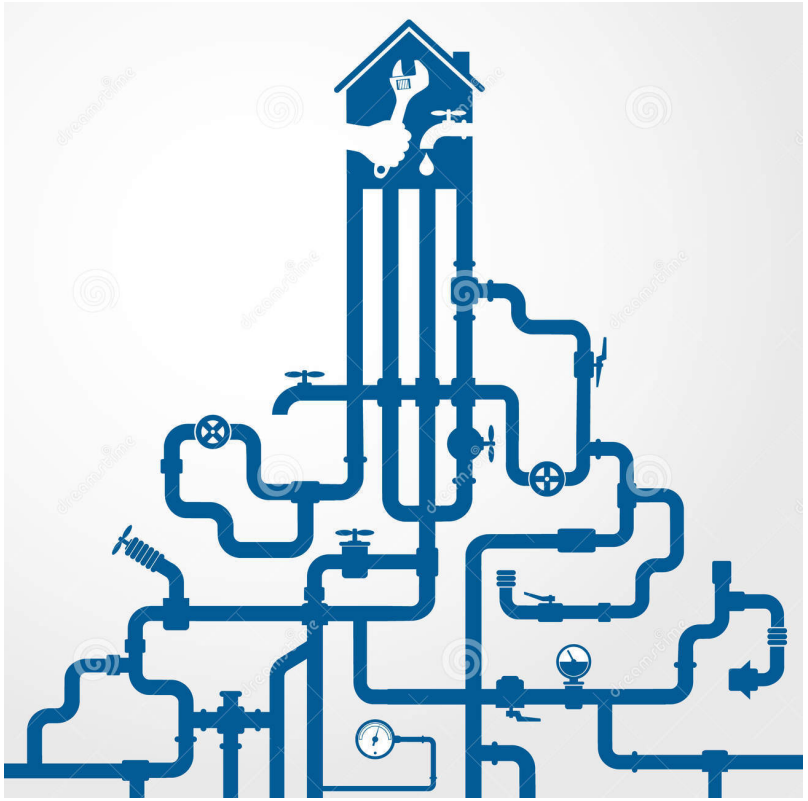
高阶Linux指令

Small is beautiful

- 管道
- 过滤器指令
  - 比较，抽取，统计
  - 选取，排序，组合，变换



Unix 设计准则

- 共性问题
  - 所有的程序都需要在屏幕上进行输出
  - 输出的内容超过一屏，需要分屏显示
  - 某些时候，还需要将输出结果写到文件中去

Unix 设计准则

- 解决方法：
  1. 所有程序中都包含自己的分屏显示和保存功能
     - 加重了程序员的负担
  2. 提供一个单独的工具来分屏显示（`more`），提供一个单独的工具来保存文件（`>`）
     - 保证工具的简单性
     - 程序员可以更专注地实现这个工具
     - 用户只要学会使用一个工具

Unix 设计准则

- 每个程序或者命令应该是一个工具，它只完成一件事情，但一定要完成好这件事情
- 当需要新工具时，最好对现有工具进行组合，而不是编写一个新工具

Small is beautiful

~~Bigger than Bigger~~

管道与过滤器

```
$ cat file1 file2 file3 | grep linux | wc -l
```

pipe：一个程序的标准输出成为另一个标准输入

- **标准输出与错误连在一起，则 `$ foo 2>&1 | bar`**

管道与过滤器

```
$ cat file1 file2 file3 | grep linux | wc -l
```

pipeline
（管道线）

pipeline

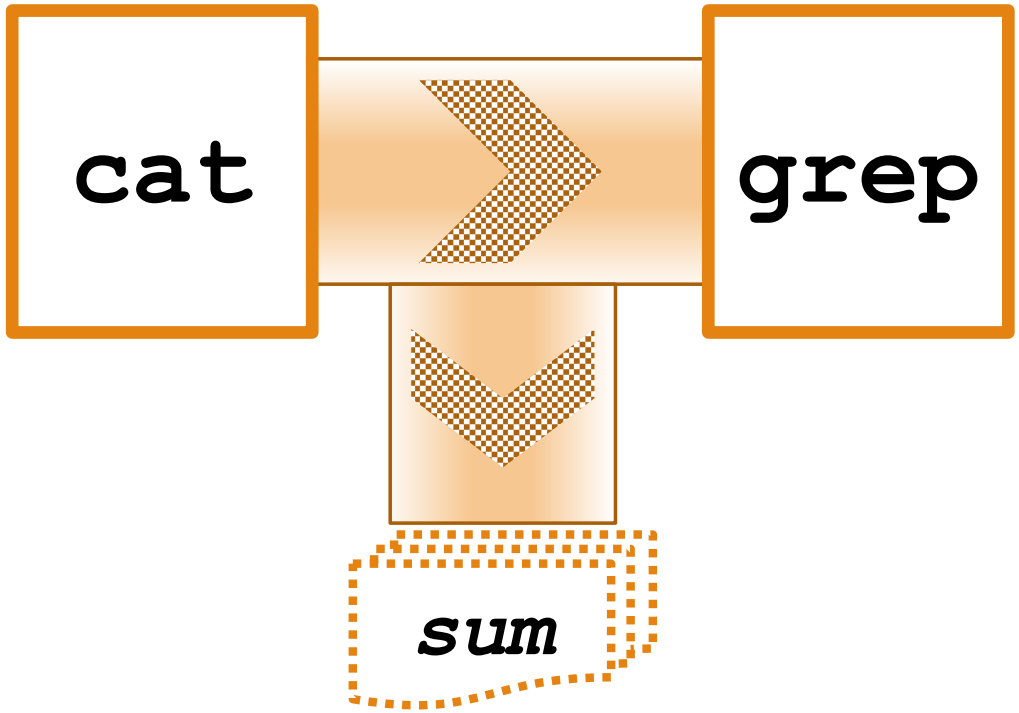
filter
（过滤器）

filter

filter

管线分流： tee

- 既保存结果到文件，又发送到其它过滤器
  - `$ cat file1 file2 file3 | tee sum | grep linux`



`$ who | tee status`

- 既显示命令结果，又能将结果保存到状态文件中去

用Unix来解决问题

- 学习使用 **Unix** 过滤器
- 分解问题**(Think!)**
- 选择工具**(RTFM)**
- 与他人讨论**(Google! Google! Google!)**
- 选择选项

常见的Unix 过滤器

过滤器	参阅	作用	重要性
awk	perl	编程语言：操作文本	★★★★
cat	split tac	组合文件	★★★★★
colrm	cut join	删除指定数据列	★★
comm	cmp diff	比较两个有序文件	★★
cmp	comm diff	比较两个文件	★★★
cut	colrm join	抽取指定字段列	★★★★
diff	cmp comm	比较两个文件，显示不同	★★★

常见的Unix 过滤器

过滤器	参阅	作用	重要性
expand	unexpand	将制表符转为空格	★★
fold	fmt pr	将长行转为短行	★★
fmt	fold pr	格式化段落	★★
grep	strings	查询包含指定模式的行	★★★★★
head	tail	显示数据的开头几行	★★★★
join	colrm cut	基于公用字段，组合数据列	★★
look	grep	选择以指定模式开头的行	★★★

常见的Unix 过滤器

过滤器	参阅	作用	重要性
nl	wc	创建行号	★★
paste	colrm cut	组合数据列	★★
perl	awk	编程语言： 操作文件，进程	★★★
pr	fold fmt	将文本格式化为页或列	★★
rev	cat tac	将每行数据的字符反序排列	★★
sdiff	cmp diff		
sed	tr	非交互式文本编辑	★★★★

常见的Unix 过滤器

过滤器	参阅	作用	重要性
sort	uniq	排序数据；检查是否有序	★★★★
split	cat	将大文件分割成小文件	★★★
strings	grep	在二进制文件中搜索字符串	★★★
tac	cat rev	组合并反序文件	★★★
tail	head	显示数据末尾行	★★★★
tr	sed	改变或删除选定的字符	★★★★
tsort	sort	根据偏序创建全序	★★★

常见的Unix 过滤器

过滤器	参阅	作用	重要性
`unexpand`	`expand`	将空格转变为制表符	★★
`uniq`	`sort`	选择重复/唯一行	★★★
`wc`	`nl`		★★★★

RTFM！

cat split ; tac rev

- **$ split [-d] [-a *num*] [-l *lines*] [*file* [*prefix*]]**
  - **file**: 目标文件;
  - **prefix**: 拆分后的文件名前缀
  - **-l**: 各拆分文件的行数（默认为1000行）
  - **-d**: **digits**, 拆分文件名后缀用数字表示
  - **-a**: 文件名后缀的长度

cat split ; tac rev

- `$ for i in {1..50000}; do echo "hello, this is line $i" >> hello_split; done`
- `$ cat hello_split | split - -d small_`

  `# 将 hello_split 文件分割为若干以“small_”为文件名前缀的小文件`
- `$ tac hello_split | rev | head`

  `# 将hello_split 文件从尾至头，且逐行反转，显示前10项`

pwd: ~/demo/ch07/filter

文件的比较

过滤器	作用	文件类型	重要性
cmp	比较两个文件	二进制或文件	★★★
comm	比较两个有序文件，显示区别	文件：有序	★★
diff	比较两个文件，显示区别	文本	★★★
sdiff	比较两个文件，显示区别	文本	★★

cmp

- **$ cmp *cmp1 cmp2***
- 逐字节地比较两个文件
- 比较在第一个不同处停止
- 显示不同的行号，及字节号
  - 每行末尾存在 '\n'

./ch07/filter/cmp1

```
a b c d e f g
h i j k l m n
o p q   r s t
u v w   x y z
```

./ch07/filter/cmp2

```
a b c d e f g
h i j   l m n
o p q   r s t
o p     r s t
u v w   x y z
```


pwd: ~/demo/ch07/filter

cmp

- **cmp** 是逐字节地比较两个文件，因此它可以比较任意类型的文件：文本，图片，程序，音乐 . . .
- 通常用于快速查询两个文件是否一致

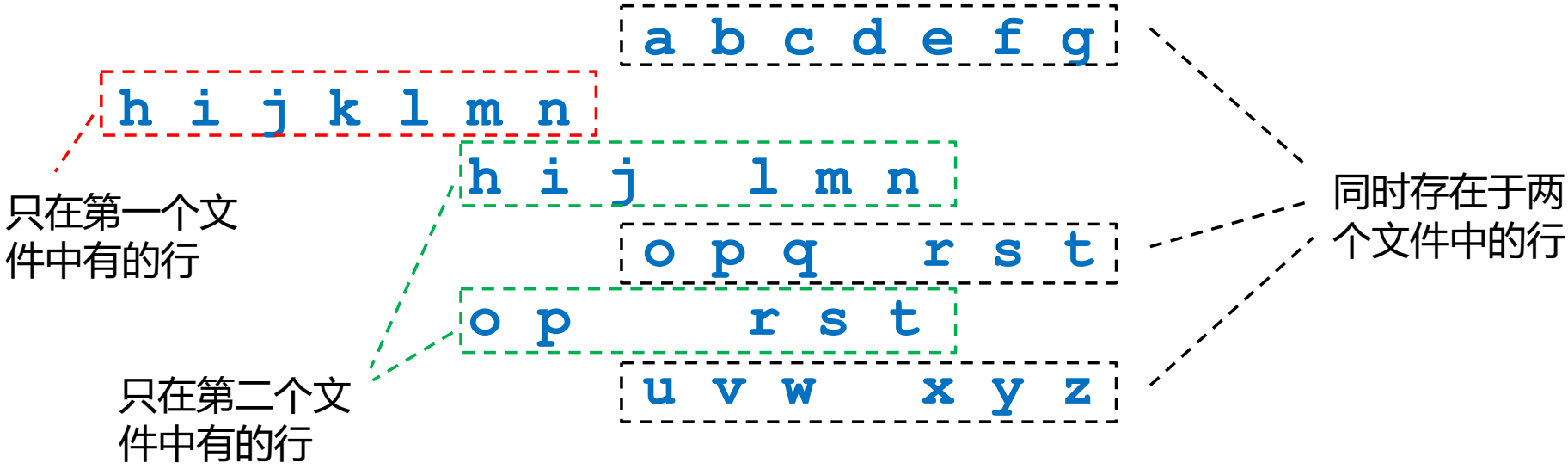
comm

- **`$ comm [-123] file1 file2`**
  - `-1,2,3`： 去除相应的结果列
- 逐行进行两个有序文件的比较
- 结果分三列显示
  1. 只在第一个文件中有的行
  2. 只在第二个文件中有的行
  3. 在两个文件中都有的行

朋友圈
交易记录
统计数据
歌曲列表
… …

comm

- 若原始文件没有排序，可通过进程替换来实现排序
- `$ comm <(sort cmp1) <(sort cmp2)`
  - 进程替换：创建临时文件，来接收进程结果



pwd: ~/demo/ch07/filter

diff; sdiff; patch

- 很多文件（如源代码）是无序的
- **comm** 将比较结果都打印出来，对大文件容易混乱
- **$ diff [OPTION]... FILES**
- 逐行比较无序文件

diff; sdiff; patch

- **$ diff cmp1 cmp2**

```
2c2
< h i j k l m n
---
> h i j   l m n
3a4
> o p     r s t
```

- **c:** change
- **d:** delete
- **a:** append

./ch07/filter/cmp1

```
a b c d e f g
h i j k l m n
o p q   r s t
u v w   x y z
```

./ch07/filter/cmp2

```
a b c d e f g
h i j   l m n
o p q   r s t
o p     r s t
u v w   x y z
```

diff 结果：

将第一个文件修改为第二个文件所需遵循的指示

diff; sdiff; patch

- `-i`：忽略大小写
- `-w`：忽略空白符
  - `"xx" "x  x"` 相同
- `-b`：忽略空白符数量
  - `"xx" "x  x"` 不同
  - `"x x" "x  x"` 相同
- `-B`：忽略空白行

diff; sdiff; patch

- sdiff: diff for mortal; diff -y
- $ sdiff cmp1 cmp2

```
a b c d e f g                 a b c d e f g
h i j k l m n               | h i j   l m n
o p q   r s t                 o p q   r s t
                            > o p     r s t
u v w   x y z                 u v w   x y z
```


pwd: ~/demo/ch07/filter

diff; sdiff; patch

- `$ diff foo-2.0.c foo-2.1.c > foo-diff-2.1`
- `foo-diff-2.1`：两个文件间的差分（`diff`）
- 利用差分文件，结合`patch`命令，将基础文件恢复到任意版本
- 进行软件更新分发时，只需发布差分（补丁）文件即可

cut; paste

- **$ cut -c list [*file...*]**
- 按字符，从文件中抽取指定列
- **$ cut -c2 hello_cut**
- **$ cut -c2-5 hello_cut**
- **$ cut -c2,5 hello_cut**

./ch07/filter/hello_cut

```
123456789
123456789
123456789
123456789
123456789
123456789
123456789
123456789
123456789
123456789
123456789
123456789
```

cut; paste

- 适用于程序读取的文件包含：
  - 记录(record, tuple)：一行仅存储一条数据信息
  - 字段(field, attribute)：每个记录由若干部分组成
  - 定界符(delimiter, separator)：字段间的分隔字符
    - `$ cut -d DELIM # DELIM : 指定间隔符`

CSV (Comma-separated value) file

```
1   , 234, 56, 789
12 , 34 , 5 , 6789
123, 456, 7,  89
12 , 34 , 56, 789
123, 45 , 67, 89
```

cut; paste

- **/etc/passwd** 存放着所有用户帐号的信息
- 文件包含若干由7个字段构成的记录，由：分隔
- **username:password:UserID:GroupID:comment:home directory:shell**
- **$ cut -f4 -d':' /etc/passwd | sort | uniq -c | grep -v "^[ ]*1 "**

  # 统计当前包含2个以上成员的用户组

cut; paste

- **`$ paste [-d char...] [file...]`**
- 用于组合多个文件的字段列
- **`-d`**：指定分隔符（默认为**TAB**）
  - 可以指定多个字符，然后**paste**将轮流使用每个定界符

WC

- **$ wc [-clw] [*file...*]**
- **-c (character)**：按字符计数
  - 字母、数字、标点符号、空格、**TAB**、新行字符
- **-w (word)**：按单词计数
  - 连续字符，以空格、**TAB**、新行字符间隔
- **-l (line)**：按行计数

sort; uniq

- **`$ sort [-dfnru] [-o outfile] [infile...]`**
- 对输入文件进行排序
- **Note!!** **`$ sort file > file`**
  - 会将文件置空，因为 `>` 会将输出文件先置空，再运行命令
- **`$ sort -o file file`**

sort; uniq

- **-d (dictionary)**：只查看字母、数字、空白符（含空格和**TAB**），忽略标点符号等
- **-f (fold)**：忽略大小写
- **-n (numeric)**：依据行开头的数字数字排序
  - `$ echo -e "1\n5\n10\n2\n110"  | sort -n`
- **-r (reverse)**：反向排序

sort; uniq

- 早期计算机多采用的是 **ASCII** 来对字符进行编码
  - **#0 ~ #31** ： 控制字符； **#9** 为 **TAB**
  - **#32 ~ #126** ： 可显示字符
  - **#127** ： **null**
- 排序顺序的依赖于**ASCII**中的字符顺序

sort; uniq

- **ASCII** 字符顺序的基本原则：
  1. 空格最前
  2. 紧跟数字
  3. 先大后小
- **Try this !!**
  - `$ echo -e "A\na\nB\nb"  | sort`

sort; uniq

- **ASCII** 字符顺序的基本原则：
  1. 空格最前
  2. 紧跟数字
  3. 先大后小

- **Try this !!**
  - `$ echo -e "A\na\nB\nb"  | sort`

sort; uniq

- **POSIX 1003.2** 标准开始，支持区域设置 **locale**
- 区域设置包含了一系列的环境变量，用于定义：
  - 标识语言，日期格式，时间格式，货币符号，**排序序列**
- **$LC_COLLATE** ： 决定了排序序列
- **$ locale # 查看当前系统的区域设置**

sort; uniq

- **LC_COLLATE="en_US.UTF-8"**
  - **COLLATE: collation order 校对顺序**
- **en.US 是美国英语的区域设置，其排序方式采用的是字典排序序列(dictionary collating sequence)：**
  - **aAbBcCdD...zZ**
- **C 是C语言设置，排序依据于ASCII字符编号**

sort; uniq

- `$ echo -e "A\na\nB\nb"  | sort`
- `$ export LC_COLLATE=C`
- `$ echo -e "A\na\nB\nb"  | sort`

sort; uniq

- **$ uniq [-cdu] [*infile* [*outfile*]]**
- 查找连续重复的行
- **echo -e "a\na\na\nb\nc" | uniq**
- **-d: (repeated)** 只查看重复的行
- **-u: (unique)** 只查看非重复行
- **-c: (count)** 统计重复出现的次数

tr; sed

- **$ tr [-cds] [*set1* [*set2*]]**
- **tr (translate)** 转换，将输入中的所有**set1**中的字符，改变为相应**set2**集中的字符
- **tr** 没有输入文件参数，因为它是一个纯过滤器
- **$ echo -e "a\nb\nc" | tr "ab" "A"**

tr; sed

- 若想进行大小写替换，则可以：
- `$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ < hello_tr`
- `$ tr a-z A-Z < hello_tr`
- `$ tr [:lower:] [:upper:] < hello_tr`

pwd: ~/demo/ch07/filter

返回和换行

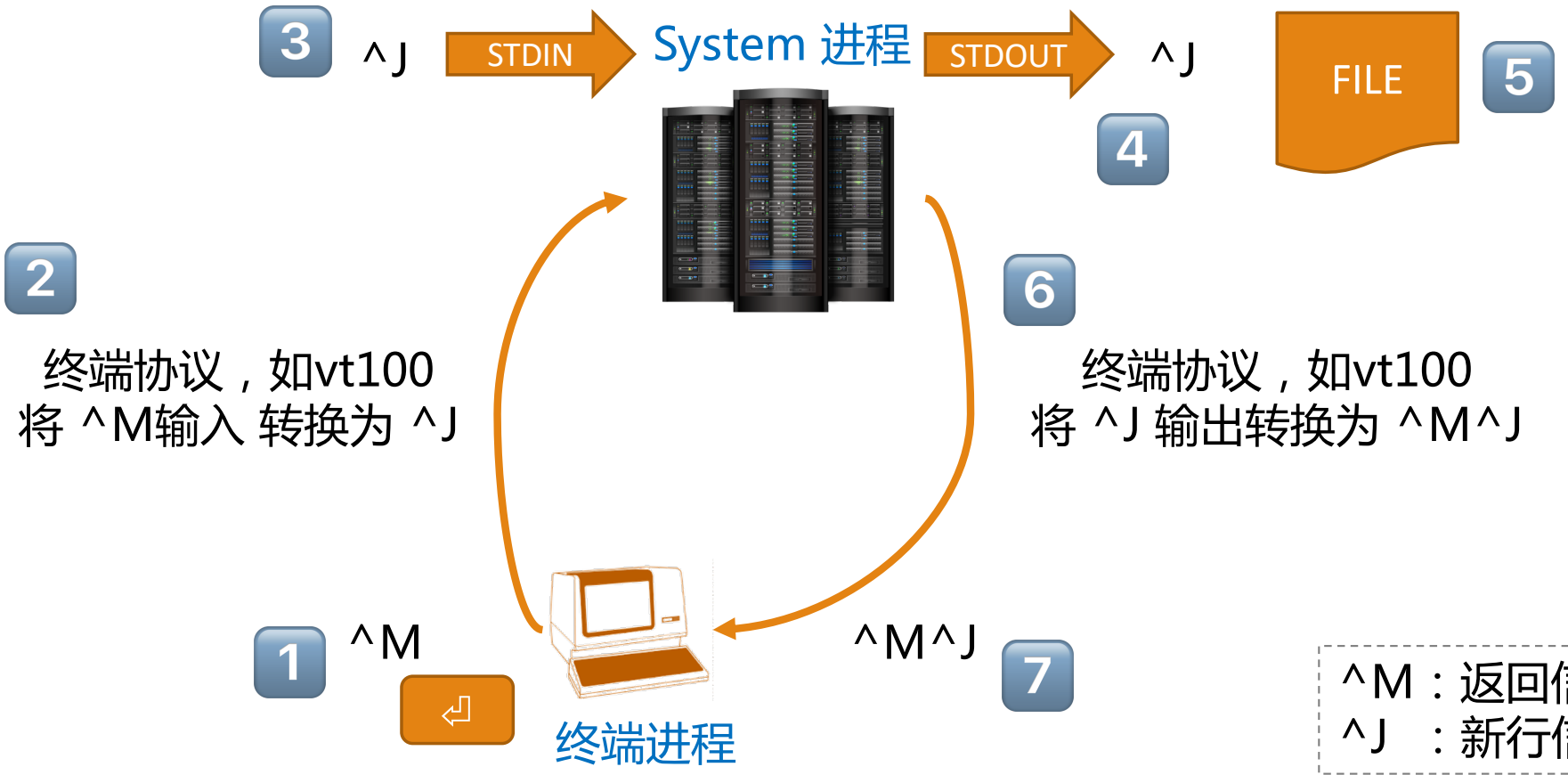
- Teletype ASR33中的两个键/信号
  - Return (CR: carriage return) ^M 托架返回
  - Linefeed (LF: line feed) ^J 新行/打印纸上移
- 使用返回与换行的场景
  - 终端上命令键入结束后，是 ^M， 即返回
  - 文本文件中的数据，每行结束后，是 ^J，即开启新行
  - 将文本数据显示在终端上时，是^M^J， 即返回并开启新行

Unix 下的返回换行处理方式

- 键入过程中，**<Return>**键（终端命令行下或文本编辑器内），其**^M**被改变为**^J**
- 将数据写到终端上时，**^J** 被改变为**^M^J**
- **STDIN**接收到的换行，永远是 **^J**
- **STDOUT**写到文件中，仍然是**^J**
- **STDOUT**写到终端上，才会是**^M^J**，以保证显示正确

Unix 下的返回换行处理方式

保存在文件中的永远是新行符 \n，不必担心数据是用于显示的，还是用于程序处理的



^M：返回信号
^J ：新行信号

返回和换行

- **Unix** 下的文本文件使用 **^J** 表示换行
- **Windows** 下的文本文件使用 **^M^J** 表示换行
- **Mac OS 9** 前的文本文件使用 **^M** 表示新行
- **Mac OSX** 下的文本文件使用 **^J** 表示换行

tr; sed

- 利用tr进行不可显示字符的转换
- ^M : \r (return)
- ^J : \n (new line)
- ^I : \t (tabulate)
- $ cat hello_tr_win.txt | tr '\r' 'R'

tr; sed

- **tr [-cds] [*set1* [*set2*]]**
- **-d: delete** 删除指定的字符
- **-c: complement** 补集，除指定字符外的所有字符
- **-s: squeeze** 替换字符集内的多个连续字符
  - **$ echo -e "a\nabcdefg\na\nb\na\nc" | tr -s [:lower:] "A"**

tr; sed

- **$ sed [-i] *command* | -e *command…* [file...]**
- 非交互式的文本编辑程序，因为 **sed** 从管道线中接收数据，也可被称为是流编辑器**(stream editor)**
- **sed command** 是**shell**无关的，非常强大
- **$ sed 's/abc/ABC/g' file**
- **-i : (in-place)** 修改结果直接写入原文件，**无法还原**
- **-e : (script expression)** 连接多个命令，形成**sed**脚本

Small is beautiful