终端命令和系统信息

主要内容

- 进程替换
- 终端命令序列
- 系统信息相关命令
- 编辑器vim

进程替换(Process Substitution)

```
filter <(cmd1) <(cmd2)
cmd1 > t1
cmd2 > t2
filter t1 t2
rm t1 t2
```

- 在支持命名管道(FIFO)或支持/dev/fd的系统中可以进行进程替换
- 进程替换的格式为： <list) 或>(list)
  - <或>与括号之间不能有空格；list为命令或管道
  - 替换为**一个文件名**，对应于命名管道的一端，另一端为list
  - 第一种格式>(list)表示往该文件中写，其内容作为list的输入
  - 第二种格式<(list)表示从该文件中读，读的内容来自于list的输出
- 如果一个命令只有一个参数采用进程替换，一般可采用传统的管道机制
- 进程替换是对于传统管道机制的扩展，常用于一个命令需要有多个输入或多个输出的情形，否则需要通过创建多个临时文件来实现

```
$ echo >(true) <(true)
/dev/fd/63 /dev/fd/62
$ cat <(ls -ld /bin/*)
$ cat <(ls -ld /bin/*) <(ls -ld /usr/bin/*) > files
```

```
$ ls -ld /bin/* | cat
$ ls -ld /bin/* >tmp1
$ ls -ld /usr/bin/* >tmp2
$ cat tmp1 tmp2 >files
$ rm tmp1 tmp2
```

终端能力(ESC序列)

- 终端可改变背景色(setab)、前景色(setaf)，设置字符的显示样式(加粗/下划线)、移动光标等
- 通过往终端设备中写入相应的ESC序列或其他特殊字符串执行终端命令
- 不同类型的终端所采用的终端命令序列稍微有所区别
  - 环境变量TERM设置为登录使用的终端类型
  - 终端能力(terminal capabilities)描述：ESC序列与终端能力的对应关系， man terminfo
  - infocmp [TERM] 查看（当前或指定）终端对应的终端能力文件

```
demo@mars:~$ infocmp linux
#       Reconstructed via infocmp from file: /lib/terminfo/l/linux
linux|linux console,
        am, bce, ccc, eo, mir, msgr, xenl, xon,
        colors#8, it#8, ncv#18, pairs#64,
        acsc=+\020\,\021-
\030.^Y0\333`\004a\261f\370g\361h\260i\316j\331k\277l\332m\300n\305o~p\304q\304r\304s_t\303u\264v\301w\
302x\263y\363z\362{\343|\330}\234~\376,
        bel=^G, blink=\E[5m, bold=\E[1m, civis=\E[?25l\E[?1c,
        clear=\E[H\E[J, cnorm=\E[?25h\E[?0c, cr=^M,
…
        rmul=\E[24m, rs1=\Ec\E]R, sc=\E7, setab=\E[4%p1%dm,
        setaf=\E[3%p1%dm,
sgr=%?%p9%t\E(0%e\E(B%;\E[0%?%p6%t;1%;%?%p5%t;2%;%?%p2%t;4%;%?%p1%p3%|%t;7%;%?%p4%t;5%;%?%p7%t;8%;m,…
```

```
$ echo -e '\E[32m'
# 设置文本为绿色
```

\E表示ESC字符
^G表示 Ctrl-G
%p1%d 表示这里为一个整数

终端命令序列: 光标位置相关

\033表示ESC字符
^H 表示<Ctrl-H>

ESC序列	名称	tput	含义
终端左上角的坐标对于tput为(0,0), 对于ESC序列为(1,1)		tput lines	终端的行数
		tput cols	终端的列数
\033[s	Save Cursor Position)	tput sc	保存光标位置
\033[u	Restore Cursor Position)	tput rc	恢复光标位置
\033[?25l	CIVIS	tput civis	隐藏光标，这里l为小写的L
\033[?25h	CNORM	tput cnorm	显示光标
\033[n;m H	CUP(Cursor Position)	tput cup Y X	光标移动到m行n列，编号从1开始，数字省略时为1
\033[nG	HPA(Cursor Horizontal Position Absolute)	tput hpa X	光标移动到n列，数字省略时为1
\033[nA	CUU(Cursor Up)	tput cuu N tput cuu1	光标向上移动n行，缺省为1
\033[nB 或^J	CUD(Cursor Down)	tput cud N tput cud1	光标向下移动n行，缺省为1
\033[nC	CUF(Cursor Forward)	tput cuf N tput cuf1	光标向前移动n列，缺省为1
\033[nD 或^H	CUB(Cursor Backward)	tput cub N tput cub1(可能删除字符)	光标向后移动n列，缺省为1

终端命令转义序列: 删除、插入和其他

ESC序列	名称	tput	含义
\033[?1049h	smcup	tput smcup	保存屏幕内容
\033[?1049l	rmcup	tput rmcup	恢复屏幕内容
\033[H\033[2J	Clear	tput clear	清除屏幕，光标到左上角
\033[nK	EL(Erase Line)	tput el tput el1	清除一行，n为0或无表示从光标到行尾，1表示光标到行首，2表示该行所有字符
\033[nJ	ED(Erase Display)	tput ed	清空屏幕, n=0或者无表示从光标开始到之后所有屏幕，n=1表示光标开始之前所有屏幕，n=2清空所有屏幕
\033[nX	ECH(Erase Chars)	tput ech N	删除n个字符
\033[n@	ICH(Insert Chars)	tput ich N	插入n个字符(空格)
\033[nL	IL(Insert Lines)	tput il tput il1	插入n行，n无表示1行
^G		tput bel	响铃
\033]0;title\007	OSC(operating system controls)		设置窗口标题为title

\033表示ESC字符， 如果使用echo –e命令，还可表示为\E \e
^G 表示<Ctrl-G>
\007 表示BEL字符,如果使用echo –e命令，还可表示为\a

系统缺省提示符PS1就是使用OSC设置窗口标题

终端命令转义序列:文本和颜色

ESC序列	名称	tput	含义
\033[<n>m	SGR(Select Graphic Rendition)	tput sgr arg1 … arg9 设置多个文本属性	m之前可包括**多个以;分隔的数字<n>**，这些数字分别设置各种图形参数（颜色等）
\033[0m	**sgr0**	**tput sgr0**	**恢复到缺省(正常)模式**
\033[1m	**bold**	**tput bold**	**加粗模式**
\033[2m	dim	tput dim	半加粗模式
\033[3m \033[23m	italic	tput sitm tput ritm	进入和退出斜体模式
\033[4m **\033[24m**	**underline**	**tput smul** **tput rmul**	**进入和退出下划线模式**
\033[5m	blink	tput blink	进入闪烁模式
\033[7m **\033[27m**	**standout or rev**	**tput smso或tput rev** **tput rmso**	**进入和退出突出模式或者反显(前景背景调换)模式**
\033[8m	invis	tput invis	进入不可见模式
\033[3<n>m	foreground color	tput setaf N	设置文本前景颜色(黑红绿黄蓝紫青白)
\033[4<n>m	background color	tput setab N	设置文本背景颜色

```
$ echo -e '\033[01;32mGreen and Bold\n\033[0mNormal'
Green and Bold
Normal
```

Color table[15]

Intensity	0	1	2	3	4	5	6	7
Normal	Black	Red	Green	Yellow[16]	Blue	Magenta	Cyan	White
Bright	Black	Red	Green	Yellow	Blue	Magenta	Cyan	White

执行终端命令

```
$ echo -e '\033[01;32mGreen and Bold\n\033[0mNormal'
Green and Bold
Normal
$ printf '\033[01;32mGreen and Bold\n\033[0mNormal\n'
```

- 往终端设备中写入相应的ESC序列或其他特殊字符串
- echo命令的-e选项支持转义方式，可输入ESC等特殊字符
- printf命令的格式化字符串中支持转义方式，%b可解释后面参数里面的转义方式

- 在Shell脚本中可用变量保存**没有解析过**的ESC序列字符串
- 在通过echo输出到终端时采用**-e选项，**解释成为ESC序列
- 最后由终端设备解释

```
#!/usr/bin/env bash
RED="\033[0;31m"
GREEN="\033[1;32m"
LIGHT_BLUE="\033[1;34m"
WHITE="\033[1;37m"
NO_COLOR="\033[0m"
UNDERLINE="\033[4m"
NO_UNDERLINE="\033[24m"
echo -e "I ${LIGHT_BLUE}love${NO_COLOR} Linux"
echo -ne ${GREEN}$UNDERLINE
echo "Green and UnderLine"
echo -ne ${NO_UNDERLINE}${RED}
echo "Red Text"
echo -ne $NO_COLOR
```

-n选项表示不换行

tput-demo-esc

tput命令

```
dlmao@mars:~$ RED=$(tput setaf 1)
dlmao@mars:~$ printf '%q\n' $RED
$'\E[31m'
```

tput命令使用终端能力名称来代替ESC序列

tput caps [args…]

- 将caps所代表的终端能力(有些命令可能要传递额外的参数)转换为该终端能力对应的特殊字符序列，写到标准输出中
- 如果tput命令的标准输出为终端设备，就会执行实际的终端命令

```
tput setaf 1   # 设置前景色为红色
```

- 下面的例子中，执行tput命令将输出保存在变量中，即变量保存了特殊的字符序列
- 接下来的echo命令不需要使用-e选项

tput-demo

```
#!/usr/bin/env bash
RED=`tput setaf 1`
GREEN=$(tput setaf 2) RESET=$(tput sgr0)
echo "${RED}red text ${GREEN}green text${RESET}"

echo "$(tput setaf 1)Red text $(tput setab 7)and white
background$(tput sgr 0)"
```

日期信息 date and cal

date命令查看或设置日期

- date [MMDDhhmm[[CC]YY] 设置日期
- date [+FORMAT] 以FORMAT格式显示当前时间
- date还支持选项--date='DATESTR'来描述其他时间

cal [options] [[month] year]

- -3 当前以及前后两个月的日历
- -y 显示年历

```
dlmao@mars:~$ cal
      三月 2020
日 一 二 三 四 五 六
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

cal	显示当前月
cal -3	显示当月及前后3月
cal -y	显示今年的日历
cal 2021	显示2021年的日历
cal 1 2021	显示2021年1月的日历

```
dlmao@mars:~$ date
2020年 03月 31日 星期二 17:02:26 CST
dlmao@mars:~$ LC_TIME=C date
Tue Mar 31 17:03:20 CST 2020
dlmao@mars:~$ date +%Y-%m-%d
2020-03-31
dlmao@mars:~$ date --date='last friday'
2020年 03月 27日 星期五 00:00:00 CST
```

- FORMAT

%y 年份最后两位

%Y 年

%d 日

%m 月份

%H 小时

%M 分钟

%S 秒数

%s 距epoch (1970年1月1日零点）的秒数

%w 星期几(0..6)

系统信息

- tty: 查看当前所连接的终端设备文件
- whoami: 登录用户名
- uptime: 查看系统运行时间和负载等信息
- free 查看系统内存空闲和使用情况。 -h 对人友好形式显示， --si 1k=1000字节

- swap: 内存不够时, 将某些进程使用的内存中数据暂时保存到硬盘的swap中
- total: 内存总量
- free: 空闲的， 尚未使用的
- used: 已经使用的
- buff/cache: 内核和cache使用的
- free = total - used - buffers/cache
- available: 启动的新进程可以使用的最大内存，这个时候不需要交换

```
demo@mars:~$ tty
/dev/pts/8
demo@mars:~$ whoami
demo
demo@mars:~$ uptime
 12:46:25 up  4:16,  5 users,  load average: 0.16, 0.13, 0.09
demo@mars:~$ free
               total        used        free      shared  buff/cache   available
Mem:         4030684     1107968     1386024       15176     1536692     2644212
Swap:         969960           0      969960
demo@mars:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8G        1.1G        1.3G        14M        1.5G        2.5G
Swap:          947M          0B        947M
```