Linux重定向 管道命令 环境变量PATH

1. 了解Linux目录配置标准FHS
2. Linux数据重定向的理解与操作
3. Linux管道命令的理解与操作
4. Linux 环境变量与PATH

了解Linux目录配置标准FHS

是什么？ FHS本质一套规定Linux目录结构，软件建议安装位置的标准

为什么？ 使用Linux来开发产品或者发布软件的公司、个人太多，如果每家公司或者个人都按照自己的意愿来配置文

件或者软件的存放位置，这无疑是一场灾难。我们可不想学完我们的Linux之后，去公司发现看不懂别人的目录结

构，更别谈开发了。

怎么办?

```
#进入根目录
[root@localhost home]# cd /
#查看目录树（两层，只看目录，不看文件）
[root@localhost /]# tree -d -L 1
.
├── bin -> usr/bin #可执行文件目录，linux自带命令在这里
├── boot #开机会使用到的文件，包括Linux核心文件以及开机菜单与开机所需配置文件等等。
├── dev #设备目录
├── etc #配置文件几乎都放置在这个目录内，例如人员的帐号密码档、 各种服务的启始档等等。
├── home #家目录，系统默认的使用者主文件夹，用户的个人文件都可放在这里
├── lib -> usr/lib #放置在开机时会用到的函数库，以及在/bin或/sbin下面的指令会调用的函数库
├── lib64 -> usr/lib64 #与lib类似
├── media #放置可移除的设备，包括软盘、光盘、DVD等等设备都暂时挂载于此。
├── mnt #设备临时挂载目录
├── opt #第三方协力软件放置的目录
├── proc #是一个虚拟文件系统，放置内存中的数据，不占用硬盘空间
├── root #系统管理员的主文件夹
├── run #放置系统开机后所产生的各项信息
├── sbin -> usr/sbin #里面包括了开机、修复、还原系统所需要的指令，root用户使用
├── srv #service是一些网络服务启动之后，这些服务所需要取用的数据目录。
├── sys #与proc类似
├── tmp #temp,让一般使用者或者是正在执行的程序暂时放置文件的地方
```

```
├── usr #unix software resource：与软件安装/执行有关
└── var #variable,与系统运行过程有关,主要为变动性较大的数据
19 directories
————————————————————
```

Linux数据重定向的理解与操作

基本准备

基本大部分计算机，要与人交互，都要默认打开三个设备(文件)

- 标准输入，stdin，代码是0
- 标准错误，stdout，代码是1
- 标准错误输出，stderr，代码是2

为什么？Linux一切皆文件，交互需求，输入输出信息分类

除了上面的三个标准设备(文件)， 我们可能经常有从文件读取数据，或者将数据写入文件的场景

重定向的理解

是什么?

输出/追加重定向：本来应该显示到显示器(通常)文件的内容，写入到文件当中。

输入重定向：或者本来应该从显示器(通常)文件读取数据，转化成从指定文件读取数据

输出重定向

```
#输出重定向
[whb@VM-0-3-centos test]$ ls
```

```
[whb@VM-0-3-centos test]$ whoami          # 该命令输出的结果，默认是直接打印到标准输出stdout文件中
```

```
[whb@VM-0-3-centos test]$ whoami>file4 # 输出重定向，不在显示出来
[whb@VM-0-3-centos test]$ ll               # 新创建文件file4
total 4
-rw-rw-r-- 1 whb whb 0 Mar   2 15:22 file1
-rw-rw-r-- 1 whb whb 0 Mar   2 15:22 file2
-rw-rw-r-- 1 whb whb 0 Mar   2 15:22 file3
-rw-rw-r-- 1 whb whb 4 Mar   2 15:23 file4
[whb@VM-0-3-centos test]$ cat file4      #本来应该显示到显示器的内容，写到了file4文件中
```

追加重定向

```
[whb@VM-0-3-centos test]$ cat file4
whb
[whb@VM-0-3-centos test]$ echo "hello bit" > file4 #将新内容重定向到文件file4中
[whb@VM-0-3-centos test]$ cat file4                          #发现之前的"whb"被覆盖了
```

```
[whb@VM-0-3-centos test]$ echo "hello bit" > file4
[whb@VM-0-3-centos test]$ echo "hello bit" > file4
[whb@VM-0-3-centos test]$ echo "hello bit" > file4
[whb@VM-0-3-centos test]$ cat file4                          #连续重定向多次，发现文件内容并没有变化
hello bit                                                                    #其实不是没有变化，而是每次写入都是从头开始
[whb@VM-0-3-centos test]$ cat file4
hello bit
[whb@VM-0-3-centos test]$ echo "hello bit1" >> file4
[whb@VM-0-3-centos test]$ echo "hello bit2" >> file4
[whb@VM-0-3-centos test]$ echo "hello bit3" >> file4
[whb@VM-0-3-centos test]$ cat file4      #通过多次>>，我们发现内容是被追加到文件结尾的，叫做追加重定向
hello bit
hello bit1
hello bit2
hello bit3
```

输入重定向

```
[whb@VM-0-3-centos test]$ cat > file1
hello
bit
best #ctrl+d结束
[whb@VM-0-3-centos test]$ cat file1 #可以看出，如果cat后面没有紧跟数据源文件，默认从标准输入获取数据
hello
bit
best
[whb@VM-0-3-centos test]$ ll
total 4
```

```
-rw-rw-r-- 1 whb whb 15 Mar   2 15:22 file1
-rw-rw-r-- 1 whb whb  0 Mar   2 15:22 file2
-rw-rw-r-- 1 whb whb  0 Mar   2 15:22 file3
-rw-rw-r-- 1 whb whb 43 Mar   2 15:30 file4
[whb@VM-0-3-centos test]$ cat file4 #也可以这样输出，想象一下，file4的内容打印到屏幕的具体过程
hello bit
hello bit1
hello bit2
hello bit3
[whb@VM-0-3-centos test]$ cat < file4 #既然cat要打印文件，前提也是先读取file4，所以也可以这样写
hello bit
hello bit1
hello bit2
hello bit3
[whb@VM-0-3-centos test]$ cat < file4 >file3 #理解一下
[whb@VM-0-3-centos test]$ cat file3 #拷贝了file4到file3
hello bit
hello bit1
hello bit2
hello bit3
```

代码0,1,2？什么鬼

```
[whb@VM-0-3-centos test]$ find /home -name test.c
find: ‘/home/wl’: Permission denied    #报错信息
/home/wudu/work/linux-57/linux-lesson7/signal_test/test.c          #正常信息
/home/wudu/work/linux-57/linux-lesson7/sigaction_test/test.c     #正常信息
...
[whb@VM-0-3-centos test]$ find /home -name test.c > test_list #发现并不是所有信息都被重定向
find: ‘/home/wl’: Permission denied                                                #报错信息并没有被写入目标文件
...
[whb@VM-0-3-centos test]$ cat test_list
/home/wudu/work/linux-57/linux-lesson7/signal_test/test.c        #正常信息被写入了
/home/wudu/work/linux-57/linux-lesson7/sigaction_test/test.c
```

```
...
# 1：代表标准输出
# 2：代表标准错误
# 但是他们两个默认都会往显示器打印，像上面的情况，打印输出到显示器会发生混乱，so
#只打印正常信息
[whb@VM-0-3-centos test]$ find /home -name test.c 2>err.list
/home/wudu/work/linux-57/linux-lesson7/signal_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigaction_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigmask/test.c
/home/wudu/work/linux-57/linux-lesson15/test/test.c
/home/wudu/work/linux-57/linux-lesson4/pipesize/test.c
/home/wudu/work/linux-57/linux-lesson5/testmkfifo/test.c
/home/wudu/work/linux-57/linux-lesson5/shmtest/test.c
/home/wudu/work/linux-57/linux-lesson3/dynamic/test.c
/home/wudu/work/linux-57/linux-lesson3/static/test.c
...
[whb@VM-0-3-centos test]$ cat err.list    #错误信息被单独分离出来
find: ‘/home/wl’: Permission denied
find: ‘/home/sly’: Permission denied
find: ‘/home/zwc’: Permission denied
find: ‘/home/cpx’: Permission denied
find: ‘/home/gb’: Permission denied
find: ‘/home/bss’: Permission denied
# 只打印错误信息
# 上面默认就是
# 正常和错误信息都重定向到文件中 2>&1
[whb@VM-0-3-centos test]$ find /home -name test.c >info.list 2>&1
[whb@VM-0-3-centos test]$ cat info.list
find: ‘/home/wl’: Permission denied
find: ‘/home/wudu/.local’: Permission denied
/home/wudu/work/linux-57/linux-lesson7/signal_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigaction_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigmask/test.c
...
```

/dev/null

垃圾桶黑洞设备，如果我知道有错误信息，但是我不想要，我想凡是错误信息直接丢弃，习惯写法是

```
[whb@VM-0-3-centos test]$ find /home -name test.c 2>/dev/null #便只显示正常信息了
/home/wudu/work/linux-57/linux-lesson7/signal_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigaction_test/test.c
/home/wudu/work/linux-57/linux-lesson7/sigmask/test.c
/home/wudu/work/linux-57/linux-lesson15/test/test.c
/home/wudu/work/linux-57/linux-lesson4/pipesize/test.c
/home/wudu/work/linux-57/linux-lesson5/testmkfifo/test.c
```

Linux管道命令的理解与操作

命令是可以产生数据的，如果我们还要多输出数据进行加工，甚至想多次加工，就需要使用管道

样例

```
[whb@VM-0-3-centos ~]$ last #显示正在或者最近登录linux的用户信息
```

```
#如果我只想看到前5条信息呢？
[whb@VM-0-3-centos ~]$ last | head -5 #其中'|'就是管道,head是一个截取文本行的工具，可以单独讲
```

```
#如果我只想看到我的历史登录信息呢？
[whb@VM-0-3-centos ~]$ last | grep 'felixg'
```

显示一共多少行

```
#如果我只想看到我的历史登录时间呢？有点难，了解一下就行,其中awk是一个文本处理工具，要到比特很后期的课程才有
```

可见，管道是可以级联多条命令的，每条命令的结果输出，都作为输入，导入下一条命令。有点像流水线

理论上，管道的基本操作就完了，实际上，如果站在日常使用就够了，但是如果想面试大厂，至少有两方面问题需要深究？

- 除了上面的文本处理工具，还有哪些文本处理工具你都使用过，各自有什么特征？
- 站在操作系统层面，管道的底层原理是什么？

Linux 环境变量与PATH

假设我们想用C写一个输出 ‘hello world’ 的程序

#等等,这个 './' 是什么东西？mycmd从构成上，也可以认为是一个命令
#像ls，就在/bin/ls or /usr/bin/ls, 为什么执行ls命令不需要带路径呢？（尽管也可以）

环境变量PATH

是一个路径集，命令再被执行时，系统会在环境变量PATH中进行路径查找，如果找到，就停止查找，执行命令

PATH：保存了多条路径！路径之间用“:”隔开,从左到右依次进行程序搜索，找不到就继续下一条路径，找到了就停下来，找不到就报错

```
[whb@VM-0-3-centos test]$ echo $PATH     #不同环境，平台可能内容会有不同
```

```
$: 用来显示PATH环境变量的内容
#各个路径以 : 作为分隔符，每个区域代表一个搜索路径
```

这也就解释了，为何ls不需要带路径，因为ls所在的路径，本身就在PATH环境变量列表当中。

```
[whb@VM-0-3-centos test]$ which ls #which 显示指定命令所在路径
```

那么问题来了，我也想让我的‘ hello world ‘程序执行的时候不带 ‘./‘ 这样的路径, 怎么办呢?

```
#方法一
[whb@VM-0-3-centos test]$ mycmd
```

```
[whb@VM-0-3-centos test]$ sudo cp mycmd /usr/bin
[whb@VM-0-3-centos test]$ ls /usr/bin/mycmd
[whb@VM-0-3-centos test]$ mycmd #但是千万不要这样干，因为会‘污染’指令集
```

```
[whb@VM-0-3-centos test]$ sudo rm /usr/bin/mycmd #删掉在/usr/bin路径下的可执行程序
[whb@VM-0-3-centos test]$ which mycmd #mycmd已经能通过环境变量被找到
~/test/mycmd
```

```
#方法二
[whb@VM-0-3-centos test]$ echo $PATH
```

```
[whb@VM-0-3-centos test]$ pwd #查看自己当前处于哪个目录
[whb@VM-0-3-centos test]$ export PATH=$PATH:/home/whb/test#导出新的环境变量，加上程序所在的路径
[whb@VM-0-3-centos test]$ echo $PATH
[whb@VM-0-3-centos test]$ mycmd #可以不带路径
```

还有其他环境变量吗?

```
#了解一下即可
[whb@VM-0-3-centos test]$ env #显示当前用户环境变量
```

windows当中有没有类似的概念呢?

有的，我们可以现场找一下

我们在思考一下，几乎可以得出如下结论：

- 系统本身会提供某种全局查找属性，帮我们找到特定的模块/程序

其实，编译器内部也有类似的功能，想想，为何我们 #include<头文件> ,并没有指明头文件在哪里，但是编译器也能帮我们找到并展开，包括我们也并没有指明我们的程序依赖哪些第三方库，编译器也能找到。