Unix/Linux

Unix 只有 2 entities, files and processes.

Files are the passive entities, represent data, provide input and receive output. (no life)

streams of bytes on disk

interfaces to devices

input output streams of running programs, directories

Processes are the active entities, reading processing writing data. (have life)

instances of running programs

instructions for the cpu

Each command is executed as a single process.

Each invocations of a command gives rise to a new process.

Every process starts life as a file.

Basic Command

date
pwd
who am I
who
ls

KeyBoard Control

^C	Interrupted
^Z	Suspend command
^D	End of file
^H / ^? / DELETE	Delete last character
^W	Delete last word
^U	Delete line
^S	Suspend output(rarely)
^Q	Continue output(rarely)

stty to establish specific key settings

eg: stty erase <BACKSPACE>

File System Command

cd to user home directory

cd ~userA to userA home directory

cd ~userA/B/C to userA 的 / B/C directory

cd -和 cd $OLDPWD 都可以在最近所操作的两个目录之间进行切换

```
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ cd -
/Users/robin/Desktop/Note
robin@Robins-MacBook-Air:~/Desktop/Note$ cd -
/Users/robin/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ cd $OLDPWD
robin@Robins-MacBook-Air:~/Desktop/Note$ cd $OLDPWD
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ echo $OLDPWD
/Users/robin/Desktop/Note
```

pushed & poped

```
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ dirs
~/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ pushd /Applications/
/Applications ~/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:/Applications$ pushd .
/Applications /Applications ~/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:/Applications$ popd
/Applications ~/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:/Applications$ popd
~/Desktop/Note/AlgorithmNote
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ popd
-bash: popd: directory stack empty
robin@Robins-MacBook-Air:~/Desktop/Note/AlgorithmNote$ dirs
~/Desktop/Note/AlgorithmNote
```

dirs : 列出当前堆栈中保存的目录列表

pushd：切换到指定目录（参数） 把原目录和当前指定目录压入虚拟的堆栈中。如果不指定参数，则会切换回到前一个目录，把前一个目录优先级放栈顶

popd： 弹出堆栈中最近的目录

cat view contents of files (append files in file-system to end of device file, display)

-n add lines to the displayed file

wc counts number of words, lines, characters in the files

-l display the number of lines

Process Command

ps currently running processes

	-f	
	-u uidA	all processes owned by uidA
	-ef	display all processes on the system
	-elf	see the size of all programs
more		view the text files page by page, displaying one screen at a time in case the file is large (For example log files). 若 it fits 满 string, 则 buffer file context. It allows the user do scroll up and down through the page. <SPACE>翻页， <return>下行
less		同 more 可 buffer 也可 go back
ls		display content of the specified directory
	-l	display detail info (UID GID …
	-la	display detail info + detial hidden files
	-A	files
	-a	files + hidden files
	-F	display file with their type, directory with /
	/	display root dir
	-l dirA	show dirA's context info line by line
	-ld dirA	show dirA itself info in a line

Shell

Shell is the ***process*** which **read characters** typed in from the keyboard, and eventually **invokes the corresponding program**. Dies when the user logs out. Part of job of the shell is to find the location of a requested command by searching PATH (local variable).

Commands exist as ***program files*** somewhere within the logical file system.

Shells help users generate lists of filenames, redirect command input output, construct command pipelines.

Shells are not build into Unix, they are just programs in file system, like ls or cat.

cd(改变 shell 的 status), pwd, sleep 是 build into shell 的 ,Done by shell itself.

Shell family

bourne shell	sh	$

korn shell	ksh	$
C shell	csh	%
Bourne Again Shell	bash	$

more public domain shell : ash , zsh …

Shell Wildcards

(由 shell 去 expend, 可辅助 user 去 auto complete or generate filenames)

*	any number of any character
?	all single character
[ab]	a or b or a range of specifies character / character class
[a-z]	a letter
[^a-z]	not a letter

shell expands the wildcards before passing the generated arguments to commands.

ls p* * tell shell to generate filenames, the shell searches the specific directory to find the files.

echo p* shell see *, then hierarchical to match the list of arguments beginning with p, then invoke echo.

Shell interprets wildcards, generated an argument list, and then calls the specific command.

Regular expressions(RE)/grep wildcards

.	any character
*	>= 0 occurrences of previous character
[abc]	any one of a, b or c
[a-z]	any character in range
^	beginning of line
$	end of line

Create/Remove/Copy/Move directories

mkdir A

mkdir –p A/B/C	able to create missing parent directories as needed

rmdir A

rm –r A

cp [-ip] f1 f2

cp [-ip] f1 f2 … fn dir arguments > 2, the last one must be directory.

cp –r [-ip] dir1 dir2

cp *.cc copy wildcards should be careful when there's no matched files. Shell 会 leave the pattern with wildcards unexpanded. 会去找名为*.cc 的文件, 没找到则 err

mv [-i] f1 f2 -i if there's a danger of overwriting existing files

mv [-i] f1 f2 … fn dir

mv [-i] d1 d2 replace p1 with name p2

I/O redirect

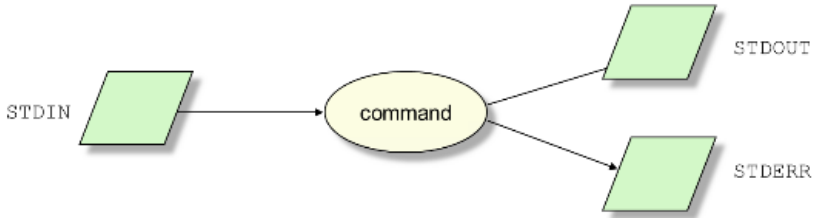
shell default input is read from keyboard, output is written to display.

> write output to specific file

< read input from a file

| output of left is the input of the right command

Unix 里啥都是 files，shell 读 input 只有遇到 end of file character 才停止，通常^D 便可从 terminal 产生该 end of file character.



Shell open 3 files STDIN(file descriptor 0), STDOUT(1), STDERR(2) for every command first.

The standard input/output of a process are NOT command line parameters.

- I/O directives change the bindings

	> output	redirect STDOUT to output
	>> output	append STDOUT to output
	< input	redirect STDIN from input
bash, ksh and sh →	2> outerr	redirect STDERR to outerr
	2>&1	redirect STDERR to STDOUT's file
csh →	>& outerr	redirect STDOUT and STDERR to outerr

> A create file A

> A cat redirect（output 写进 A） + cat (shell 一直 open3 个文件，cat 改 output 了)

hello

cat A show hello

cat A nonExistB > output output 里只有 A 的内容，err goes to screen

cat << EOT << EOT 告诉 cat, here is your data, End of Text is EOT

>hello

>EOT

hello

sort < input > output 从 input 读并 sort 再写入 output

dangerous： sort A > A clear A then sort

solution: 1) sort –o A A 2) sort A > B

STDERR

- redirected to a file 2> err
- merger with std output 2>&1

Appending Output

>> A append to the end of A

<< used for 'here' documents

Unix file system (organized as an inverted tree)

Unix contain only one logical file system, which may be composed of many physical devices and networks.

There is only one hierarchical tree in Unix, diff from those provided in DOS, VMS. Hide everything.

Directories are just files which hold the name of entries of those files inside the directory. (aim to group related files)

Directories – branches
Files – leaves

Directory Path

Absolute path names start from root (path name unique): /report/style/book

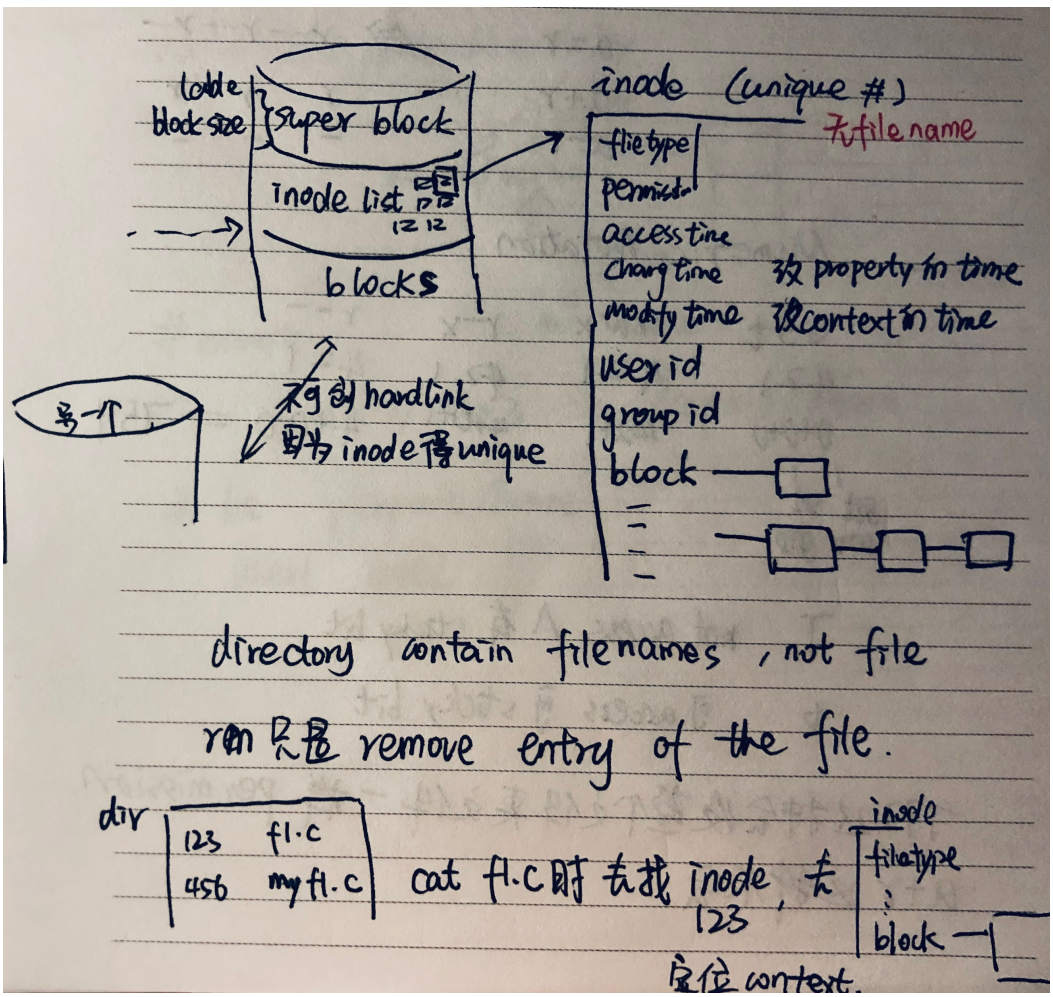
Relative path names start from current directory (not unique): style/book

. current dir .. parent dir allow traverse back up

Onion Diagram Structure of Unix

the heart is hardware, surrounding is kernel

Kernel shields applications from hardware details.



Linking Files

Unix create directory 时会 auto 产. .. link

Hard Links

every file 有个 unique ID 号也叫 inode number, 用 ls –il 即可查看 inode#.

ln afile newName

afile 与 newName 是一个 file, 它们的 inode# 会相同, link 数会+1

rm 某个 link, 该 undelying i-node 下的 link count 会-1, 其余 link 仍 valid.

- a file will exist, unless there is a link point to it.

ln [-s] f1 f2

can create hardlink to a directory, if you’re a root.

can’t, if you’re not a root.

symbolic link

区别 hard link, 不用 inode#, 用 file type.
does not refer to a files i-node, but rather to a new file whose content refers to the file.

ln –s file new

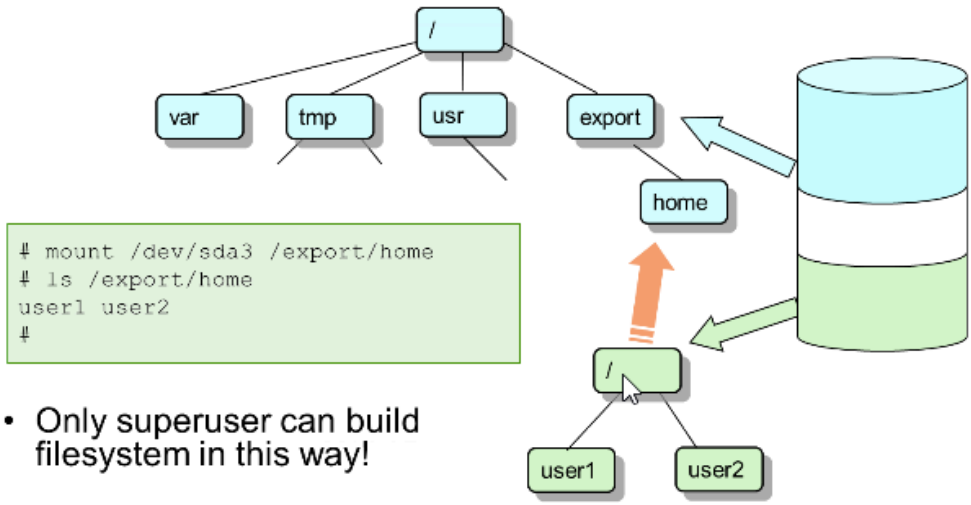
new　有新的 inode#, 在 block 里，指向同一个 file context.

rm 原 file, link new 仍在，但是 break 了，cat new 会报 no such file.

恢复原 file, link 会再次生效。

Build File system hierarchy (mount)

- Each disk device has its own hierarchical file system
  - joined together into one hierarchy with `mount` command



- Only superuser can build filesystem in this way!

Process & Access Unix

The first process in Unix is called init.

1. Init's child process set up machine and ultimately prompt user to login
2. username + password will associate user with a sub-directory of the file system (their home directory
3. once logged in, a new child process (the shell) is created to enable user to enter command.4
4. The shell is stamped with the users personal UID and shared GID.

All Processes have a parent process.

Each command is executed as a new process, and is the child of the process which invoked it.

Almost every command you run and every file you create is stamped with your UID and GID.

Create process 2 种方式

1) fork from parent -> child
2) exec run from file

System call is functions implemented inside the kernel. eg: fork, exec, exit, wait.

User has no direct access to these operation, but C API help us build applications that make use of them.

kill -9 18475	force stop job (9 hard kill, 2 interrupt, 15 Termination requested, 11 memory fault detected, 17 suspend requested)
kill	只是 send signal to jobs / process
kill %1	terminate a background jobs
kill siginal PPID	
kill siginal %JobID	
kill –stop %1	bash
stop %1	ksh
fg	resume a stopped job

Unix User

file & process 才是 Unix 的掌控者，user 只是 file process 的 attributes.

Every file & process must be owned by someone and exist in a group.

Every subsequent file or process created by user is stamped with users' identity UID(unique) and GID(shared)

GID 存在使得 user 可 access to shared files and commands, user default 至少 16 个 GID 用于 access permissions only.

Standard Unix model for access control

```
if UID (process) == 0
   allow access
else if UID (process) == UID (file)
   apply user's (owner's) access rights
else if GID (process) == GID (file)
   apply group access rights
else
   apply others access rights
```

Permission for files

user	group	other
rwx/rws	rwx/rws	rwx/rws

r		read contents
w		alter contents
x		exe/run program
s	setID	change the UID or GID of process

- etc/passwd 文件内容:

  ```
  root:*:0:0:System Administrator:/var/root:/bin/sh
  ```

  system : password : UID : GID : gecos : home_directory : loginShell

当 etc/passwd 改密码的文件权限只有 root 才有时，user 怎样改密码呢？

/usr/bin 下名叫 passwd 的 process（负责改 passwd），它的 UID，GID 本应该继承自其 parent process, 但这里若设了/usr/bin/passed 中的 set ID，即-rwsr-xr-x，则 execution UID & GID 可以 not take from parent process, but from user SetID.

Permission for directories

user	group	other
rwx	rwx	rwx

r	list contents
w	可 add/delete any files 的权利

x search, access the directory with a path, cd

t sticky group 对 directories 可 write access, A 在 dir 下创了 file 并对 group 成员不可 w, 但是 others can still delete, 因她们对 dir 可 w, 设了 sticky bit 则只有 owner 可 delete.

Chmod – change permissions

chmod [-R] <permission-mode> file [files …]

-R recursion through a dir

permission-mode

symbolic notation (user group others all)

```
chmod  u  +  r  files ...
       g  -  w
       o  =  x
       a     s
             t
```

go+r g-r ug+rx,o-w a=r a+r a-r

numeric notation

```
s s t     r w x     r - x     r - -
4 2 1     4 2 1     4 2 1     4 2 1

0+0+0     4+2+1     4+0+1     4+0+0     = 754
```

7777 enable everything, rwsrwsrwt

777 这种会使整个文件夹文件一样 permission，但 a+r 不会

Default Permission

umask 在 create 时，改 default， 事先设好 rwx 位，default 便是它们的补位。通常 umask=022, files 644 directories 755

change ownership

chown chgrp 改 UID GID of a file

```
chown fred file
chown -R fred dir
chgrp staff file1 file2
chgrp -R staff dir1 dir2
chown fred:staff newfile
```

login su+新 user 名 newgrp + 新 group 名 改 UID GID of a process

```
newgrp infotech
```

su 可 switch user to another， ^D 推出该 user.

sudo as root/specific user member , 这些用户列在 sudoers 里， 只有这些人才可用 sudo.

Unix Files

在 unix file system 里存着, 可以是 data, text, executable code

streams of bytes without structure

directories are files 存着它里面的 files 的 the name of entry

Devices files 在/dev 下

Display part of files

head		[files…]	show first 10 lines of file
	-n	[files…]	show n lines from the start
tail			show last 10 lines of file
	-n	[files…]	show n lines to the end
	+n	[files…]	skip n lines to the start, then show all

head 和 tail open file 但不 close file. (看 log 常用)

Search files

grep [-cilnvw] <RE> [files…] search **files/input** for *all lines* which **match RE** and then print them

-c	display count of matching lines
-i	case insensitive
-l	list names of files containing match lines
-n	show each line with line number
-v	only display lines that do not match
-w	search for the expression as a word

<RE> are string templates

.	any character
*	>= 0 occurrences of previous character
[abc]	any one of a, b or c

[a-z] any character in range

^ beginning of line

$ end of line

grep –l string * display the filename in which the match is found

grep '^csc' userlist show 以 csc 开头的行

grep –v ')$' userlist show 没有以)结尾的行

grep '\$' userlist show 含有$符号的行

grep ro.er userlist show 含有 ro XXX er 这种格式的词的行

Quoting (leave special character alone)

shell 见* 会 file name expansion， program 见*会知道时 regular expression, 都因为 quoting 作用。

"..."	quotes spaces & wildcards
'...'	also quotes $, ^, "
\.	quotes next character

"" 可 capture . * , ? [] space

' ' 可 capture 上述，也可 capture $, ^, "

\ 转义字符

grep '(csc.*)' userlist show all lines inside usrlist that contain "(csc" followed by any number of any characters followed by ")"

```
$ ls "p*"
p* not found
$ echo $home
/home/sparc2/dpm
$ echo "$home"
/home/sparc2/dpm
$ echo '$home'
$home
$ echo "it costs $10"
it costs
$ echo "it costs \$10"
it costs $10
```

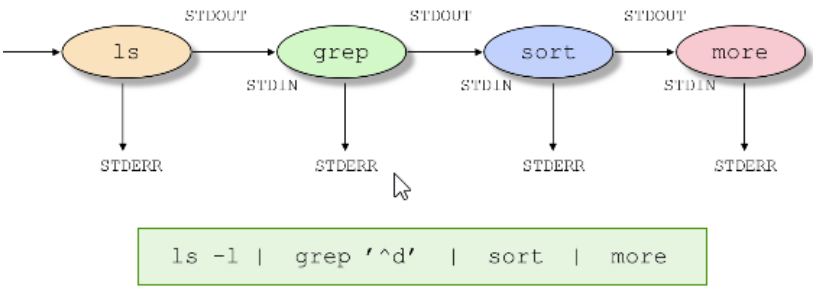
sorting files

sort –nr –k4,5 userlist

-n	sort on numeric value
-r	reverse the sort
-ka,b	sort key starts at field a (1-based) and ends at field b

Command Pipelines

The output of one command is directed into the input of another to form a pipeline




```
ls -l |  grep '^d'  |  sort  |  more
```

By default, STDERR is not passed down the pipe

Command 同时进行，shell 会 buffer output.

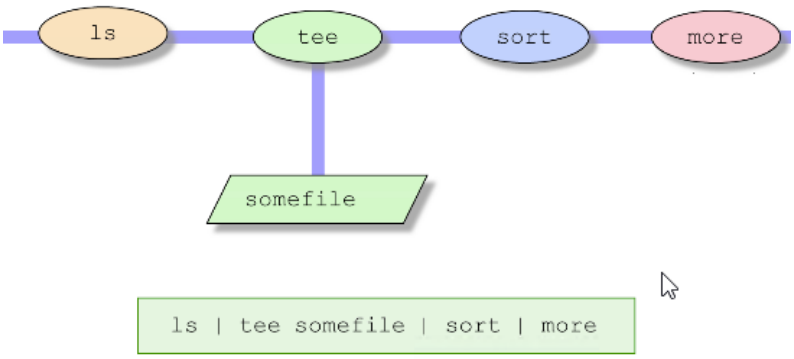
区别：

pipeline	|	cat f1 | cat	cat f1 的 output, to a command as its input
redirect	>	cat f1 > f2	cat f1 的 output, to a file
head -27 afile | tail -1			show line 27
ls –l | sort –nr –k5 | head			list 10 largest files in current dir
ls –l | grep '^d'			list only the sub-directories in current dir
who | cut –d" " –f1			tell me who are login, 找 name
command 2>&1 | command2			standard error output

ls | tee copyfile | sort | more

tee read from STDIN and write STDOUT to any arguments files. This enable multi-pipeline, use tail –f process can see in another window.

`tee` enables the stream to be diverted



cat < dev/tty | tee –a log| sh| tee –a log　　把 input output 都存在 log 中

Translate Command tr

tr abc ABC < myfile　　把小写变大写

```
$ tr '[a-z]' '[A-Z]'
hello
HELLO

$ tr '[aeiou]' '[AEIOU]' < input
ThIs Is sOmE InpUt thAt I shAll UsE tO dEmOstrAtE thE UtIlIty
Of sEvErAl Of thE stAndArd fIltErs AvAIlAblE In UnIx. ThE fIlE
Is nOt vEry lOng bUt shOUld bE sUffIcIEnt tO shOw hOw thE
cOmmAnds OpErAtE.

$ tr ' ' '\n' < input
This
is
some
input
that
I
```

tr –d ‘a’　　删 a

tr –cd ‘a’　　删非 a

- Delete characters from an input stream

```
$ tr -d '[aeiou]'
Hello world
Hll wrld
```

- Use complement of input character pattern

```
$ tr -cd '[aeiou]'
Hello world
eoo
```

- Character classes for commonly occurring character types

```
$ tr '[:lower:]' '[:upper:]'
Hello, world
HELLO, WORLD
```

- Different sized match and replacement strings
  - map all input characters to same output

```
$ tr 'aeiou' '-'
The moon's a balloon
Th- m--n's - b-ll--n
```

  - map some input to one char, others to another

```
$ tr 'aeiou' '-+'
The moon's a balloon, for sure it is
Th+ m++n's - b-ll++n, f+r s+r+ +t +s
```

first char in input mapped to '-', others to '+'

  - use –s (squeeze) option to remove duplicate replaced characters

```
$ tr -s 'aeiou' '-'
The moon's a balloon
Th- m-n's - b-ll-n
```

remove adjacent '-' in the output

Compare files

diff f1 f2 show diff between 2 lines

diff –e f1 f2 how f2 can be generated from f1

cmp

comm

diff3

cutting fields (remove fields form each line of the file)

cut –d" " –f1 userlist select field 1 from userlist, where fields are delimited(界定) by a single space. (default, the field separator is a single tab) , 选第 1 field

cut –c17-22 userlist show 每行的第 17-22 个字符

ls –l | sed 1d | awk ‘{print $9}’ 删了第一 line 再 print 第九列

Pasting File Contents (组合多个文件内容串联成 single line in output)

cat f2 | paste f1 - f1 f2 的形式 output

Duplicate Lines (remove dup)

`uniq -c`	count number of duplicate line
`uniq -d`	only print repeated lines
`uniq -u`	only print non-repeated lines

sort filename | uniq

Counting word

`wc -l`	just report lines
`wc -w`	just report words
`wc -c`	just report characters

ls –l | wc –l show 当前 dir 下的文件个数 + 1,

ls | wc –l show 当前 dir 下的文件个数

ls –l | sed 1d |wc -l 删了第一 line 再执行 count

print files

`lpr` sends files to a line or laser printer

```
$ lpr -Plw story1
```

`lpq` queries the state of the print queue

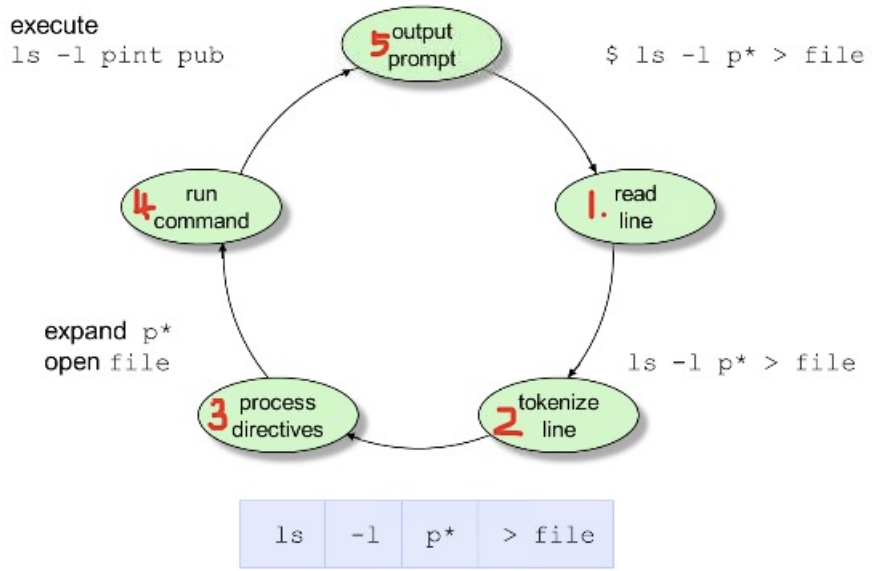
```
$ lpq -Plw
lw is ready and printing
Rank    Owner   Job     Files       Total size
active  dpm     555     userlist          385 bytes
1st     dpm     556     story1      456 bytes
```

`lprm` dequeues jobs from the print queue

```
$ lprm -Plw 556
suna2: dfA556sparc2 dequeued
suna2: dfA556sparc2 dequeued
```

How shell works?

exe 这里 父 shell 唤醒子 ls, 子 run 并 return, 父等到 return 则 output



Script 会执行把所有 command 操作及结果显示道 typescript 文件中，^D 退出 Script。

A file cannot act upon another file or upon a process.

A process can act upon a file or upon a process.

```
process > file ✓
file > process
file < file
process < process
process | process ✓
process | file
file | process
file | file
process < file ✓
```

Job: a single unit for the shell to group and manage all processes together with a command

jobs	tell still running or done
bg %1	put to background
fg %1	bring to foreground
stop %1	
^C	terminate
^Z	suspend

Background Commands &

```
$ cc bigprog.c &
[1] 18581
```

```
$ cc bigprog.c > diag 2>&1 &
```

merge output+error in 后台

Composing Commands

com1; com2; com3	顺序执行
com1& com2&com3	all 执行 at the same time,(1, 2 后台执行，show 3 前台)
com1 && com2 && com3	1 return 0(success) then 2 exec, if 2 return 0(success), 3 exec
com1 || com2 ||com3	1 return 0 then done, else 2 exec, if all return 0, 3 exec
(ls | sort | more) 2 > err	redirect STDERR for an entire pipeline

Grouping Commands

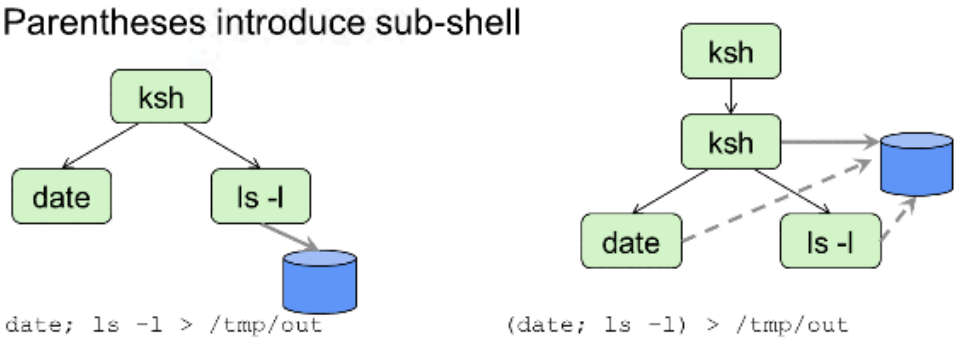
- Capturing output from a sequence of commands

```
$ (date; ls -l) > /tmp/out
```

- Running a sequence of commands in the background

```
$ (sleep 600; echo All Done) &
```

- Parentheses introduce sub-shell



Local & Environment Variables

local variables	part of shell, 只在 current shell visible, not visible to any command that are invoked
environment variables	externally to shell, available in subshells and other command

Some variables set by most shells

HOME	the path of the users home directory
LOGNAME	the users login name
PATH	the command search path for the shell
PS1	the shell's primary prompt
PS2	the shell's secondary prompt
SHELL	the path name of the shell
TERM	the type of the terminal
IFS	input field separator
MAILCHECK	the frequency e-mail is checked
DISPLAY	X server for GUI applications

set	display all variables local + env (internal to shell)
env	display env (those outside the shell)
set A=B	change local variables, no spaces, 因其并非 command
setenv A=B	change environment variables
export var	an env variable var 被 var contents 取代

The PATH Variable

Shell default never search current dir

Path is used for shell to find commands, searches each directory in sequence.

- Extending the PATH

```
$ echo $PATH
/usr/bin:/usr/ucb:/usr/local/bin:/usr/X11/bin
$ PATH=${PATH}:/home/fred/bin
echo $PATH
/usr/bin:/usr/ucb:/usr/local/bin:/usr/X11/bin:/home/fred/bin
```

prepend PATH /etc	ksh/bash short-cut
append PATH /etc	ksh/bash short-cut

pathname ./program	在 current dir 下强制 run pathname 里的 program
echo $?	Echoes (prints) the exit value for the previous command

myprog a1 a2 a3

echo $0 myprog

echo $1 a1

…

Command Substitution &(...)

useful for setting values of variables

ls | rm　　fails, 因 rm 不要 list files 为 input, 它要 command line arguments

rm ‘ls’　　works

echo ‘date’ 旧写法　　←等价→　　echo $(date)　shell 执行 date，后给$( ) output

Set Editor／History

set history = 50 enable history mechanism

history display command history

!! re-run last command

!n re-run command n

!str re-run last command beginning n

^old^new^ substitute new for old in last command

!n:s/old/new/ substitute new for old in command n

Command Aliasing(自创 symbolic names)

```
$ alias laser='lpr -Plw'
$ alias dir='ls'
$ alias r='fc -e - '
$ alias mroe=more
$ alias rm='rm -i'
$ alias ls='ls -F'
$ alias
$ unalias rm
```

requires the trailing space

redefines rm

lists aliases

Set Prompt

set prompt = “[\!]%”　　allow history number

alias cd ‘cd \!*; set prompt = “[\!]$cwd%”’　　each time user changes cwd, prompt is also updated.

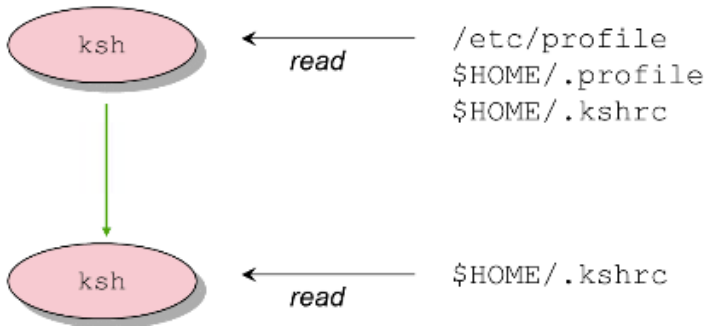
Unix Startup Files

The ENV environment variable must be configured

```
ENV=$HOME/.kshrc; export ENV
```

Used to configure the shell automatically

- when the user logs in
- when a new shell (perhaps in a window) is created



$HOME/.custom/.profile mainly environment settings, defines user's environment

```
PATH=${PATH}:${HOME}/bin ; export PATH
PS1="[!]$ " ; export PS1

umask 077
PAGER=/usr/local/bin/less ; export PAGER
MANPATH=/usr/share/man:/usr/local/man
export MANPATH
LD_LIBRARY_PATH=/usr/X11/lib:/usr/lib
export LD_LIBRARY_PATH
HISTSIZE=50 ; export HISTSIZE
HISTFILE=${HOME}/.sh_history ; export HISTFILE
```

$HOME/.custom/.envfile also env + used on a per-shell basis

```
module load fsf/gmake
module load perl5/core/5.8
module load perl5/devel/5.8
module load fsf/gcc/default
module load msjava/sunjdk/1.5.0
set -o noclobber
```

$HOME/.custom/.interactive modify interactive session behaves

```
EDITOR=vi #need to set here for vi mode cmd line editing
export EDITOR
alias gerp=grep
alias what=ls
```

$ . .interactive 使上述改在 current shell 生效，而非./.interactive,（它在 subshell 执行，设置只在 subshell 生效，不是 current shell）

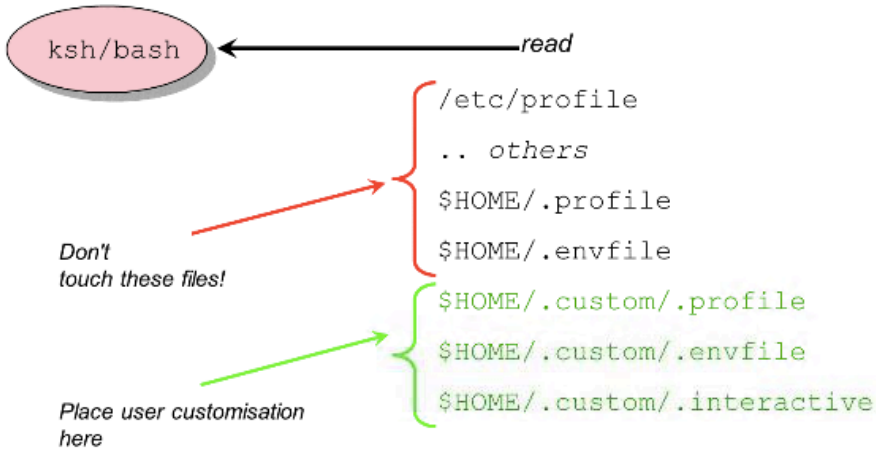
set -o ignoreof

set +o ingnoreof

set –o noclobber 不允许 overwrite file

Aurora Startup Files

- Chain of 16 or more startup scripts
  - mainly depending on user *model (programmer, trader, etc)*
  - model setup when user created, stored in `.model`

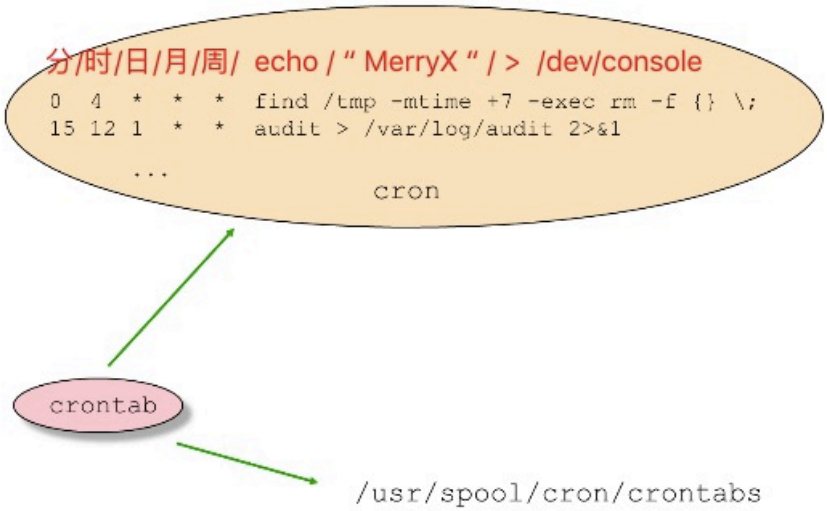


Corn is a system daemon which dispatches jobs (commands or scripts of commands) according to a given timetable.

Scheduling Repetitive Tasks

- cron runs commands at pre-defined times

  在设定的时间run，(min若设*30表示，每30分钟run



user communicate their timetables to cron using crontab command.

crontab also stores timetable information in files within the spool directory.

- crontab sends a timetable of commands to cron

```
crontab [-elr] [filename] [username]

    e        edit crontab entries
    l        list crontab entries
    r        remove contab entries
```

- crontab entries follow the same format

```
0  0 25 12  * echo "Merry Xmas" > /dev/console
```

min hour month-day month weekday

at

Scheduling One-off Tasks

- One off tasks should be scheduled with `at`
  - front end to crontab

```
$ at 22:00
at> cc SomeBigJob.c
at> ^D
job 5397 at Sat Jan 30 22:00:00 1995
$
```

- Examining the at queue

```
$ atq
Rank   Execution Date         Owner     Job#  Queue   Job Name
 1st   Jan 30th, 1995 22:00   fred      5397   a        stdin
$
```

Unlike cron which is orientated towards repetitive tasks, at is useful for one-off tasks. The syntax is quite powerful and more intuitive than most Unix commands:

```
at now + 1 hour < someComm
and
at 0930 Mar 10
echo "echo weekend" | at 5
 pm Friday
```

at commands are managed by cron.

find Searching Command

```
$ find / -name foobar -print
```

where to start looking | what to look for | what to do with the found files

`find` understands a variety of predicates for selecting files

```
-name <filename pattern>
-user <user name>
-group <group name>
-size [+|-] <size in blocks>
-perm [-] <octal number>
-atime [+|-] <days>
-mtime [+|-] <days>
-ctime [+|-] <days>
-type [d|f|l]
-inum <i-node number>
```

find Examples

- List details of all files owned by root

```
find / -user root -ls
```

- Print pathnames of all files older than 3 months, which are larger than 100K

```
find / -atime +90 -size +200 -print
```

- Print pathnames of all C programs from the current directory

```
find . -name "*.c" -print
```

- Find all SUID root programs and e-mail administrator their pathnames

```
find . -user root -perm -4000 -print \
   | mail root@pluto
```