

## 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**

```
[xifeng@VM-16-14-centos lesson9]$ echo PATH
PATH
[xifeng@VM-16-14-centos lesson9]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/xifeng/.local/bin:/home/xifeng/bin
[xifeng@VM-16-14-centos lesson9]$
```

---

## 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)
  - 将本地变量导出成环境变量
    - 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

```
1 Linux学习服务器 x +
-rw-rw-r-- 1 xifeng xifeng 808 Mar 20 2022 test.c
-rw-rw-r-- 1 xifeng xifeng 250 Mar 19 2022 test.cc
-rw-r--r-- 1 xifeng xifeng 3940 Mar 20 2022 环境变量.md
[xifeng@VM-16-14-centos lesson10]$ env
XDG_SESSION_ID=590665
HOSTNAME=VM-16-14-centos
TERM=xterm
SHELL=/bin/bash
HISTSIZE=3000
SSH_CLIENT=218.22.58.78 11928 22
SSH_TTY=/dev/pts/2
USER=xifeng
LD_LIBRARY_PATH=/home/xifeng/.VimForCpp/vim/bundle/YCM.so/el7.x86_64
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:c
d=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=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=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.t
zo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lr
z=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;
31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7
z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35
:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=
01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35
:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.og
m=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35
:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.c
gm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=01;3
6:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mp
c=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;36:*.oga=01;36:*.spx=01;36:
*.xspf=01;36:
MAIL=/var/spool/mail/xifeng
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/xifeng/.local/bin:
/home/xifeng/bin
PWD=/home/xifeng/lesson10
LANG=en_US.utf8
SHLVL=1
HOME=/home/xifeng
LOGNAME=xifeng
SSH_CONNECTION=218.22.58.78 11928 10.0.16.14 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
PROMPT_COMMAND=history -a; history -a; printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%.*}" "${PWD/#$HOME/~}"
XDG_RUNTIME_DIR=/run/user/1001
HISTTIMEFORMAT=%F %T
_=/usr/bin/env
OLDPWD=/home/xifeng
```

### 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

```
[xifeng@VM-16-14-centos lesson10]$ myval=1000
[xifeng@VM-16-14-centos lesson10]$ set | grep myval
myval=1000
[xifeng@VM-16-14-centos lesson10]$ env | grep myval
[xifeng@VM-16-14-centos lesson10]$ export myval
[xifeng@VM-16-14-centos lesson10]$ env | grep myval
myval=1000
[xifeng@VM-16-14-centos lesson10]$ unset myval
[xifeng@VM-16-14-centos lesson10]$ env | grep myval
[xifeng@VM-16-14-centos lesson10]$ set | grep myval
[xifeng@VM-16-14-centos lesson10]$
```

定义一个本地变量

可以看到set可以查到但是env查不到，说明myval还不是环境变量

通过export之后，可以看到env能够查到了，说明export可以将本地变量转换成环境变量

unset之后，env和set都查不到，说明是删除本地变量和环境变量

## 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

```
1: test.c
1 #include<unistd.h>
2 #include<stdio.h>
3 int main(int argc, char *argv[])
4 {
5     for(int i=0; i < argc; i++)
6     {
7         printf("argv[%d]-->%s\n", i, argv[i]);
8     }
9     return 0;
10 }
```

argc表明的是个数

程序获取的是一个字符串（以空格为分隔符）

也就是得到的是"./test" "-a" "-b" "-d"

```
#!/test
argv[0]-->./test
#!/test -a -b -c -d
argv[0]-->./test
argv[1]-->-a
argv[2]-->-b
argv[3]-->-c
argv[4]-->-d
```

```
1: test.c
1 #include<unistd.h>
2 #include<string.h>
3 #include<stdio.h>
4 int main(int argc, char *argv[])
5 {
6     if(argc!=2)
7     {
8         printf("Usage:./test -[a|b]\n");
9         return 1;
10    }
11    if(strcmp(argv[1], "-a")==0)
12    {
13        printf("hello all\n");
14    }
15    else if(strcmp(argv[1], "-b")==0)
16    {
17        printf("hello linux\n");
18    }
19    else{
20        printf("hello word\n");
21    }
22    return 0;
23 }
```

```

[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)

```

GETENV(3)                                Linux Programmer's Manual                                GETENV

NAME
    getenv, secure_getenv - get an environment variable

SYNOPSIS
    #include <stdlib.h>

    char *getenv(const char *name);
    char *secure_getenv(const char *name);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

    secure_getenv(): _GNU_SOURCE

```

```

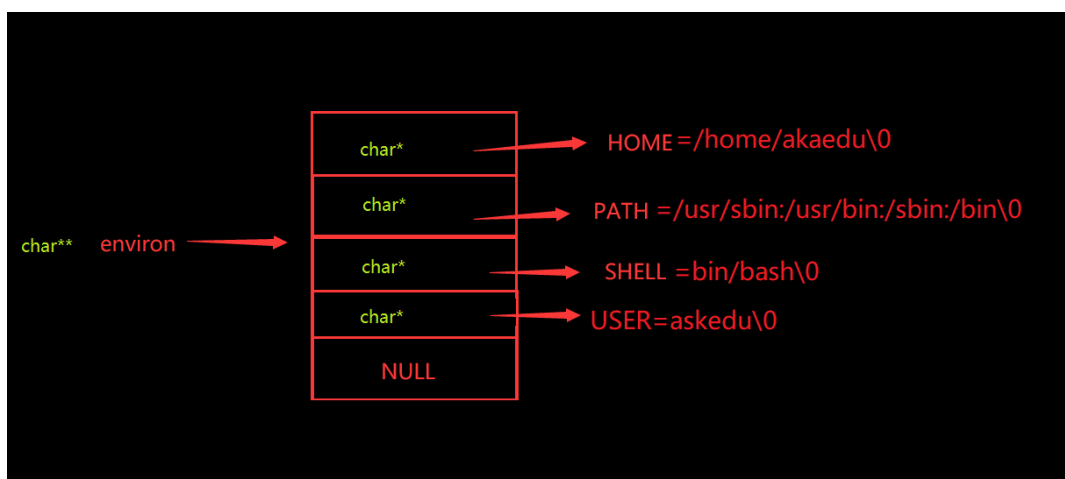
1: test.c
1 #include<unistd.h>
2 #include<string.h>
3 #include<stdio.h>
4 #include<stdlib.h>
5 //int main(int argc,char *argv[],char *env[])
6 int main()
7 {
8     printf("PATH:%s\n",getenv("PATH"));
9     printf("HOME:%s\n",getenv("HOME"));
10    printf("SHELL:%s\n",getenv("SHELL"));
11    return 0;

```

头文件

用法

- 代码有几种形式



```

int main()
{
    extern char **environ; //environ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]);
    }
}

```

buffers

量，因为是系统提供的所以不需要类似于argc的计数功能)

```

6-->OLDPWD=/home/xifeng
7-->SSH_TTY=/dev/pts/0
8-->USER=xifeng
9-->LD_LIBRARY_PATH=/home/xifeng/.VimForCxx/vim/bundle/YCM.so:el
x86_64
10-->LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=0
11:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:s
12:g=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*
13:tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=0
14:31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:
15:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:
16:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.b2=01;31:*.bz=01;
17:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.j
18:ar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;
19:31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.ca
20:p=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;
21:35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.t
22:if=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=0
23:1;35:*.pcx=01;35:*.mov=01;35:*.mpeg=01;35:*.mjpeg=01;35:*.m2v=01;35:
24:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4
25:v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35

```

- 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

```
[xifeng@VM-16-14-centos lesson10]$ my_env_sum=99999
[xifeng@VM-16-14-centos lesson10]$ echo $my_env_sum
99999
[xifeng@VM-16-14-centos lesson10]$ env |grep my_env_sum
[xifeng@VM-16-14-centos lesson10]$ ./test
my_env_sum=(null)
[xifeng@VM-16-14-centos lesson10]$ export my_env_sum
[xifeng@VM-16-14-centos lesson10]$ env |grep my_env_sum
my_env_sum=99999
[xifeng@VM-16-14-centos lesson10]$ ./test
my_env_sum=99999
[xifeng@VM-16-14-centos lesson10]$
```

设置本地变量

本地变量可以找到

把本地变量导出成环境变量

环境变量找不到

程序没有打印出来，因为没有找到这个环境变量

程序可以正常打印

环境变量可以被子进程所继承

- 可以查看vim ~/.bashrc 和 vim ~/.bash\_profile 和 vim /etc/bashrc