五级项目 2：设置环境变量

1　学时

- 1 学时

2　实验目的

- 理解、掌握、应用 Linux 下环境变量的使用。

3　实验内容

- 本项目通过命令行设置环境变量的值，因此进程的命令行参数至少要有 2 个，因为第一个参数必然是可执行程序名，第二个参数必须指明环境变量的名字，第三个参数如果存在则是环境变量的值，如果不存在则要获取当前时间作为值。

4　实验原理

（1）Shell 变量

Shell 是一个特殊的进程，是用户与内核之间的接口，Shell 启动后拥有多个自己的变量，以列表形式保存，也称为环境变量。变量列表由若干字符串组成，并以 NULL 作为结尾。在 Shell 中启动进程后，进程使用特殊全局变量“char**environ”继承 Shell 的环境变量，这个变量是一个特殊的全局变量，不需要用户自己定义，只需要使用“extern char**environ;”引用即可使用。

环境变量列表形式如图所示。

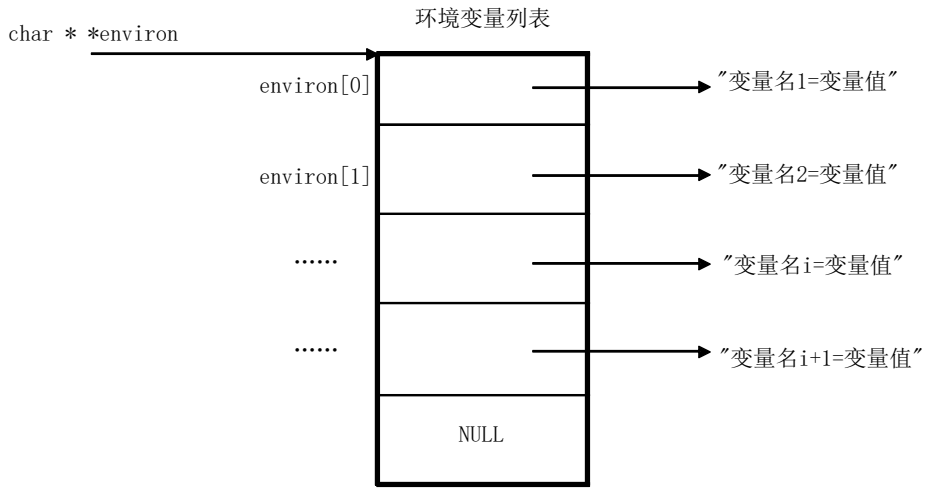

图　环境变量列表

（2）环境变量访问命令

在 Shell 中对环境变量的访问主要包括查看、定义、修改、删除。

（a）查看

用户可以使用 echo 命令查看某个变量的值，格式为：

echo　$变量名

如“echo　$HOME”将输出登录用户的主目录。

如果查看当前 Shell 中所有的环境变量信息，可使用“set”命令。常用的环境变量有 HOME（用户主目录），PWD（当前工作目录），PATH（默认可执行程序搜索路径）等。这些变量在用户登录系统后，由系统自动设置。

（b）定义

Shell 允许用户自定义环境变量，BASH 中定义变量的语法如下：

变量名=变量值（“=”左右无空格）

如：

myvar=hello

这样定义的环境变量只能被 Shell 本身访问，不能被在 Shell 中启动的其它应用程序访问。

如输入命令“sh”启动另一个 Shell 程序，再次输入命令“echo $ myvar”查看 myvar 变量的值，结果输出一个空行。这表明 sh 这个 Shell 中没有定义变量 myvar，因为 myvar 是原来的 Shell 定义的，不能被在原来 Shell 中启动的 sh 进程访问。记得输入“exit”命令退出 sh 这个 Shell。

如果在变量前加上 export 命令，则该变量可以被 Shell 中启动的其它应用程序访问。可以在定义变量的同时使用 export 命令，也可以在定义变量后，单独使用。

如：# export myvar=hello

或者

myvar=hello

export myvar

这样就可以使用新运行的 sh 去访问 myvar 变量了。

（c）修改

修改环境变量的格式与定义环境变量的格式相同，对于 export 命令的使用也相同。如果修改变量的同时还需要包含原来的值，则需要事先引用原来的值，如 PATH 变量中保存了可执行文件的搜索路径，使用冒号分隔，原来保存了一些路径值，如果要增加一个当前目录，则要进行如下修改：export PATH=$PATH:.。

（d）删除

删除环境变量的命令为 unset，命令格式为：

unset 变量名

如删除 myvar 变量可使用“unset myvar”。

需要注意的是使用命令方式对变量做出的改变，无论增加、修改、删除等都是临时的，当系统重新启动后将恢复到改变之前的状况。这是因为 Linux 系统启动过程中是通过读取系统配置文件来确定各变量的值，使用命令方式对环境变量表的改动并没有记录入文件，所以重新启动后将恢复原状。如果用户想在系统重新启动后仍然保持对环境变量表的改动，需要将改动写入文件中，相关的文件有/etc/profile 和用户主目录中的.bash_profile。前者文件中的修改结果所有登录用户都能访问，而后者文件中的修改只有指定某个用户才能访问。

(3)环境变量访问函数

使用命令方式访问环境变量是要在命令行上输入命令的，并且修改的是 Shell 的环境变量。在编程过程中访问进程的环境变量时需要使用专门函数。常用的函数有 getenv、putenv、setenv 和 unsetenv，见表。

表 函数 getenv

项目	描述
头文件	#include <stdlib.h>

原型	char *getenv(const char *name);
功能	获取环境变量
参数	name：要获取的环境变量名
返回值	成功获取环境变量时，返回一个指向环境变量值的指针；如果没有获取环境变量则返回空指针 NULL

表　函数 putenv

项目	描述
头文件	#include <stdlib.h>
原型	int putenv(char *string);
功能	改变或者增加环境变量
参数	string：要改变或增加的环境变量表达式
返回值	执行成功时，返回 0；失败时返回-1。

表　函数 setenv

项目	描述
头文件	#include <stdlib.h>
原型	int setenv(const char *name, const char *value, int overwrite);
功能	改变或者增加一个环境变量
参数	name：环境变量名 value：环境变量的值 overwrite：是否覆盖环境变量原值。0：不覆盖；非 0：覆盖
返回值	执行成功时，返回 0；失败时返回-1。

表　函数 unsetenv

项目	描述
头文件	#include <stdlib.h>
原型	void unsetenv(const char *name);
功能	删除一个环境变量
参数	name：要删除的环境变量名
返回值	执行成功时，返回 0；失败时返回-1，并设置 errno。

（4）程序结构设计

由于本项目是通过命令行设置环境变量的值，因此进程的命令行参数至少要有 2 个，因为第一个参数必然是可执行程序名，第二个参数必须指明环境变量的名字，第三个参数如果存在则是环境变量的值，如果不存在则要获取当前时间作为值。所以程序首先要判断命令行参数的个数，首先如果小于 2 的话程序不能正常执行，需要做出运行方式的提示并退出进程;

如果大于或等于 2 则根据参数个数进行下一步操作。这里要利用分支结构。

（5）程序数据设计

由于可能要获取当前时间作为环境变量的值，值是字符串，因此时间必须得到字符串形式的，可以通过 struct tm 结构体自行定义时间字符串，还可以直接使用时间的字符串形式，本项目实现时采用时间的字符串形式。

根据项目构思，本项目定义如下数据：

```
char  *curtimechar;   //保存当前时间字符串指针
time_t  curtime;   //保存当前时间，time_t 类型
```

（6）程序基本流程

程序的流程图如图 所示。

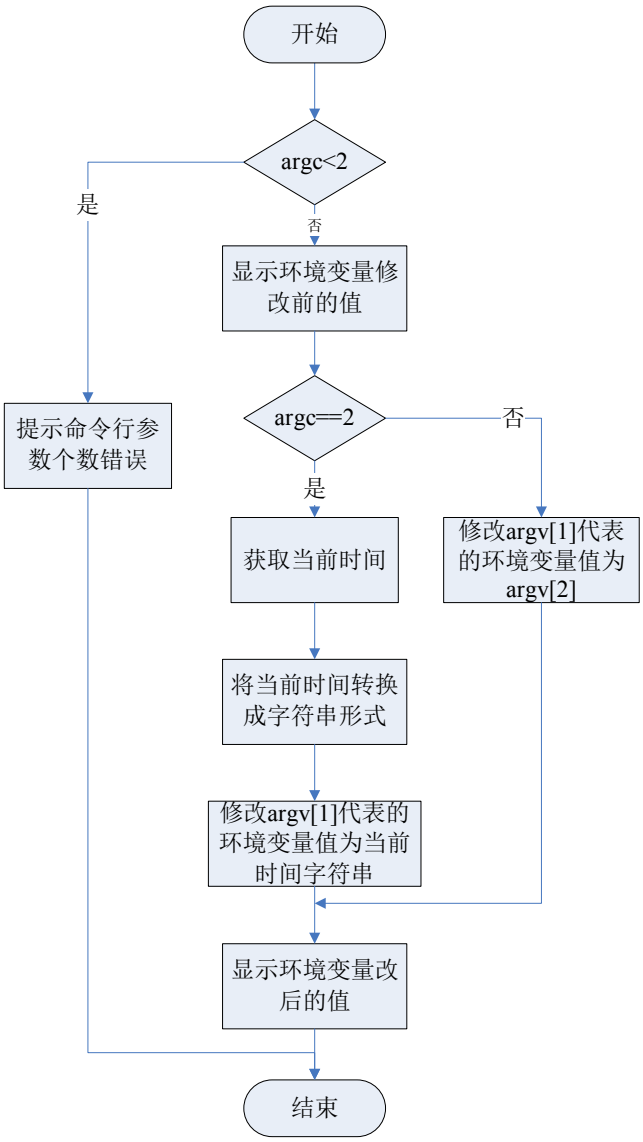

图　设置环境变量流程图

5　预习要求和技术准备工作

- 掌握 Linux 基本操作
- 掌握 C 语言开发工具的使用
- 掌握环境变量相关函数的使用

6 实验环境

- PC 机
- 在 Windows 环境中的 VMware 虚拟机上运行 Ubuntu 操作系统或者独立的 Ubuntu 操作系统
- 基于 Linux 的 vi 编辑器和 gcc 编译器

7 实验设计及操作步骤

7.1 以 root 身份登录系统，在/home 目录中创建目录 exp52

```
cd  /home
mkdir  exp52
```

7.2 进入刚创建的目录

```
cd  exp52
```

7.3 使用 vi 编辑文件，文件名是 setenvvar.c。

7.4 编写程序，实现要求的功能

7.5 编译可执行文件。

```
#gcc  setenvvar.c  -o  setenvvar
```

7.6 运行程序

```
./setenvvar
```

可能的运行界面如图所示。

```
[root@localhost jiaocai]# ./setenvvar
usage:./setenvvar varname [value]
```

图 运行界面 1

```
[root@localhost jiaocai]# ./setenvvar test
before test = (null)
after test = Mon May  2 13:57:36 2011
```

图运行界面 2

```
[root@localhost jiaocai]# ./setenvvar test hello
before test = (null)
after test = hello
```

图 运行界面 3

8 实验报告提交要求：

将实验操作每个步骤中的命令、源程序以及截图写入实验报告，实验报告命

名为“学号姓名-实验 52.doc”，交给指定人员。

9 项目思考

本例创建的环境变量在进程退出后是否还能继续访问，请同学自行测试一下，然后思考出现测试结果的原因。