Lập trình Hệ thống

Unix Programming

Part 1: Shell Introduction

Nguyễn Quốc Tuấn

Network and Communication System Department
Faculty of Electronics and Communications
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Shell Intro

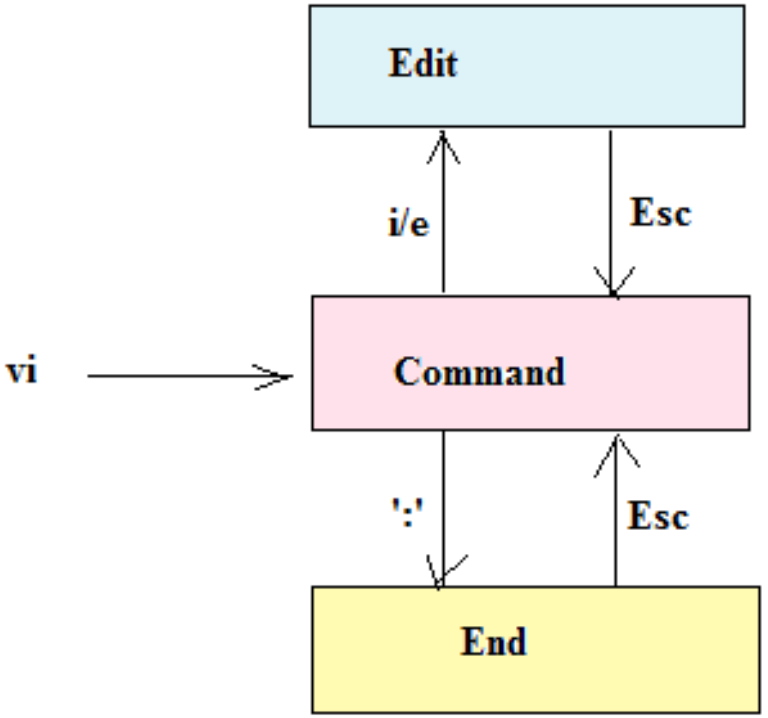
❑ SOẠN THẢO FILE

- Cho phép viết các chương trình Shell / ngôn ngữ C
- Truy cập từ xa (terminal) sử dụng

$ vi file // Soạn thảo file

Thực hiện 3 chế độ

+ Edit : soạn thảo
+ Command : điều khiển
+ End : Cất /thoát



Shell Intro

❑ SOẠN THẢO FILE

- **Chế độ lệnh**
  - i: chèn
  - e: hiệu đính
  - x: xóa trước
  - dd: xóa dòng
  - h: di chuyển con trỏ lên
  - l: di chuyển con trỏ xuống
  - j: di chuyển con trỏ sang phải
  - k: di chuyển con trỏ sang trái

- **Chế độ chấm dứt**
  - ! : phủ định
  - q: thoát

Shell Intro

❑ SOẠN THẢO FILE

- **Chế độ soạn thảo**
  - Soạn thảo thông thường
  - Khi có lỗi trở về chế độ lệnh

- Sau khi soạn thảo file chương trình shell
  - File có thể đổi mode để có thể vận hành được

    $ chmod 755 myfile

Shell Intro

❑ CÁC LỆNH LÀM VIỆC VỚI FILE

- ***Sinh viên tự đọc các lệnh***

- Lệnh so sánh nội dung 2 file

  $ ***cmp [options] file1 file2***

  - *thông báo vị trí khác nhau đầu tiên giữa chúng*

- Sai khác giữa 2 file

  $ diff [option] file1 file2

  - liệt kê sự khác nhau giữa 2 file

- Lựa chọn hàng của 1 phần của file

  $cut [option] file

- Gép file

  $paste [option] file1 file2

Shell Intro

❑ CÁC LỆNH LÀM VIỆC VỚI FILE

- *Sinh viên tự đọc các lệnh*

- Lệnh xắp xếp file theo nội dung file

  $sort [options] file

- Lệnh hủy bỏ hàng kép

  $ uniq [option] file [new-file]

- Lệnh xác định loại file

  $file [option] file

Shell Intro

❑ CÁC LỆNH LÀM VIỆC VỚI FILE

- Ví dụ

```
$ cut –c 1-3,8-12  file1
Ngaai oi
Bo
Daic Quoc
Kho
```

```
$ cat file1
Ngay mai oi di choi
Bo ho
Dai hoc Quoc gia
Khong

$ cat file2
Ngay mai toi di choi
Bo Ho
Dai hoc Quoc gia
Khong
Co
```

```
$ diff filel file2
    1,2c1,2
    < Ngay mai oi di choi
    < Bo ho
    ---
    > Ngay mai toi di choi
    > Bo Ho
    4a5
    > Co
```

Shell Intro

❑ CÁC LỆNH LÀM VIỆC VỚI FILE

- Ví dụ

```
$ sort file1
Bo ho
Dai hoc Quoc gia
Khong
Ngay mai oi di choi
```

```
$ uniq  data newda
$ cat newda
        1  2   3  4  10
        9  12  3  23 67
        9  7   5  44 1
        5  2   23 11 4
$ uniq -1 data
        1  2   3  4  10
        9  12  3  23 67
        5  2   23 11 4
```

```
$ cat data
        1  2   3  4  10
        9  12  3  23 67
        9  7   5  44 1
        9  12  3  23 67
        5  2   23 11 4
```

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Đặc trưng tiến trình**
  - Chương trình là một file thực hiện được
  - Tiến trình là một chương trình đang được thực hiện
  - Từ một chương trình có thể sinh ra nhiều tiến trình trong hệ thống
  - Mỗi tiến trình được xácđịnh thông qua một số nguyên duy nhất gọi là PID (process identification)

Một tiến trình đang được vận hành

- Dựa trên
  - + Số nhận dạng (pid)
  - + Sở hữu (uid)
  - + đường truy cập

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Đường truy cập**

- Mỗi tiến trình luôn được xuất phát từ một nguồn → điểm truy cập

  Ví dụ ;

  Tiến trình từ file trên ổ dĩa →

  từ trên mạng qua giao diện Ethernet., Wifi. →

- Theo dõi tiến trình

  $ ***top*** để theo dõi trạng thái các tiến trình được cập nhật liên tục

  $ ***ps*** để xem trạng thái các tiến trình tại một thời điểm

```
$ sleep(10)
$ ps
        PID     TTY    TIME    CMD
      27628    pts/7   0:00    sleep
      27619    pts/7   0:00    bash
      27629    pts/7   0:00    ps
```

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Bộ mô tả file**

- Tất cả I/O cho tệp, pipe và socket được xử lý bởi kernel qua một cơ chế gọi là bộ mô tả tệp. Bộ mô tả tệp là một số nguyên không dấu được kernel sử dụng để tham chiếu các file đang mở và các luồng I/O.
- Mỗi tiến trình kế thừa mô tả tệp của chính nó từ cha nó. Ba mô tả tệp đầu tiên**: 0 là stdin, 1 là stdout và 2 là stderr.** Khi mở một file, mô tả có sẵn tiếp theo là 3 và nó sẽ được gán cho tệp mới. Nếu tất cả các mô tả tệp có sẵn đang được sử dụng, thì không thể mở tệp mới.

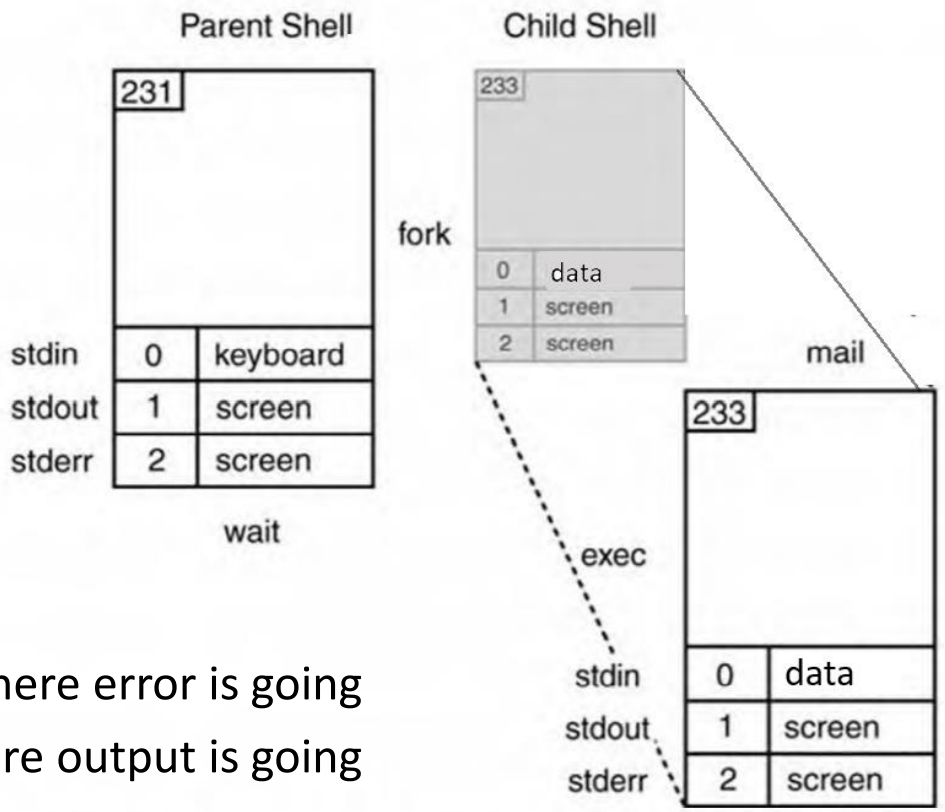
- **Định hướng lại**

- Khi một bộ mô tả tệp được gán cho thứ gì đó không phải là thiết bị đầu cuối, nó được gọi là chuyển hướng I/O. Shell thực hiện chuyển hướng đầu ra tới một tệp bằng cách đóng bộ mô tả tệp đầu ra chuẩn, 1 (thiết bị đầu cuối), rồi gán bộ mô tả đó cho tệp. Khi chuyển hướng đầu vào chuẩn, trình bao đóng bộ mô tả tệp 0 (thiết bị đầu cuối) và gán bộ mô tả đó cho một tệp

Shell Intro

❑ TIẾN TRÌNH SHELL

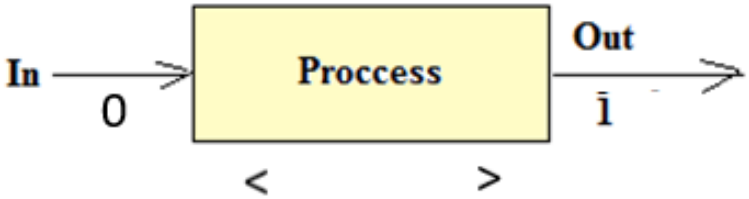
- Ví dụ: $mail qtuan < data



Toán tử	*Thực hiện*
<	Redirect input
>	Redirect output
>>	Append output
2>	Redirect error
1>&2	Redirect output to where error is going
2>&1	Redirect error to where output is going

Shell Intro

❑ TIẾN TRÌNH SHELL



- **Định hướng của tiến trình**
  - Định hướng ra mặc định là màn hình
  - Đinh hướng vào mặc định là bàn phím

- **Định hướng lại tiến trình**

- Định hướng lại lối ra

  " > "

  Ví dụ: $date > ngay

- định hướng lại lối vào

  " < "

  Ví dụ $mail qtuan < tam

- ">>" cho phép viết tiếp

Vi dụ:

$ who > file

$ cat filel file2 » fileB

$ mail torn < file

$ find / -name file -print 2> errors

% (find / -name file -print > /dev/tty) > &errors

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Ví dụ**

```
$ cat <<- DONE
> Hello there
> What's up?
> Bye now The time is `date`
> DONE
    Hello there
    What's up?
    Bye now The time is Thu Sep 20 19:48:23 PST 2020.
$
```

```
$ sort «D0NE     // DONE đầu cuối người dung
> pears
> apples
> bananas
DONE            //  kết thúc BONE
    apples
    bananas
    pears
$
```

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Lệnh exe và định hướng lại**

- Lệnh execute sử dụng chạy chương trình mới mà không cần bắt đầu mở tiến trình mới. Đầu ra/vào chuẩn được thay đổi bằng lệnh exe mà không cần tạo shell con.
- Nếu một file được mở bằng exe, lệnh đọc tiếp sẽ di chuyển con trỏ file mỗi lần một dòng cho đến cuối tệp. File phải đóng lại để bắt đầu đọc lại từ đầu. Nếu sử dụng tiện ích UNIX như cat and sort, hệ điều hành tự đóng tệp khi lệnh hoàn thành.

Lệnh	*Thực thi*
exec ls	*Execute ls. When ls is finished, shell in which it was started does not return.*
exec < filea	*Open filea for reading standard input.*
exec > filex	*Open filex for writing standard output.*
exec 3< datfile	*Open datfile as file descriptor 3 for reading input.*
sort <&3	*Sort datfile.*
exec 4>newfile	*Open newfile as file descriptor (fd) 4 for writing.*
ls >&4	*Output of ls is redirected to newfile.*
exec 5<&4	*Make fd 5 a copy of fd 4.*
exec 3<&-	*Close fd 3.*

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Lệnh exe và định hướng lại**

- Ví dụ

1 *$ exec date*

Sun Sep 20 10:07:34 POT 2020

<Login prompt appear if you are in your login shell>

2 *$ exec > temp*

$ ls

$ pwd

$ echo Hello

3 *$ exec > /dev/tty*

4 *$ echo Hello*

Hello

1. *Vì lệnh date được thực hiện thay cho shell hiện tại, khi lệnh date thoát ra, shell sẽ kết thúc. Nếu một B.shell đã được khởi động từ C.shell, thì B.shell thoát và lời nhắc C.shell*
2. *Lệnh exe mở đầu ra tiêu chuẩn cho shell hiện tại vào file tempi. Đầu ra từ ls, pwd và echo sẽ không còn xuất hiện trên màn hình nữa mà cất vào file temp.*
3. *Lệnh exe mở lại đầu ra tiêu chuẩn cho thiết bị đầu cuối. Bây giờ, đầu ra sẽ chuyển đến màn hình*
4. *Đầu ra tiêu chuẩn đã được chuyển hướng trở lại thiết bị đầu cuối (/dev/tty).*

Shell Intro

❑ TIẾN TRÌNH SHELL

- **Lệnh exe và định hướng lại**

- Ví dụ:

```
1   $ cat doit
    pwd
    echo hello
    date
2   $ exec < doit
    /home/Admin1/qtuan/SV-1802345
     hello
     Sun Sep 20 10:07:34 PDT 2020
3   %
```

```
1  $ exec 3> filex
2  $ who >& 3
3  $ date >& 3
4  $ exec 3>&-
5  $ exec 3< filex
6  $ cat <&3
       qtuan   console   Sep   20 09:53
       qtuan   ttyp0     Sep   20 09:54
       qtuan   ttypl     Sep   20 09:59
       qtuan   ttyp2     Sep   20 15:42
       Sun Sep 20 13:31:31 PDT 2020
7  $ exec 3<&-
8  $ date >& 3
       Sun Sep 20 13:41:14 PDT 2020
```

Shell Intro

❑ TIẾN TRÌNH

- Thường khi thực hiện một lệnh, lệnh đó sẽ chạy ở chế độ foreground và dấu nhắc chỉ xuất hiện khi lệnh hoàn tất.
- Nhiều tiến trình không thuận tiện đợi lệnh hoàn thành. Bằng cách đặt dấu (&) ở cuối dòng lệnh, shell thực hiện lênh ở chế độ background.
- $! biến chứa số PID của công việc cuối cùng được đặt trong chế độ background.
- Các tiến trình không phải là background là tiến trình foreground

Ví dụ trong thực tế:

Nhân viên ở quầy giao dịch hàng không: foreground

Nhân viên kỹ thuật radar: background

- **Tiến trình Foreground (mặt)**
  - Chỉ vận hành 1 chương trình tại 1 thời điểm

Ví dụ: *$ vi tam.c*

$ ls

Shell Intro

❑ TIẾN TRÌNH

- **Tiến trình nền**

- Cho phép vận hành nhiều nhiều chương trình đồng thời
- Loại tiến trình vận hành lâu

  Ví dụ: $sleep 500&

  [1] PID 1234

- **Bậc tiến trình nền**

- Có nhiều bậc ưu tiên chi tiến trình nền
- **$nice** dùng để chạy 1 tiến trình nền bậc ưu tiên thấp hơn

  Ví dụ: *$nice sort afile > asorted&*
- **$nohup** cho phép chạy tiến trình nền liên tục thậm chí ngay cả khi sở hữu logout

  Ví dụ: *$nohup tam&*

Shell Intro

❑ TIẾN TRÌNH

- **Tiến trình nền**

- Ví dụ:

```
# Lệnh in manual shell
  $ man sh | lp& 2
        [1] 1557 3
  ......
  $ kill -9 $!
```

```
#!/bin/sh
for person in qtuan tiennd congls anhph
do
      mail Sperson < memo
done &
```

Shell Intro

❑ TIẾN TRÌNH

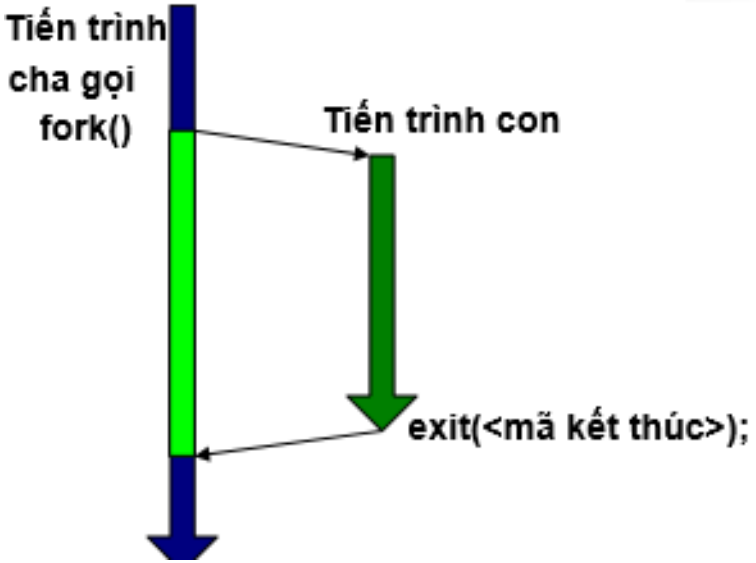
- **Tiến trình cha / con**

- Tiến trình shell thông thường
- Về bản chất, một tiến trình được sinh ra khi có một tiến trình gọi tới hàm fork()

Tiến trình được sinh ra là tiến trình con, tiến trình gọi đến fork() là tiến trình cha (PPID)

Tiến trình đầu tiên trong hệ thống là init có PID=1

init sinh ra các tiến trình khác trong hệ thống



Shell Intro

❑ BÁO HIỆU TIẾN TRÌNH

- **Tiến trình liên lạc với nhau**

- Các tiến trình được xem là hoạt động "song song" với nhau
- Nếu nhiều tiến trình hợp tác giải quyết cùng một bài toán, cần có các cơ chế liên lạc để trao đổi thông tin
- Một trong các cơ chế đó là báo hiệu (signal)

signal là cơ chế cho phép các tiến trình thông báo cho nhau về sự xuất các yếu tố không được xác định trước

Shell Intro

❑ HỦY BỎ TIẾN TRÌNH

- Một tiến trình có thể được kết thúc bằng cách sử dụng tổ hợp phím như
  Ctrl-C hoặc Ctrl-\
  Sử dụng lệnh kill của shell
- Lệnh này sử dụng để loại bỏ các ứng dụng nền hoặc nếu thiết bị đầu cuối của bạn ở trạng thái đóng băng, không phản hồi- loại bỏ tiến trình đang gây ra sự cố.
- Khuôn dạng lênh

$kill [-signal] PID [-<số hiệu của signal>] <PID>

Ví dụ

```
$ sleep 60&
$ ps
   PID    TTY    TIME   CMD
 27628   pts/7  0:00    sleep
 27619   pts/7   0:00    bash
 27629    pts/7  0:00    ps 3
```

```
$ kill 27628
$ ps
   PID    TTY    TIME    CMD
 27631   pts/7  0:00     ps
 27619    pts/7 0:00     bash
   [1]+ Terminated sleep 60
```

Shell Intro

❑ SIGNALS

- Một báo hiệu gửi thông báo đến một tiến trình khiến tiến trình kết thúc do:
  + Sự kiện không mong muốn như treo máy, lỗi bus hoặc mất điện hoặc do lỗi chương trình chẳng hạn như chia cho số 0 hoặc tham chiếu bộ nhớ không hợp lệ .
  + Bằng cách nhấn các chuỗi phím nhất định. Ví dụ: nhấn các phím Break, Delete, Quit hoặc Stop
- Mỗi tiến trình có đáp ứng với báo hiệu cho trước như
  + Báo hiệu bị tảng lờ
  + Tiến trình có thể dừng lại
  + Tiến trình có thể vẫn tiếp tục
  + Báo hiệu lấy một hàm được định nghĩa trong chương trình

Shell Intro

❑ SIGNALS

Số	Tên	Miêu tả	Hoạt động do
0	EXIT	Thoát khỏi Shell	Termination
1	SICHUP	Ngắt kết nối đầu cuối	Termination
2	SIGINT	Người dùng nhấn Ctrl-C	Termination
3	SIGQUIT	Người dùng nhấn Ctrl\	Termination
4	SIGILL	Phần cứng bất hợp pháp	Lỗi chương trình
5	SICTRAP	Tạo ra bởi trình gỡ lỗi	Lỗi chương trình
8	SIGFPE;	Lỗi ví dụ: chia cho 0	Lỗi chương trình
9	SIGKILL	Bỏ qua	Termination
10	SIGUSR1	Báo hiệu do ứng dụng xác định	
11	SIGSECV	Tham chiếu bộ nhớ không hợp lệ	Lỗi chương trình
12	SICUSR2	Báo hiệu do ứng dụng xác định	

Shell Intro

❑ SIGNALS

Số	Tên	Miêu tả	Hoạt động
13	SIGPIPE	Kết nối đường ống bị hỏng	Lỗi người vận hành
14	SICALRM	Quá hạn	Alarm gửi
15	SIGTERM	Chấm dứt chương trình	Termination
17	SIGCHLD	Tiến trình con đã dừng hoặc chết	Bỏ qua
18	SIGCONT	Bắt đầu công việc đã dừng;	Tiếp tục nếu bị dừng lại
19	SIGSTOP	Dừng một công việc;	Dừng tiến trình
20	SIGSTP	Trạm dừng tương tác; nhấn Ctrl-Z	Dừng tiến trình
21	SIGII1N	Background đang cố gắng đọc kiểm soát thiết bị đầu cuối	Dừng tiến trình
21	SIGTTOU	Background đang cố gắng ghi kiểm soát thiết bị đầu cuố	Dừng tiến trình

Shell Intro

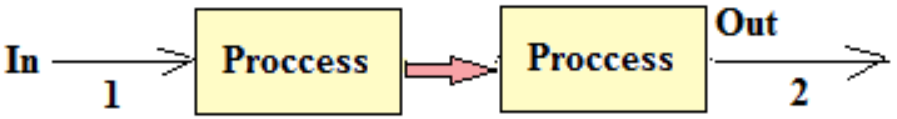
❑ TIẾN TRÌNH SHELL

- **Đường ống tiến trình**

- Đường ống là hình thức *giao tiếp liên tiến trình* - cho phép các tiến trình giao tiếp với nhau. Đó là một cơ chế đầu ra của một lệnh được gửi làm đầu vào cho một lệnh khác, với hạn chế là dữ liệu chỉ có thể chảy theo một hướng, thường là giữa cha và con.
- Shell thực hiện đường ống bằng cách đóng và mở các bộ mô tả tệp; tuy nhiên, thay vì gán các bộ mô tả cho một tệp, nó sẽ gán chúng cho một bộ mô tả ống được tạo bằng lệnh gọi hệ thống ống. Sau khi cha tạo các bộ mô tả tệp đường ống, nó sẽ tạo một tiến trình con cho mỗi lệnh trong đường ống. Do vậy một tiến trình sẽ ghi vào đường ống và người kia sẽ đọc từ đó.
- Để đơn giản hóa mọi thứ, đường ống chỉ là một bộ đệm kernel mà từ đó cả hai tiến trình có thể chia sẻ dữ liệu, do đó loại bỏ cần thiết các file tạm thời trung gian. Kernel đồng bộ hóa các hoạt động để một tiến trình đợi trong khi tiến trình kia đọc hoặc ghi vào bộ đệm

Shell Intro

❑ TIẾN TRÌNH SHELL



- Cú pháp lệnh đường ống

  $ who I wc

- Để thực hiện lệnh đường ống trên tương đương 3 lênh

  $ who > tmp

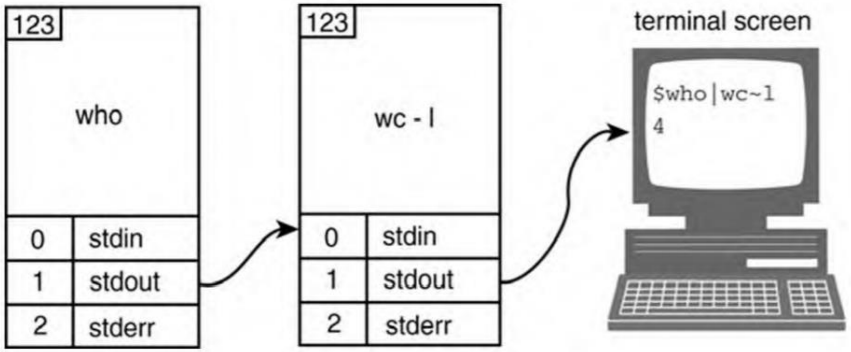
  $ wc –l tmp

  4 tmp

  $ rm tmp

--> *$ who | wc -l*

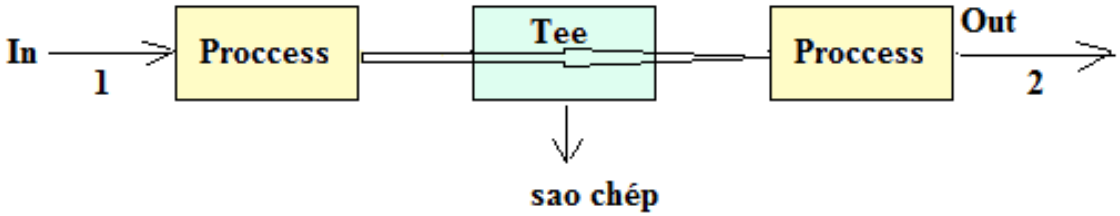
4



Shell Intro

❑ Tiến trình shell

- Tách thông tin đường ống
  - ***$ Tee***
  - Lệnh tee sao chép lối ra chuẩn đưa ra đường ống tới lối vào chuẩn sinh ra file
  - Khuôn dạng : S command | tee file | command



Ví dụ :

$ date | tee myfile

S ls | tee file1 | sort

Shell Intro

❑ Tiến trình shell

- Cấu trúc cây tiến trình

$***pstree [options]***

Kết quả hiển thị cây gốc là tiến trình init (PID=1)

```
pstree
init---4*[getty]
init-+-atd
     |-bash---startx---xinit-+-X
     |                       `-fvwm2-+-FvwmButtons
     |                               |-FvwmPager
     |-cardmgr                       `-FvwmTaskBar
     |-crond
     |-gpm
     |-httpd---10*[httpd]
     |-ifup-ppp---pppd---chat
     |-inetd
     |-kerneld
     |-kflushd
     |-klogd
     |-kswapd
     |-lpd
     |-2*[md_thread]
     |-5*[mingetty]
     |-nmbd
     |-nxterm---bash---tcsh---pstree
     |-portmap
     |-sendmail
     |-smbd
     |-syslogd
     |-update
     |-xclock
     `-xload
```

Thanks