二、环境变量

1. 环境变量的概念

环境变量：存放和编程环境/系统环境相关信息的变量叫做环境变量。

--》env(1) 察看系统的环境变量表

--》echo $环境变量：将环境变量的值输出到显示器

一些具体的环境变量：

USER 当前用户的名字

PATH 指定命令(ls,cp,pwd等)的查找路径

(按路径顺序，先访问到的路径里的同名命令就执行，自己可以编写命令放到PATH以使用)

-----------------------------

2. 环境变量的格式

环境变量的格式：在等号的两边坚决不允许出现空格。

name=value

name是环境变量的名字

value是环境变量的值

----------------------

3. 环境变量的配置

(1) Windows Xp系统中的配置

我的电脑 => 右键，选择属性 => 高级 => 环境变量 => 系统变量 => 选中Path,点击编辑 => 在Path变量值的最后增加分号(;),再增加新的路径(千万不要改动之前的变量值) => 一路点击确定即可

(2) Linux系统下的配置

打开任意一个终端，执行以下命令：

export PATH=$PATH:.

----------------------

注解：$环境变量： 获取环境变量的值

$PATH:. 在$PATH环境变量的后边添加上当前路径。

PATH=$PATH:. 在$PATH环境变量的后边添加上当前路径，然后更新PATH的值。

: 路径的分隔符，类似于Windows系统中的分号

export 表示导入/修改的意思，就是将后边的变量导出为环境变量。----------------------

如果需要设置环境变量为永久有效，则执行以下指令：

vi ~/.bashrc文件，在文件末尾增加代码：export PATH=$PATH:.

source ~/.bashrc(或者bash)表示让文件的配置立即生效

~代表当前用户的工作主目录

----------------------------------------

补充：

export PS1="\W\$ " (只显示当前目录的省略，参考《鸟哥私房菜》)

bash有外部命令和内部命令

跟bash独立的应用程序称为外部命令

命令如果是bash中的程序，称为内部命令

type command 用于察看命令的内部还是外部。

————————————————————

4. 编程相关的环境变量

PATH - 存放不需要再增加路径就可以直接运行可执行文件的目录

CPATH - 主要表示C语言中头文件所在的路径

CPLUS\_INCLUDE\_PATH - 主要表示C++头文件所在的路径

LIBRARY\_PATH - 编译链接时查找静态库和共享库的路径

LD\_LIBRARY\_PATH - 运行时查找共享库的路径

5. 头文件的定位方式

(1)#include <>

表示去系统默认的路径中查找指定的头文件 /usr/include

(2)#include "<目录>/xxx.h"

缺点：头文件路径发生变化，必须修改源程序

(3)export CPATH=$CPATH:<目录>

缺点：同时构建多个工程时，容易引发冲突

1. gcc -I编译选项：gcc -I<目录>

优点：既不会修改源程序，也不会有任何冲突，推荐使用此方法

==============================

1. 静态库和动态库

1. 静态库和动态库

什么是库？什么是静态库？

函数的仓库，就称为函数库。函数库分为静态和动态两种

静态库，静态库中的函数，在编译阶段链接。称为静态库。

动态库，动态库中的函数，在程序执行的时候链接。成为动态库。

2. 动态库和静态库的区别

1）使用静态库链接的可执行文件：文件比较大，不需要跳转，执行的时候就不再依赖于静态库文件。

2）使用动态库链接的可执行文件：文件比较小，需要跳转，执行的时候依赖于动态库文件。

————————

补充：

静态链接：发生在编译阶段(把函数代码完全拷贝到可执行文件中,变成T)

动态链接：程序运行的时候，才发生的链接(例如：printf)，使用时需要链接库文件

静态库和动态库的命名规则：

静态库：lib库名.a

动态库：lib库名.so

---------------------

3.静态库的创建：

a) 先编译源文件为目标文件

gcc -c tmath.c process.c

b) 将目标文件打包到静态库文件

ar -r libtmath.a tmath.o process.o

ar -t libtmath.a 察看静态库里包含的目标文件

c) 使用静态库编译链接文件，生成可执行文件

gcc test.o -L. -ltmath -o tt

-L路径 库文件所在路径

-l库名 找仓库的名字(不需要后缀)

-----------------------

补充：

制作跟系统一样的静态库文件

第一步：将头文件移动到系统指定的路径下

/usr/include

(sudo mv 所有头文件 /usr/include)

第二步：将库文件移动到程序链接时的默认的路径下

/usr/lib

(sudo mv libtmath.a /usr/lib )

第三步：将程序和库文件链接

gcc test.c -ltmath -o tt

---------------------------

4.动态库的制作：

a) 将源文件编译为目标文件

gcc -fPIC -c process.c tmath.c

-fPIC表示多个进程共享库文件时，只加载一份

b) 生成动态库

gcc -shared \*.o -o libtmath.so

c) 动态库的使用

gcc test.c -L. -ltmath

d) 链接时，失败

原因：加载器找不到库文件，与静态库的链接器不同

e) 解决办法：需要让加载器知道库文件的所在路径。

1)通过配置环境变量LD\_LIBRARY\_PATH告诉加载器的搜索路径

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:.

2)将库文件移动到/usr/lib或者/lib文件夹下

sudo mv libtmath.so /usr/lib

------------------------------

补充：

/usr/lib和/lib是链接器查找库文件的默认路径，同时也是加载器查找库文件的默认路径

客户端更新本质是动态库文件更新

使用新库文件替换旧的库文件，可执行文件不做更改。

第一步：将tmath.c文件中的add函数内部，使用return x-y；

第二步：制作库文件libtmath.so

gcc -fpic -shared tmath.c process.c -o libtmath.so

第三步：更新库文件。

sudo mv libtmath.so /usr/lib

-------------------------------

5.动态加载库函数

在程序中，根据自己的需求加载动态库，使用到了一系列的函数

dlopen(3)

#include <dlfcn.h>

void \*dlopen(const char \*filename,int flag);

功能：加载一个动态库文件

参数：

filename 指定要加载的库文件的名字

flag RTLD\_LAZY 延迟绑定 RTLD\_NOW 立即加载

返回值：返回一个地址。NULL 失败

dlerror(3)

char \*dlerror(void);

功能：返回最近的错误信息。错误信息由dlopen、dlclose、dlsym函数调用产生。

返回值：NULL 代表没有错误

dlsym(3)

void \*dlsym(void \*handle,const char \*symbol);

功能：将handle指向的动态库中的symbol(函数名)加载到内存中

参数：handle 是dlopen函数的返回值,symbol 是要加载到内存的符号

返回值：是symbol被加载到内存中的地址；NULL 失败

dlclose(3)

int dlclose(void \*handle);

功能：关闭指定动态库文件

参数：handle 是dlopen函数的返回值

返回值：0正确返回,非0失败返回

注意：Link with -ldl.编译时 -ldl，表示链接指定库文件

例：gcc dynamic.c -ldl

举例，代码参见dynamic.c

------------------

使用动态库编译链接和动态加载区别：动态加载看不到库函数名

gcc test.c -ltmath

a.out

nm a.out

使用

gcc dynamic.c -ldl

a.out

nm a.out