[544] Linux Pipelines

Tyler Caraza-Harter

Learning Objectives

- chain multiple Linux programs together into a pipeline
- redirect process output to a file
- observe resource consumption on Linux

Unix Philosophy

1. "Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new 'features'."
2. "Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input."

Supplemental Reading:

Designing Data Intensive Applications ("Batch Processing with Unix Tools" of Chapter 10)

The Pipe

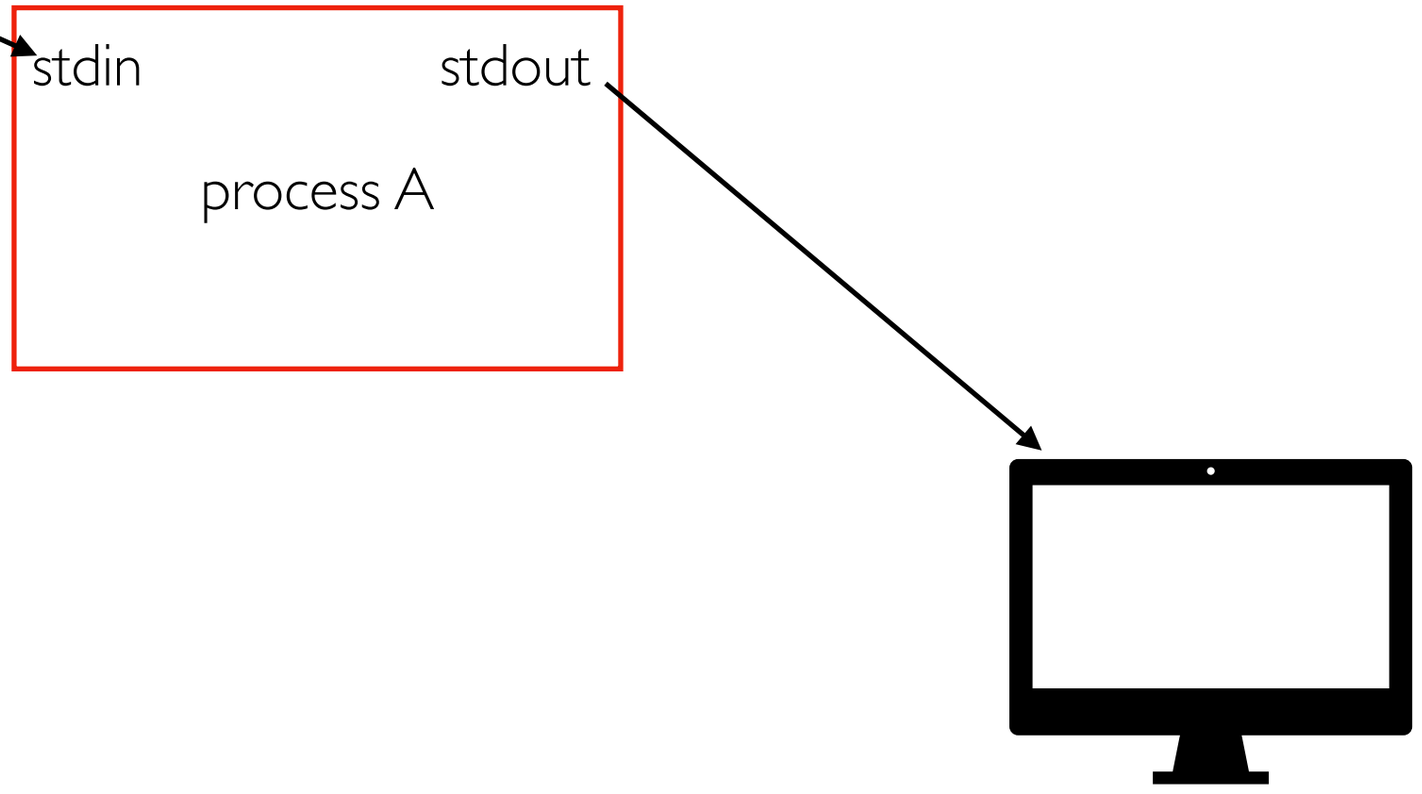
Simple Log Analysis

Various tools can take these log files and produce pretty reports about your web-site traffic, but for the sake of exercise, let's build our own, using basic Unix tools. For example, say you want to find the five most popular pages on your website. You can do this in a Unix shell as follows:[i]

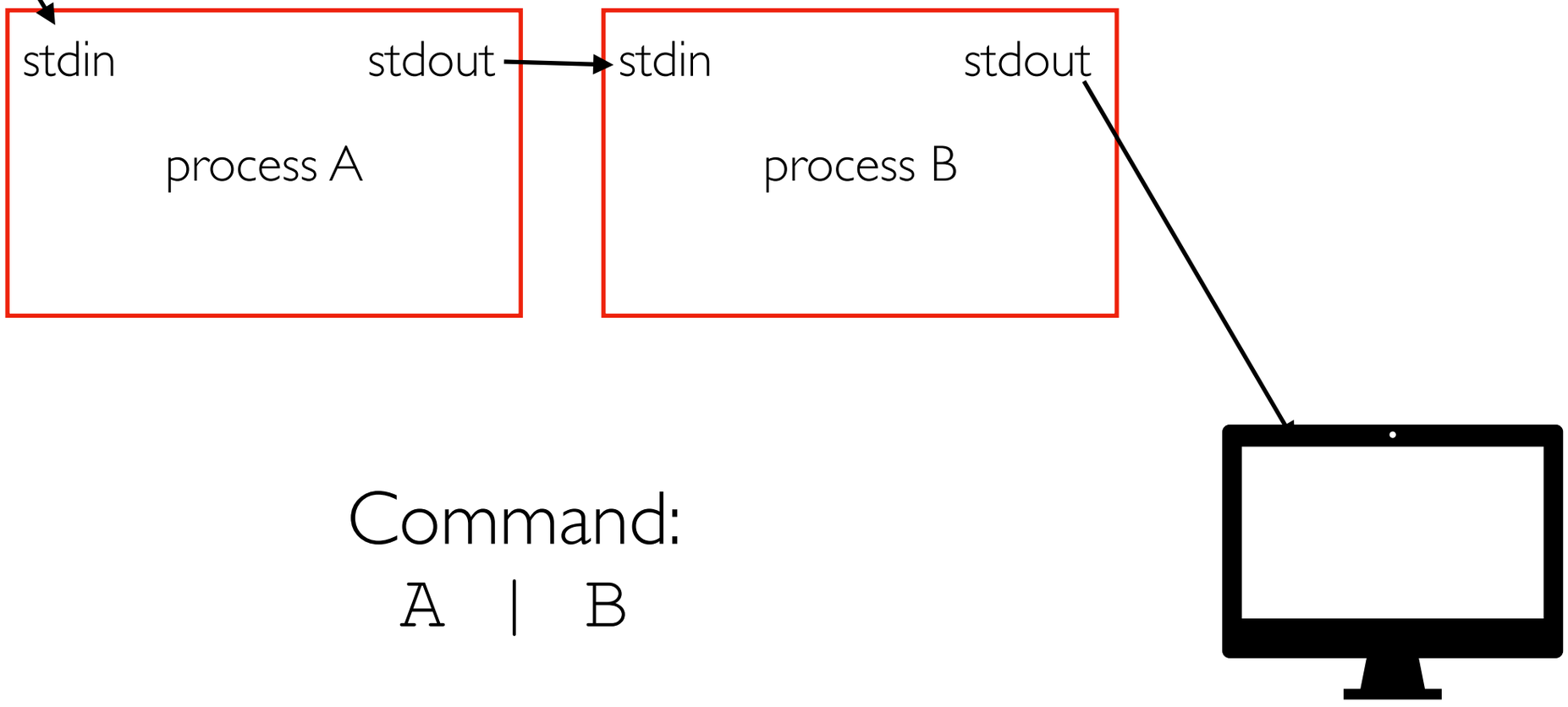
```
cat /var/log/nginx/access.log |  ❶
  awk '{print $7}' |             ❷
  sort             |             ❸
  uniq -c          |             ❹
  sort -r -n       |             ❺
  head -n 5                      ❻
```

the pipe connects output of one process to input of the next

Standand Input and Output (I/O)



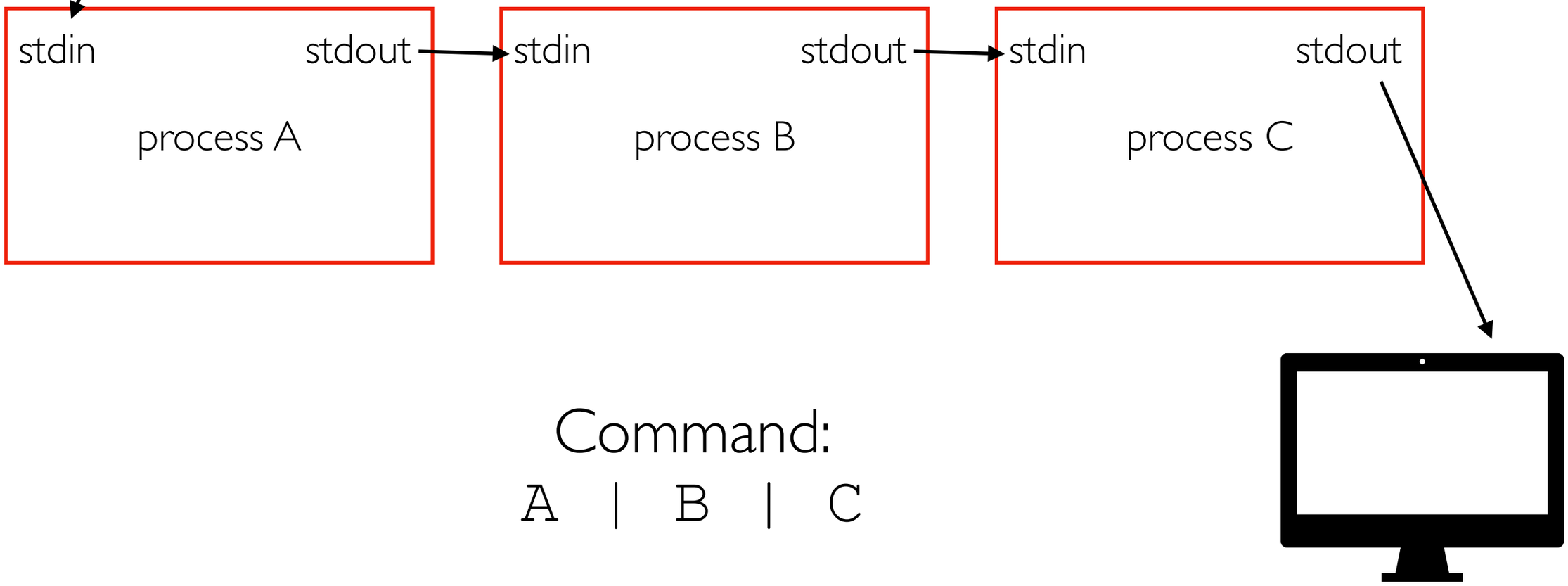
stdout => stdin



Command:

```
A | B
```

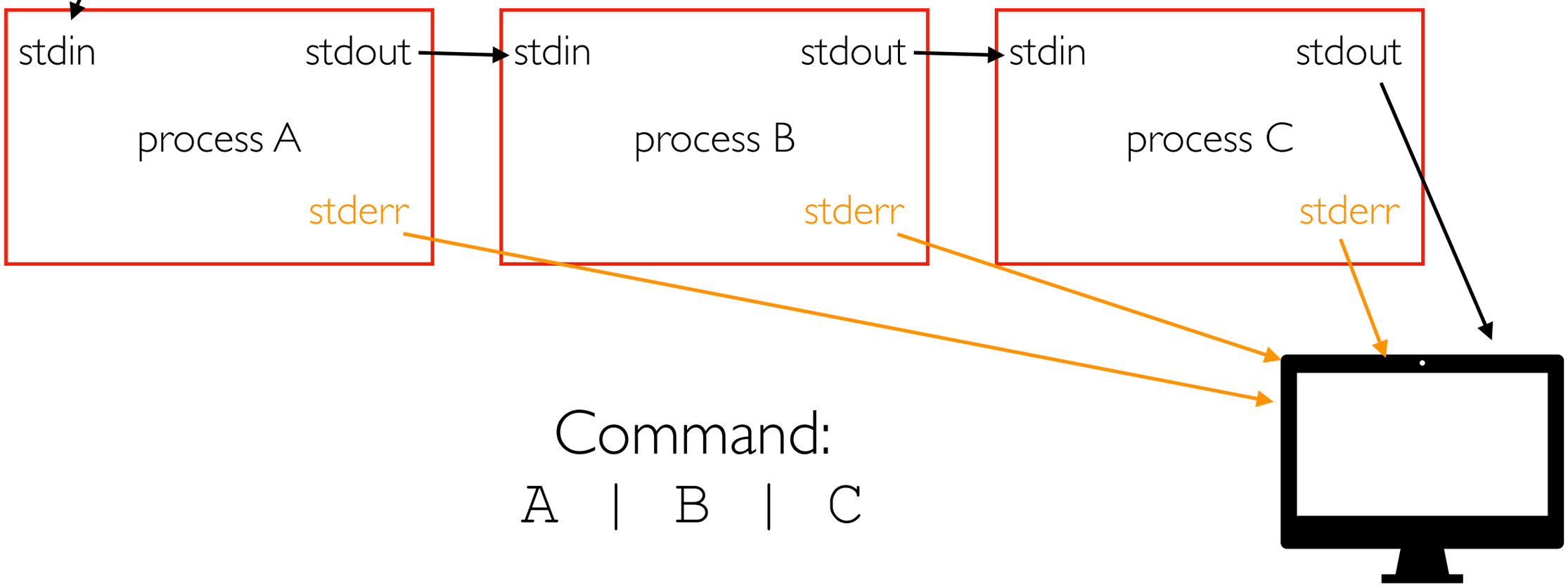
Chains can be long



Command:

```
A | B | C
```

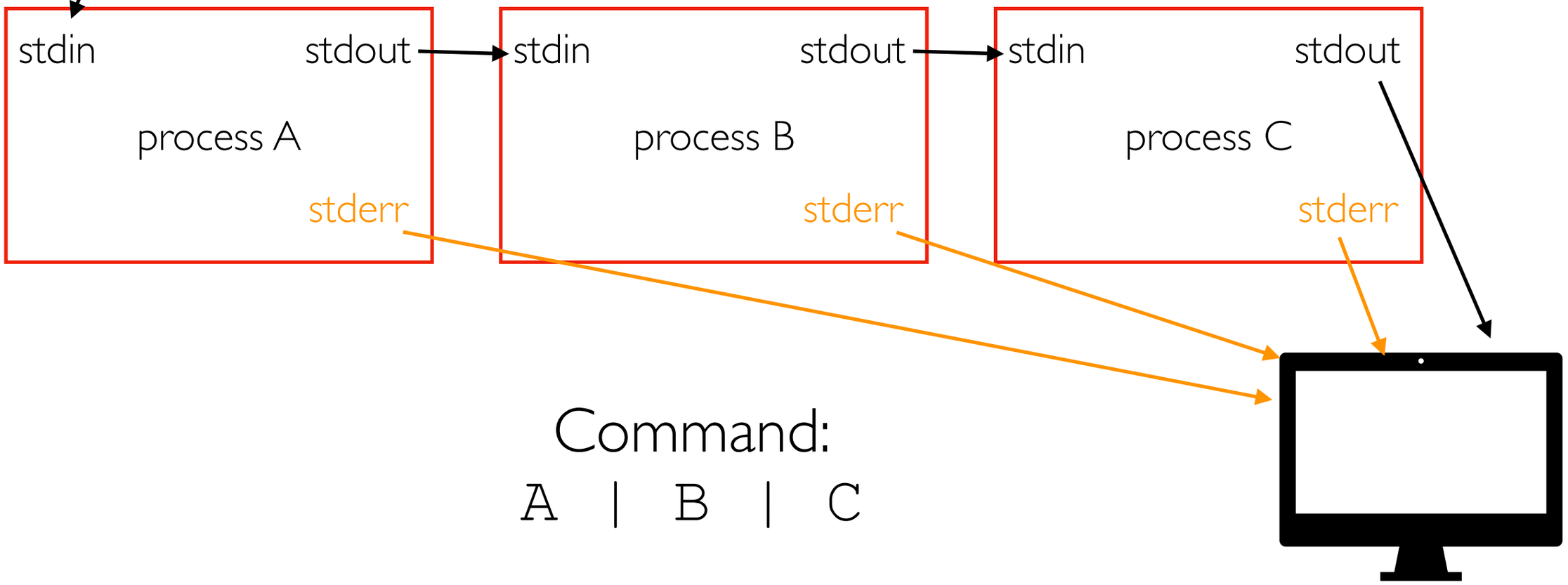
stderr (for things like warnings that shouldn't be chained)



Command:

```
A | B | C
```

stderr (for things like warnings that shouldn't be chained)

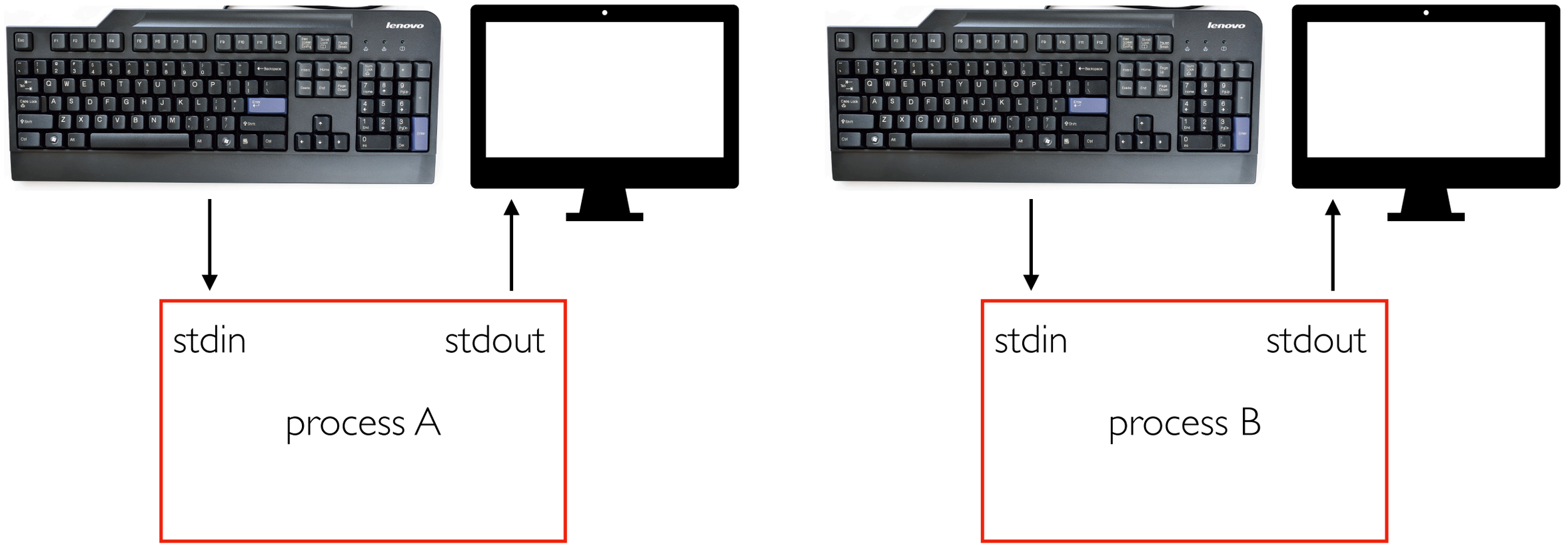


Command:

```
A | B | C
```

What if you want to pass intermediate data through files?

Logical AND: run sequentially



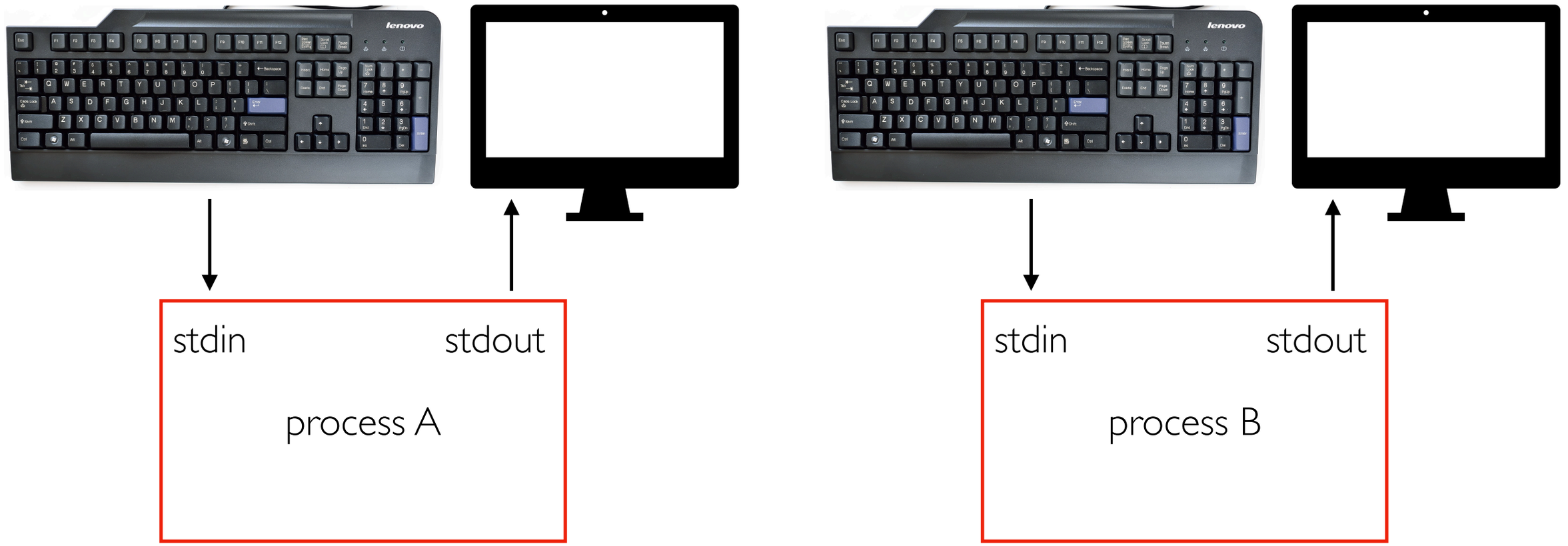
first

second
(if first succeeds)

Command:

```
A && B
```

Logical AND: run sequentially



first

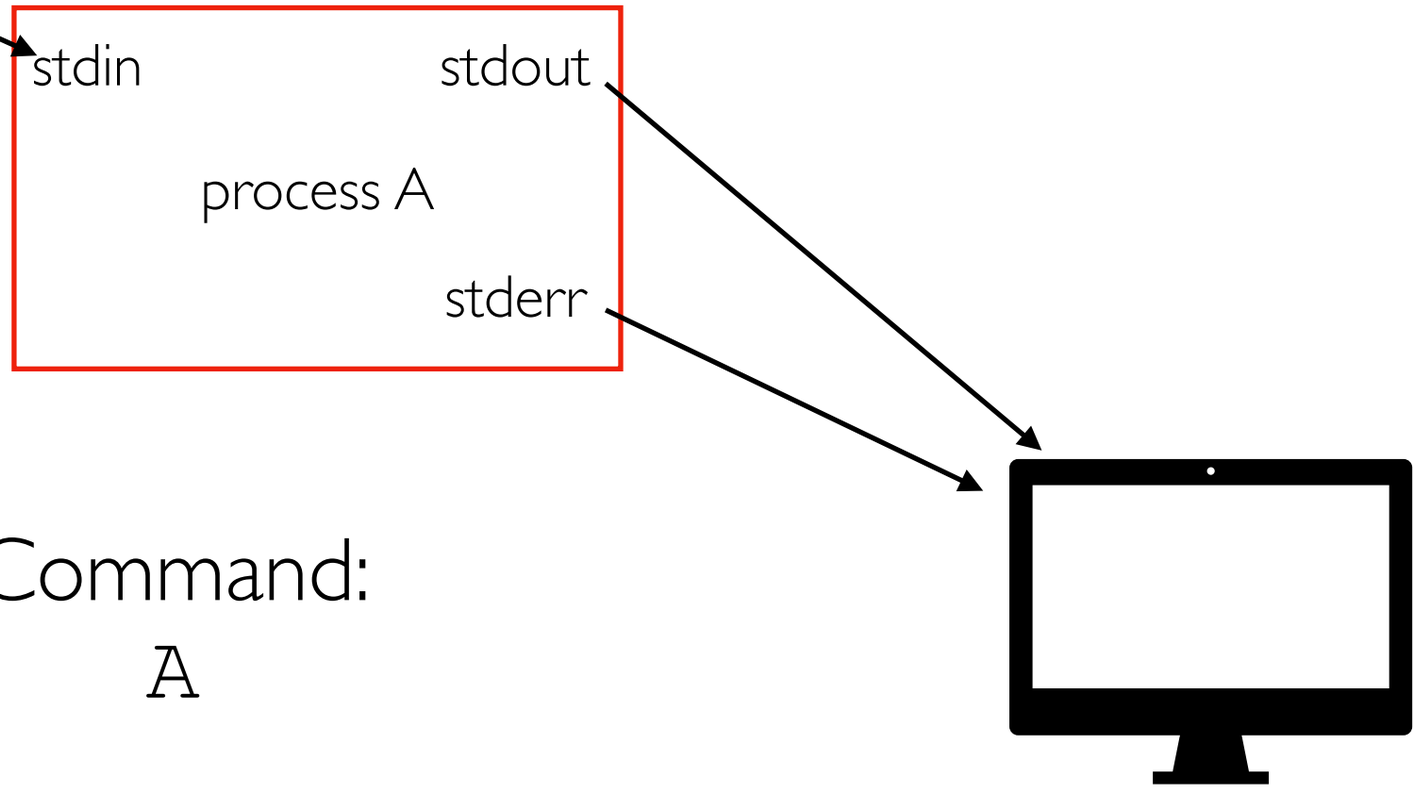
second
(if first succeeds)

Command:

`A && B`

Example: wget http://URL/stations.txt && cat stations.txt && rm stations.txt

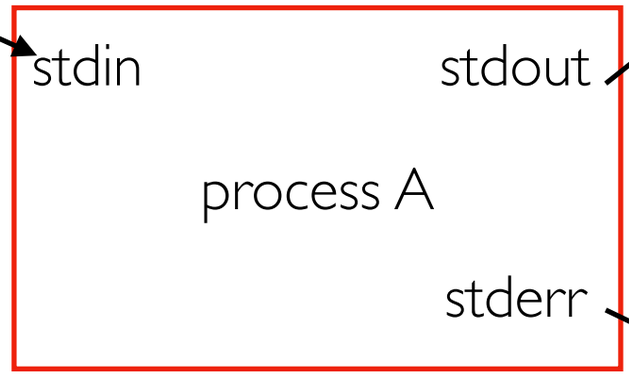
Redirection



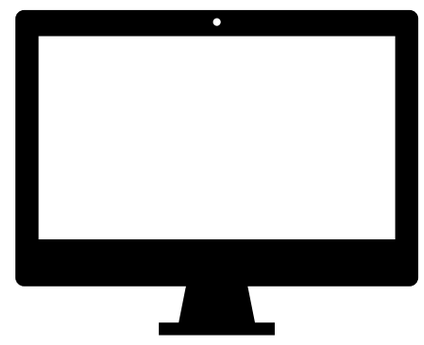
Command:

`A`

Redirection



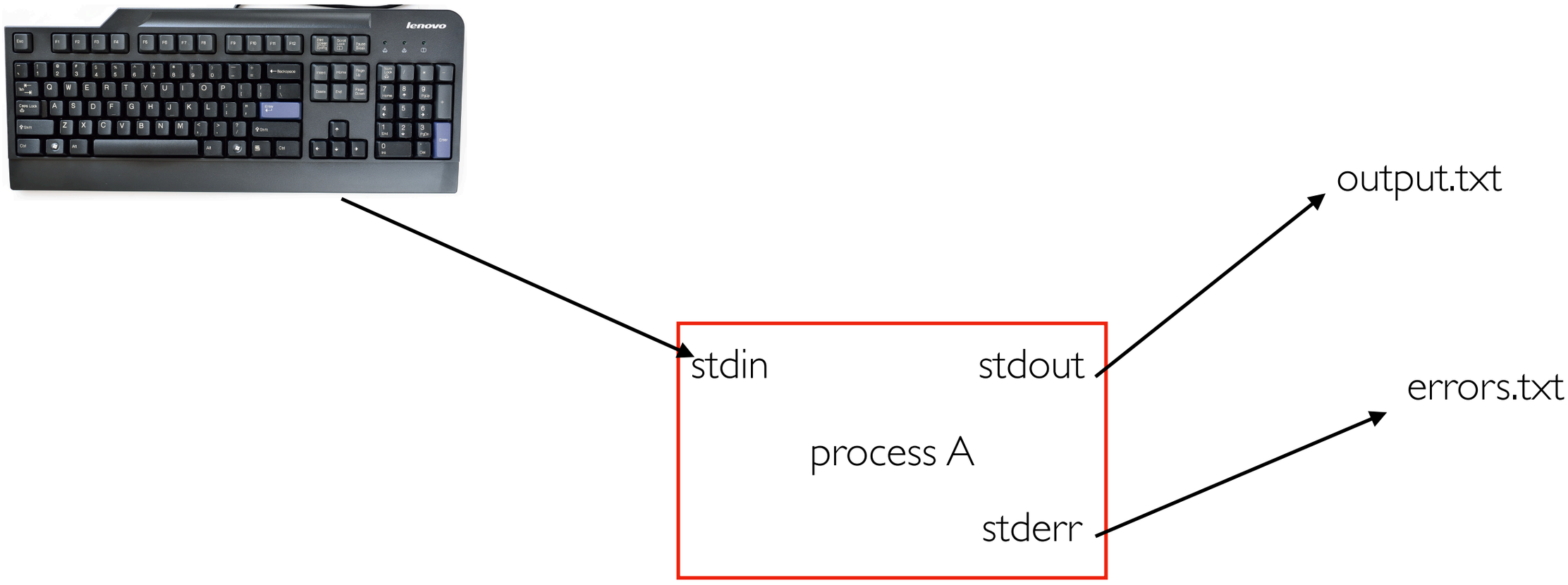
output.txt



Command:

```
A > output.txt
```

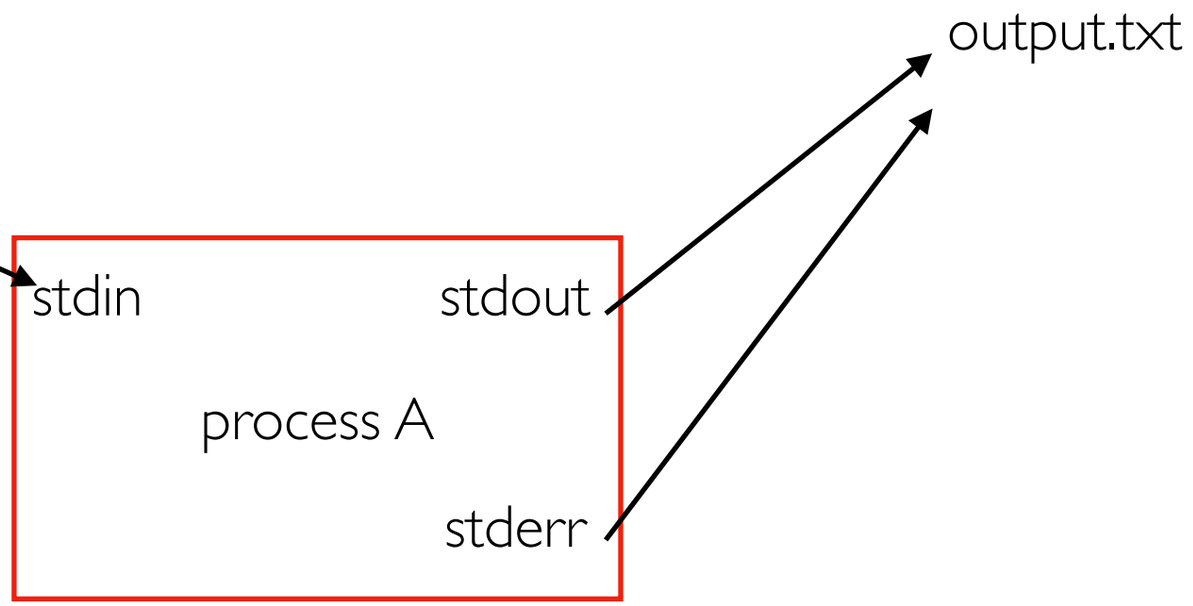
Redirection



Command:

```
A > output.txt 2> errors.txt
```

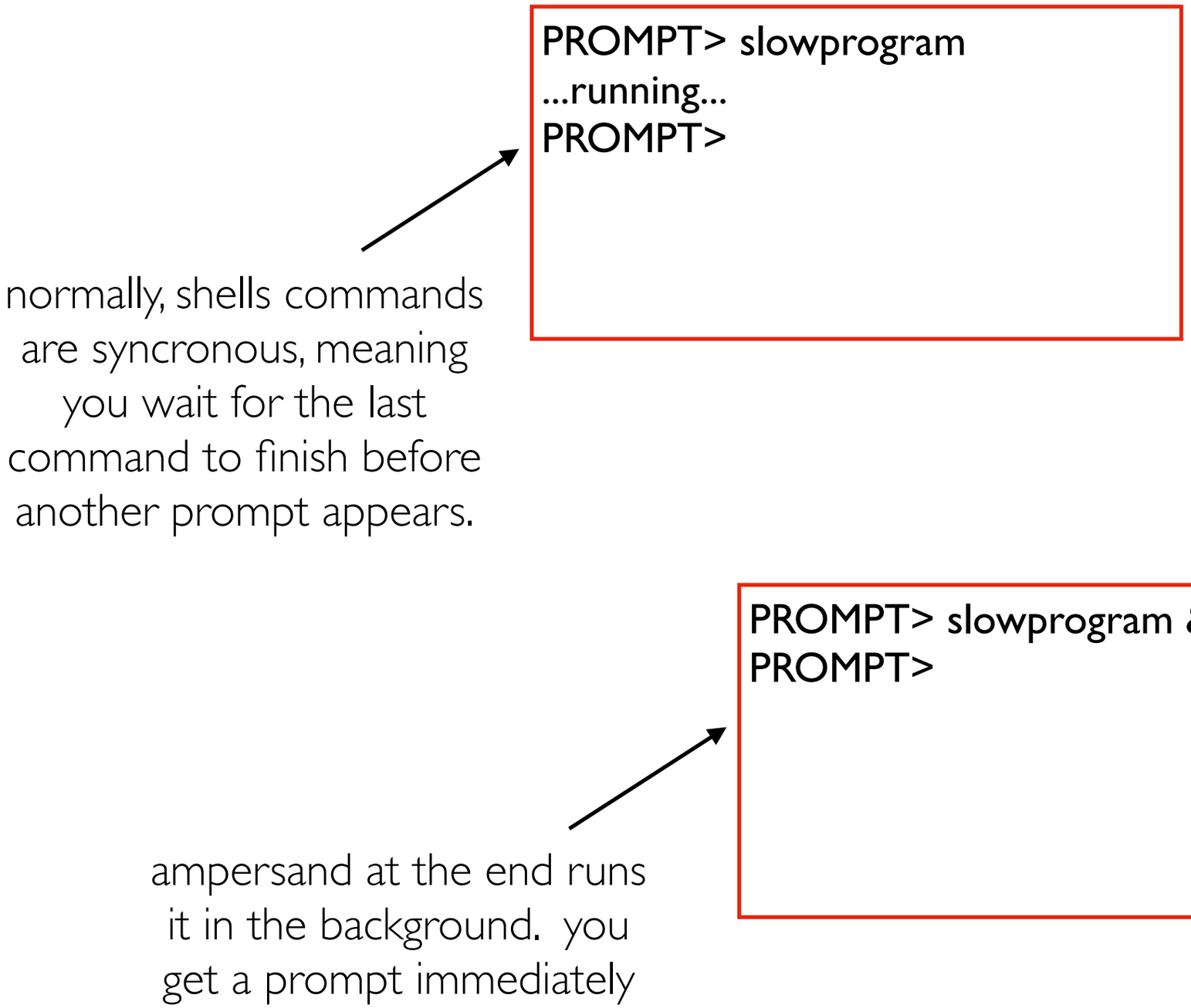
Redirection



Command:

```
A &> output.txt
```

Async


PROMPT> slowprogram
...running...
PROMPT>
normally, shells commands are syncronous, meaning you wait for the last command to finish before another prompt appears.
PROMPT> slowprogram &
PROMPT>
ampersand at the end runs it in the background. you get a prompt immediately

All together

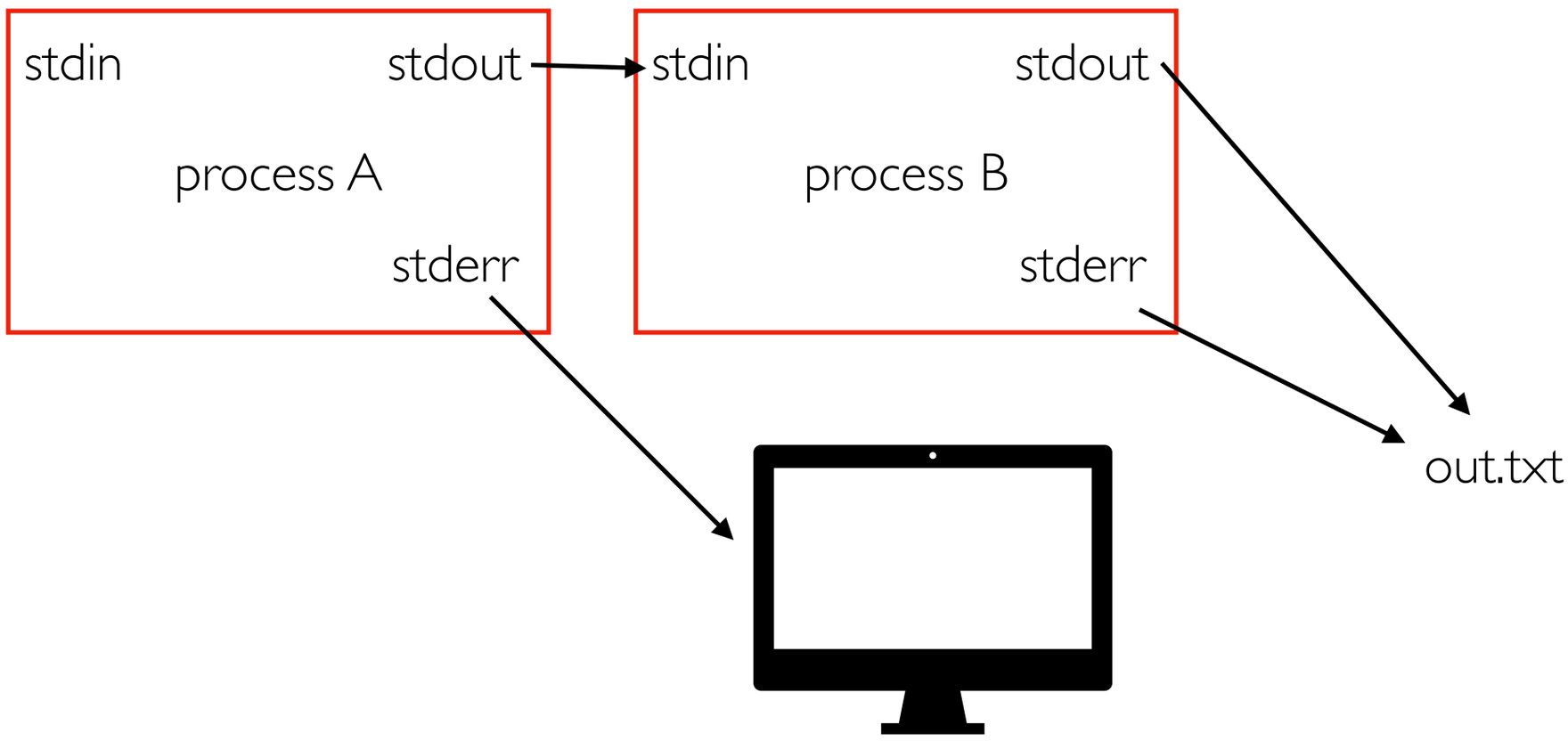
Command:

```
A | B &> out.txt &
```

All together

Command:

```
A | B &> out.txt &
```



This pipeline will run in the background (perhaps for a long time), and we won't see the output. BUT we can find it later in the out.txt file.

TopHat + Demos...