** getenv

**功 能:** 从环境中取字符串,获取[环境变量](https://baike.baidu.com/item/%E7%8E%AF%E5%A2%83%E5%8F%98%E9%87%8F)的值

**头文件:**[stdlib.h](https://baike.baidu.com/item/stdlib.h)

**用 法:**char \*getenv(char \*envvar);

**函数说明:getenv()用来取得参数envvar环境变量的内容。参数envvar为环境变量的名称，如果该变量存在则会返回指向该内容的**[**指针**](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)**。环境变量的格式为envvar=value。getenv函数的返回值存储在一个全局二维数组里，当你再次使用getenv函数时不用担心会覆盖上次的调用结果。**

**返回值:**执行成功则返回指向该内容的指针，找不到符合的[环境变量](https://baike.baidu.com/item/%E7%8E%AF%E5%A2%83%E5%8F%98%E9%87%8F)名称则返回NULL。如果变量存在但无关联值，它将运行成功并返回一个[空字符串](https://baike.baidu.com/item/%E7%A9%BA%E5%AD%97%E7%AC%A6%E4%B8%B2)，即该字符的第一个字节是null。

**函数定义**

#include <stdio.h>

FILE \* popen(const char \*command , const char \*type );

int pclose(FILE \*stream);

**函数说明**

　　popen()函数通过创建一个管道，调用fork()产生一个子进程，执行一个shell以运行命令来开启一个进程。这个管道必须由pclose()函数关闭，而不是fclose()函数。pclose()函数关闭标准I/O流，等待命令执行结束，然后返回shell的终止状态。如果shell不能被执行，则pclose()返回的终止状态与shell已执行exit一样。

　　type参数只能是读或者写中的一种，得到的返回值（标准I/O流）也具有和type相应的只读或只写类型。如果type是"r"则文件指针连接到command的标准输出；如果type是"w"则文件指针连接到command的标准输入。

　　command参数是一个指向以NULL结束的shell命令字符串的指针。这行命令将被传到bin/sh并使用-c标志，shell将执行这个命令。

　　popen()的返回值是个标准I/O流，必须由pclose来终止。前面提到这个流是单向的（只能用于读或写）。向这个流写内容相当于写入该命令的标准输入，命令的标准输出和调用popen()的进程相同；与之相反的，从流中读数据相当于读取命令的标准输出，命令的标准输入和调用popen()的进程相同。

**返回值**

　　如果调用fork()或pipe()失败，或者不能分配内存将返回NULL，否则返回标准I/O流。popen()没有为内存分配失败设置errno值。如果调用fork()或pipe()时出现错误，errno被设为相应的错误类型。如果type参数不合法，errno将返回EINVAL。

附上一个例子：

[复制代码](javascript:void(0);)

//execute shell command  
//执行一个shell命令，输出结果逐行存储在resvec中，并返回行数

int32\_t myexec(const char \*cmd, vector<string> &resvec) {

resvec.clear();

FILE \*pp = popen(cmd, "r"); //建立管道

if (!pp) {

return -1;

}  
 char tmp[1024]; //设置一个合适的长度，以存储每一行输出

while (fgets(tmp, sizeof(tmp), pp) != NULL) {

if (tmp[strlen(tmp) - 1] == '\n') {

tmp[strlen(tmp) - 1] = '\0'; //去除换行符

}

resvec.push\_back(tmp);

}

pclose(pp); //关闭管道

return resvec.size();

}

#### Strtok()函数详解：

  该函数包含在**"string.h"**头文件中   
**函数原型：**

1. char\* strtok (char\* str,constchar\* delimiters );

**函数功能：**   
  切割字符串，将str切分成一个个子串   
**函数参数：**   
  str：在第一次被调用的时间str是传入需要被切割字符串的首地址；在后面调用的时间传入NULL。   
  delimiters：表示切割字符串（字符串中每个字符都会 当作分割符）。   
**函数返回值：**   
  当s中的字符查找到末尾时，返回NULL;   
  如果查不到delimiter所标示的字符，则返回当前strtok的字符串的指针。

包含文件：[string.h](http://baike.baidu.com/view/1728868.htm)

函数名: strstr

函数原型：

     extern char \*strstr(char \*str1, const char \*str2);

语法：

\* strstr(str1,str2)

str1: 被查找目标　string expression to search.

str2: 要查找对象　The string expression to find.

返回值：若str2是str1的子串，则返回str2在str1的首次出现的地址；如果str2不是str1的子串，则返回NULL。

int parseIndex(const char\* pStr)

{

int nodeIndex = 0;

//char \*nodeName = "list[5]"; //只可读，不可写 指针指向常量字符串"list[5]"

//char nodeName[] = "list[5]"; //可读写 nodeName数组重新开辟空间存贮"list[5]"

char str[20] = {0};

strcpy\_s(str,strlen(pStr)+1,pStr);

char \*pLeftSquare = strstr(str, "[");

char \*pRightSquare = strstr(str, "]");

if(pLeftSquare && pRightSquare && pRightSquare - pLeftSquare > 1)

{

// nodeName contains '[' and ']', get the inner string as index

\*pLeftSquare = NULL; // trim the [] start

pLeftSquare++;

\*pRightSquare = NULL; // trim the [] end

nodeIndex = atoi(pLeftSquare); // convert to integer

if(nodeIndex < 1)

{

throw "Index cannot be less than 1";

}

}

cout<<"string:"<<pStr<<",Index:"<<nodeIndex<<endl;

return nodeIndex;

}

在读文件的时候往往需要遍历文件夹，python的os.path包含了很多文件、文件夹操作的方法。下面列出：

os.path.abspath(path) #返回绝对路径

os.path.basename(path) #返回文件名

os.path.commonprefix(list) #返回多个路径中，所有path共有的最长的路径。

os.path.dirname(path) #返回文件路径

os.path.exists(path)  #路径存在则返回True,路径损坏返回False

os.path.lexists  #路径存在则返回True,路径损坏也返回True

os.path.expanduser(path)  #把path中包含的"~"和"~user"转换成用户目录

os.path.expandvars(path)  #根据环境变量的值替换path中包含的”$name”和”${name}”

os.path.getatime(path)  #返回最后一次进入此path的时间。

os.path.getmtime(path)  #返回在此path下最后一次修改的时间。

os.path.getctime(path)  #返回path的大小

os.path.getsize(path)  #返回文件大小，如果文件不存在就返回错误

os.path.isabs(path)  #判断是否为绝对路径

os.path.isfile(path)  #判断路径是否为文件

os.path.isdir(path)  #判断路径是否为目录

os.path.islink(path)  #判断路径是否为链接

os.path.ismount(path)  #判断路径是否为挂载点（）

os.path.join(path1[, path2[, ...]])  #把目录和文件名合成一个路径

os.path.normcase(path)  #转换path的大小写和斜杠

os.path.normpath(path)  #规范path字符串形式

os.path.realpath(path)  #返回path的真实路径

os.path.relpath(path[, start])  #从start开始计算相对路径

os.path.samefile(path1, path2)  #判断目录或文件是否相同

os.path.sameopenfile(fp1, fp2)  #判断fp1和fp2是否指向同一文件

os.path.samestat(stat1, stat2)  #判断stat tuple stat1和stat2是否指向同一个文件

os.path.split(path)  #把路径分割成dirname和basename，返回一个元组

os.path.splitdrive(path)   #一般用在windows下，返回驱动器名和路径组成的元组

os.path.splitext(path)  #分割路径，返回路径名和文件扩展名的元组

os.path.splitunc(path)  #把路径分割为加载点与文件

os.path.walk(path, visit, arg)  #遍历path，进入每个目录都调用visit函数，visit函数必须有3个参数(arg, dirname, names)，dirname表示当前目录的目录名，names代表当前目录下的所有文件名，args则为walk的第三个参数

os.path.supports\_unicode\_filenames  #设置是否支持unicode路径名

下面给出一种遍历的实现：

1 rootdir = 'F:\data'

2 list = os.listdir(rootdir) #列出文件夹下所有的目录与文件

3 for i in range(0,len(list)):

4 path = os.path.join(rootdir,list[i])

5 if os.path.isfile(path):

6 #你想对文件的操作

**grep pattern [file...]  
(1)grep 搜索字符串 [filename]  
(2)grep 正则表达式 [filename]**

[options]主要参数：  
－c：只输出匹配行的计数。  
－I：不区分大 小写(只适用于单字符)。  
－h：查询多文件时不显示文件名。  
－l：查询多文件时只输出包含匹配字符的文件名。  
－n：显示匹配行及 行号。  
－s：不显示不存在或无匹配文本的错误信息。  
－v：显示不包含匹配文本的所有行。  
pattern正则表达式主要参数：  
\： 忽略正则表达式中特殊字符的原有含义。  
^：匹配正则表达式的开始行。  
$: 匹配正则表达式的结束行。  
\<：从匹配正则表达 式的行开始。  
\>：到匹配正则表达式的行结束。  
[ ]：单个字符，如[A]即A符合要求 。  
[ - ]：范围，如[A-Z]，即A、B、C一直到Z都符合要求 。  
。：所有的单个字符。  
\* ：有字符，长度可以为0。

grep –n xfabReqCarrierCurrentFABJudge \*

find ./ -print|xargs grep xfabReqCarrierCurrentFABJudge

查找文件名不区分大小写

find ./ -name "\*"|grep -i "xfabReqCarrierCurrentFABJudge.cpp"

数据库select语句 date类型转char

select to\_char(datetargetlate, 'yyyyMMdd HH24:mi:ss') from XsiteWorkOrder WHERE workordernumber='P0000002'

chmod -R xxx \*

锁表查询

SELECT object\_name, machine, s.sid, s.serial#

FROM gv$locked\_object l, dba\_objects o, gv$session s

WHERE l.object\_id　= o.object\_id

AND l.session\_id = s.sid;

锁表解锁

（gv$session.sid, gv$session.serial #）

ALTER system kill session '1623, 2581';

Oracle union 命令：

上下两个select 的字段个数必须一致，相对应的字段类型必须一致，数据会自动去重.

Eg:

select lot.PhysicalLocation EqpId, carr.carriercategory carriercategory, lot.Appid CarrierId, decode(carr.iscarrierOut, 'T', 'WaitForCarrierOut','') LoadState, '' TxnTime

from FabLotCarrierExt carr, fwlot lot

where carr.LotRef= lot.sysid and lot.appid='T4'

union

select sysid,NAME , CLASSID ,TABLENAME,OWNER

from fwclassinfo

where classid ='00003190'

delete from fablistrule where sysid='000039c0.ca650a0a.58be7b4e.00002f9d.322'

update fablistrule set LENGTH='40,40,40,40,40' WHERE sysid ='000039c0.ca650a0a.58be7b4e.00002f9d.323'

insert into fablistrule(sysid, RULENAME,LISTITEMS,LENGTH,HEADER,FWTIMESTAMP) values('000039c0.ca650a0a.58be7b4e.00002f9d.321','LISTCARRIERINBUFFER','lot.Appid, lot.PhysicalLocation, carr.State','40,40,40','CarrierId,LOADEDFOUPID,LOADFOUPSTATUS',0)

decode用法：最后一个字段是默认值，即前面的条件不满足则选择最后一个值

select lot.PHYSICALSTATUS , lot.PhysicalLocation EqpId, carr.carriercategory carriercategory, lot.Appid CarrierId, decode(carr.iscarrierOut, 'T', 'WaitForCarrierOut','') LoadState, '' TxnTime,

carr.sysid

from FabLotCarrierExt carr, fwlot lot

where carr.LotRef= lot.sysid and lot.appid='T4'

查询数据库中是否存在一个字段

select \*   
from user\_tab\_columns   
where column\_name like '%SHAREDEQPSYNCSTATUS%'

查询数据库中是否存在一张表

select \* from cat where table\_name like ‘%RESER%’

***oracle 序列***

1、创建、删除

create sequence seq\_newsId

increment by 1

start with 1

maxvalue 999999999;

得到序列的SQL语句

select seq\_newsid.nextval from sys.dual;

删除序列的SQL

DROP SEQUENCE seq\_newsId；

2、说明

序列是一数据库对象，利用它可生成唯一的整数。一般使用序列自动地生成主码值。一个序列的值是由特别的Oracle程序自动生成，因而序列避免了在运用层实现序列而引起的性能瓶颈。Oracle序列允许同时生成多个序列号，而每一个序列号是唯一的。 当一个序列号生成时，序列是递增，独立于事务的提交或回滚。容许设计缺省序列，不需指定任何子句。该序列为上升序列，由1开始，增量为1，没有上限。

    1） 建立序列命令

    CREATE SEQUENCE [user.]sequence\_name

    [increment by n]

    [start with n]

    [maxvalue n | nomaxvalue]

    [minvalue n | nominvalue];

    INCREMENT BY： 指定序列号之间的间隔，该值可为正的或负的整数，但不可为0。序列为升序。忽略该子句时，缺省值为1。

    START WITH：指定生成的第一个序列号。在升序时，序列可从比最小值大的值开始，缺省值为序列的最小值。对于降序，序列可由比最大值小的值开始，缺省值为序列的最大值。

    MAXVALUE：指定序列可生成的最大值。

    NOMAXVALUE：为升序指定最大值为1027，为降序指定最大值为-1。

    MINVALUE：指定序列的最小值。

    NOMINVALUE：为升序指定最小值为1。为降序指定最小值为-1026。

    2） 更改序列命令

    ALTERSEQUENCE [user.]sequence\_name

    [INCREMENT BY n]

    [MAXVALUE n| NOMAXVALUE ]

    [MINVALUE n | NOMINVALUE]；

    修改序列可以：

    1) 修改未来序列值的增量。

    2) 设置或撤消最小值或最大值。

    3) 转变缓冲序列的数目。

    4) 指定序列号是否是有序。

    5) 删除序列命令

    DROP SEQUENCE [user.]sequence\_name；

    从数据库中删除一序列。

   树立一个序列号的语句：

   CREATE SEQUENCE EXAM\_NO\_SEQ

      START WITH 1484

      MAXVALUE 9999999999

      MINVALUE 1

      CYCLE

      CACHE 20

      NOORDER;

left join :左连接，返回左表中所有的记录以及右表中连接字段相等的记录。

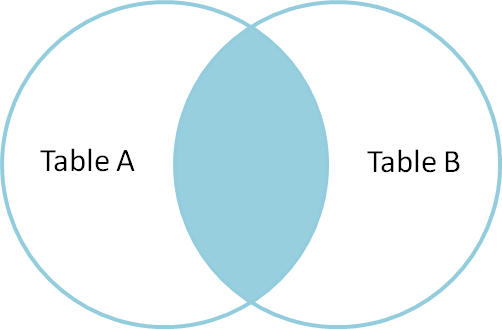
right join :右连接，返回右表中所有的记录以及左表中连接字段相等的记录。

inner join: 内连接，又叫等值连接，只返回两个表中连接字段相等的行。

full join:外连接，返回两个表中的行：left join + right join。

cross join:结果是笛卡尔积，就是第一个表的行数乘以第二个表的行数。

注意：INNER JOIN可以用使用简写JOIN方式



EXISTS用于检查子查询是否至少会返回一行数据，该子查询实际上并不返回任何数据，而是返回值True或False

如果我们需要将两个select语句的结果作为一个整体显示出来，我们就需要用到union或者union all关键字。

union(或称为联合)的作用是将多个结果合并在一起显示出来。

union和union all的区别是,union会自动压缩多个结果集合中的重复结果，而union all则将所有的结果全部显示出来，不管是不是重复。

with as 相当于虚拟视图。相当于建立了一张临时表

with as短语，也叫做子查询部分(subquery factoring)，可以让你做很多事情，定义一个sql片断，该sql片断会被整个sql语句所用到。有的时候，是为了让sql语句的可读性更高些，也有可能是在union all的不同部分，作为提供数据的部分

[**with as**](http://www.tieguanyin168.com/index.php/with-as-1191.html)**语法**  
–针对一个别名  
with tmp as (select \* from tb\_name)

–针对多个别名  
with  
   tmp as (select \* from tb\_name),  
   tmp2 as (select \* from tb\_name2),  
   tmp3 as (select \* from tb\_name3),

Oracle中的不等于号：

在Oracle中，   
  
<>   
!=   
~=   
^=   
  
都是不等于号的意思.

记住：null只能通过is null或者is not null来判断，其它操作符与null操作都是false.

 select \* from test where nvl(name,'xx')<>'xn' 来查询name字段不等于'xn'的记录.

或者

 select \* from test where instr(concat(name,'xx'),'xn') = 0 来查询name字段不等于'xn'的记录。

1.extern关键字解释:

(1)extern置于变量或者函数前，以标示变量或者函数的定义在别的文件中，提示编译器遇到此变量和函数时在其他模块中寻找其定义。

(2)当它与"C"一起连用时，如: extern "C" void fun(int a, int b);则告诉编译器在编译fun这个函数名时按着C的规则去翻译相应的函数名而不是C++的.

static 关键字

表示静态的变量，分配内存的时候, 存储在静态区,不存储在栈上面. static修饰全局变量的作用域只能是本身的编译单元（内部链接,参见 [存储持续性、作用域和链接性](http://www.cnblogs.com/yyxt/p/4011893.html)），也就是说它的“全局”只对本编译单元有效，其他编译单元则看不到它，**一般定义static全局变量时，都把它放在原文件中而不是头文件，这样就不会给其他模块造成不必要的信息污染**

2. C++中的一个空类的sizeof()大小是1，是被编译器安插进去的一个char，这样可以使得这个class的两个objects在内存中配置独一无二的地址。class, struct都有对齐补齐原则.

3. class, struct都有对齐补齐三条原则：  
**第一条：第一个成员的首地址为0  
第二条：每个成员的首地址是自身大小的整数倍  
       第二条补充：以4字节对齐为例，如果自身大小大于4字节，都以4字节整数倍为基准对齐。  
第三条：最后以结构总体对齐。  
        第三条补充：以4字节对齐为例，取结构体中最大成员类型倍数，如果超过4字节，都以4字节整数倍为基准对齐。（其中这一条还有个名字叫：“补齐”，补齐的目的就是多个结构变量挨着摆放的时候也满足对齐的要求。）**

4. 引用与指针有什么区别?  
1)引用必须被初始化，指针不必。(const变量也必须在一开始定义的时候就完成初始化)  
2)引用初始化以后不能被改变，指针可以改变所指的对象。(也即引用初始化完成之后,就永远指向初始化时指定的对象;但指针可以动态修改指向,使之指向其他对象.)  
3)不存在指向空值的引用，但是存在指向空值的指针。

5. strcpy和memcpy主要有以下4方面的区别:  
1、复制的内容不同。strcpy只能复制字符串，而memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。  
2、复制的方法不同。strcpy不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy则是根据其第3个参数决定复制的长度。  
3、用途不同。通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy

4、执行效率不同,memcpy 执行效率高,strcpy 次之

6. main 函数执行之前/之后还会执行什么代码?

main 函数之前执行:

一些全局变量、对象和静态变量、对象的空间分配和赋初值就是在执行main函数之前;

main函数执行完后，还要去执行一些诸如释放空间、释放资源使用权等操作

全局对象的析构函数会在main函数之后执行；

用atexit注册的函数也会在main之后执行。

7. 谈谈你对拷贝构造函数和赋值运算符的认识  
拷贝构造函数和赋值运算符重载有以下两个不同之处:  
(1)拷贝构造函数生成新的类对象,而赋值运算符不能。  
(2)由于拷贝构造函数是直接构造一个新的类对象,所以在初始化这个对象之前不用检验源对象是否和新建对象相同。而赋值运算符则需要这个操作,另外赋值运算中如果原来的对象中有内存分配要先把内存释放掉.  
注意:当有类中有指针类型的成员变量时,一定要重写拷贝构造函数和赋值运算符,不要使用默认的。

8. 用 C++设计一个不能被继承的类

类的构造函数和析构函数声明为private则改基类不能被继承，因为每当子类对象构造时首先调用父类的构造函数，再调用自己的构造函数.

9. 简述类成员函数的重写、重载和隐藏的区别

(1)重写和重载主要有以下几点不同。

范围的区别:被重写的和重写的函数在两个类(基类和派生类)中,而重载和被重载的函数在同一个类中。

参数的区别:被重写函数和重写函数的参数列表一定相同,而被重载函数和重载函数的参数列表一定不同。

virtual 的区别:重写的基类中被重写的函数必须要有 virtual 修饰,而重载函数和被重载函数可以被virtual 修饰,也可以没有。

10.谈谈你对面向对象的认识  
面向对象可以理解成对待每一个问题,都是首先要确定这个问题由几个部分组成,而每一个部分其实就是一个对象。然后再分别设计这些对象,最后得到整个程序。传统的程序设计多是基于功能的思想来进行考虑和设计的,而面向对象的程序设计则是基于对象的角度来考虑问题。这样做能够使得程序更加的简洁清晰。

11. 简述多态实现的原理  
编译器发现一个类中有虚函数,便会立即为此类生成虚函数表 vtable。虚函数表的各表项为指向对应虚函数的指针。编译器还会在此类中隐含插入一个指针 vptr(对 vc 编译器来说,它插在类的第一个位置上)指向虚函数表。调用此类的构造函数时,在类的构造函数中,编译器会隐含执行 vptr 与 vtable 的关联代码,将 vptr 指向对应的 vtable,将类与此类的 vtable 联系了起来。另外在调用类的构造函数时,  
指向基础类的指针此时已经变成指向具体的类的 this 指针,这样依靠此 this 指针即可得到正确的 vtable。如此才能真正与函数体进行连接,这就是动态联编,实现多态的基本原理。

12. :链表和数组有什么区别  
数组和链表有以下几点不同:  
(1)存储形式:数组是一块连续的空间,声明时就要确定长度。链表是一块可不连续的动态空间,长度可变,每个结点要保存相邻结点指针。  
(2)数据查找:数组的线性查找速度快,查找操作直接使用偏移地址。链表需要按顺序检索结点,效率低。  
(3)数据插入或删除:链表可以快速插入和删除结点,而数组则可能需要大量数据移动。  
(4)越界问题:链表不存在越界问题,数组有越界问题。  
说明:在选择数组或链表数据结构时,一定要根据实际需要进行选择。数组便于查询,链表便于插入删除。数组节省空间但是长度固定,链表虽然变长但是占了更多的存储空间。

13. windows vc6.0编译器会下类的成员函数最多可以有1158个(包括static 和 普通成员函数)，否则报编译错误fatal error C1067: compiler limit : debug information module size exceeded

14. create table classes(

id number(9) not null primary key,

classname varchar2(40) not null,

title varchar2(10)

)

desc classes;

select \* from classes;

insert into classes values(1,'gaoxinyizhong','number one');

insert into classes values(2,'gaoxinerzhong','number two');

update classes set address='xian' where id =1;

alter table classes add address varchar2(125);

alter table classes modify address varchar2(45);

alter table classes drop column address;

alter table classes add (test1 varchar2(33),test2 varchar2(22));

alter table classes modify (test1 varchar2(55),test2 varchar2(66));

alter table classes drop (test1 ,test2 );

15.float类型与零值的比较

**const float EP = 0.000001;   
 if ((x >= - EP) && (x <= EP)** )

16. sizeof 和 strlen 的区别  
(1) sizeof 是一个操作符,strlen 是库函数。  
(2) 参数的不同.sizeof 的参数可以是数据的类型,也可以是变量,而 strlen 只能以结尾为‘\0‘的字符串作参数。  
(3) 编译器在编译时就计算出了 sizeof 的结果。而 strlen 函数必须在运行时才能计算出来。并且 sizeof计算的是数据类型占内存的大小,而 strlen 计算的是字符串实际的长度。  
(4) 数组做 sizeof 的参数不退化,传递给 strlen 就退化为指针了。

17. C中的 malloc 和C++中的 new 有什么区别  
(1)new、delete 是操作符,可以重载,只能在 C++中使用。  
(2)malloc、free 是函数, C、C++中都可以使用。  
(3)new 可以调用对象的构造函数,对应的 delete 调用相应的析构函数。  
(4)malloc 仅仅分配内存,free 仅仅回收内存,并不执行构造和析构函数  
(5)new、delete 返回的是某种数据类型指针,malloc、free 返回的是 void 指针。

18. C++中不能重载的运算符只有5个：

.  (成员访问运算符)  
.\*  (成员指针访问运算符)  
::  (域运算符)  
sizeof  (长度运算符)  
?:  (条件运算符）  
  
前两个运算符不能重载是为了保证访问成员的功能不能被改变，域运算符和sizeof 运算符的运算对象是类型而不是变量或一般表达式，不具备重载的特征。

19. 编码实现某一变量某位清 0 或置 1  
给定一个整型变量 a,写两段代码,第一个设置 a 的 bit 3,第二个清 a 的 bit 3,在以上两个操作中,要保持其他位不变。

static int a;

a |= (0x1<<2);//将 a 第 3 位置 1

 a &= ~(0x1<<2);  //将 a 第 3 位清零

20.dbx调试方法

cc -g -o hello checkmem.c

dbx –C hello

打开内存检测

Check –memuse

Check –access

Chaeck –leaks

Check –all

显示内存泄漏信息:showleaks

显示内存使用情况:showmemuse

打断点:stop at checkmem.c:36 //在checkmem.c文件的36行打断点

执行:run

下一步:next

步入调用:step

21. const指针 与指向const的指针的区别

（1）指向const的指针：指针指向的内容是不能被修改的，但是指针的指向可以改变

const int\* p;

int const \*p;

（2）const指针：它的意思是指针本身的值是不能被修改的(类似于引用)，但是指针指向的值可以改变

 int\* const p=一个地址; (因为指针本身的值是不能被修改的所以它必须被初始化）

(3) 指向const的const指针

const int\* const p = 一个地址

22.六大设计模式

(1) 单一职责原则:

一个类只负责一个功能领域中的相应职责

(2) 里氏替换原则:

所有引用基类（父类）的地方必须能透明地使用其子类的对象。

(3) 接口隔离原则:

使用多个专门的接口，而不使用单一的总接口，即客户端不应该依赖那些它不需要的接口。

(4) 最少知道原则(迪米特原则):

一个软件实体应当尽可能少地与其他实体发生相互作用

(5) 依赖倒置原则:

抽象不应该依赖于细节，细节应当依赖于抽象。换言之，要针对接口编程，而不是针对实现编程。

(6) 开闭原则:

对扩展开放，对修改关闭

DBX：

设置断点：

stop at filename:n

stop in function

如果函数 draw 定义在几个不同的类中，要在每个函数中置入断点，请键入：

stop inmember draw

要在于类 shape 中定义的所有成员函数中设置断点，请键入：

stop inclass shape

要在于类形状中定义的所有成员函数中以及从类继承的函数中都设置断点，请键入：

stop inclass shape –recurse

要在带重载名称 （相同名、不同类型或参数数量不同）的非成员函数(sort函数)中设置多个断点，  
请使用 stop infunction 命令。

stop infunction sort

要在指定变量的值更改时停止程序执行，请键入：

stop change variable

要在条件语句的求值为真时停止程序执行，请键入

stop cond condition

打印C++类型对象：

Print tmpStr;

tmpStr = {

p = 0x12e7248

caseSensitive = 1

paranoidCheck = 0

skipWhite = 1

initialCapac = 63

resizeInc = 64

freeboard = 63

}

print \*tmpStr.p

\*tmpStr.p = {

array = 0x1557f90 "qwert"

nchars = 5U

capacity = 63U

refs = 1U

}

print tmpStr.p.array

tmpStr.p->array = 0x1557f90 "qwert"

监视表达式：

Display display expression, ...

取消监视：

要关闭特定变量或表达式的显示，请键入：

undisplay expression

要关闭所有当前被监视的变量的显示，请键入：

undisplay 0

要给变量赋值，请键入

assign variable = expression

C 和 C++ 的数组分片语法  
对各维数组而言，使用 print 命令进行数组分片的完整语法为：

print array-expression [first-expression .. last-expression :stride-expression]

eg: print arr[2..6:2]arr[2..6:2] =  
[2] = 2  
[4] = 4  
[6] = 6

要打开内存使用和内存泄漏检查，请键入：

check -memuse

若只打开内存访问检查，请键入：

check -access

要打开内存泄漏、内存使用和内存访问检查，请键入：

check –all

要完全关闭运行时检查，请键入：

uncheck -all