

【1-7】重定向、管道与环境变量

笔记本： 备课_linux

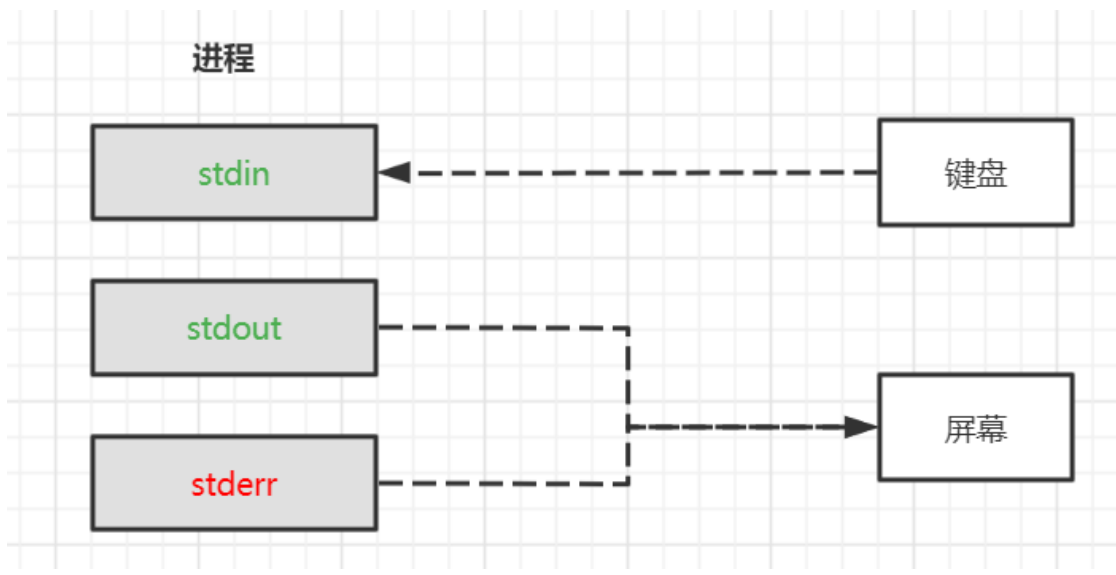
创建时间： 2022/4/17 8:52

更新时间： 2022/4/28 11:03

作者： 兰鸣人花道

1、标准输入、标准输出、标准错误

- 执行一个shell命令行的时候都会自动打开三个标准文件
 - 标准输入stdin，通常对应终端键盘，程序默认从stdin读取数据
 - 标准输出stdout，程序默认向stdout写入/输出数据
 - 标准错误输出stderr，程序会向stderr中写入错误信息
 - stdout和stderr，都对应终端的屏幕
- 文件描述符
 - stdin， 对应0
 - stdout， 对应1
 - stderr， 对应2
- 进程将从标准输入文件中得到输入数据，将正常输出数据输出到标准输出文件，而将错误信息送入标准错误文件中



2、重定向

- 输出重定向
 - 有时候，不想输出到终端上/屏幕上，命令结果需要被进一步处理
 - 这时可以把输出结果重定向到某个文件
 - 语法：
 - 命令 > 文件 也可以这样写 命令 1> 文件
 - 如果文件不存在，会创建出这个新文件
 - 如果文件存在，会覆盖掉以前的内容，以前的内容都不在了，所以很危险，慎用!!!
 - 命令 >> 文件 也可以这样写 命令 1>> 文件
 - 如果文件不存在，会创建出这个新文件
 - 如果文件存在，会追加内容到原来的文件，不会对原来的文件进行覆盖
 - 举例：
 - ps -ef > myfile 表示把ps -ef 的结果保存到myfile中去
 - 输出重定向会覆盖文件内容，如果不想覆盖文件内容，请用>>

- 错误重定向
 - 默认情况下
 - 命令 `> 文件`, 表示会将输出重定向到`file`
 - 命令 `< 文件`, 表示会将`file`内容作为输入
 - 如果希望将错误重定向到文件, 可以这样写
 - 命令 `2> file`
 - 命令 `2>> file`
 - 同时重定向输出与错误
 - 同时将输出与错误重定向到同一个文件
 - 语法
 - `COMMAND > file 2>&1` : 把标准错误和标准输出都重定向到`file`中
 - 比如, `ps -ef > out.log 2>&1`
 - 同时将输出与错误重定向到不同的文件
 - 同时将输出与错误重定向到不同的文件
 - 语法
 - `COMMAND > file1 2>file2` : 把标准输出重定向到`file1`, 把标准错误输出重定向到`file2`
 - 比如, `ps -ef > out.log 2>err.log`
- 输入重定向
 - 命令也可以从文件获取输入, 这就是输入重定向
 - 语法
 - 命令 `< 文件`
 - 比如, `rm -i file01 < file02` 表示从`file02`文件中读取内容作为命令的输入

```
# 表示将执行aaa.sh脚本的正确输出(输出重定向)到file1文件, 将标准错误输出到error1文件, 这是两个不同的文件
[root@localhost ~]# sh aaa.sh > file1 2> error1
[root@localhost ~]# cat file1
stdout ....
[root@localhost ~]# cat error1
cat: xxxxxxxxxxxx: No such file or directory
[root@localhost ~]# cat aaa.sh
#!/bin/sh
```

```
echo "stdout ...."
cat xxxxxxxxxxxx
[root@localhost ~]#

# 将输出重定向和输出错误都放到同一个文件
[root@localhost ~]# sh aaa.sh >name1 2>&1
[root@localhost ~]# cat name1
stdout ....
cat: xxxxxxxxxxxx: No such file or directory
[root@localhost ~]#
```

实例1、删除文件的时候要输入一个y

```
[root@lanhai link_test]# echo "y" > input_test
[root@lanhai link_test]# cat input_test
y
[root@lanhai link_test]# rm hello < input_test
rm: remove regular empty file 'hello'? [root@lanhai link_test]#
[root@lanhai link_test]#
```

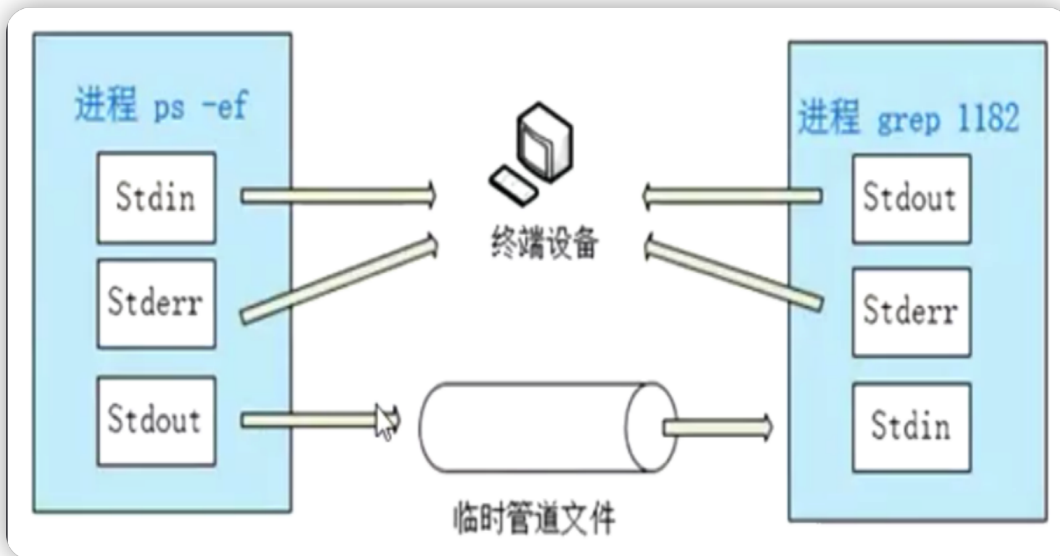
实例2, 可视化输入, 然后写入文件

```
[root@lanhai link_test]# cat >myfile <<EOF
> hello world
> hello python
> hello java
> EOF
[root@lanhai link_test]# cat myfile
hello world
hello python
hello java
[root@lanhai link_test]#
```

EOF表示一个输入结束的标志，后面我输入完成了，就用EOF，就结束输入了

3、管道

- 将一个命令的输出的内容，作为下一个命令的输入
- 比如先用`ps -ef` 查看系统所有的进程，然后再用`grep 888` 过滤id为1182的进程
- 可以这样写
 - `ps -ef > tmp_file`
 - `grep 1182 tmp_file`
 - 连招如下：
 - `ps -ef | grep 1182`
- 这里的 `|` 就是管道符，表示将`ps -ef` 的输出作为 `grep 888`的输入
- 管道原理示意图
 - 将前面的`ps -ef` 命令的stdout(这个本来是输出到终端设备里的) 重定向到一个临时的管道设备里
 - 同时将后一个命令`grep 1182`的stdin重定向到这个临时的管道设备里



- 管道的一些用法
 - 连续使用
 - `ps -ef | grep python | wc -l`

4、环境变量

- 变量?
- windows中的环境变量?
- linux中为什么不指定绝对路径就可以执行?
- linux中的环境变量
 - 多用户操作系统

- 每个用户登录系统后，都会有一个专用的运行环境，比如\$HOME
 - 这个运行环境就是用一组环境变量来定义的
 - 用户也可以对自己的运行环境做一个定制，方法就是修改对应的系统环境变量
- 相关环境变量的命令
 - env
 - printenv
- 场景的环境变量
 - HOME
 - PWD
 - PATH
 - SHELL
 - HISTSIZE，保存历史命令大小
 - ...
- 查看环境变量
 - echo \$环境变量名
- 环境变量PATH的作用
 - 当输入命令的时候，linux会去查找PATH里面记录的路径
 - 根目录下可以输入ls，在/usr下也可以输入ls
 - 但是ls命令不在这些目录下，但是为什么能使用呢？
 - 就是说输入命令的时候，linux会去/bin，/usr/bin，/sbin等这些目录下查找输入的命令
 - 而PATH的值就是/bin:/usr/bin:/sbin:...
 - 冒号:是分隔符，将目录分割开
- 配置环境变量
 - 临时生效
 - 设置环境变量的命令export
 - 比如修改PATH的值，增加一个/home/tom
 - export PATH=\$PATH:/home/tom
 - echo \$PATH
 - 直接在终端里执行这个命令，只是临时生效，重新登录系统/复制一个会话就失效
 - 永久生效
 - 如果要永久生效，就要修改配置文件
 - /etc/profile
 - 将export PATH=\$PATH:/home/tom 追加到配置文件的末尾，最后保存退出
 - /etc/profile 对所有用户生效
 - ~/.bashrc 对当前用户生效
 - 执行如下命令使配置生效
 - source /etc/profile
 - . /etc/profile
- 实例一
 - 在/home/tom目录下有个test.sh脚本
 - 为什么在/home/tom路径下直接运行test.sh命令无法执行
 - 查看环境变量的值： echo \$PATH
 - 修改环境变量
 - 再次执行
- 实例二
 - 新增环境变量
 - vi /etc/profile
 - DIR=/var/charles
 - source /etc/profile
 - cd \$DIR

```
# 自己先创建了一个很长的路径后
[root@localhost dir0004]# pwd
/root/dir001/dir0001/dir0002/dir0003/dir0004
[root@localhost dir0004]# cd

# 切换到了别的目录，想进入那个很深层级的目录
[root@localhost ~]# cd /tmp
[root@localhost tmp]# cd /root/dir001/^C

# 在/etc/profile文件里面加了环境变量的配置
[root@localhost tmp]# vi /etc/profile
```

```
# 在没有source或者.重新加载配置文件的情况下，不生效
[root@localhost tmp]# cd $mydir

# 重新加载配置文件之后，就可以直接cd $dir的方式快速进入我那个目录
[root@localhost ~]# source /etc/profile
[root@localhost ~]# cd $mydir
[root@localhost dir0004]# pwd
/root/dir001/dir0001/dir0002/dir0003/dir0004
[root@localhost dir0004]# tail -5 /etc/profile

export PATH=$PATH:/root/

mydir=/root/dir001/dir0001/dir0002/dir0003/dir0004
export mydir
[root@localhost dir0004]#
```

- 实例三
 - 临时修改环境变量，把ps、ls等命令临时取消了
 - 这时只能使用绝对路径了
- 实例四
 - 新增一个路径，手写一个ls可执行文件，放在PATH路径最前面
 - 那么ls用不了

作业：