

[544] Deploying on Linux

Tyler Caraza-Harter

A Shell: the most helpful program?

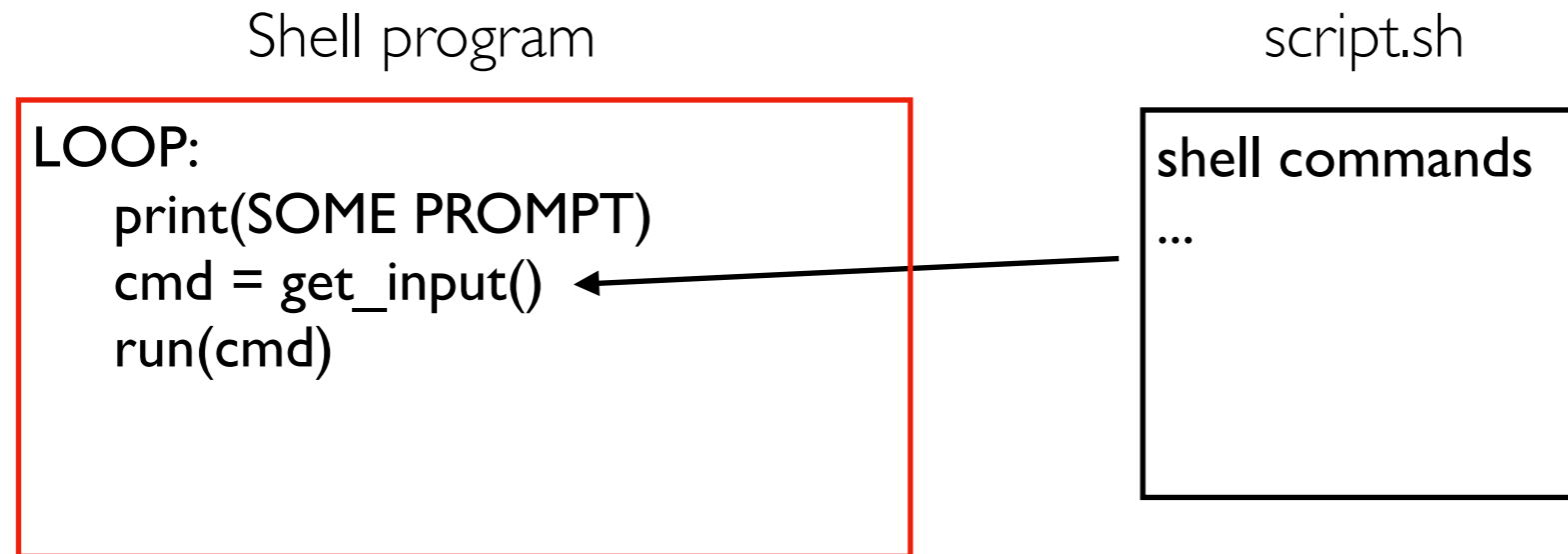
Shell program

LOOP:

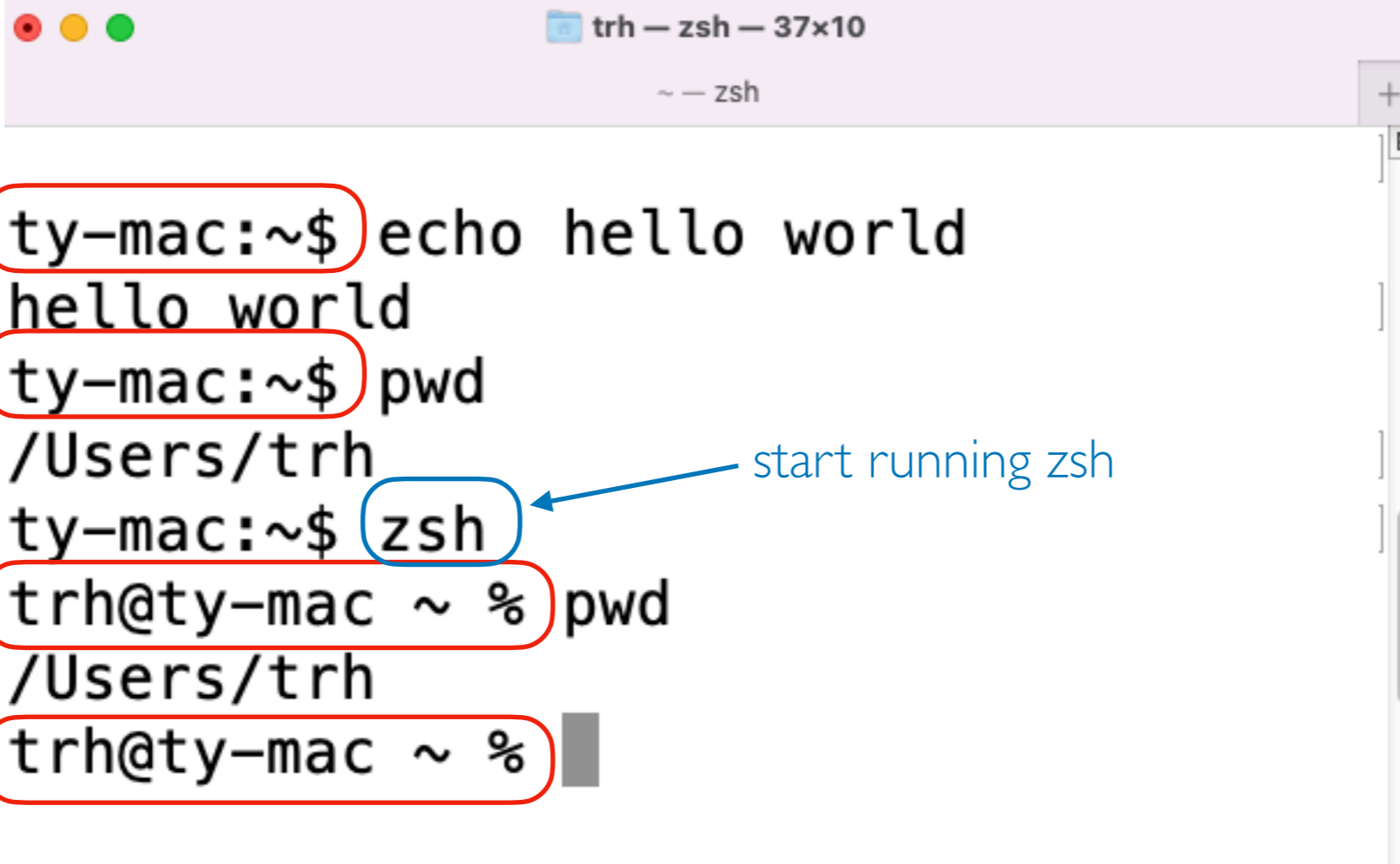
```
print(SOME PROMPT)
cmd = get_input()
run(cmd)
```



If you can type it, you can automate it.



You can start running a shell inside a shell



The image shows a terminal window titled "trh — zsh — 37x10" with a subtitle "~ — zsh". The terminal output is as follows:

```
ty-mac:~$ echo hello world
hello world
ty-mac:~$ pwd
/Users/trh
ty-mac:~$ zsh
trh@ty-mac ~ % pwd
/Users/trh
trh@ty-mac ~ %
```

Annotations in the image include:

- Red arrows pointing to the prompts `ty-mac:~$` and `trh@ty-mac ~ %`, labeled "bash prompt" and "zsh prompt" respectively.
- A blue arrow pointing to the `zsh` command, labeled "start running zsh".

SSH: Secure Shell

running on
my laptop →

```
trh — trh@instance-1: ~ — ssh trh@34.27.165.44 — 80x29
~ — trh@instance-1: ~ — ssh trh@34.27.165.44

ty-mac:~$ hostname
ty-mac
ty-mac:~$ ssh trh@34.27.165.44
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1027-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Jan 24 16:13:32 UTC 2023

System load:  0.0          Processes:            102
Usage of /:   12.4% of 24.05GB  Users logged in:    0
Memory usage: 13%          IPv4 address for ens4: 10.128.0.36
Swap usage:  0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Last login: Tue Jan 24 15:59:38 2023 from 72.33.0.184
trh@instance-1:~$ hostname
instance-1
trh@instance-1:~$ █
```

running on
my VM →

Demos...

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

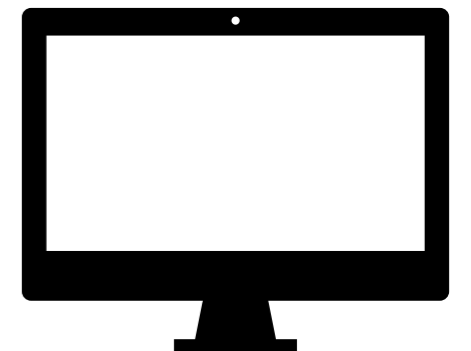
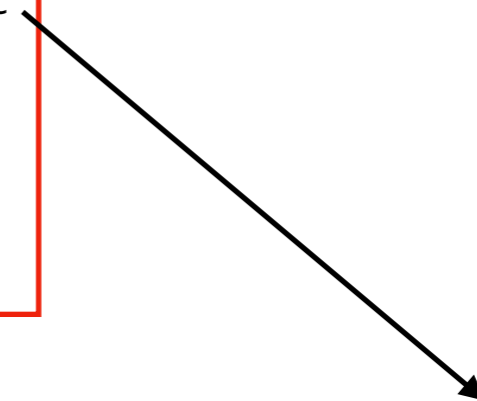
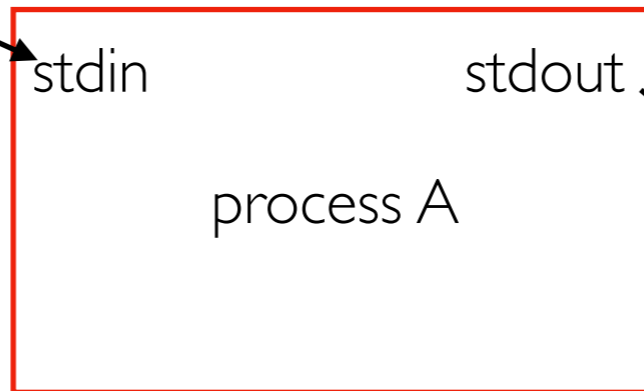
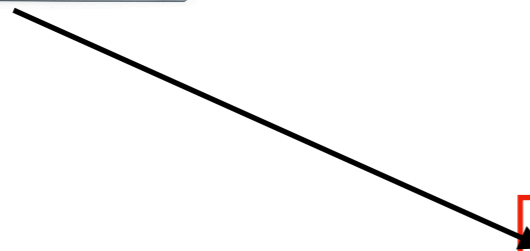
Various tools can take these log files and produce pretty reports about your website traffic, but for the sake of exercise, let's build our own, using basic Unix tools. For example, say you want to find the five most popular pages on your website. You can do this in a Unix shell as follows:¹

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

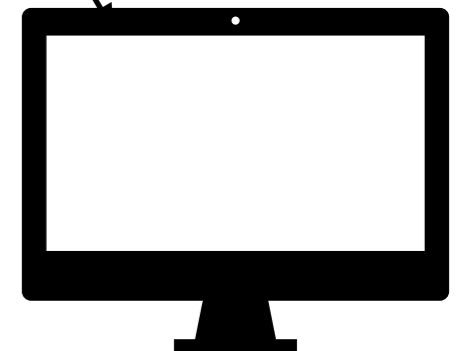
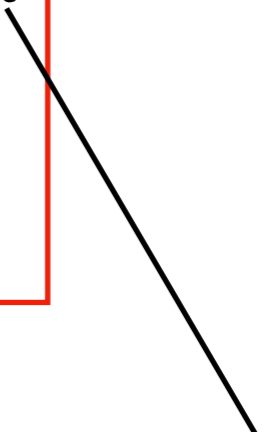
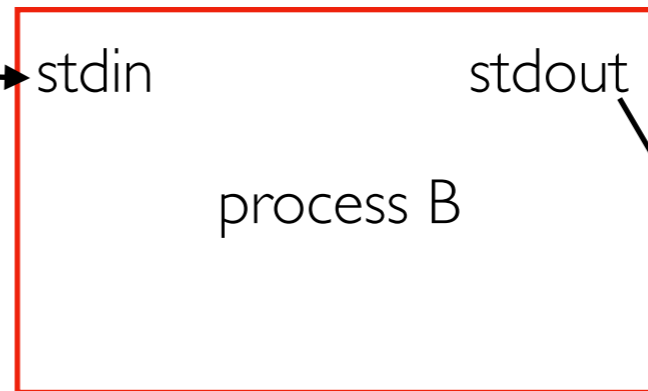
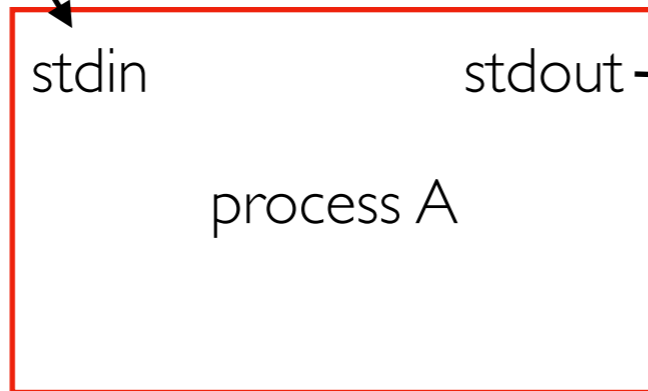


the pipe connects output of one process to input of the next

Standard Input and Output (I/O)



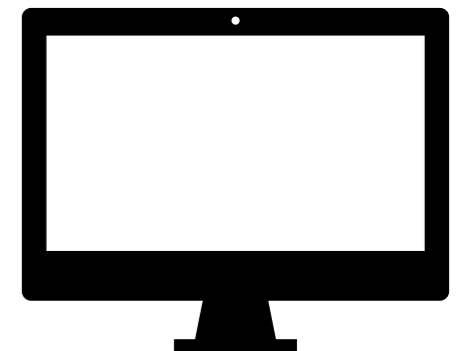
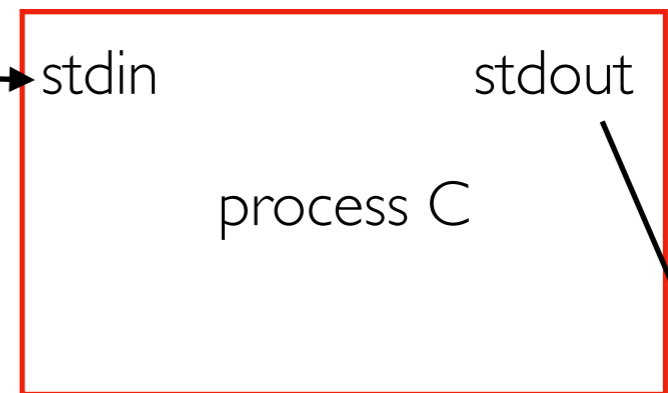
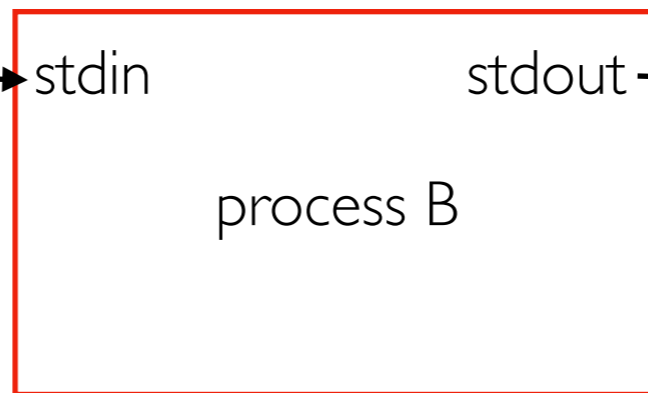
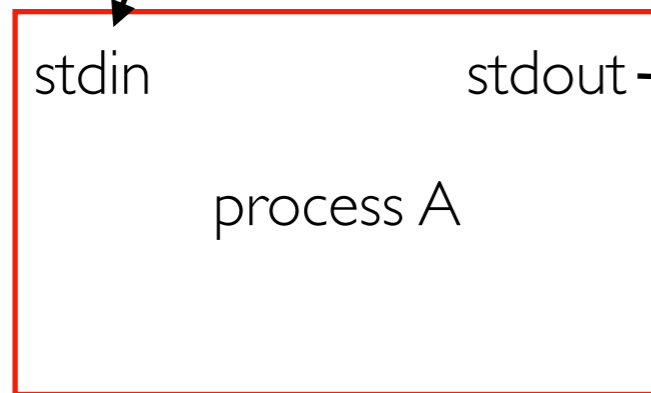
stdout => stdin



Command:

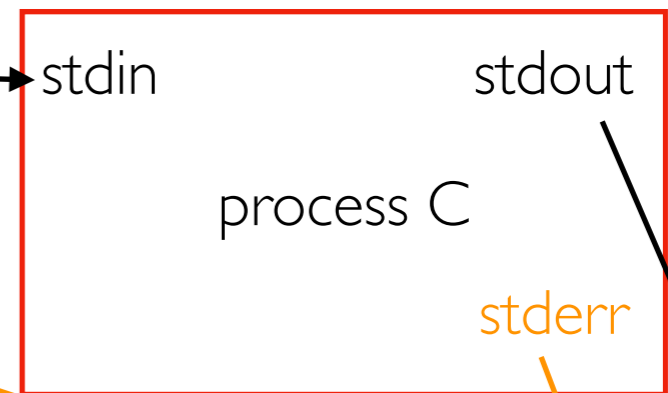
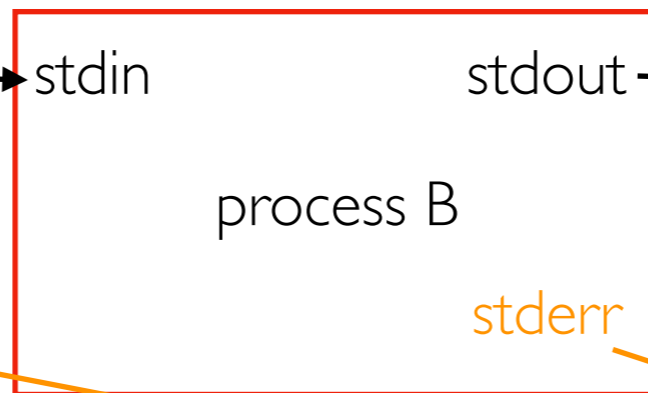
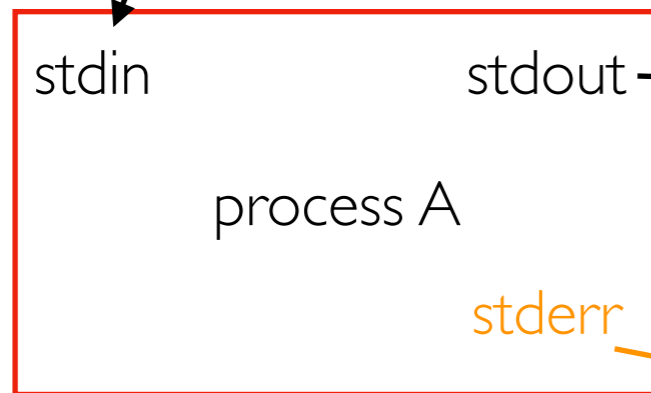
A | B

Chains can be long

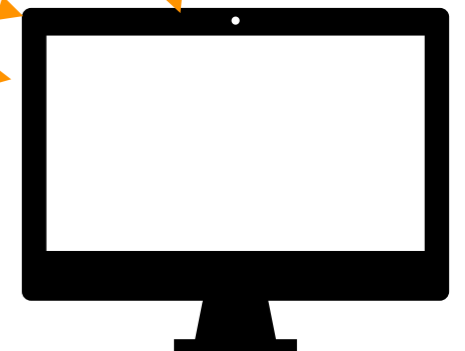


Command:
A | B | C

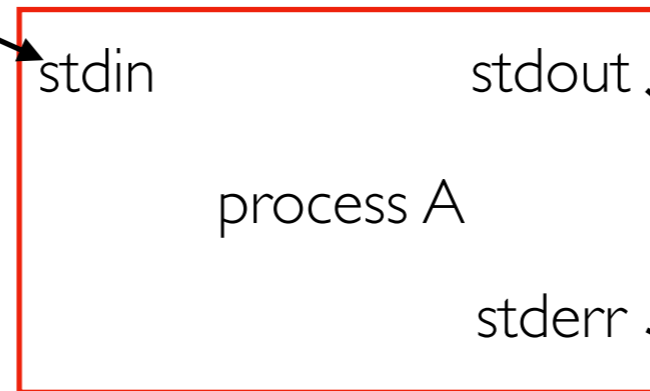
stderr (for things like warnings that shouldn't be chained)



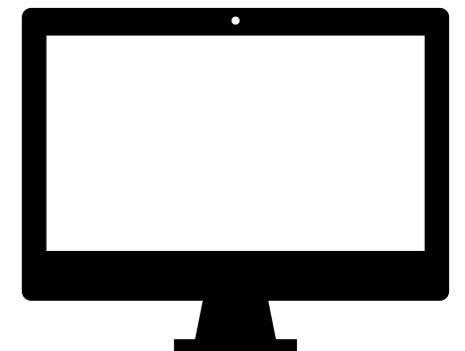
Command:
A | B | C



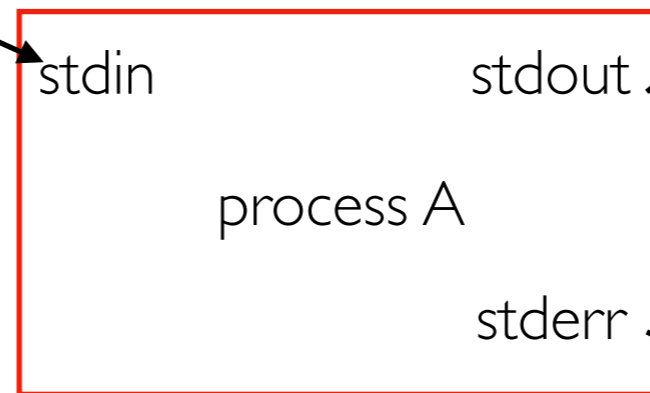
Redirection



Command:
A

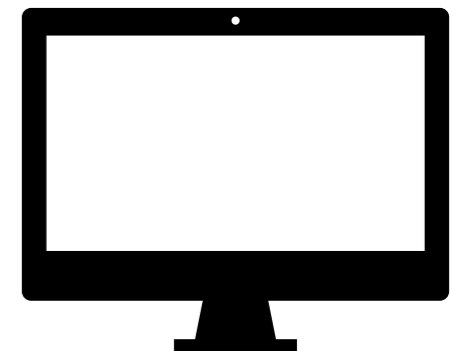


Redirection

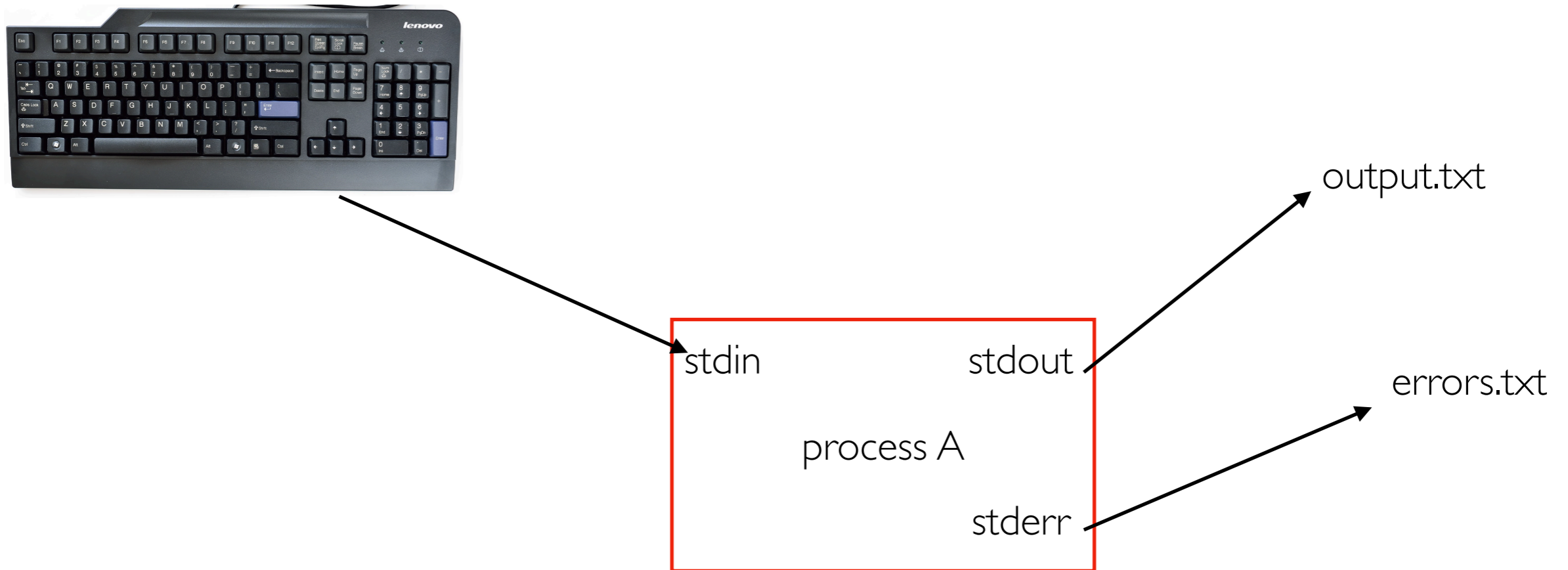


output.txt

Command:
A > output.txt



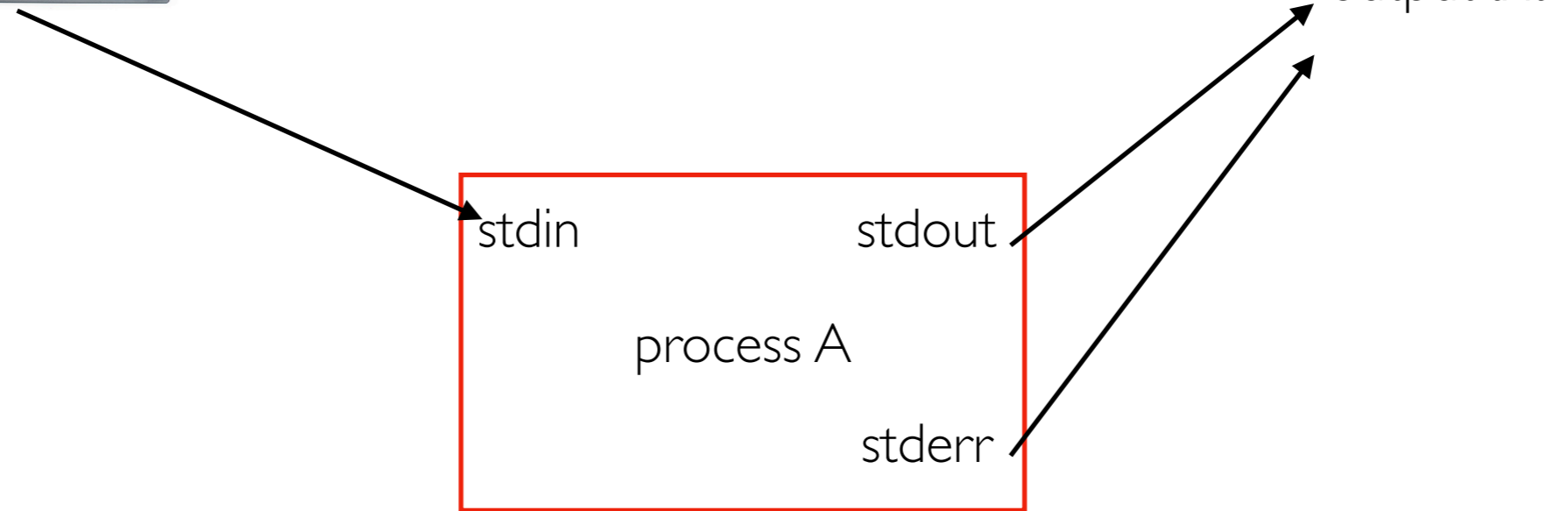
Redirection



Command:

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

Redirection

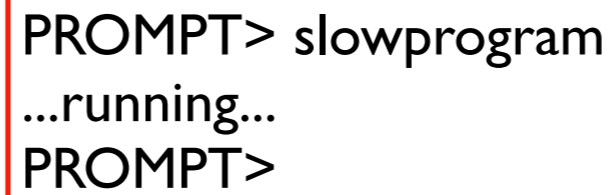


Command:

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


Async

normally, shells commands are synchronous, meaning you wait for the last command to finish before another prompt appears.



```
PROMPT> slowprogram
...running...
PROMPT>
```

A terminal window with a red border showing the command 'slowprogram' being executed. The output is '...running...' followed by a new prompt 'PROMPT>' on the next line. An arrow points from the explanatory text to this terminal window.

ampersand at the end runs it in the background. you get a prompt immediately



```
PROMPT> slowprogram &
PROMPT>
```

A terminal window with a red border showing the command 'slowprogram &' being executed. The prompt 'PROMPT>' appears immediately on the next line. An arrow points from the explanatory text to this terminal window.

All together

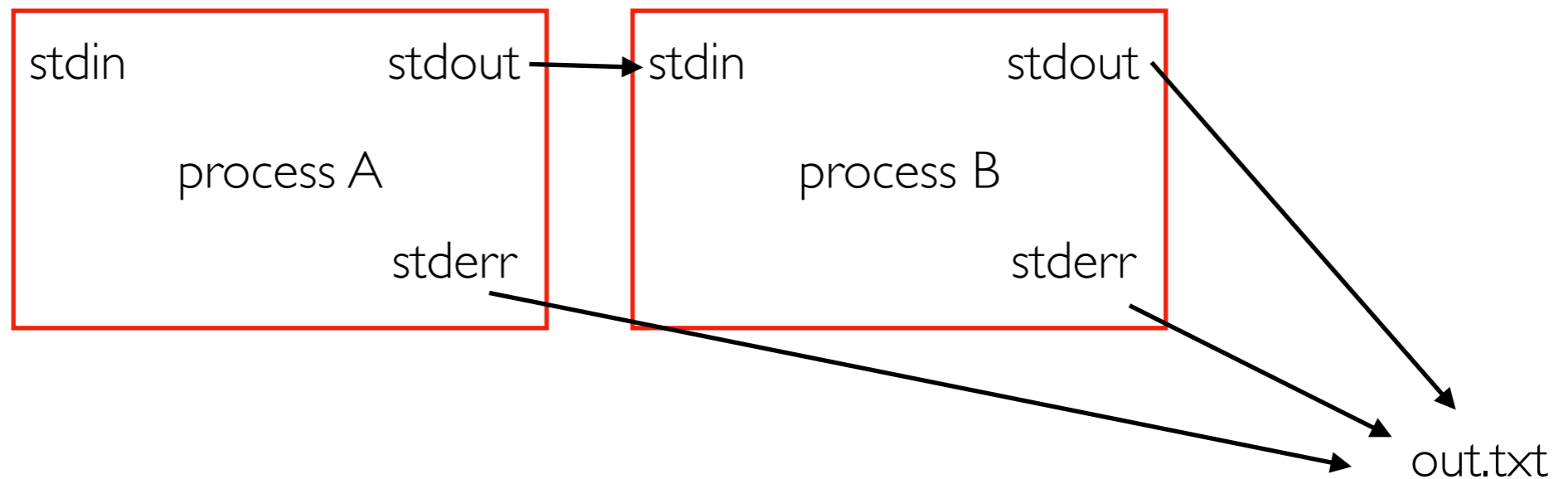
Command:

```
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...