@TOC

环境变量

为什么系统的命令不需要带路径呢？系统可以找到也就是**环境变量(PATH)**

- 用：当分隔符，而且要查看需要加上$，不加查看不了**echo $PATH**

常见环境变量

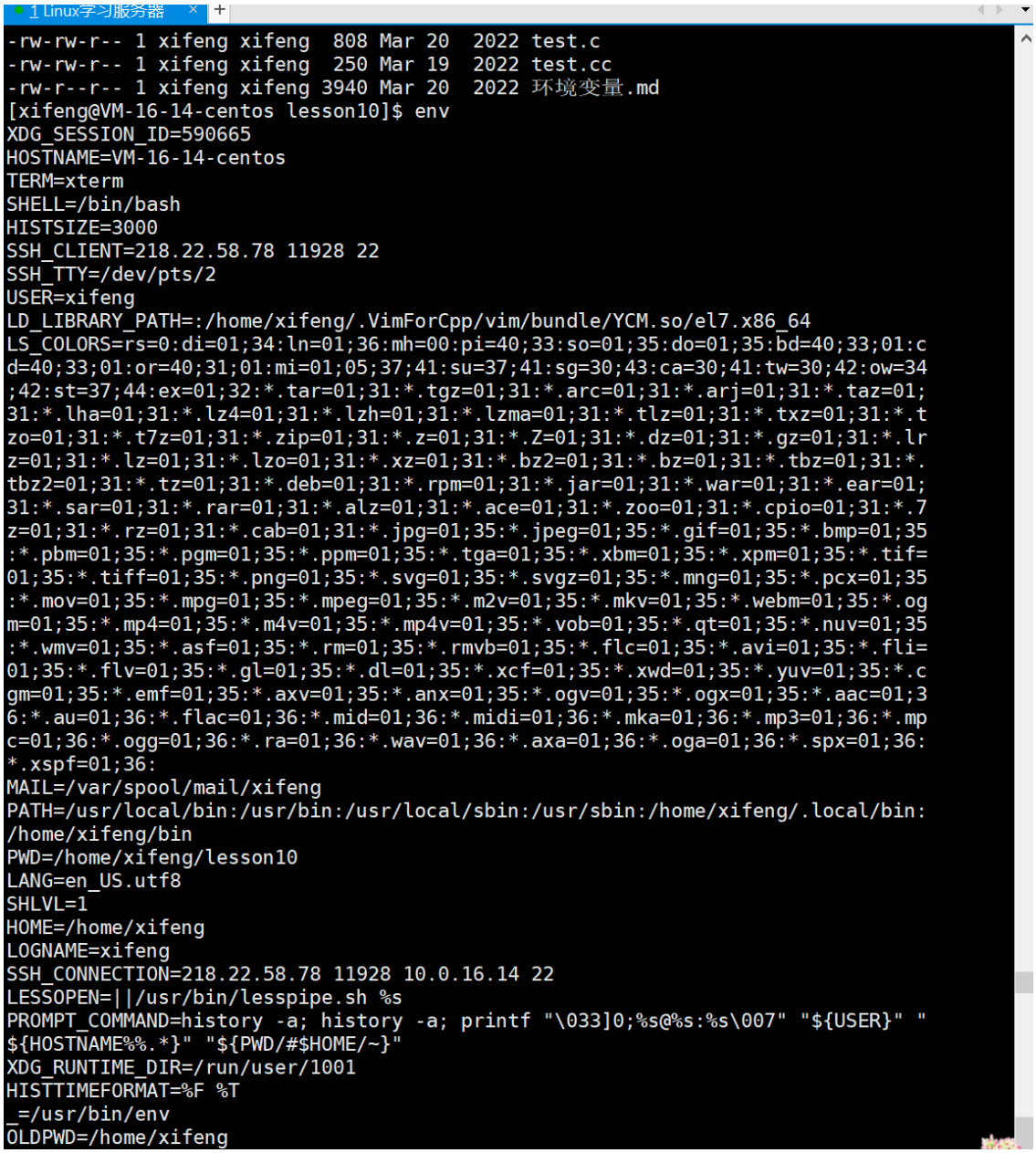
环境变量本质是OS在内存/磁盘文件中开辟的空间，用来保存系统相关数据(简单理解就是变量名+变量内容)

- **PATH**：指定命令的搜索路径
- **HOME** : 指定用户的主工作目录(即用户登陆到Linux系统中时,默认的目录)
- **SHELL** : 当前Shell,它的值通常是/bin/bash
- 系统上还存在一种变量，是与本次登录(session)有关的变量，只在本次登录有效(本地变量),像之前写的一些脚本就是创建了一个本地变量(例：count=0; while [$count -le 10]; do echo $count; let count++; done 这就定义了一个本地变量echo $count可以查看值)

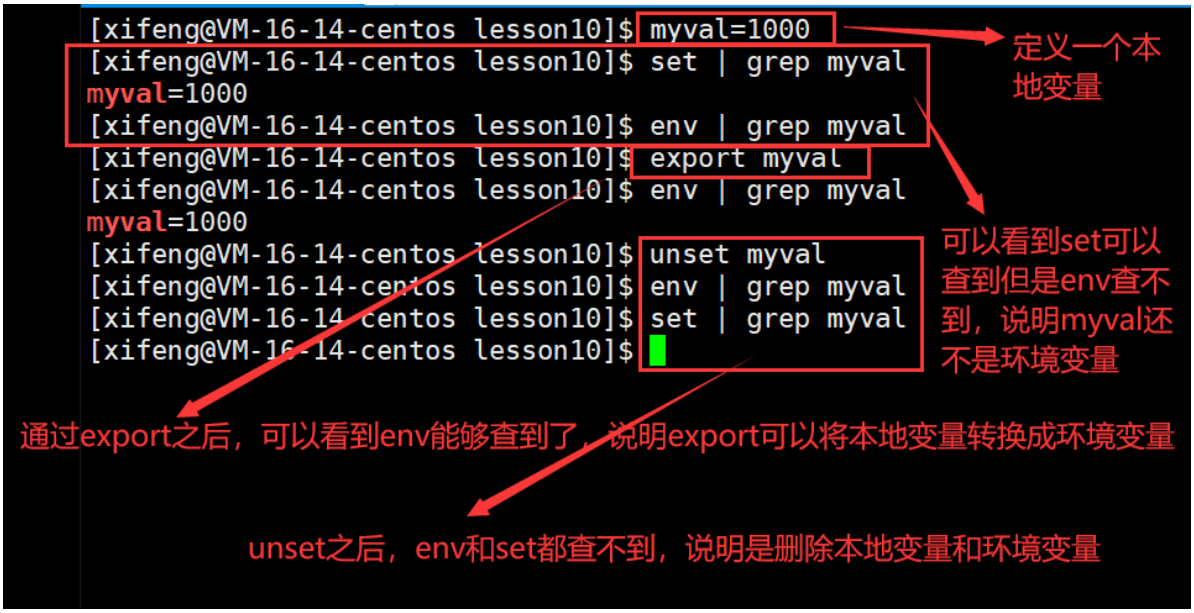
和环境变量的相关指令

1. echo: 显示某个环境变量或者本地变量值
    - 查看本地变量：echo $my_env
    - 查看环境变量：echo $PATH
2. export: 设置一个新的环境变量 (内存级)
    - 将本地变量导成环境变量
        - export myval(本地变量名)
    - 用法:**export PATH=$PATH: /home/xifeng/...（路径）**

3. env: 显示所有环境变量

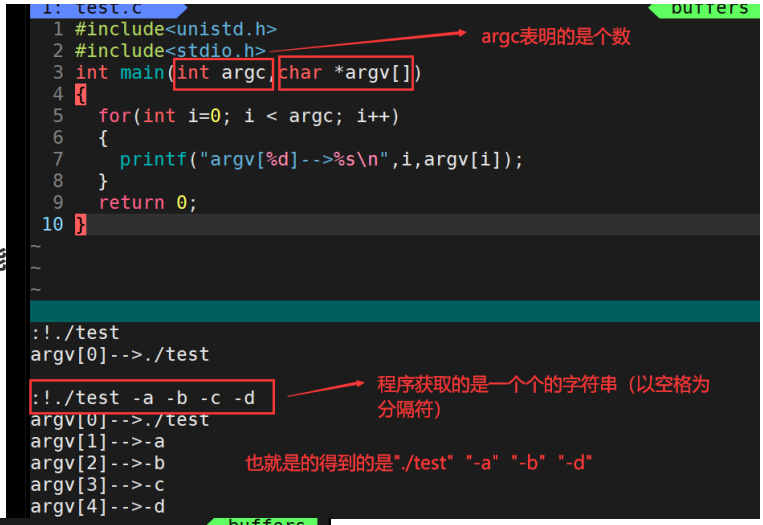


4. unset: 清除环境变量
    - 取消本地或者环境变量
5. set: 显示本地定义的shell变量(本地变量)和环境变量
6. myval(本地变量) 只对本次登录有效



环境变量的获取和操作

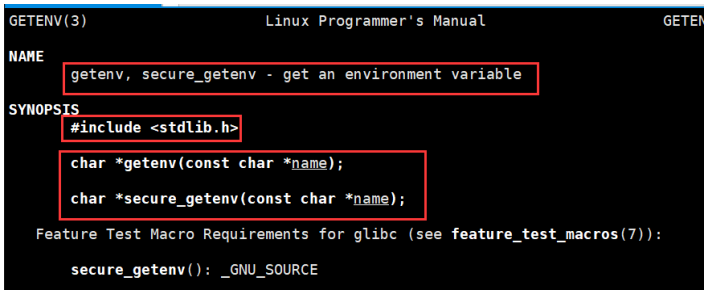
首先，先引入几个点：我们知道main函数是可以带参数的，一般是：int main(int argc,char *argv[])，我们可以在命令行上传各种参数------那么为什么要这样传呢？---->**同一个程序可以通过带不同的命令行参数来使其执行不同的功能**

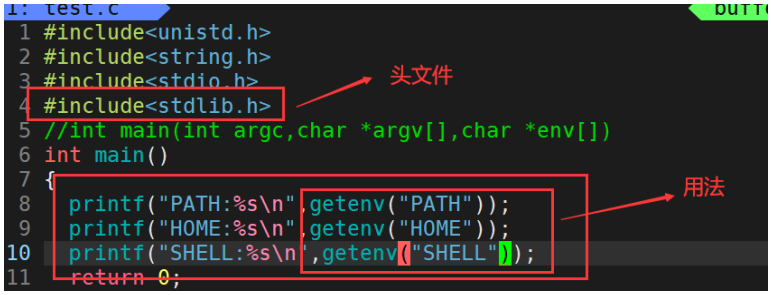



```c
#include<unistd.h>
#include<string.h>
#include<stdio.h>
int main(int argc,char *argv[])
{
  if(argc!=2)
  {
    printf("Usage:./test -[a|b]\n");
    return 1;
  }
  if(strcmp(argv[1],"-a")==0)
  {
    printf("hello all\n");
  }
  else if(strcmp(argv[1],"-b")==0)
  {
    printf("hello linux\n");
  }
  else{
    printf("hello word\n");
  }
  return 0;
}
```

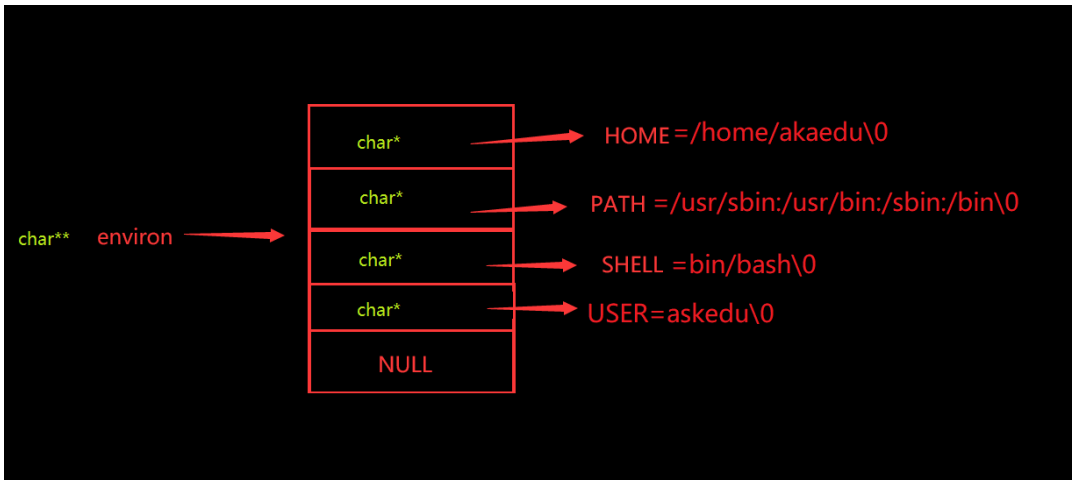
```
[xifeng@VM-16-14-centos lesson10]$ gcc -o test test.c
[xifeng@VM-16-14-centos lesson10]$ vim test.c
[xifeng@VM-16-14-centos lesson10]$ ./test
Usage:./test -[a|b]
[xifeng@VM-16-14-centos lesson10]$ ./test -a
hello all
[xifeng@VM-16-14-centos lesson10]$ ./test -b
hello linux
[xifeng@VM-16-14-centos lesson10]$ ./test -c
hello word
```

- 指令有很多选项，用来完成同一个命令的不同子功能，选项底层使用的就是**命令行参数**！！
- 获取环境变量
  - getenv()获取环境变量(推荐)





  - 代码有几种形式




```c
int main()
{
    extern char **environ;//envrion是系统提供的外部变量，可以直接去找到环境变量的地址
    for(int i=0;environ[i];i++)
    {
        printf("%d-->%s\n",i,environ[i]);
    }
}
```

```c
test.c                                                    buffers
#include<unistd.h>
#include<string.h>
#include<stdio.h>
int main(int argc,char *argv[],char *env[])
{
  for(int i=0;env[i];i++)
  {
    printf("%d-->%s\n",i,env[i]);
  }
  return 0;
```

(env[]是导入系统的环境变量，因为是系统提供的所以不需要类似于argc的计数功能)

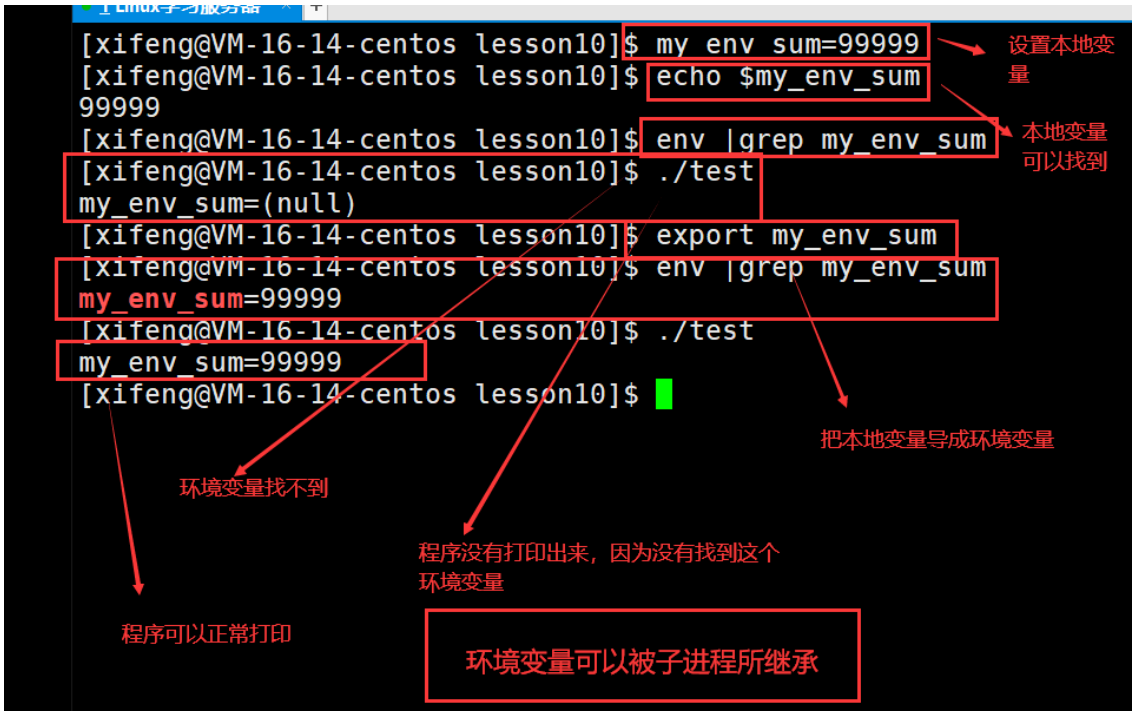
```
-->OLDPWD=/home/xifeng
-->SSH_TTY=/dev/pts/0
-->USER=xifeng
-->LD_LIBRARY_PATH=:/home/xifeng/.VimForCpp/vim/bundle/YCM.so/el7
x86_64
0-->LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=0
;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:s
=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*
tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=0
;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:
.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:
.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;
1:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.j
r=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;
1:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.ca
=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;
5:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.t
f=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=0
;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:
.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4
=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35
```

环境变量通常是具有全局属性的

- 使用shell时，默认的环境变量放在/etc/profile.d
- **命令行上启动的进程，子进程pid会变，但父进程都是bash（子进程的环境变量都是来自于bash）**

```
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2822,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2825,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2831,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2837,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2840,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ./testc
pid:2842,ppid:16298
[xifeng@VM-16-14-centos lesson10]$ ps axj | grep 16298
16298  2872  2872 16298 pts/2     2872 R+    1001   0:00 ps axj
16298  2873  2872 16298 pts/2     2872 R+    1001   0:00 grep --color=auto 16298
16297 16298 16298 16298 pts/2     2872 Ss    1001   0:00 -bash
[xifeng@VM-16-14-centos lesson10]$
```

- **环境变量具有“全局属性”根本原因是因为：环境变量是可以被子进程继承的**



- 可以查看vim ~/.bashrc 和 vim ~/.bash_profile 和 vim /ect/bashrc