复习

Linux操作系统概述

- 了解操作系统的概念： 内核+系统程序（命令行和GUI用户界面以及实用工具）
- 了解Linux操作系统：为什么称为GNU/Linux
- 了解什么是Linux发行版，发行版中主要包括哪些内容
- 了解如何查看以前登录信息，last和lastb的区别(成功登录和不成功登录，谁可以查看）
- 了解退出shell的三种方法：Ctrl-D/logout/exit
- 了解如何关闭和重启系统（shutdown的用法）

  shutdown [OPTIONS...] [TIME] [WALL...]

  -P 关机（缺省） -r 重启

  -c 取消前面的shutdown命令

  TIME的两种格式，多少分钟之后，以及hh:mm
- 了解查看系统信息的方法：hostname/uname/lsb_release
- 了解分页查看文本文件以及编辑文本文件命令：less和more、nano；但不用管其中内部的命令使用

图形用户界面GUI

- 不考

命令帮助

command [options] arguments　比如 ls –lt *.txt

- 了解命令格式，即命令、选项和参数部分
- 了解如何找到要执行的命令，包含路径部分和不包含路径部分的处理
- 了解内置命令（当前shell环境下执行）和外部命令（启动新的子进程，环境变量如何继承）的区别
- 了解如何查找外部命令(PATH变量，如何设置PATH变量)
- 了解短选项和长选项
  - 多个短选项一般可以合并，短选项和长选项也可以有自身的参数
  - 通常采用--表示选项部分的结束。也可通过绝对路径或者相对路径来表示接下来的部分是文件参数而不是选项部分
- 参数中的单个连字符-通常表示对应于标准输入STDIN的文件

命令帮助

- 了解help命令与man命令的区别: help type/man which
- 了解man/whatis/apropos的区别, -S, -s, --sections=LIST 在那些section里面查找

  man -f command/whatis command　命令名查找

  man -k key1 key2 /apropos -a key1 key2 / apropos key1 key2 在命令名或单行描述中查找

  man 1 kill/man kill.2/ man -a kill
- info不考

命令历史和命令别名

- 命令行方式（交互式）执行时： 首先 history expansion，然后alias substitution
- 了解命令历史以及history [n]命令：查看最近命令历史。fc命令不考
- 浏览命令历史: 箭头以及Ctrl-N/Ctrl-P
- 历史事件扩展
  - !!
  - !-n
  - !n
  - !string !?string
  - :s/pattern/replacement; ^pattern^replacement
  - :<n>
  - :<x>-<y>
  - :*
- 命令别名的作用是什么？如何查看别名？如何设置别名？如何取消别名？ 如何临时不使用别名扩展？ **alias [name[=value]]** **unalias [-a] name [name…]**

终端组合键

- 掌握主要的终端组合键以及其发送的信号：CTRL-D(EOF)、CTRL-C(SIGINT)、CTRL-\(SIGQUIT)、CTRL-Z(SIGTSTP)
- Ctrl-S和Ctrl-Q 不考；bash组合键和bind命令不考
- stty命令：了解stty命令，不需要记忆具体的选项，比如echo ixon等
  - stty -a
  - stty sane
  - stty echo 打开回显， stty -echo关闭回显

文件系统初步

- 整个Linux文件系统通过目录组织成一颗大的层次结构的树
- 掌握工作目录、绝对路径和相对路径
  - .和..表示当前目录和父目录
  - /的父目录仍然是/
  - pwd查看工作目录
  - cd 改变工作目录：cd dir　cd ~user　 cd -
- 了解dirname/basename/realpath命令的作用
- 掌握波浪号扩展：~　~root，什么情况下进行扩展
- 文件名不能命名为单点和双点，也不能包含斜线和NULL字符（ASCII码为0）字符，最长255个字符。路径名最长4096个字符 (不用记忆这些数字)
- 文件名中包含特殊字符(比如*)怎么办？文件名以-开始怎么办？
- 什么是隐藏文件？ 使用ls命令如何查看隐藏文件？

文件系统初步: ls

列目录ls

- 标准输出为终端时缺省多列显示，不是终端时一行一个文件名输出
- 给出一个列表输出结果，能够知道长列表的各个字段的作用，主要是inode/类型和权限部分/用户和用户组等，能够知道该输出是采用哪些选项获得的
- 常用选项：
  - -a 包括隐藏文件， -A 与-a类似，但不包括.和..
  - –F 文件名后面附加后缀，了解后缀为*/@，表示什么含义
  - -i inode编号
  - –l 长选项
    - ls -l 显示修改时间 ls -lu 显示访问时间 ls -lc显示状态改变时间
  - –R 递归
  - –d 查看目录本身而非目录的内容
  - 排序相关选项：缺省从**大到小，从新到旧**
    - -r(逆序)
    - –S（大小）
    - 时间排序：–t（修改时间） –ltc（改变时间） –ltu（访问时间）

文件系统初步

- touch FILE 改变时间命令，不存在时创建文件
  - -a -m 改变访问时间或修改时间
  - -t time 改变为time指定的时刻 [[CC]YY]MMDDhhmm[.ss]
  - -d, --date=DATESTR 改变时间为DATESTR指定的时间
- tree命令不考
- mkdir: 创建目录，如何创建目录树? 了解mkdir src/p/t1与mkdir -p src/p/t1的区别
- rmdir 删除空目录，删除一串空目录，了解rmdir src/p/t1与rmdir -p src/pt/t1的区别，递归删除目录：rm -R dir
- rm 删除，文件不可写时会询问用户
  - -i 删除前询问 -f 不询问删除 -R 递归删除

文件系统初步

- cp复制，了解下述命令的作用
  - cp file1 src/file2
  - cp -t dest_dir file1 file2
  - cp file1 file2 dest_dir
  - cp -R dir1 new_dir，递归复制。掌握new_dir存在或者不存在时本命令的作用
  - -i 覆盖时询问 -f 不询问直接覆盖 -n不覆盖
  - -a 归档方式，保留拥有者和用户组、访问模式和时间属性
- mv 移动或重命名
  - mv file1 file
  - mv -t dest_dir file1 file2 dir1
  - mv file1 file2 dir1 dest_dir 如果dest_dir也有dir1时，dir1为空目录时才会覆盖，如果不空，则不会移动该目录
  - -i 覆盖时询问 -f 不询问直接覆盖 -n不覆盖

文件系统初步

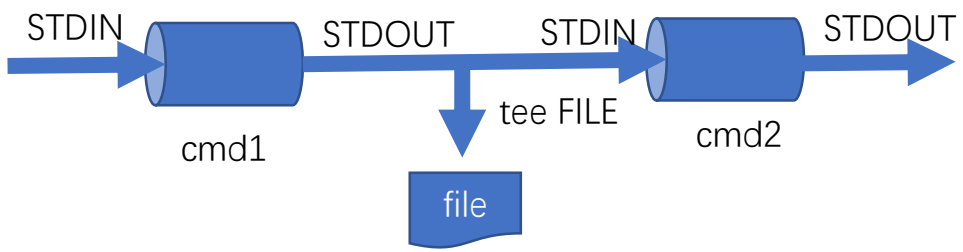
- file 查看文件类型
- du:查看目录使用情况，du -sch /home/* 2>/dev/null
  - 缺省显示各级目录的使用情况
  - -s 仅仅显示每个参数的总使用情况
  - -c 所有参数的总使用情况
- df: 查看文件系统，df ~ 查看目录所在文件系统
- Linux系统目录Filesystem Hierarchy Standard不考

文件系统初步

- 文件通配符扩展：
  - 匹配不是针对完整的路径，而是路径中的各个部分，比如/home/dl*/t*.txt
  - 匹配为完全匹配，而不是部分匹配，比如a*不匹配 cat
  - * 匹配0个以上任意字符，?匹配1个字符，但注意不能匹配最开始的点
    - 如果要匹配隐藏文件怎么实现？
  - 匹配字符集中某个字符，[abc] [^abc] [[:digit:]a-z] 不需要记忆预定义字符类的名称
  - 掌握怎么使用文件通配符扩展，给出文件通配符扩展，如何展开
  - nullglob/globstar等选项不考
- 扩展通配符不考

重定向和管道

- 重定向:
  - 输入/输出/标准错误重定向 <file >file 2>file >>file 2>>file
  - 重定向到文件描述字n所对应的文件(用&n表示) >&n
  - 标准输出和错误重定向到同一个文件 >file 2>&1 &> file >& file
  - noclobber选项不考
  - 标准输入重定向为文本块: HERE文本和HERE字符串
- 管道: cat /etc/passwd | wc -l
- 管道线分流tee命令 -a 选项表示附加




```
cat <<< "nice to meet you"
```

```
$ cat <<EOF
This is a test log file
blah...
EOF
```

字符转义和引用

- 首先基于shell元字符(包括空格类字符)进行单词分割
- 引入字符转义：
  - 如果\后面跟元字符，则表示忽略其特殊含义，比如 * 表示字符*
  - **转义仅针对元字符。比如\n并不表示换行字符**
  - 如果\后面跟普通字符，则表示忽略其特殊含义，因为没有特殊含义，所以剩下字符本身，比如 \n表示n
- 如果分析到一个引号部分，则直到另外一个匹配的引号部分，引号中的内容作为一个单词
  - 如果为单引号，则引号中的内容不作进一步的解释 '$HOME'
  - 如果为双引号，则双引号中允许参数和变量扩展、算术扩展和命令替换 "$HOME"
- Bash支持$'…' 与传统单引号类似，但是单引号里面允许转义字符，比如$'\n' 表示换行符

echo和printf命令

- echo命令： 参数显示到标准输出，参数之间以空格隔开，最后附加换行
  - -n 不附加换行
  - -e 参数中支持转义字符，了解 \n \t \e \033的用法
- printf命令：类似于printf()函数的方法格式化
  - %b: 解释\n \t \e等
  - %q: 进行转义使其可用于shell

变量

- 什么是合法的变量名？ 字母数字和下划线，首字母必须是字母或下划线，大小写敏感
- 变量赋值和访问： var=value echo $var
- 删除变量： unset var
- 普通(shell)变量和环境变量，环境变量为子进程继承的变量，但不会反馈回父进程
  - export var
  - export var=value
  - 了解常用环境变量：HOME PATH SHELL USER等
- 查看环境变量： env/printenv/export/declare -x
- 查看所有变量 set/declare -p
- 执行外部命令时指定环境变量：
  - env var=value command….
  - var=value command…..
- 特殊变量： $$ $? $!

shell变量和shell选项

- 了解常用shell变量：IFS/PS1/PS2
- shell选项： 了解如何查看、设置和关闭，不用记忆具体的选项名字
  - set -o /set +o /set -o option /set +o option
  - shopt / shopt -p /shopt -s option /shopt -u option

本地化设置locale

- locale： zh_CN.UTF-8/en_US.UTF-8，为C表示没有本地化
- 不用记忆12类locale子类
- 本地化环境变量的优先级： LC_ALL>LC_*>LANG
- 用户进一步可在其bash配置文件(.bashrc)中设置环境变量来定制自己的locale设置，

```
export LANG=zh_CN.UTF-8
export LC_COLLATE=C.UTF-8
export LANGUAGE=en_US  LC_MESSAGES=C.UTF-8
```

- 建议设置LANG来给出LC_*的默认值，然后通过LC_XXX具体微调某个大类习惯
- LC_ALL一般仅在执行某个命令时使用

```
LC_ALL=C command
```

LANGUAGE	消息(出错、菜单等)的本地化翻译,可同时设置多种语言信息，以冒号隔开，比如en_US:zh_CN
LC_ALL	替代除LANGUAGE外的其他locale环境设置，优先级最高
LANG	LC_*没有设置时使用的默认值

内置命令和外部命令

- 命令解析顺序:
  - 如果命令名前包含路径部分，明确指定是要执行哪个目录中的哪个命令。如/usr/bin/lscpu 或./args
  - 否则：首先看是否为命令别名，接下来看是否为shell函数，然后看是否为shell内置命令，如果都找不到则在PATH指出的目录中查找
- 了解type/which/whereis命令的区别
  - type -a 可列出所有可能匹配的命令,包括别名、函数、内置命令以及外部命令
  - type -P 只查找外部命令，等价于which

shell环境

- 了解shell的分类方式:
  - 通过身份验证登录的shell一般为login shell，不需要验证身份的一般为non-login shell。bash -l 以login shell方式
  - 脚本的执行或者bash -c 'cmd'执行某个命令时为非交互方式，等待用户输入命令的bash为交互式方式。 bash -i 以交互式shell方式
- login shell初始化时会执行.profile类型的文件，在该文件中会调用bashrc类型的文件
- non-login shell初始化时执行bashrc类型的文件
- 脚本如何执行?
  - #shebang行的含义: #!/usr/bin/env bash
  - chmod u+x script; ./script
  - source script ; . script
  - bash script
  - 上述方式之间的区别（当前shell环境还是子进程）？

bash：解析命令

普通命令：[NAME=VALUE]... [COMMAND [ARGS]...]

- 首先基于分割单词的元字符 | < > & ; ( ) space tab newline将命令分解成单词<word>
- 如果为**交互式shell**，首先进行
  - 历史事件扩展
  - 别名替换
- 接下来按照如下顺序进行shell扩展：
  - 花括号扩展{1,2,3}: 按照预定的模式生成一组字符串
  - 波浪号扩展~:展开用户主目录
  - 参数和变量扩展$：获得参数或变量的值
  - 算术扩展$(( ))：获得表达式运算的值
  - **命令替换``or $( )：嵌套另一条命令**
  - 进程替换：进程的输入或输出替换为一个文件名
  - 单词分隔(word splitting)：前面**参数扩展、算术扩展、命令替换执行后的结果**再根据IFS(空格、Tab、换行等字符)分隔单词
  - (文件名)通配符扩展：展开成匹配的一个或多个文件名
  - 注：上述某个扩展展开后不再适用**当前以及之前**的扩展。花括号和命令替换允许嵌套
- 扩展完后决定命令部分和参数部分，进行环境变量设置和重定向，然后执行相应的命令

✓ 最初用于分割单词的元字符如下：
| < > & ; ( ) space tab newline
✓ 参数/算术/命令替换扩展的结果再基于IFS(缺省**space tab newline**)进行单词分割

花括号扩展

- 如何通过花括号扩展为多个参数
- 两种扩展模式都要了解：
  - file{1,2,3,4}.txt
  - file{1..4}.txt
- 允许嵌套以及连续花括号扩展，掌握平行和嵌套花括号扩展如何展开
- 花括号扩展无法识别时并不会报错，而是原样输出
- 花括号扩展是最早展开的，其他扩展(包括参数扩展、文件通配符扩展等）都是在花括号扩展之后展开的

```
$ echo {a..b} {01..2}
a b 01 02
$ echo {a..b}{1..2}
a1 a2 b1 b2
$ echo {a..b}{x,y}{1..2}
ax1 ax2 ay1 ay2 bx1 bx2 by1 by2
$ echo src/{a{1..2},b{3,4}}.py
src/a1.py src/a2.py src/b3.py src/b4.py
$ echo src/{a{1..2},b{3,4}}.{py,c}
src/a1.py src/a1.c src/a2.py src/a2.c src/b3.py src/b3.c src/b4.py src/b4.c
```

命令替换

- 了解命令替换的作用，支持递归命令替换
- 两种方式都要掌握，但不需要考虑反引号方式的嵌套问题
- 双引号内允许命令替换，但是不允许单词分割
- 不在双引号时，命令替换后的结果会进行单词分割，这时空格、制表和换行符作为分隔符，多行的结果变为多个参数

```
$ ls -l `which cp`
$ cd $(dirname $(which touch))
$ echo "$(ls -d /usr/bin/* | grep zip)"
/usr/bin/funzip
…
$ file $(ls -d /usr/bin/* | grep zip)
/usr/bin/funzip:     ELF 64-bit LSB executable,…  stripped
/usr/bin/gpg-zip:    POSIX shell script, ASCII text executable
…
$ cat patterns
??*
.?*
$ ls -d  $(cat patterns)  #等价于 ls -d ??* .?*
```

单词分割

- 对通过read等读取的输入，参数扩展、算术扩展及命令替换扩展后的内容，根据IFS(**Internal Field Separator)** 分割成一个一个的单词
  - IFS缺省为空格、制表、换行符。如果IFS未设置，使用缺省值
  - IFS为空时表示不进行单词分割
- 了解read命令:如果变量个数不够时，剩下的那些单词都赋值给最后一个变量varN

read [-s] [-n nchars] [-t timeout] [-p prompt] [var1 var2 …varN]

-p prompt首先输出提示符

-t timeout等待用户输入的时间

-n nchars等待用户输入nchars个字符或已输入换行符

-s 不回显，用于输入密码等隐私信息

- 赋值语句**不支持花括号/单词分割/通配符，但支持波浪号、参数、命令替换，扩展后结果不会进行单词分割**

shell命令结构

- 简单命令： [NAME=VALUE]... [COMMAND [ARGS]...]
- 管道：管道中的每个命令都是启动一个新的进程来执行的；管道的返回状态码缺省为最后一个命令执行后的返回状态码。不用管pipefail选项
- 命令执行的状态码： echo $? 0表示成功
- 条件执行： cmd1 && cmd2 cmd1 || cmd2 ！cmd，命令执行是否成功，在此基础上进行逻辑运算，采用短路逻辑
- 组命令 { cmd1;cmd2;…cmdN;} > file : 当前shell环境下执行， {后应该有空格， }前面为分号或换行
- 子shell (cmd1;cmd2;…cmdN) > file： 子shell环境下执行，括号不需要空格

```
test -r ~/.bash_aliases && cat ~/.bash_aliases
#只有在创建目录成功时才切换到该目录
mkdir d && cd d
test -r ~/.bash_aliases || echo ".bash_aliases doesn't exist"
rm file || { echo 'Could not delete file!' >&2; exit 1; }
```

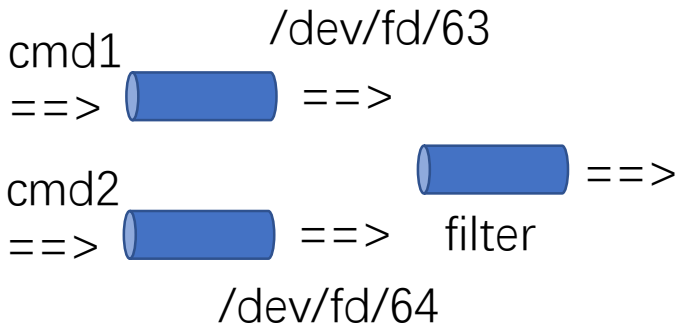
```
{ cd /usr/bin; ls;} > ~/bin_list.txt
(cd /usr/bin; ls ) >  ~/bin_list2.txt
```

进程替换

- 替换为某个文件名，该文件是一个命名管道文件，另一端由list对应的进程进行读写
- 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

```
$ cat <(ls -ld /bin/*) <(ls -ld /usr/bin/*) > files
```

filter <(cmd1) <(cmd2)
替换为
filter /dev/fd/63 /dev/fd/64




```
cmd1 > t1
cmd2 > t2
filter t1 t2
rm t1 t2
```

ESC序列

- 终端ESC序列和tput命令：
  - 不需要记忆具体的ESC序列与tput命令
  - 主要知道ESC序列如何与echo/printf等结合一起使用，如何使用tput命

```
#!/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)"
```

```
#!/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
```

查看系统信息

- 了解查看和设置日期信息的命令： date/cal
  - date [MMDDhhmm[[CC]YY] 设置日期
  - date [+FORMAT] 以FORMAT格式显示当前时间
- 了解内存使用情况 free
- 了解当前登录用户和终端 whoami/tty

vim 编辑器

- 掌握主要模式，如何进入到该模式，如何退出该模式(ESC切换到正常模式)
  - 命令或正常模式；插入模式；命令行(last line)模式和可视模式
- 掌握如何保存和退出的命令，包括 :q :w :wq :w! :q! :wq! ZZ ZQ
- 掌握光标移动命令：
  - hjkl
  - 0 $ ^，注意0和^的区别
- 不考，光标在当前屏幕移动： HML
- 不考，屏幕滚动： C-F/C-B 半屏 C-D/C-U
- 掌握移动到指定的行 gg G 10G 10gg
- 文本对象移动：知道word/WORD的区别，掌握wbe和WBE命令
  - word，字母数字和下划线组成，空格类和标点符号确定单词的边界，多个连续标点符号也是一个单词。 w下一个单词开始 b 当前单词或上一个单词的开始 e 当前单词或下一个单词的结束
  - WORD，空格类字符确定单词的边界， WBE

vim 编辑器

- 行内查找:
  - f/F 光标找到字符位置， t/T光标在字符前一位置
  - 不考重复行内查找 ;,
- 搜索:
  - / ? 查找（正则表达式）文本
  - n/N重复上次搜索
  - * 向前搜索光标所在word # 向后搜索光标所在word
  - 不考g* 与g#
- 插入文本： ilaAoO
- 删除文本 x d{motion} d4w 4dd
- 替换字符： r s
- 修改文本： c{motion}
- 合并行 J 分割行 r<CR>
- 重复最近编辑命令 .
- 撤销 u和重做 :redo C-R

```
[#1] operation [#2] target
#1: 表示重复多少次operation，缺省为1
#2: 表示一次operation对应的target有多少个
```

键	动作
i	当前位置前插入
I	当前行首第一个非空白字符处插入，等价于^i
a	当前位置后插入
A	当前行尾插入，等价于$a
o	当前行下面插入
O	当前行上方插入

vi编辑器

- y{motion} v{motion} c{motion}等支持的文本对象选择：
  - aw/aW (around word/WORD)
  - iw/iW (inner word/WORD)
  - a' a"等以及i' i"等
- 拷贝和黏贴：
  - y/p/P
  - 和寄存器结合在一起 "a10yy “ap
  - 不需要记忆特殊寄存器
- 可视模式
  - 选择并进入可视模式: v(字符模式) V(行模式) Ctrl-V(列模式)
  - gv 选择上次可视模式所选择的内容
  - 列模式，gn/gN等不考
  - 不执行操作就退出可视模式：建议采用<ESC>
  - 在可视模式中交换选择文本的开始和结束位置： o

vi编辑器

- 常用vi选项，如何打开和关闭vi选项
- number/nocompatible/expandtab/autoindent/wrap/ignorecase/smartcase等
- 常用标记： < >
- 命令行模式:
  - 如果描述命令作用的范围?
  - normal/global命令
  - substitute命令
  - 外部命令

多文件编辑以及map命令不考

:!bash %	调用bash执行当前编辑的脚本 **% 表示当前编辑的文件名**(h :_%)
:0r template.txt	在文件头部插入template.txt中的内容
:'<,'>w tmp.txt	将可视模式下最近选择的行保存到文件tmp.txt
:r !date	当前行之后插入date命令执行后的输出
:w !sudo tee % >/dev/null	当前用户没有权限写所打开的文件，调用sudo切换成超级用户来保存
:'<,'> !sort	将可视模式下最近选择的行重新排序

用户管理

- 了解passwd/shadow/group文件，其中shadow只有root可读写
- id/ id root: 查看用户的用户、用户组信息
- useradd如何增加用户，如何增加管理员用户，如何加入到其他用户组
  - useradd -m -G sudo -s /bin/bash test1
- usermod：如何加入到其他用户组
  - usermod -a -G video test1
  - usermod -G video test1
- userdel: 删除用户
- groupadd/groupdel 增加用户组和删除用户组
- passwd: 如何改变密码

su和sudo

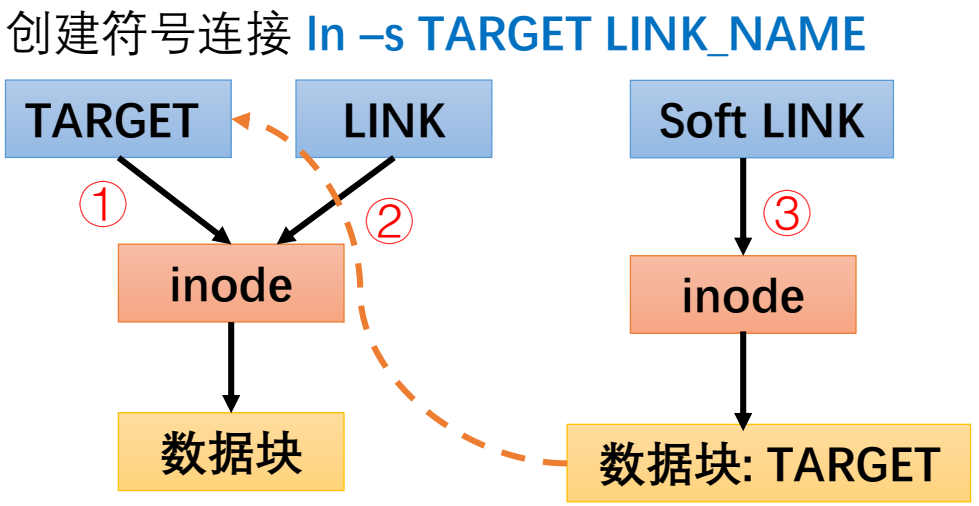
- su
  - 超级用户root一般不允许通过用户认证方式登录
  - su user:切换成另外一个用户执行，是否继承原有用户的环境?
    - 缺省继承环境
    - su -l user或su - user 以登录shell，全新环境
    - -c command 以另一个身份执行指定的command，执行完后结束
- sudo的基本特性，输入谁的密码？每次都要输入吗？如何切换为另外一个身份进入新的shell?
  - sudo -s /sudo -i 以root身份运行shell
  - sudo su

-s, --shell 启动一个新的shell(当前环境变量SHELL指定的shell) 。如果有cmd，执行cmd后退出

-i, --login 就像新的用户登录一样，以登录shell方式启动新的shell。如果有cmd，执行cmd后退出

硬连接和软连接

- 硬连接和软连接的区别
- 硬连接只能在同一个文件系统中，且只允许对文件创建，多个目录项中指向同一个inode
  - ls -i可查看inode节点编号　ln TARGET LINK 创建一个硬连接
- 软(符号）连接是一个特殊的文件，其文件的内容为路径名（可是相对路径或者绝对路径），软连接的基本特性
  - ln -s TARGET LINK创建符号连接
- 根据ls的输出来判断硬连接和符号链接的情况
- stat命令查看文件元信息
- readlink/realpath命令



权限管理： chown/chgrp/chmod/umask

- 了解rwx权限对于目录和文件的不同含义
  - 文件：rw表示可以读写，x表示执行，对于脚本而言要求有rx权限才可执行
  - 目录：r表示可以列目录(ls)，w表示可以改变目录项即创建删除和改名， x表示可以切换到该目录
- 特殊权限：SUID/SGID/Sticky权限的作用，如何添加
  - SUID/SGID对于文件，有效用户和有效用户组变为文件的拥有者和用户组
  - 对于目录： SGID表示其中创建的新的文件或目录的用户组为该目录的用户组，且也具有SGID权限；Sticky权限表示删除或者移动目录项只有超级用户或所在目录的拥有者或文件的拥有者才允许
- chown/chgrp如何改变文件拥有者和用户组
- 如何通过chmod修改权限， 两种模式都会
- umask: 如何正确设置umask（数字形式）来屏蔽某些权限, 文件和目录的缺省缺陷分别为666和777

块设备管理

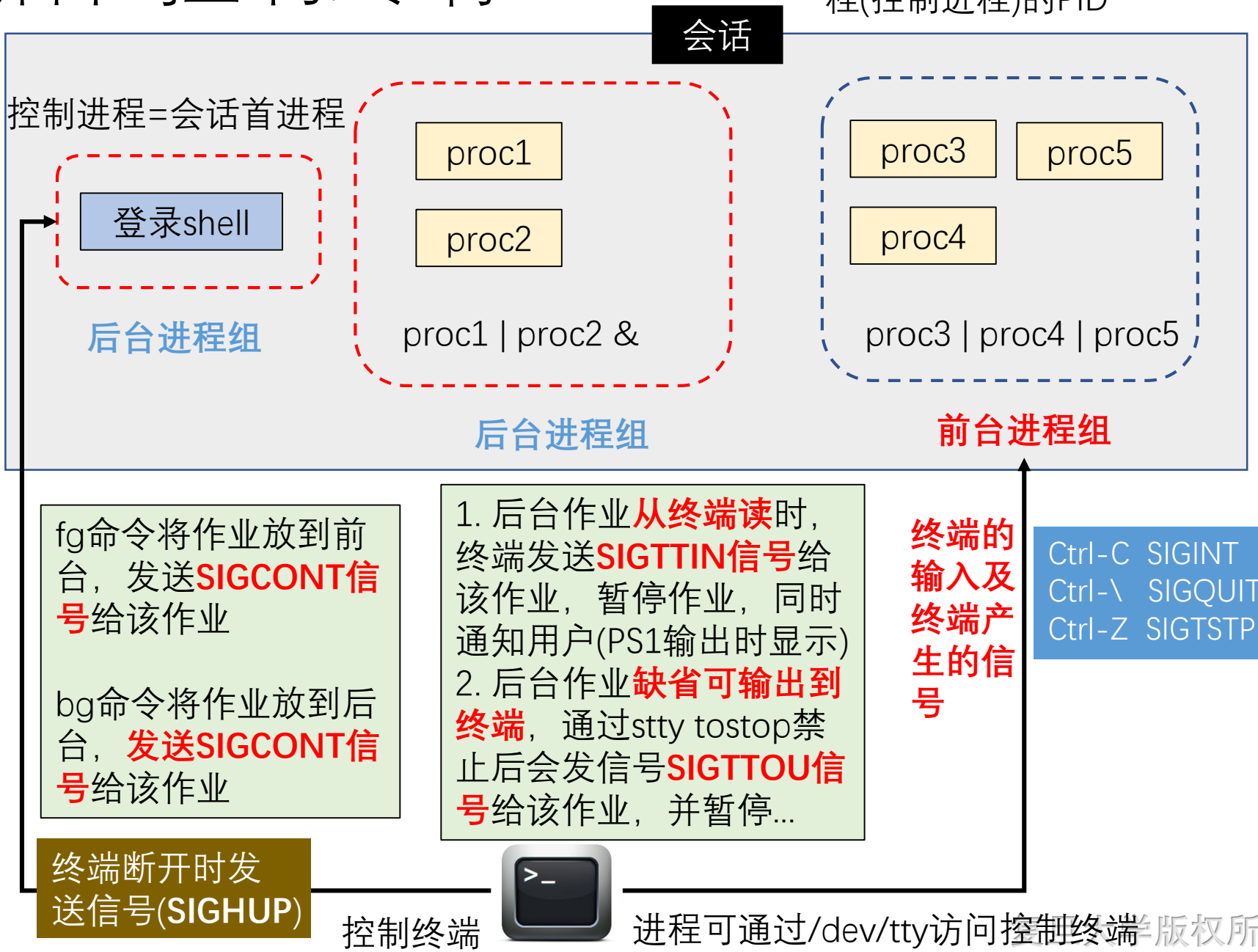
- 数据块读写命令dd: dd if=IFILE of=OFILE bs=1k count=10
- 查看块设备文件 lsblk
- 创建文件系统: mkfs file.img ; mkfs /dev/sdc2
- 加载文件系统mount/umount
  - mount /dev/sda /mnt
  - mount linux.iso /mnt 或mount -o loop linux.iso /mnt
  - umount /mnt
  - 挂载之后，挂载点下的内容暂时屏蔽掉
  - 常用挂载选项：ro/rw/user/nouser/remount/auto/noauto
- /etc/fstab 自动挂载文件系统
- FAT/NTFS文件系统的挂载不考

进程和作业管理

- 了解作业和进程的关系
  - 进程是系统维护的正在执行的程序的信息，通过父子进程的关系组成一棵进城树
  - 作业是shell维护的正在执行的命令(对应某个进程组）的信息，引入进程组的目的是方便给进程组的多个进程发送信号
  - 一个作业(进程组）可对应一个进程
  - 一个作业(进程组)可包含多个进程，比如通过管道、子shell方式等运行的多个进程属于同一个作业
  - 每个进程属于且只属于一个进程组
  - 作业分为前台和后台作业，根据其输入是否与当前的控制终端连接来决定，作业的输出可以连接到控制终端
  - **前台作业**为由控制终端控制的作业，**可以接收终端的输入**
  - **后台作业**为独立于控制终端的作业，它**无法接收终端的输入**

进程组、会话和控制终端

- 用户登录后，登录shell为控制进程，位于前台，等待用户输入
- tty命令可查看连接的终端，可通过/dev/tty访问控制终端
- 执行外部命令时附加&，该作业后台运行
- 执行外部命令缺省前台运行，此时登录shell后台运行
- 前台作业允许读写控制终端
- 后台作业缺省允许写控制终端，但不允许读
- 控制终端产生的信号发送给前台作业
- 终端断开时发送信号SIGHUP给控制进程，控制进程再发送给各个作业，各个作业收到HUP信号时缺省终止进程



作业管理

- 如何后台执行，如何停止前台进程，如何将进程转向后台执行，如何将后台进程转向前台?
  - 通过命令后加 &表示将命令放在后台执行
  - jobs [-l]可以查看当前作业列表　+表示当前作业，　-表示前一个作业
  - fg %jobid ；fg %make　fg %?game 将指定作业切换到前台
  - bg …. 将作业切换到后台继续运行
  - Ctrl-Z　给当前前台作业发送SIGTSTP暂停执行
  - kill -CONT pid or %jobid 发送SIGCONT信号恢复执行，类似于bg ….
  - suspend 或者kill -STOP $$ 暂停当前shell

进程信息查看和发送信号

- 查看进程树 pstree [pid | user]
- 查看进程ps ， 缺省当前用户且当前控制终端
  - ps a　a取消用户限制，即和任一控制终端相关的进程
  - ps x　 x取消控制终端限制，即当前用户的进程
  - ps ax　两者都取消，即所有用户所有进程
  - ps aux　u为长格式
  - pa axjf 查看进程树， j为job相关信息, f表示forest
- 给进程发信号 kill
  - 如何给进程发信号？ kill pid or kill %job
  - 如何给进程发某个信号？ kill -KILL pid / kill -SIGKILL pid/ kill -9 pid
  - 有哪些信号： kill -l
  - 常用信号的含义，哪些信号不能忽略(KILL和STOP)： SIGHUP/SIGINT(Ctrl-C)/SIGQUIT(Ctrl-\)/SIGKILL/SIGTERM/SIGTSTP(Ctrl-Z)/SIGSTOP/SIGCONT
- 根据名字来给进程发信号 killall cmd
- 基于正则表达式查找进程或者给进程发信号pgrep和pkill
- pidof program 查看命令名为program的进程pid

进程和作业管理

- 控制终端退出时发送SIGHUP信号给所有后台进程，一般收到该信号会终止
  - nohup wget url& 标准输入定向到/dev/null，标准输出定向到nohup.out，忽略SIGHUP信号
  - disown -h %jobid 作业已经运行，忽略SIGHUP信号
- 设置进程优先级nice /renice
  - 了解进程优先级的概念
  - 进程的niceness值(NI)为[-20,19]，值越低，优先级越高。普通命令的NI为0
  - 如何调整niceness，nice/renice命令的区别
  - renice时普通用户只能**设置更大(即更低)的优先级**

nice [-n niceness] command

renice priority [-g|-p|-u] identifier...

查找文件命令find

- 三个要素：查找起点、查找什么、找到了怎么办
- 掌握-mindepth <levels> /maxdepth <levels> 选项
- 文件属性选项： 文件名、文件类型、文件大小、文件时间、用户、权限
  - 文件名相关，-name pattern -iname pattern -path pattern -regex expr -lname pattern
    - 只有-name选项采用基本名，其余都采用完整文件名(包括路径部分）。在find中*可以匹配文件名开始的.和路径分隔符
  - -type f -type d -type l
  - 硬连接：-inum N -samefile NAME -links +N -links N -links -N
  - 文件大小：**-size +NK** -size NK -size -NK -empty
    - round up, +NK的形式是准确的，其他可以通过逻辑运算(not)
  - 文件时间：-mtime +N **-mtime -N** -mtime N -newer otherfile
    - round down, -N的形式是准确的，其他可以通过逻辑运算(not)
  - 拥有者：-user demo -nouser
  - 权限：-perm /7000 -perm 444 -perm -0744

查找文件命令find和locate命令

- **逻辑运算符： -and -a –or -o -not !**
  - find ~ \(-type f ! -perm 0600 \) -o \(-type d ! -perm 0700 \)
- 动作部分： -ls -delete -print –print0 -exec command \; -exec command +
  - 了解两种exec动作的区别，了解print和print0的区别
  - -prune 用于跳过哪些名字为某个模式的目录，注意其与 -or运算结合，利用短路逻辑
- 上述这些选项可以多个条件匹配，与mv/cp/chmod等结合在一起

- locate命令
  - 与find命令的区别，在哪里找，是否精确
  - 缺省匹配的完整路径名，通过-b选项匹配基本名字，locate jpg等价于 *jpg*，同样*可以匹配开头的.和目录分隔符

xargs 从标准输入构造命令并执行

- xargs: 从标准输入获得参数来构造命令并执行
  - 掌握xargs与find -exec等的区别
  - 如何与find结合一起使用
  - -r 参数为空时不执行命令　-0 用NULL字符分割参数
  - -n max-args 最多多少个参数开始执行命令
  - -I {} 表示{}对应着参数部分

```
find . -size +1G -print0 | xargs -0r -I '{}' mv '{}' ~/bigfiles
```

xargs

- echo:将命令行的内容放到标准输出
  echo arg1 arg2
- cat:将文件(包括标准输入)中的内容放到标准输出

  cat file1 file2 > large-file
- 命令替换：执行命令的标准输出的东西放到命令行

  echo $(date; ls)，注意单词分割
- 管道| 将标准输出变为标准输入
- xargs 动态构造命令，将标准输入（一般来自于管道中的前一条命令的标准输出）的内容放到命令行

压缩命令gzip/gunzip和归档命令tar

- gzip/gunzip:
  - 掌握如何进行压缩和解压，如何进行替代，其访问模式和时间有变化吗？原来的文件被移走，仅剩下压缩或解压后的文件
  - 如何保留原有文件？和重定向与管道机制结合
  - -c 选项保留原有文件，压缩或解压的结果到标准输出
  - -d 表示解压缩
  - -r 递归方式
- tar命令
  - -c 创建 -x 提取 -t 列出 -r 附加 -z 压缩或解压缩
  - -f 指定归档文件 -f - 标准输出或者标准输入
  - 创建归档文件 tar cvf tarfile dirs
    - 创建时采用相对路径，即移除路径名**最前面的斜杠**
  - 查看归档文件内容：tar tvf tarfile
  - 提取全部文件： tar xvf tarfile，或提取其中某些文件 tar xvf tarfile files
    - 解压时采用相对路径
  - 如何与find等结合：

```
find /bin -name '*zip*' -print0 | xargs -r0 tar rzvf bin-zip.tar.gz
```

grep: 正则表达式查找并输出

grep pattern [file] 如何传递pattern

- -r 递归
- -v 输出不匹配的行
- -n 输出时包含行号
- -c 输出匹配的行数
- -i 忽略大小写
- -w 匹配整个单词 -x 匹配整行
- -E 采用扩展正则表达式 -o 仅输出匹配的内容

- 扩展正则表达式: (不考虑基本正则表达式)
  - 字符集: . 单个字符 [abc] [a0-9z] [^0-9] \s \S \w \W
  - 要匹配元字符(如*)可以采用 *或者[*]
  - 边界匹配: ^ $ \< \> \b \B
  - 子模式 ( )
  - 选择 |
  - 重复匹配符: {n,m} {n,} * + ?
  - 分组引用 \1 \2 ... \9

过滤器

- 查看文本文件的开始和最后多行：head和tail
  - head -n 5 /etc/services　head -5 /etc/services
  - head -n -8 /etc/services 不要最后的8行
  - tail -n 5 /etc/services　tail -5 /etc/services
  - tail -n +5 /etc/services　不要前面的4行，从第5行开始
  - tail -f output
- 统计行、单词和字符数量wc
  - wc -l
  - wc -w
  - wc -c
  - wc如何与其他命令组合在一起来使用，统计来自于标准输入的行数和单词个数

过滤器

- 抽取数据列cut
  - 了解cut的限制，多个连续分隔符并不会合并，如果出现这种情况，前面一般通过tr -s进行压缩，或者采用sed替换或删除
  - -d DELIM 字段分隔符为DELIM而不是缺省的\t，-f LIST 取相应的字段
  - -c LIST 取相应的字符

```
ps ax |grep [s]sh |  sed 's/^\s*//'  | cut -d ' ' -f 1 | xargs
```

- 组合数据列 paste:
  - -s, --serial　各个文件中各行合并为一行
- colrm/column 不考
- 排序sort
  - -n 按照数值顺序排序
  - -r 逆序排序
  - -k field1,field2 基于指定字段进行排序　-k 3,3rn　 -k 3.7nr
  - -t DELIM 用于确定字段的分隔符DELIM，缺省为空格或制表符，且多个连续空格类字符合并
  - -u 排序规则认为相同的行仅留下一行

```
ls -l /usr/bin | sort -nr -k 5
```

过滤器

- 查找重复行 uniq ， 去除重复行，缺省为-u -d
  - -c 在行前添加重复次数
  - -u 仅仅unique的行
  - -d 仅仅重复的行
- 替换或删除字符tr
  - tr a-zA-Z A-Za-z 对应位置的字符进行替换
  - tr -s a-zA-Z A-Za-z 先替换，然后压缩多个，即字符集2中的连续字符压缩成一个
  - 替换形式中第二个字符集的字符个数不够时用最后一个字符补全，或者用[y*3]的形式表示yyy
  - tr -s '\n' 多个连续的\n压缩为一个\n
  - tr -d 0-9 删除字符
  - -c 选项表示不在字符集1的字符进行相应的操作

sed

- 可以删除某些行；可以修改某些行； 可以插入或附加一些行
- -E 采用扩展正则表达式
- sed SCRIPT FILE
- sed -e SCRIPT1 -e SCRIPT2 FILE
- -n 抑制模式空间自动输出，即读下一行到模式空间时缺省会打印当前模式空间
- 每个命令可包括3个部分[addr]X[options]
  - 哪些行匹配的情况下执行某个命令X，options部分为该命令的选项
  - 地址部分没有表示匹配所有行
  - 有行地址表示某一行
  - ADDR1, ADDR2 表示范围
  - ADDR!表示不在范围
  - 行地址可以是行号， /REGEXP/ ， 可以是+N

sed 主要命令

- s/regexp/replace/[flags]　修改模式空间中的内容
  - s命令的分隔符缺省为/，如果pattern里面有/则需要在之前加上\进行转义。也可以使用其他任意字符作为分隔符。比如s%pattern%replace%flags
  - replace中如下字符有特殊含义：
    - &表示regexp匹配的内容
    - \n，n为数字，表示regexp中的第n个子模式匹配的内容
    - \ 转义字符，比如\&表示字符&，而\\表示反斜杠\，\n表示换行符
  - flags:
    - g表示模式空间中的所有与pattern匹配的内容都替换，缺省仅替换第一次出现的模式
    - 数字 替换第几次匹配
    - p表示输出替换后的模式空间
    - i表示匹配时大小写无关
- y/set1/set2/
- d 删除模式空间 ；p 打印模式空间 = 打印当前行号 q 退出sed
- a\... i\... c\...
- r file w file

awk

- pattern可以是
  - BEGIN或END
  - 表达式，一般是比较关系+逻辑运算符结合在一起
  - 扩展正则表达式：/regexp/， 等价于 $0 ~ "regexp"
  - 范围：expr1, expr2
    - 首先寻找符合expr1的记录
    - 然后检查后续的记录，直到其满足expr2，如果无法找到，则到文件结尾为止
    - 指定范围的每条记录，执行相应的动作
- 类似于C语言， awk变量： RS/FS/OFS/NR/NF
- 支持程序结构：
  - print expr1, expr2,...,exprn或print(expr1, expr2,...)
  - 关联数组 a[idx] = value
  - for (idx in array) statement
  - `for (init;condition;update) statement`
  - `if (expr) statement [else statement]`
  - `while (expr) statement`

```
BEGIN {action}
pattern {action}
pattern {action}
  .
  .
pattern { action}
END {action}
```

shell编程

- 整数算术运算和算术运算扩展 : **let [expr…]** ((expression)) $((expression))
- 条件判断: **test expr [expr] [[expr]]**
  - 文件状态判断
  - 整数比较
  - 逻辑运算
  - 字符串比较
  - 文件通配符/正则表达式匹配
- **if COMMANDS; then COMMANDS; [elif COMMANDS; then COMMANDS;]... [else COMMANDS;] fi**
- **while COMMANDS; do COMMANDS; done**
- **for ((exp1; exp2; exp3)); do COMMANDS; done**
- **for NAME [in WORDS ...] ; do COMMANDS; done**