

Week05 Assignment

阅读教材第三章内容并查阅网络资料，回答下列问题。

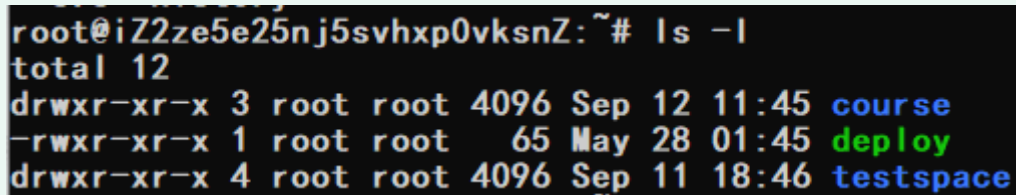
1. 权限

在使用 `ls -l` 命令后，会列出目录的很多信息，例如下述例子

```
[me@linuxbox ~]$ ls -l
total 56
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Desktop
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Documents
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Music
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Pictures
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Public
drwxrwxr-x 2 me me 4096 2007-10-26 17:20 Templates
```

请依次完成以下问题

在任一有文件的目录下使用 `ls -l` 命令并截图



```
root@iZ2ze5e25nj5svhxp0vksnZ:~# ls -l
total 12
drwxr-xr-x 3 root root 4096 Sep 12 11:45 course
-rwxr-xr-x 1 root root 65 May 28 01:45 deploy
drwxr-xr-x 4 root root 4096 Sep 11 18:46 testspace
```

在展示的内容中，第一个字段 `drwxrwxr-x` 中每一位的含义

共10位，其中：

第 1 个字符表示文件或目录的类型，其类型包括：

- p, 表示命名管道文件；
- d, 表示目录文件；
- l, 表示符号连接文件；
- -, 表示普通文件；
- s, 表示 Socket 文件；
- c, 表示字符设备文件；
- b, 表示块设备文件；

第 2 - 4 个字符表示文件或目录的所有者权限；

第 5 - 7 个字符表示文件或目录的所有者同组用户权限；

第 8 - 10 个字符表示文件或目录的其他用户权限；

当我们以普通用户身份进行一些操作时会遇到 `Permission denied` 的情况，为了解决这个问题，我们有两种办法，分别是切换到 `root` 用户，或者仅让单条指令以最高权限运行，这两个操作分别需要哪两个命令

```
su root
sudo <command>
```

当我们需要更改文件的权限(读取、写入和执行)时需要用到哪个命令

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

2. 环境变量

要实现 `echo ${name}` 的输出为 `super 2021/2121 你的学号`，需要事先用什么命令定义环境变量

```
name='super 2021 19377167'
```

3. 重定向

现有 C 文件 `test.c` 内容如下

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Number of args: %d , Args are:\n", argc);
    int i;
    for(i=0; i<argc; i++)
        printf("args[%d] %s\n", i, argv[i]);
    fprintf(stderr, "This is the message sent to stderr\n");
}
```

要将其转换成为可执行文件 `test` 需要执行什么命令

```
gcc -o test test.c
```

运行 `test` 文件，并要实现重定向输入 `test.in`，重定向输出 `test.out`，重定向错误 `error.log` 的命令为

```
test <test.in >test.out 2>error.log
```

先后执行 `./test abc 235 2>a.txt` 和 `./test abc 234 >>a.txt`，屏幕上和 `a.txt` 的内容为什么？

屏幕上的内容为

```
Number of args: 3 , Args are:
args[0] ./test
args[1] abc
args[2] 235
This is the message sent to stderr
```

`a.txt` 中的全部内容为

```
This is the message sent to stderr
Number of args: 3 , Args are:
args[0] ./test
args[1] abc
args[2] 234
```

4. 管道

请写出命令 `who | wc -l` 的结果并写出管道的用途，或者分析管道的执行过程。

结果：2

分析：

```
who > .tmp
wc -l < .tmp
```

5. Shell 编程

1. 简述 shell 中的特殊字符

转义符 `\`：使用某符号的字面意义而非其特殊含义。

单引号 `'`：其内部所有字符保持字面含义。

双引号 `"`：除了 `$`，```，`\` 特殊符号外的所有其他字符保持字面含义。

命令替换符号（反引号）：包裹内容执行后字符串，其可被 `$(command)` 替代。

示例：

```
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo #abc

root@iZ2ze5e25nj5svhxp0vksnZ:~# echo \#abc
#abc
root@iZ2ze5e25nj5svhxp0vksnZ:~# string=hello
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo $string
hello
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo '$string'
$string
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo '`'
> ^C
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo "$string"
hello
```

```
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo "$string\""  
hello"  
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo "$string\"#"  
hello\#  
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo $(pwd)  
/root  
root@iZ2ze5e25nj5svhxp0vksnZ:~# echo `pwd`  
/root
```

2. Shell 变量有哪几类，各是什么情况下使用？

1) 用户变量

```
<var_name>=<var_value>
```

变量可不用声明直接使用或者赋值，用户定义的变量名由字母和下划线组成，并且变量名第一个字母不能为数字。可采用 `${var_name}` 方法使用用户变量，该风格建议使用，因为该用法不会产生歧义。

2) 环境变量

它是定义和系统工作环境有关的变量，用户也可重新定义该类变量。

3) 内部变量

Shell 内部变量是 Shell 所定义的用户只能使用而无法重新定义的变量，主要有以下几种。

#：Shell 程序位置参数的个数。

*****：脚本的位置参数内容，该位置从 1 开始。当*在双引号中即"\$*"时，所有的位置参数被扩展为以 IFS 分割的一个字符串，即 "\$1c\$2c..."，其中 c 为 IFS 变量值的第一个字符，\$1、\$2 分别代表第一个和第二个位置参数的值。

?：上一条前台命令执行后返回的状态值。命令执行成功与失败返回的值是不同的。

\$：当前 Shell 进程的进程 ID 号，该内部变量最常见的用途是作为暂存文件的名称，以保证不会重复。

!：最后一个后台运行命令的进程号。

0：当前执行的 Shell 程序的名称。

@：脚本的位置参数内容，该位置从 1 开始。"\$@"被扩展为 "\$1"、"\$2"

_：在 Shell 启动时，为正在执行的 Shell 程序的绝对路径。之后为上一条命令的最后一个参数。

附：参数扩展

变量=\${parameter:-word}：如果 parameter 未定义或者为 null，那么用 word 置换变量的值，否则用 parameter 置换变量的值。

变量=\${parameter:=word}：如果 parameter 未定义或者为 null，则用 word 置换 parameter 的值然后置换变量的值，否则用 parameter 置换变量的值。

变量=\${parameter:?word}：如果 parameter 未定义或者为 null，word 被写至标准出错（默认情况下在屏幕显示 word 信息）然后退出，否则用 parameter 置换变量的值。

变量=\${parameter:+word}：如果 parameter 未定义或者为 null，则不进行置换，即变量值也为 null。否则用 word 置换变量的值。

3. C 语言中的分支语句有 `if-else` 和 `switch-case`，Shell 中是否有相似的语句与其对应？如何使用？

`if-else`

```
if [ _COND_ ]; then
    _COMMANDS_;
elif [ _COND_ ]; then
    _COMMANDS_;
else
    _COMMANDS_;
fi
```

示例:

```
x=1;
if [ $x -le 0 ]; then
    echo 'x<=0';
    echo $x;
elif [ $x -le 1 ]; then
    echo 'x<=1';
    echo $x;
else
    echo 'x>1';
fi
```

`switch-case`

```
case _VAR_ in
    _EXPRESSION_)
        _COMMANDS_
        ;;
    _EXPRESSION_)
        _COMMANDS_
        ;;
    *)
        _COMMANDS_
        ;;
esac
```

示例

```
x=1;
case $x in
    1|3|5|7|9)
        echo odd;
        echo $x;
        ;;
    0|2|4|6|8)
        echo even;
        echo $x;
        ;;
    *)
        echo unknown;
        echo $x;
        ;;
```

```
esac
```

4. C 语言中的循环语句有哪些？Shell 中是否有相似的语句与其对应？

while(){}

```
until [ _COND_ ]; do
    _COMMAND_;
done
```

或者

```
while [ _COND_ ]; do
    _COMMAND_;
done
```

其中，until 表示执行到 true 时退出循环，while 表示 false 时退出。

for(auto item : list)

```
for name in 参数列表
do
    _COMMAND_;
done
```

5. 写一个计算 $n!$ 的 Shell 脚本 **frac.sh**

示例输入输出：

```
[root@localhost ~]$ ./frac.sh 0
1
[root@localhost ~]$ ./frac.sh 4
24
```

你的脚本内容：

```
x=1;
i=1;
while [ $i -le $1 ]; do
    x=$((x*$i));
    i=$((i+1));
done
echo $x;
```