[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

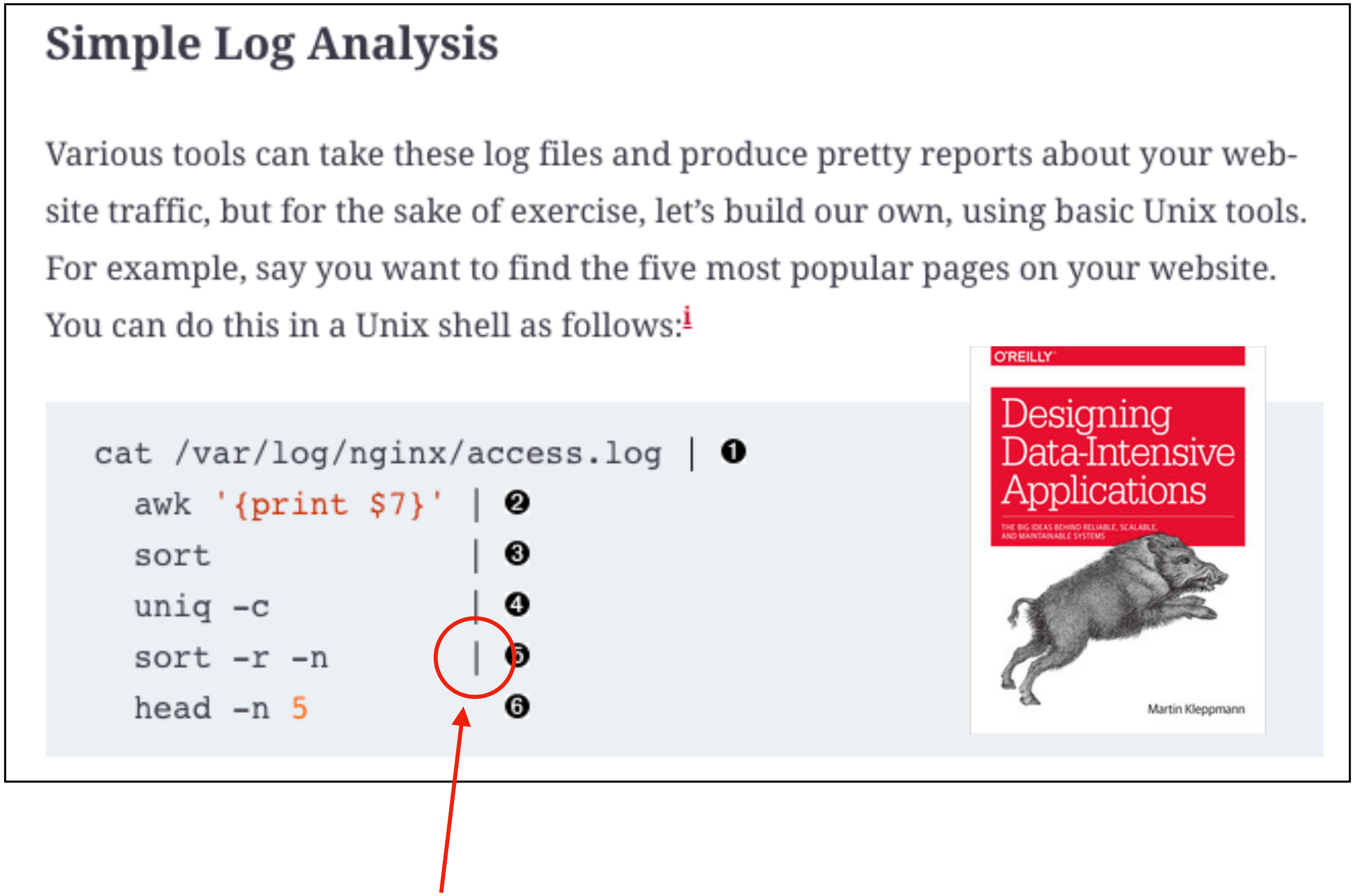
1. "Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new 'features'."
2. "Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar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

O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

Supplemental Reading:

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

The Pipe



Simple Log Analysis

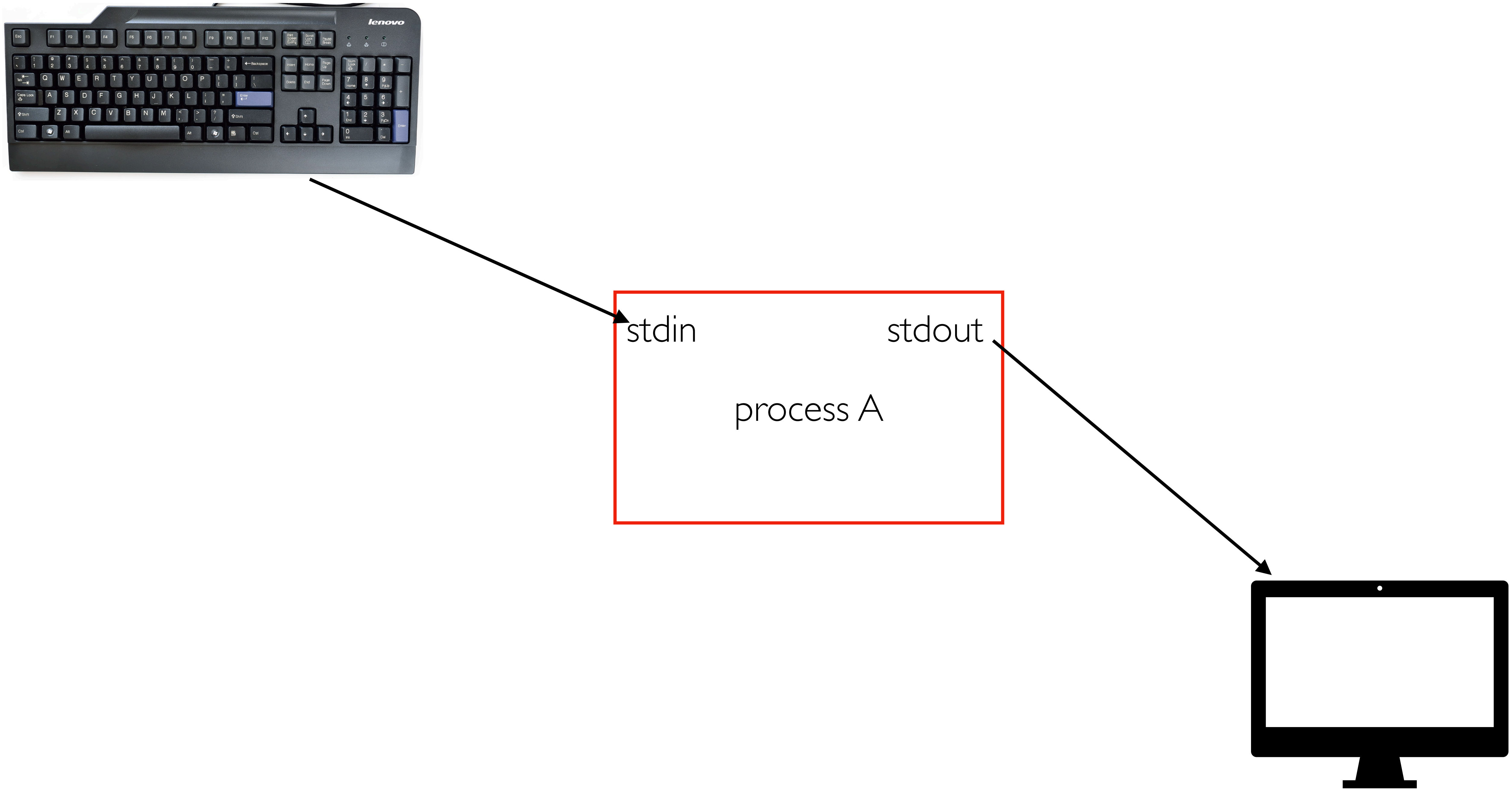
Various tools can take these log files and produce pretty reports about your website traffic, but for the sake of exercise, let's build our own, using basic Unix tools. For example, say you want to find the five most popular pages on your website. You can do this in a Unix shell as follows:[i]

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

O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

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

Standand Input and Output (I/O)

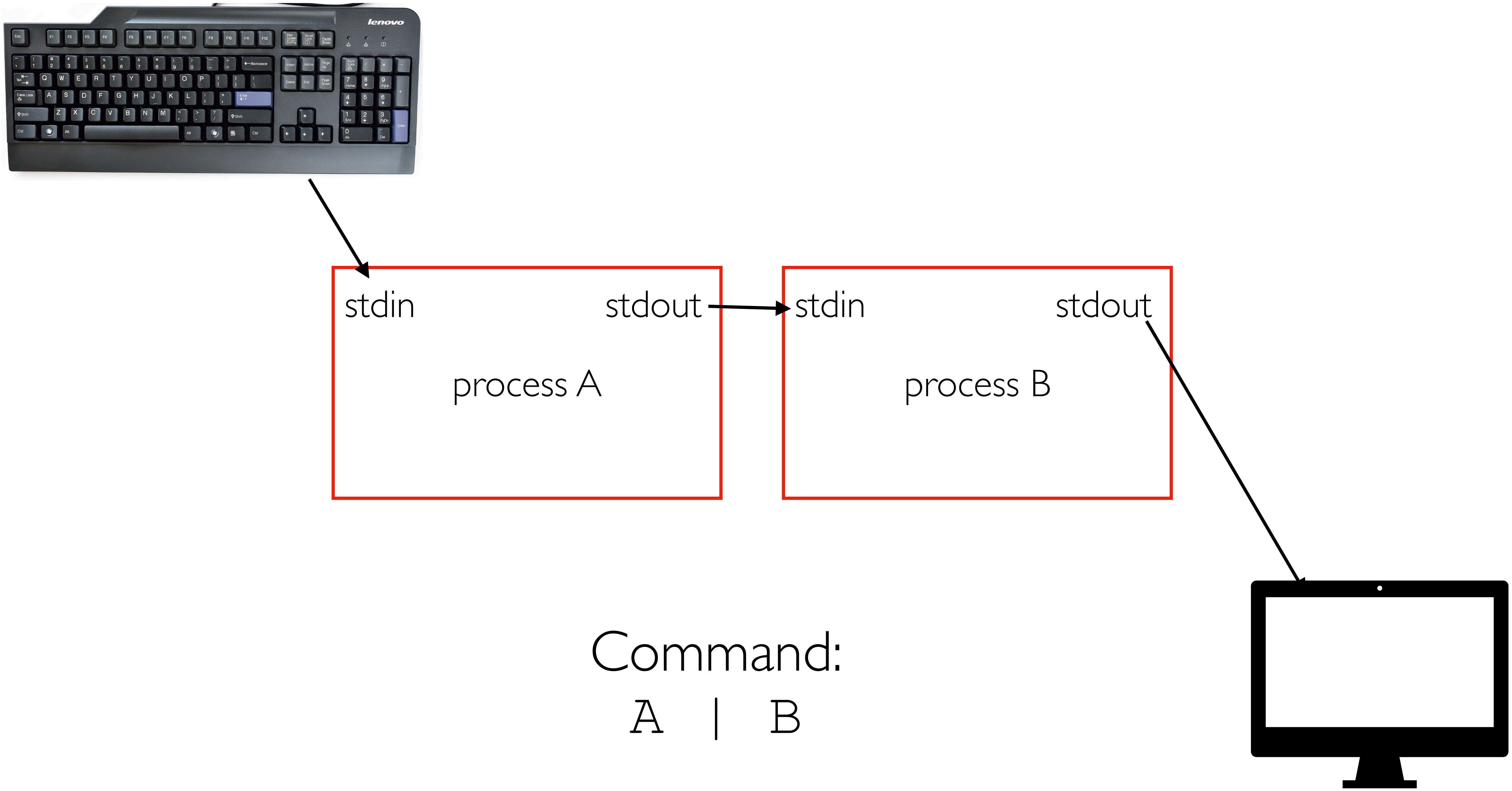


stdin

stdout

process A

stdout => stdin



stdin

stdout

process A

stdin

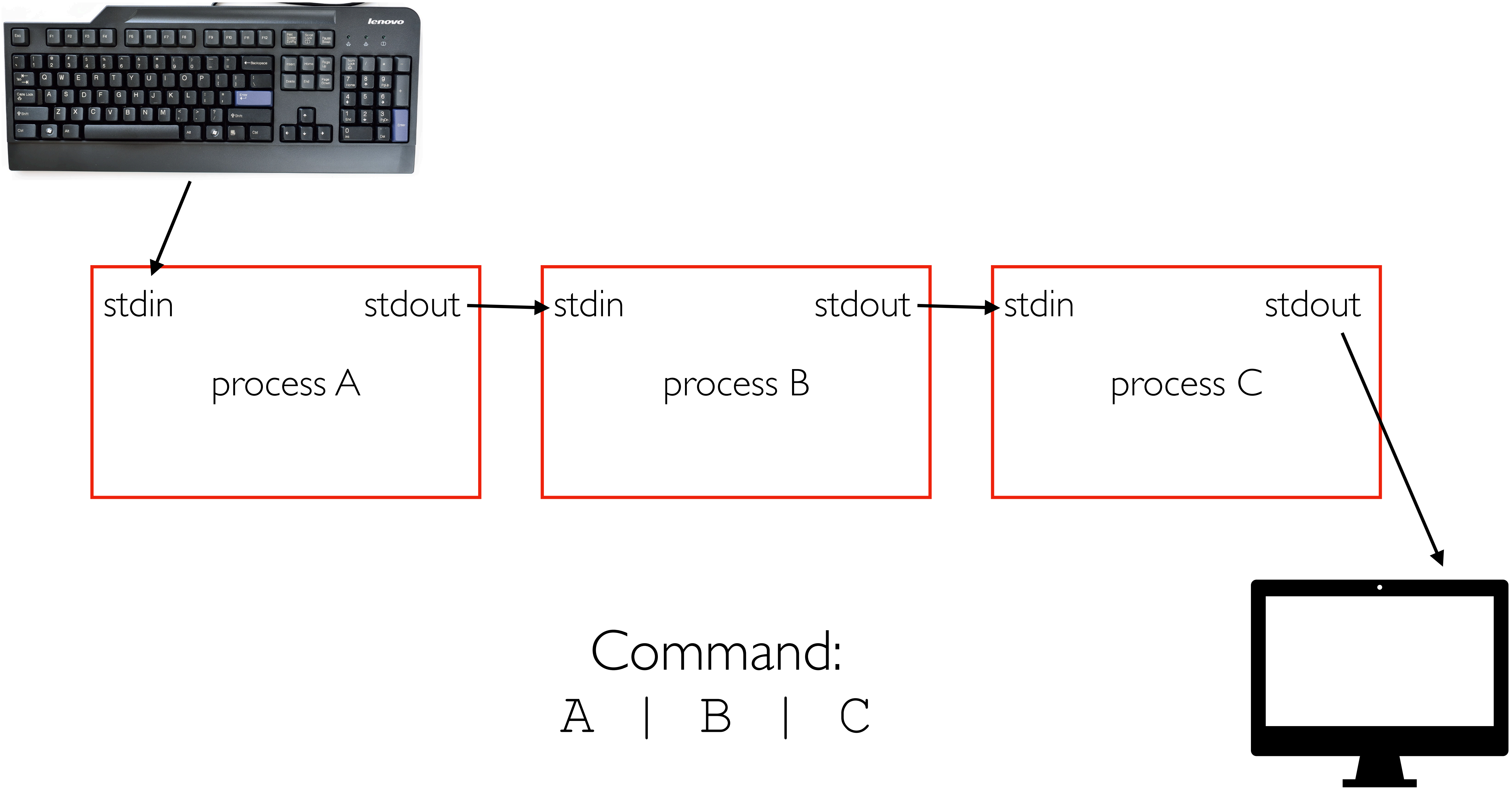
stdout

process B

Command:

```
A | B
```

Chains can be long



stdin stdout

process A

stdin stdout

process B

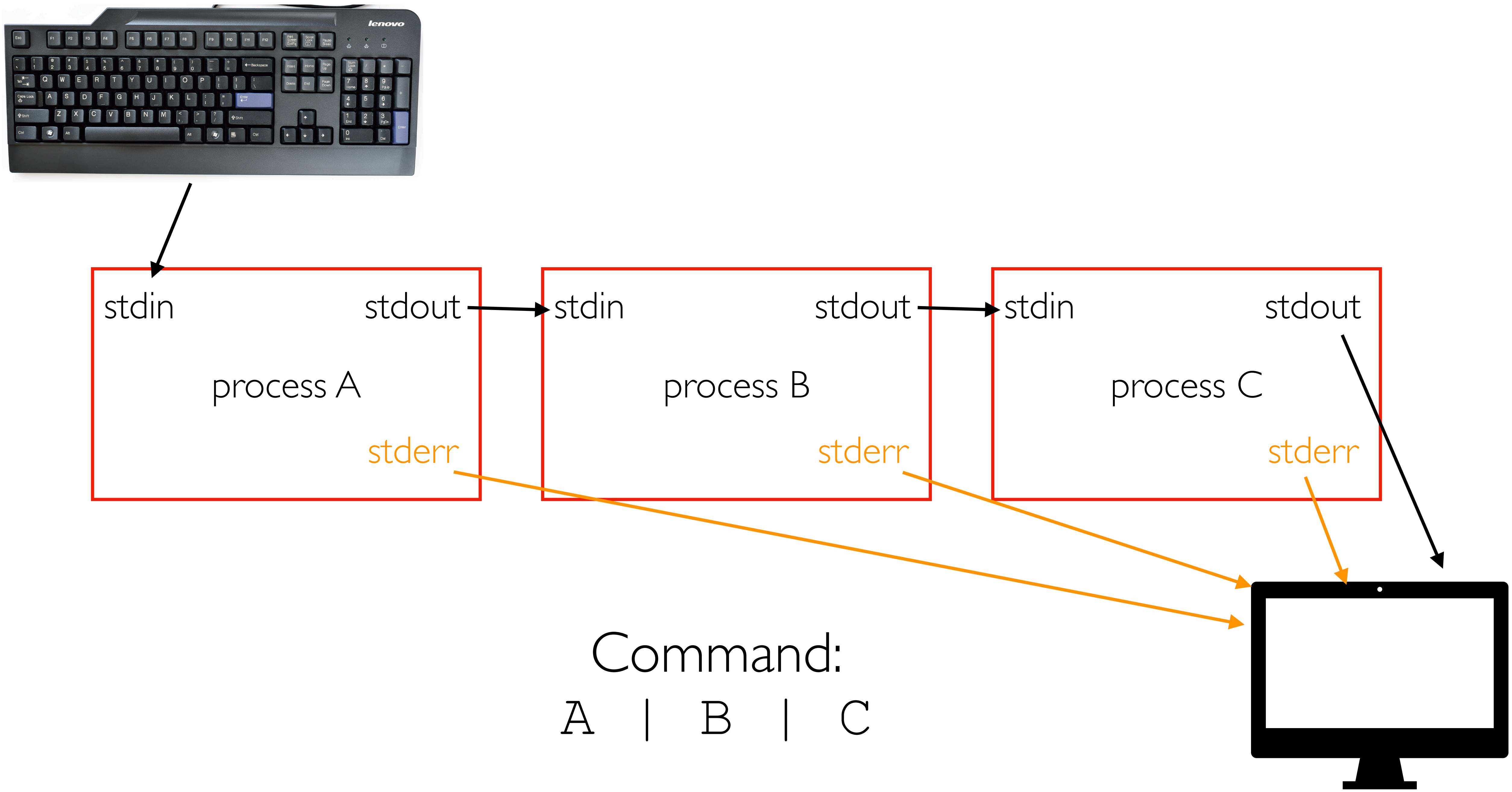
stdin stdout

process C

Command:

```
A | B | C
```

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



stdin stdout
process A
stderr

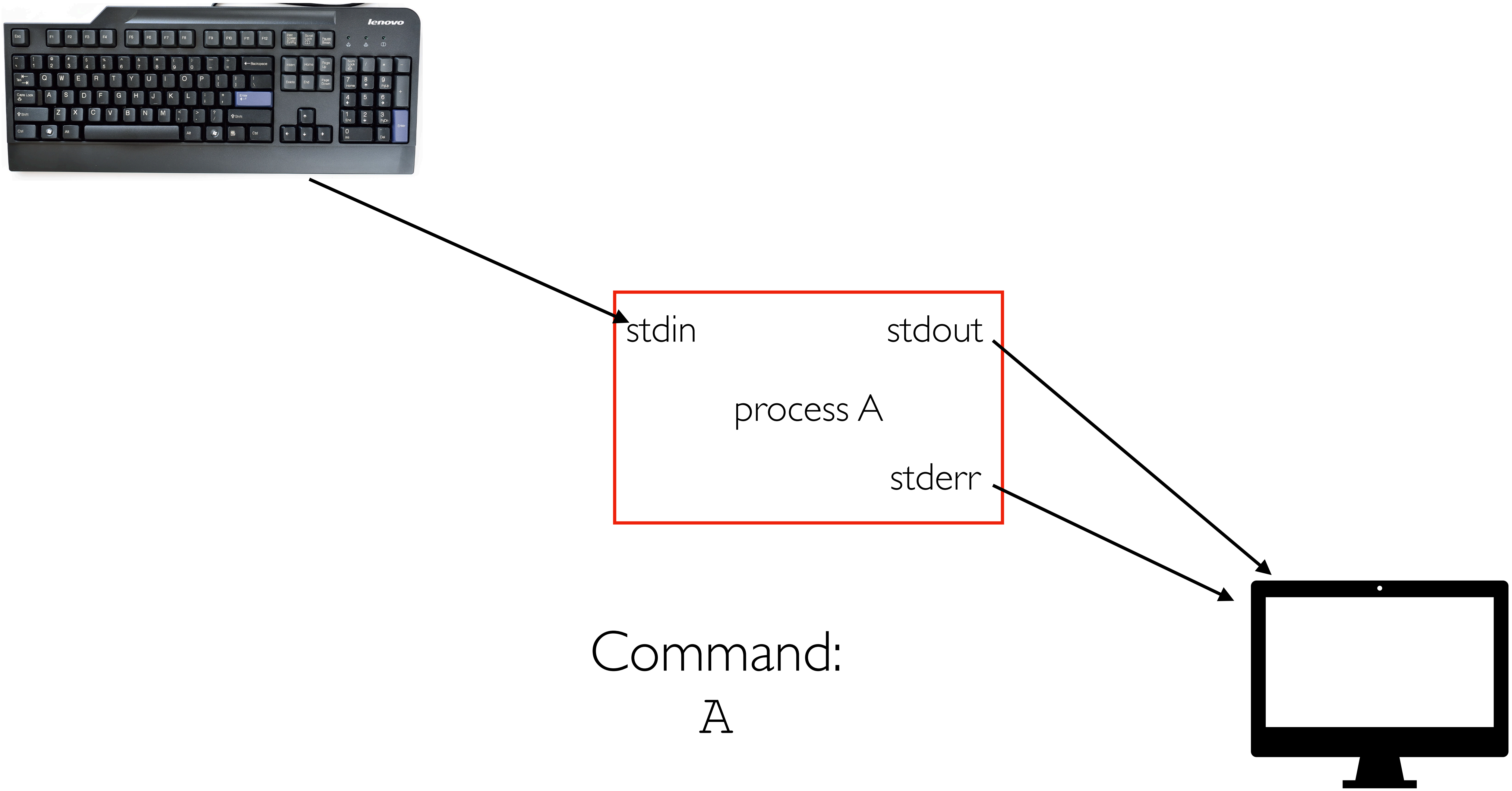
stdin stdout
process B
stderr

stdin stdout
process C
stderr

Command:

```
A | B | C
```

Redirection



stdin

stdout

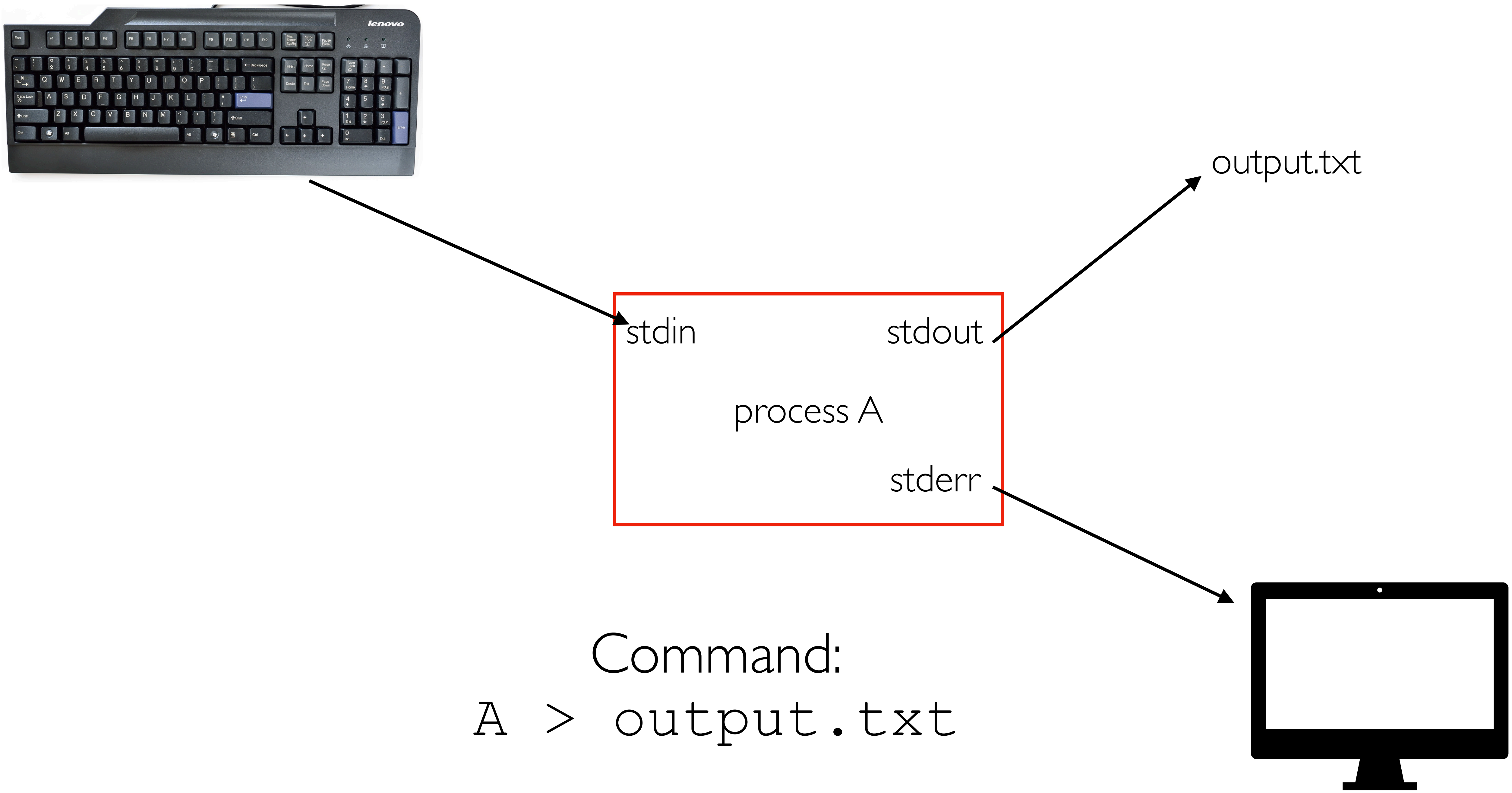
process A

stderr

Command:

```
A
```

Redirection



output.txt

stdin

stdout

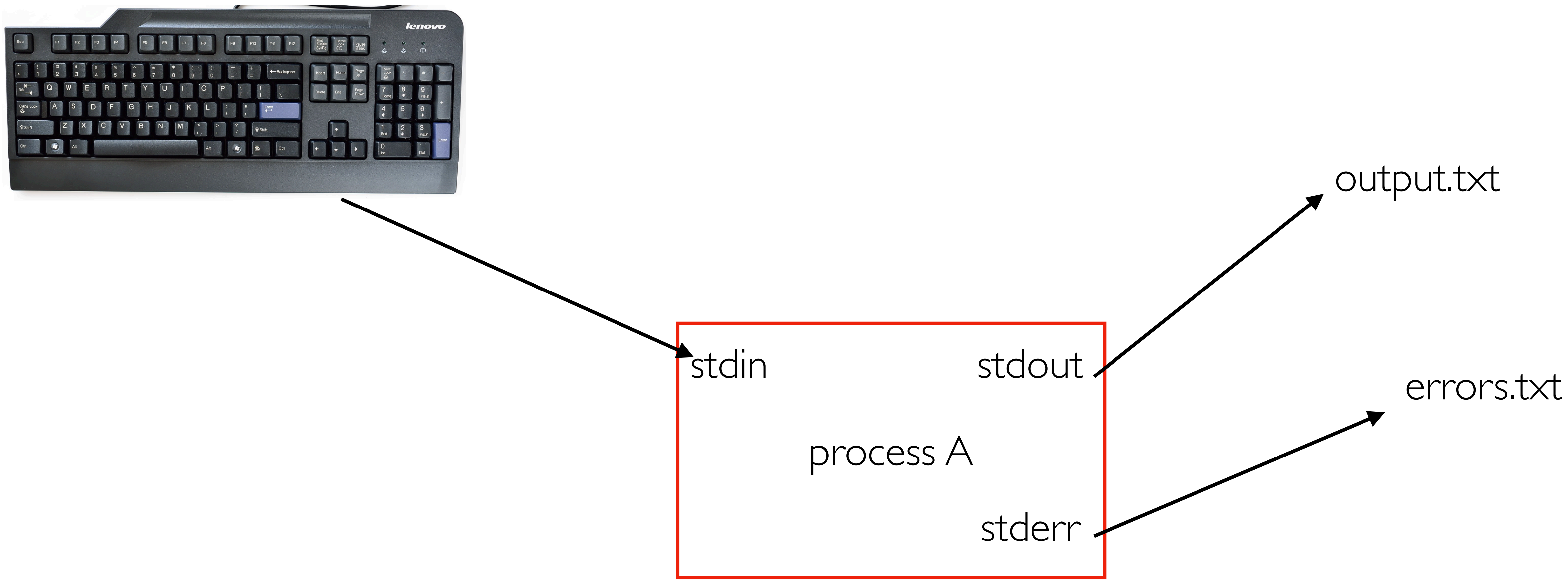
process A

stderr

Command:

```
A > output.txt
```

Redirection



stdin

stdout

process A

stderr

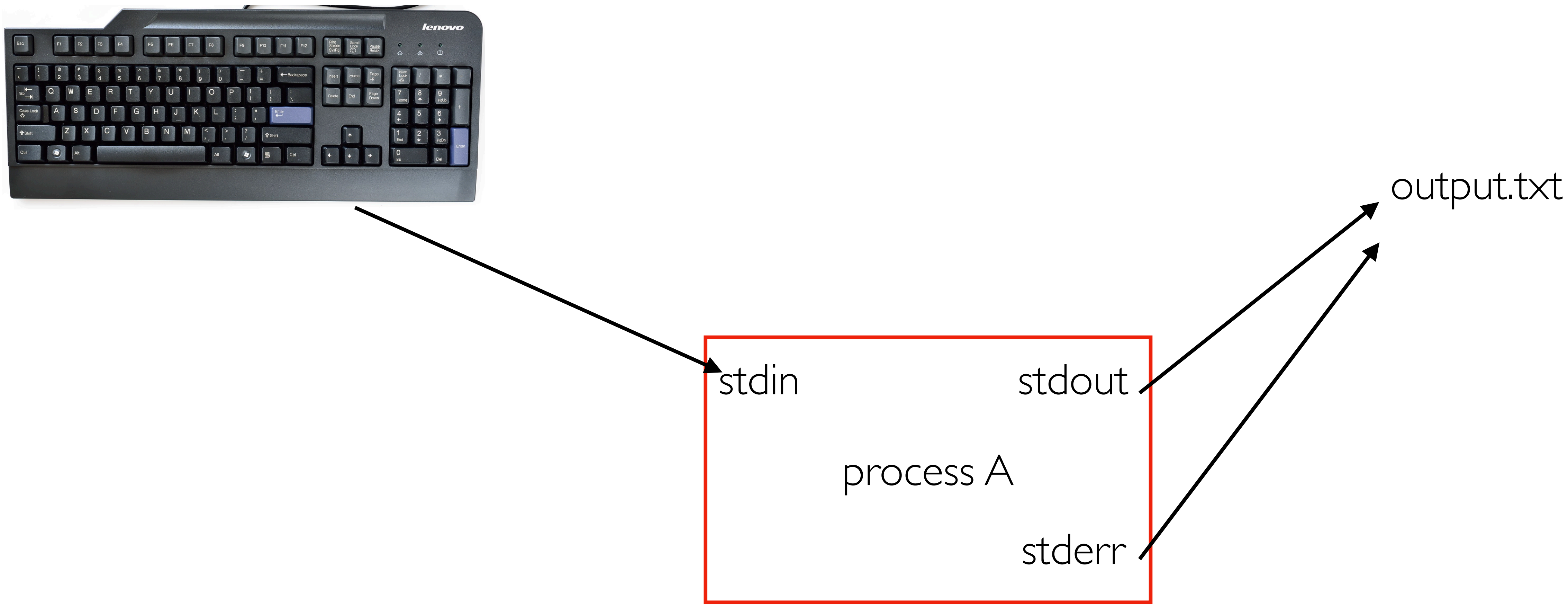
output.txt

errors.txt

Command:

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

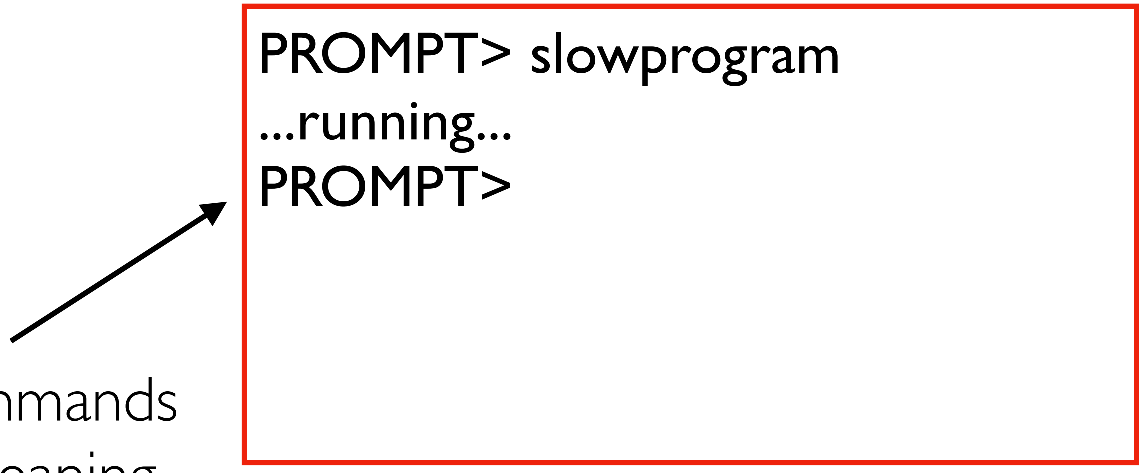
Redirection



Command:

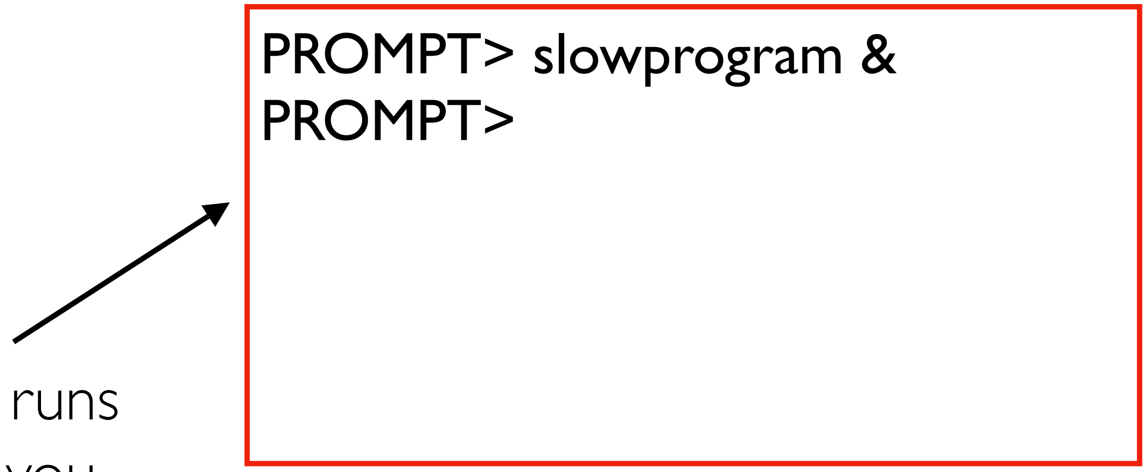
```
A &> output.txt
```

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

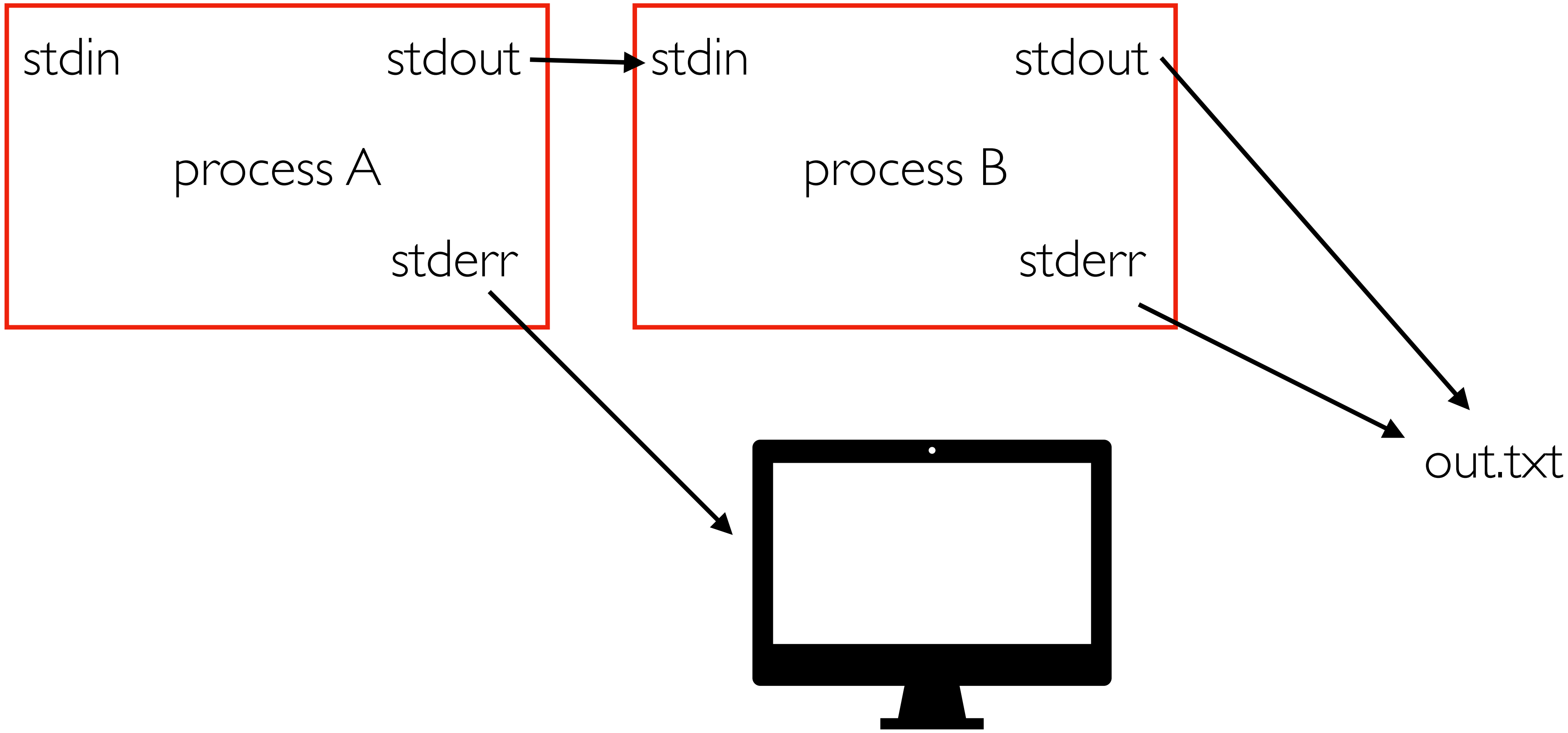
Command:

```
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.

Demos...