Linux 操作系统

# Shell 的输入与输出

# Shell 的输入与输出

- **shell** 的输入与输出主要包括：
  - **`echo`** 命令
  - **`read`** 命令
  - **`tee`** 命令
  - **`cat`** 命令
  - 管道
  - 重定向

# echo 命令

## ❑ echo

- ◆ 使用 **echo** 命令可以显示文本行、字符串或变量的值
- ◆ **echo** 命令的一些细节在 System V、BSD 和 Linux 这三种UNIX-like 系统上会有所不同，这里以 Linux 为主。
- ◆ echo 命令的一般形式：

```
echo [-e] [-n]  string
```

其中：

**string**：字符串，可以含 **shell** 变量、转义符等，
一般用双引号括起来

**-e**：让 **echo** 解释 **string** 中的转义符

**-n**：禁止 **echo** 输出后输出 **NEWLINE**（换行）。

# echo 命令

- **echo** 命令支持的转义符

| \num | ASCII码为num（八进制）的字符 | | |
|---|---|---|---|
| \a | alert (bell) 响铃 | \r | carriage return 回车 |
| \b | backspace 退格 | \t | horizontal tab<br>水平制表符 |
| \f | form feed 换页 | | |
| \c | suppress trailing newline 不换行 | \v | vertical tab<br>垂直制表符 |
| \n | new line 换行 | \\ | backslash 反斜杠 |

```
echo -e  "Hello \bworld"
```

# echo 命令举例

```
echo "your home directory is $HOME"
```

```
echo –n "your home directory is $HOME"
```

```
echo –e "your home directory is $HOME\c"
```

```
echo -e "User: $USER\tUID: $UID"
```

# read 命令

- 从键盘或文件的某一行中读取输入，并将其赋给变量。
- **read** 命令的一般形式：

```
read  variable1 variable2 ...
```

```
read -p "提示信息"  var1 var2 ...
```

- 如果只指定了一个变量，**read** 将会把输入行的所有内容赋给该变量，直至遇到第一个文件结束符或回车。

- 如果指定了多个变量，**read** 用空格作为分隔符把输入行分成多个域，分别赋给各个变量。如果输入的文本域数量多于 **read** 给出的变量数，**read** 将所有的超长部分赋予最后一个变量。

# read 命令举例

```
read name   // John Lemon Doe
```

```
read name subname   // John Lemon Doe
```

```
#!/bin/bash
echo –e "First name: \c"
read name
echo –e "Middle name: \c"
read middle
echo –e "surname: \c"
read surname
echo "the name is $name $middle $surname"
```

# cat 命令

◆ **cat** 是一个简单而通用的命令，可以用它来显示文件内容，创建文件，还可以用它来显示控制字符。

◆ **cat** 命令的一般形式：

```
cat [-n][-b][-t][-e] file1 file2 ...
```

**-n**：显示行号

**-b**：显示行号（不含空行）

**-t**：显示制表符

**-e**：显示行结束符

◆ 创建文件：

```
cat file1 file2 > newfile
```

合并文件

```
cat > newfile
```

输入文本，按 **ctrl+d** 结束输入

# 管道

- **|**：把一个命令的输出传递给另一个命令作为输入。

```
comd1 | comd2
```

**例：** 显示当前目录下的所有子目录

```
ls -l | grep ^d
```

# tee 命令

- 把输出的一个副本输送到标准输出，另一个副本拷贝到相应的文件中。
- **tee** 命令的一般形式：

```
tee [-a] filename
```

**-a**：追加到文件末尾

- **tee** 命令一般与管道结合使用
- 例：

```
ls  | tee list.out
```

# 标准输入、输出和错误

- 标准输入（**STDIN**），文件描述符为 **0**
- 标准输出（**STDOUT**），文件描述符为 **1**
- 标准错误（**STDERR**），文件描述符为 **2**

# 重定向

| | |
|---|---|
| **comd > filename** | **comd >> filename** |
| **comd 2 > filename** | **comd 2 >> filename** |
| **comd > filename 2>&1** | **comd >> filename 2>&1** |
| **comd < filename** | |
| **comd < filename >filename2** | |
| **comd << delimiter**<br>从标准输入中读入，直至遇到 **delimiter** 分界符 | |
| **comd <&m** 把文件描述符 **m** 作为标准输入 | |
| **comd >&m** 把标准输出重定向到文件描述符 **m** 中 | |
| **comd >& filename** 重定向标准输出和错误到指定文件 | |
| **comd <&-** 关闭标准输入 | |

# 命令的执行顺序

- 在执行某个命令的时候，有时需要依赖于前一个命令是否执行成功。例如，假设你希望将一个目录中的文件全部拷贝到另外一个目录中后，然后删除源目录中的全部文件。在删除之前，你希望能够确信拷贝成功，否则就有可能丢失所有的文件。

- 如果希望在成功地执行一个命令之后再执行另一个命令，或者在一个命令失败后再执行另一个命令，**&&** 和 **||** 可以完成这样的功能。

# 命令的执行顺序

- 使用 &&

```
comd1 && comd2
```

&& 左边的命令（comd1）返回真（即返回 0，成功被执行）后，&& 右边的命令（comd2）才能够被执行。

- 使用 ||

```
comd1 || comd2
```

如果 || 左边的命令（comd1）未执行成功，那么就执行 || 右边的命令（comd2）。

# ( ) 和 { } 的使用

- 几个命令合在一起执行，可以使用下面两种方法

```
( comd1; comd2; ... )
```

```
{ comd1; comd2; ... }
```

- ()、{} 一般和 && 或 || 一起使用

```
cp file1 file2 || \
( echo "cp failed" | mail jypan; exit; )
```

- 在编写 shell 脚本时，使用 && 和 ||，可根据前面命令的返回值来控制其后面命令的执行，对构造判断语句很有用。