

# The Shell

David Morgan

© David Morgan 2003-19

## What is a shell?

- a “command shell” is a program
  - defined by its function: it handles commands
  - /bin/sh, or maybe /bin/bash or /bin/csh or /bin/zsh (different ones)
- a “user shell” is also a program
  - but not defined by its function: it could do/be anything
    - *could* be a command shell
    - need not be
  - defined as the program that runs in response to a successful login
  - determined for each user in /etc/passwd (the list of defined user accounts)

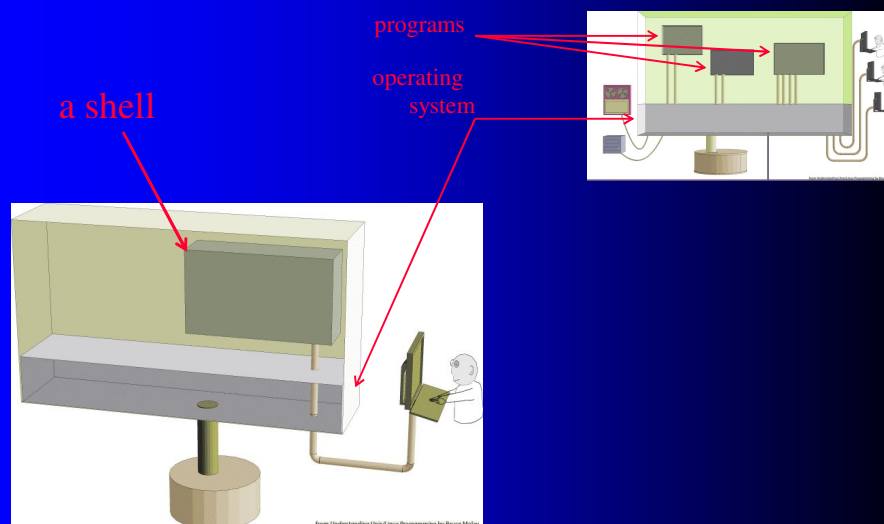
© David Morgan 2003-19

## What is “the shell” not?

- *not* the operating system
  - merely one program that can run under the OS
- *not* necessary
  - machine doesn’t necessarily even run it

© David Morgan 2003-19

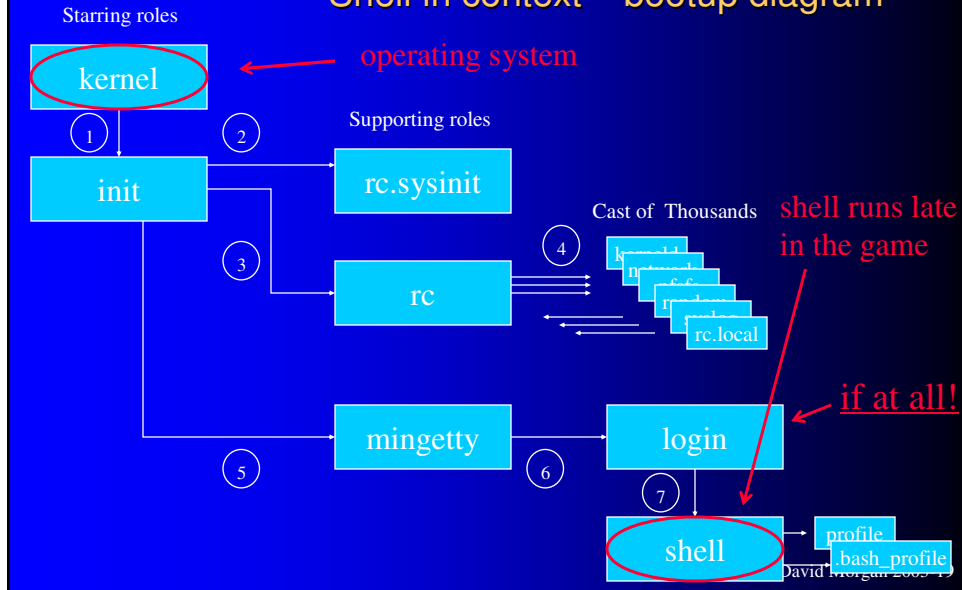
## *Not* the operating system



© David Morgan 2003-19

# Not necessary

## Shell in context – bootup diagram

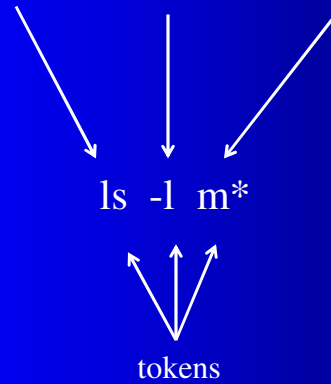


## What does the shell do?

- Command processing
  - parse
  - expand
  - execute
- I/O redirection
- Piping
- Environment control
- Background processing
- Shell scripts

## Command-line format

command [ - options ] [ arguments ]



© David Morgan 2003-19

## Command processing

- Reads what you type at the prompt
- Parses it (command/options/args token analysis)
- Optionally expands (transforms) it
- Responds by
  - locating/executing program file, or
  - emitting error message

© David Morgan 2003-19

## Command-line expansion

- transformation of command line's args
- performed by shell on each argument
- per special characters in arg (if any)
- altered result gets executed
- several transformation types

© David Morgan 2003-19

## Types of transformation

- brace expansion
- tilde expansion
- parameter expansion
- variable expansion
- command substitution
- arithmetic expansion
- word splitting
- filename (pathname) expansion

– successively, in above order

© David Morgan 2003-19

# Transformation triggers

- caused by special characters (metacharacters)
- & ; | \* ? " ' ` [ ] ( ) \$ < > { } ^ # / \ % ~
- shell has extraordinary responses to them
- ordinary characters are nouns, embedded special characters are adverbs– “how to” treat the surrounding ordinary characters

© David Morgan 2003-19

# Examples: various transformations

```
root@EMACH1:~/test
File Edit View Terminal Tabs Help
[root@EMACH1 test]# echo repeat,port,sist}er } brace expansion { }
repeater reporter resister
[root@EMACH1 test]#
[root@EMACH1 test]# echo ~/readme.txt } tilde expansion ~
/root/readme.txt
[root@EMACH1 test]#
[root@EMACH1 test]# MYBIRTHSTONE=opal
[root@EMACH1 test]# echo MYBIRTHSTONE
MYBIRTHSTONE
[root@EMACH1 test]# echo $MYBIRTHSTONE
opal
[root@EMACH1 test]#
[root@EMACH1 test]# echo ls
ls
[root@EMACH1 test]# echo $(ls)
abc areport bbc breport cbc dbc demo-filexp ebc report report1 report2 report32
reportk reportkz report.txt
[root@EMACH1 test]# echo 22+33
22+33
[root@EMACH1 test]# echo ${22+33}
55
[root@EMACH1 test]# echo report?
report1 report2 reportk
[root@EMACH1 test]#
```

command substitution  
\$( ) or ` `

© David Morgan 2003-19

# Expansion order matters

```
root@EMACH1:~/test
root@EMACH1 test:~# echo re{peat,port,sist}er
repeater reporter resister
root@EMACH1 test:~#
root@EMACH1 test:~# VAR=re{peat,port,sist}er
root@EMACH1 test:~# echo $VAR
re:peat,port,sist:er
root@EMACH1 test:~#
root@EMACH1 test:~#
root@EMACH1 test:~#
root@EMACH1 test:~# echo ${22+33}
55
root@EMACH1 test:~#
root@EMACH1 test:~# VAR=S[22-33]
root@EMACH1 test:~# echo $VAR
55
root@EMACH1 test:~#
```

different

same

brace & variable expansion  
does not “work” because  
brace expansion is already over  
when variable expansion occurs

arithmetic & variable expansion  
“works” because  
arithmetic expansion is yet to be performed  
when variable expansion occurs

© David Morgan 2003-19

# Filename expansion metachars

- \* any character(s) even none! echo \*.\*
- ? any single character cat lab?
- [...] a single one of ... ls lab[a-m]

see man 7 glob

© David Morgan 2003-19

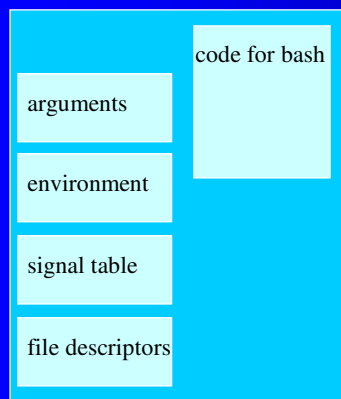
# Examples: filename expansion

```
root@EMACH1:~/test
[File Edit View Terminal Tabs Help]
[root@EMACH1 test]# ls
abc  bbc  cbc  ebc  hi  report  report2  reportk  report.txt
areport  breport  dbc  ha  ho  report1  report32  reportkz
[root@EMACH1 test]#
[root@EMACH1 test]#
[root@EMACH1 test]# echo report*[?z]
report2 report32 reportkz
[root@EMACH1 test]#
[root@EMACH1 test]#
[root@EMACH1 test]# echo h?
ha hi ho
[root@EMACH1 test]#
```

© David Morgan 2003-19

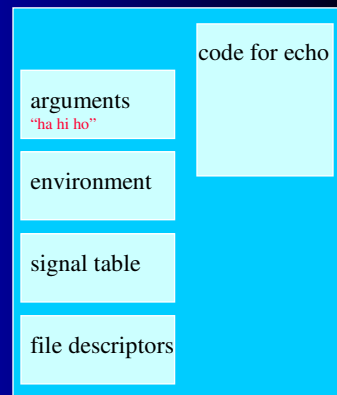
# shell vs launched (child) process: command-line before vs after

bash, as parent process



before  
user types "echo h?"  
shell passes "ha hi ho" to echo as arg list (for example)

echo, as child



after  
Child is expansion-oblivious, never sees the "?"

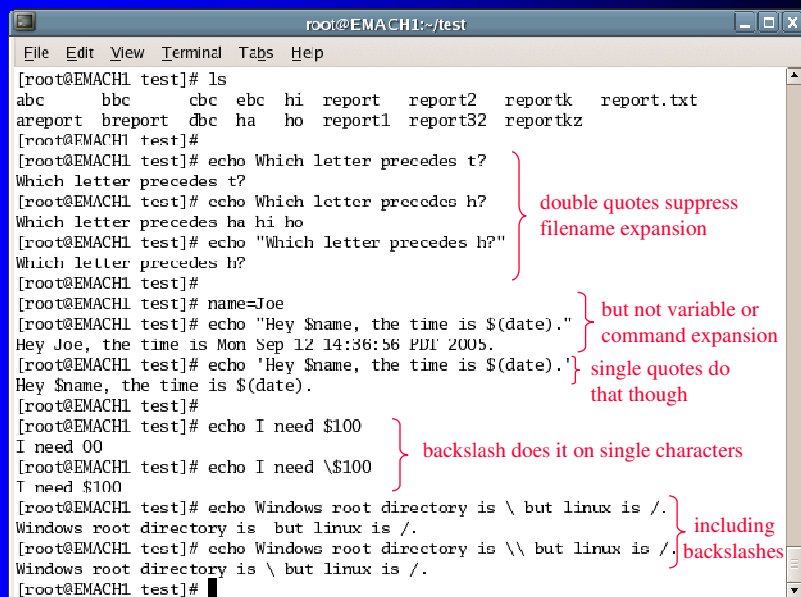


# Suppressing expansion

- prevents metacharacters' special interpretation
- single quotes
  - defeat all enclosed metacharacters
- backslash
  - defeats the metacharacter that follows it
- double quotes
  - defeat enclosed metacharacters except for variable ( \$ ) and command ( \$( ) ) expansion

© David Morgan 2003-19

# Examples: expansion suppression



```
root@EMACH1:~/test
File Edit View Terminal Tabs Help
[root@EMACH1 test]# ls
abc    bbc    cbc    ebc    hi    report  report2  reportk  report.txt
areport breport dbc    ha    ho    report1  report32  reportkz
[root@EMACH1 test]#
[root@EMACH1 test]# echo Which letter precedes t?
Which letter precedes t?
[root@EMACH1 test]# echo Which letter precedes h?
Which letter precedes ha hi ho
[root@EMACH1 test]# echo "Which letter precedes h?"
Which letter precedes h?
[root@EMACH1 test]#
[root@EMACH1 test]# name=Joe
[root@EMACH1 test]# echo "Hey $name, the time is $(date)."
```

double quotes suppress filename expansion

```
Hey Joe, the time is Mon Sep 12 14:56:56 PDT 2005.
[root@EMACH1 test]# echo 'Hey $name, the time is $(date).'
Hey $name, the time is $(date).
[root@EMACH1 test]#
[root@EMACH1 test]# echo I need $100
I need 00
[root@EMACH1 test]# echo I need \$100
I need $100
[root@EMACH1 test]# echo Windows root directory is \ but linux is /.
Windows root directory is but linux is /.
[root@EMACH1 test]# echo Windows root directory is \\ but linux is /.
Windows root directory is \ but linux is /.
[root@EMACH1 test]#
```

but not variable or command expansion

single quotes do that though

backslash does it on single characters

including backslashes

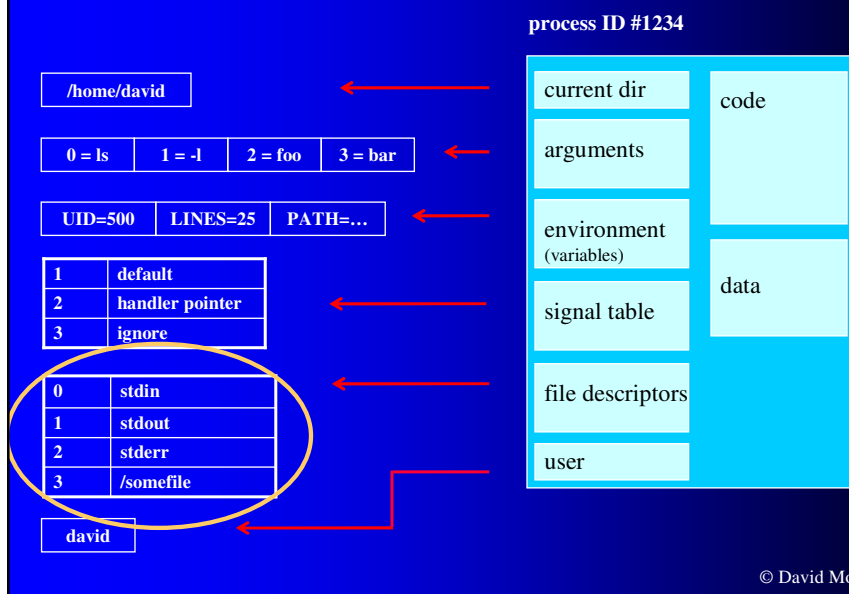
© David Morgan 2003-19

# I/O redirection

- cat lab? >> assignments
- mail dmorgan1 < prepared-message
- ls -z 2> errorlog.txt

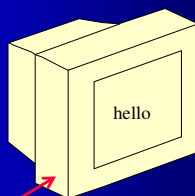
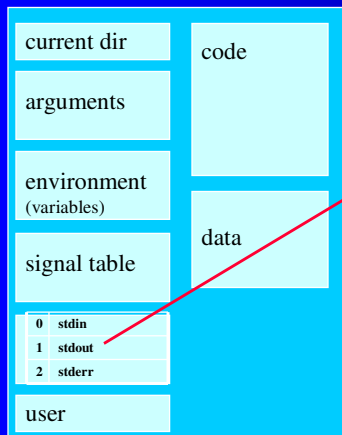
© David Morgan 2003-19

# I/O descriptors in a unix process



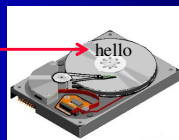
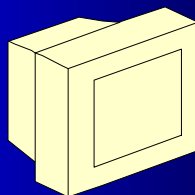
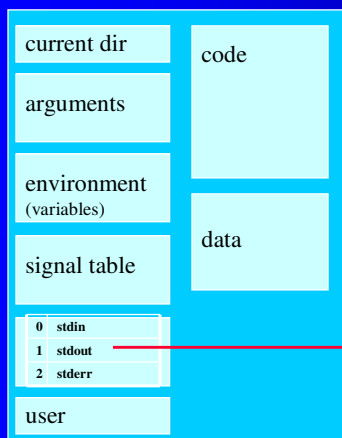
© David Morgan 2003-19

# echo "hello"



© David Morgan 2003-19

# echo "hello" > file



© David Morgan 2003-19

## myecho.c (an echo command do-alike)

```
[root@instructor ~]# > file
[root@instructor ~]# cat file ← empty file
[root@instructor ~]#
[root@instructor ~]# cat myecho.c

main (int argc, char *argv[])
{
    write(1, ← a program that writes stuff to "1"
          argv[1],      strlen(argv[1]) );
    write(1,      "\n",      1      );
}

[root@instructor ~]# ./myecho "roses are red"
roses are red ← stuff program wrote is on screen; program does not write to "screen"
[root@instructor ~]# ./myecho "roses are red" > file
[root@instructor ~]#
[root@instructor ~]# cat file
roses are red ← stuff program wrote is in file; program does not write to "file"
[root@instructor ~]# █
```

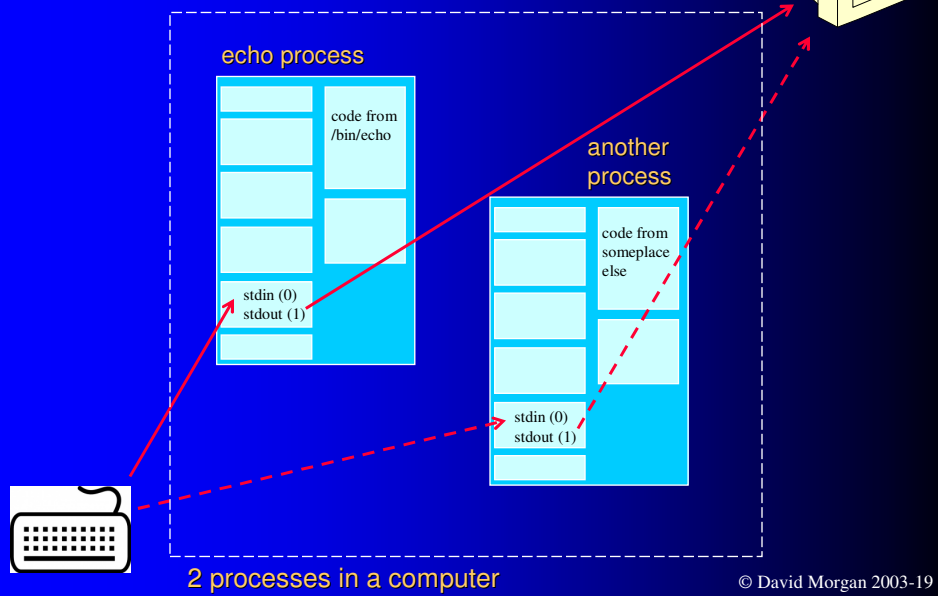
© David Morgan 2003-19

## Piping

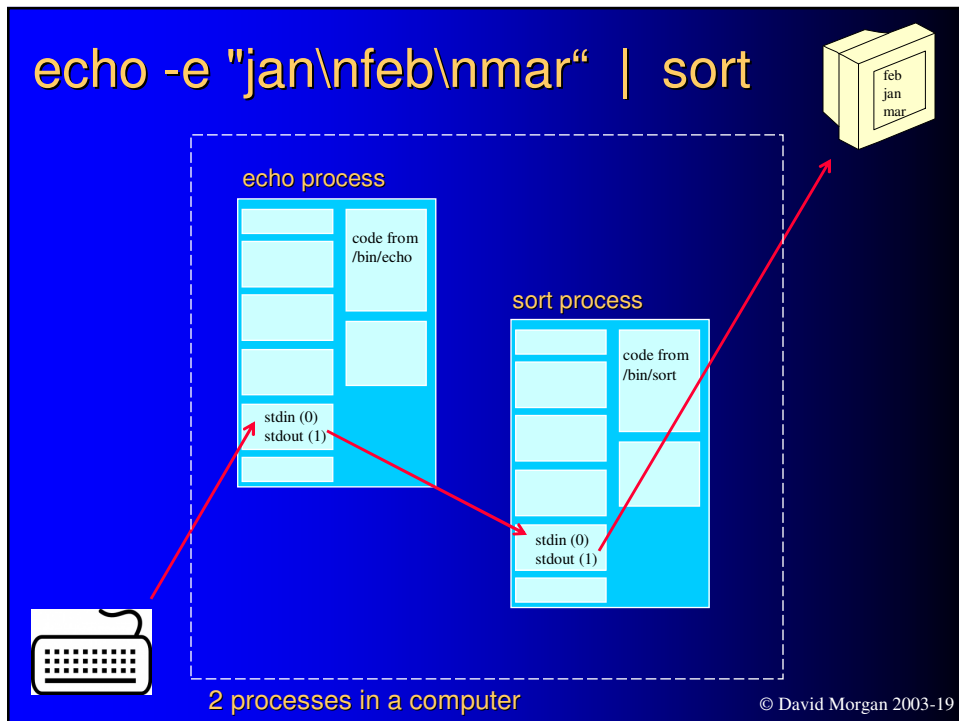
- who | sort +4
- ls -l | grep ^d

© David Morgan 2003-19

echo -e "jan\nfeb\nmar"



echo -e "jan\nfeb\nmar" | sort



## Environment control

- variables
  - environment vars (get passed to child processes)
  - local vars (don't get passed)
- setting variables
  - declare DAY=Monday , or
  - DAY=Monday
- displaying variables
  - set or declare will display all variables
  - env or printenv will display environment vars only
  - echo \$DAY to display value of single var

© David Morgan 2003-19

## Shell scripts

- Files containing a list of commands
- Permissions set to executable
- May be run as programs

© David Morgan 2003-19

## Information

- `$man bash` (man page for bash)
- Unix Shells by Example, Ellie Quigley
- A Practical Guide to Red Hat Linux, Mark Sobell, Chapter 9, section “Command Line Expansion”