Operating Systems!

OS Interaction (Part 3)

How to Get Stuff Done Using the OS, Processes, Threads, Etc.

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

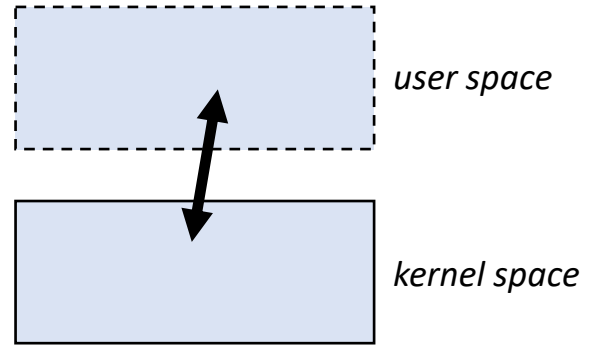
https://www.cs.montana.edu/cs460

Today

- Announcements
  - *Heads up.... PA1 due*
    Sunday [09/20/2020] @ 11:59 PM (MST)

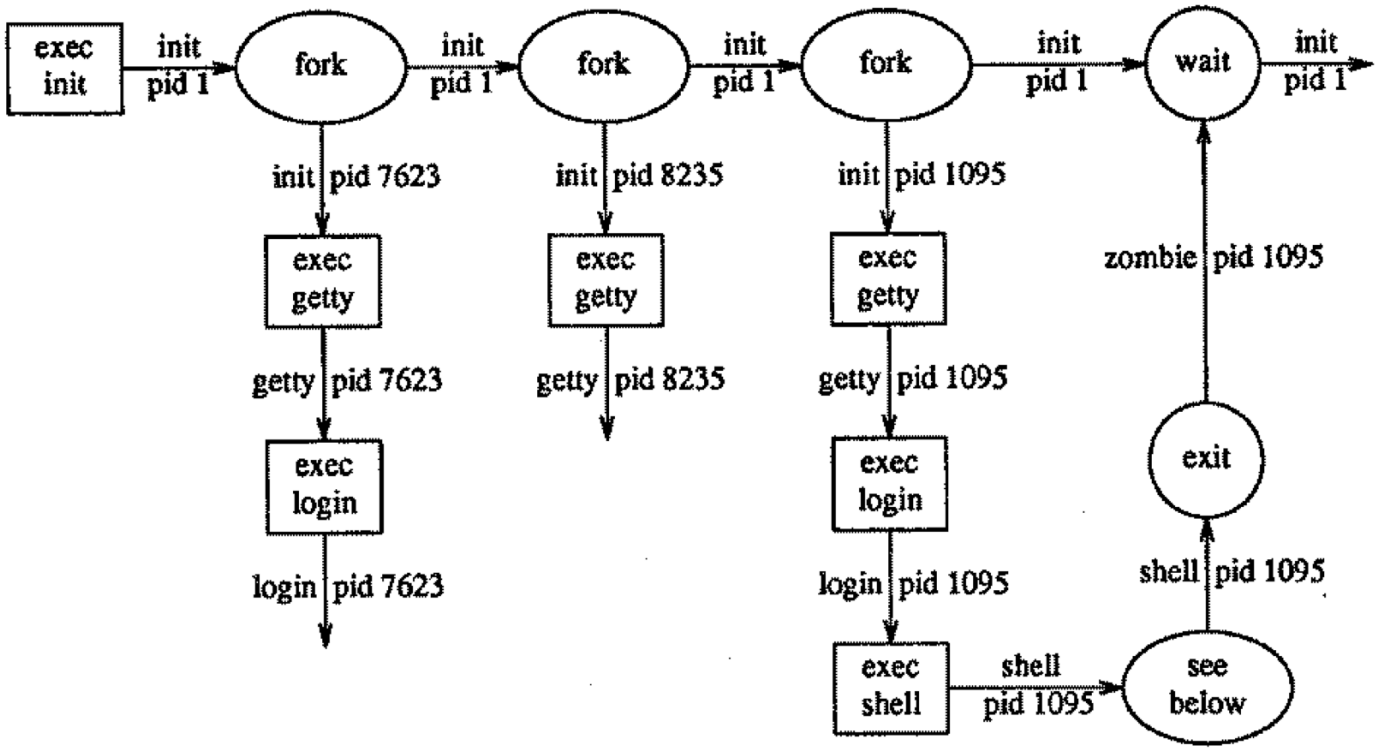
    NO D2L! *Code pushed to GitHub == submitted!*

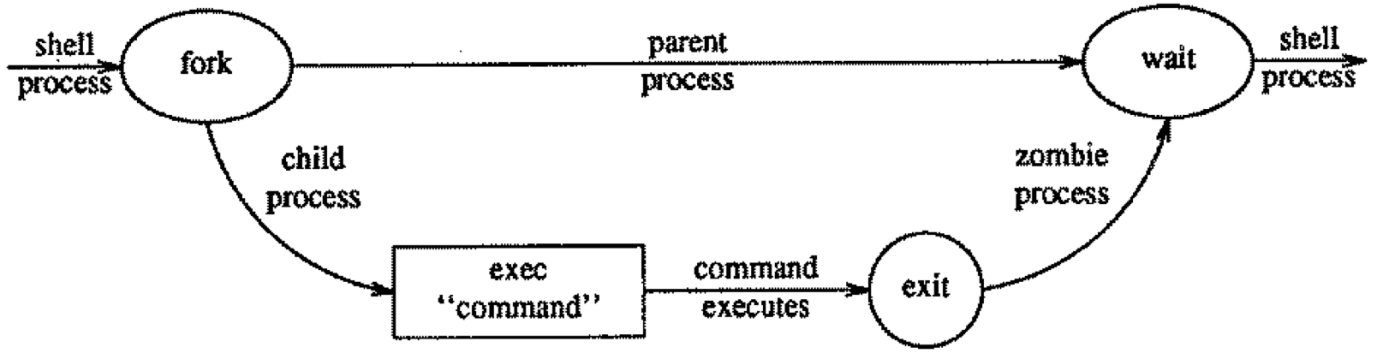
- Learning Objectives
  - Understand the big ideas behind the "OS API"
    - ~~**user vs. kernel, modes, syscalls, libraries**~~
  - Understand the big ideas behind process control
    - *[theory]* **control info, creation, termination, states, etc.**
    - *[reality]* ~~**fork, exec, getpid, waitpid, etc.**~~



Last Time...

- Process Control
  - `fork`
  - `exec`
  - `getpid`
  - `waitpid`
  - `kill`



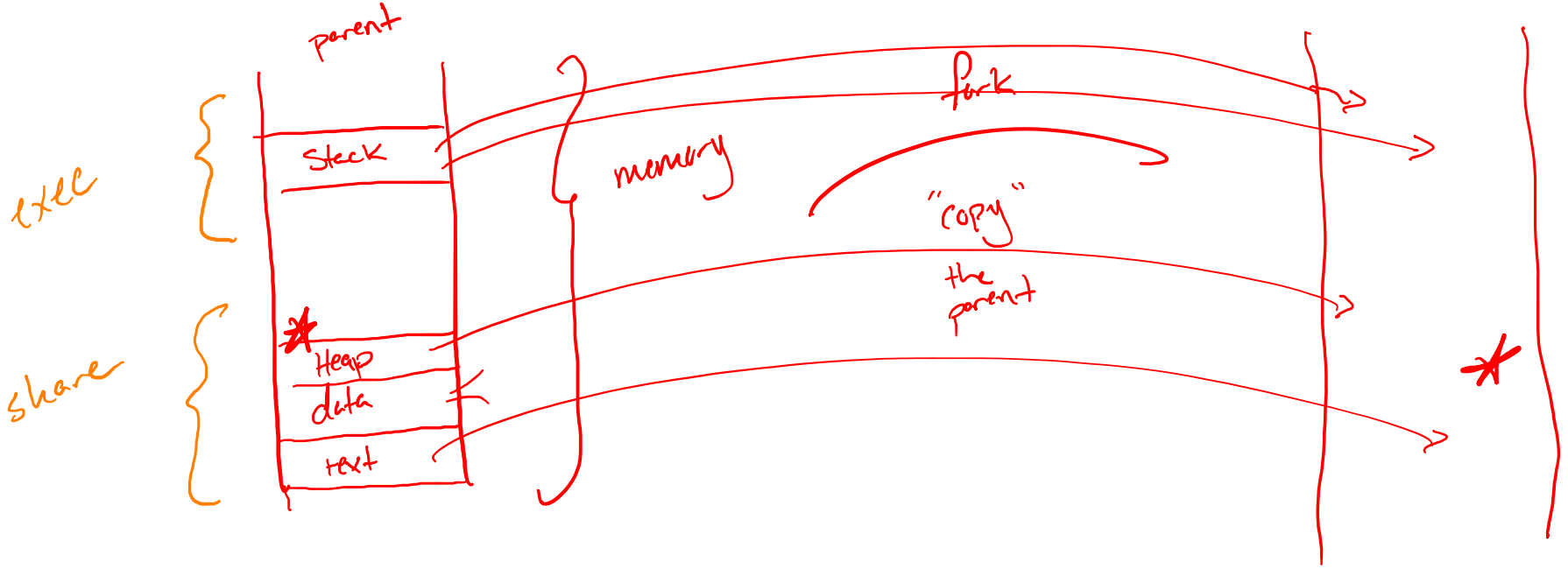


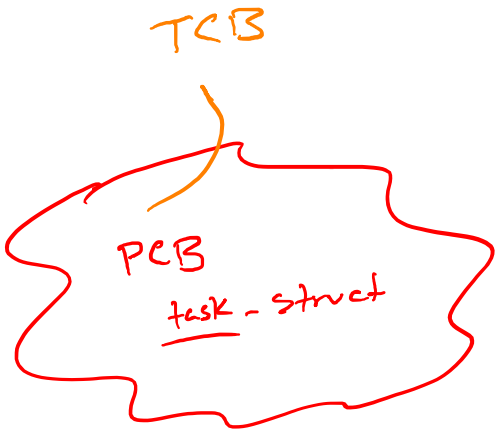
—John S. Quarterman, Abraham Silberschatz, and James L. Peterson, ``4.2BSD and 4.3BSD as Examples of the UNIX System'', Computing Surveys, Volume 17, Number 4, (December 1985), pages 379-418; translated to Japanese, Computer Science (BIT), Volume 18, Number 3, (1986), pages 175-213.

Last Time...

- Great question at the end of last class...

Q: Will a **child** see changes to a **global variable** made by the **parent**?





How can processes communicate?

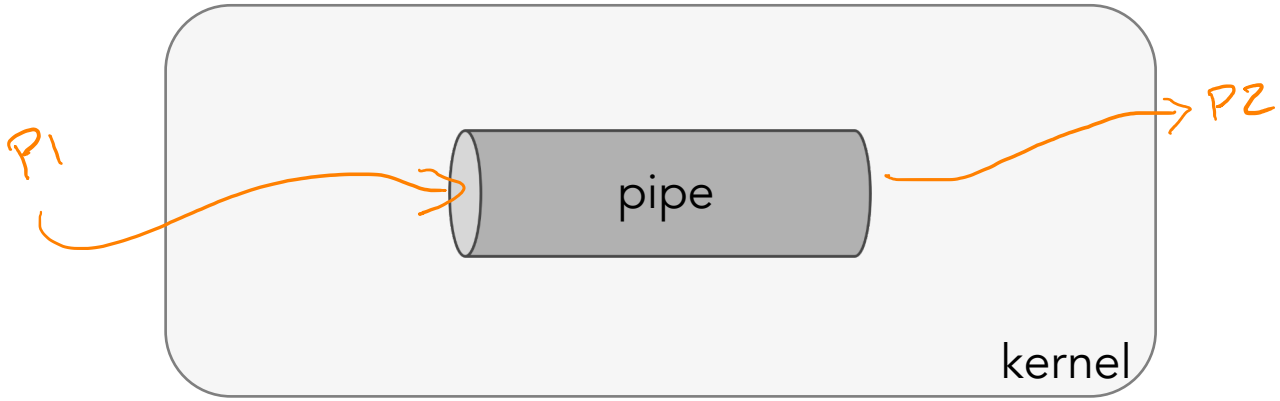
Inter-Process Communication (IPC)

Enable multiple processes to communicate & share data

- **Pipes** are used for communication between related processes *(e.g., parent-child)*

Doug McIlroy was the head of Bell Lab's Computing Techniques Research Department which created the Unix operating system. McIlroy is widely credited with creating the Unix computing concept known as **"pipes" (short for pipelines)** which allows users to connect two or more software tools together in order to perform complex computing tasks. McIlroy also authored several of Unix's core tools such as diff, sort, spell, join, graph, speak, and tr.

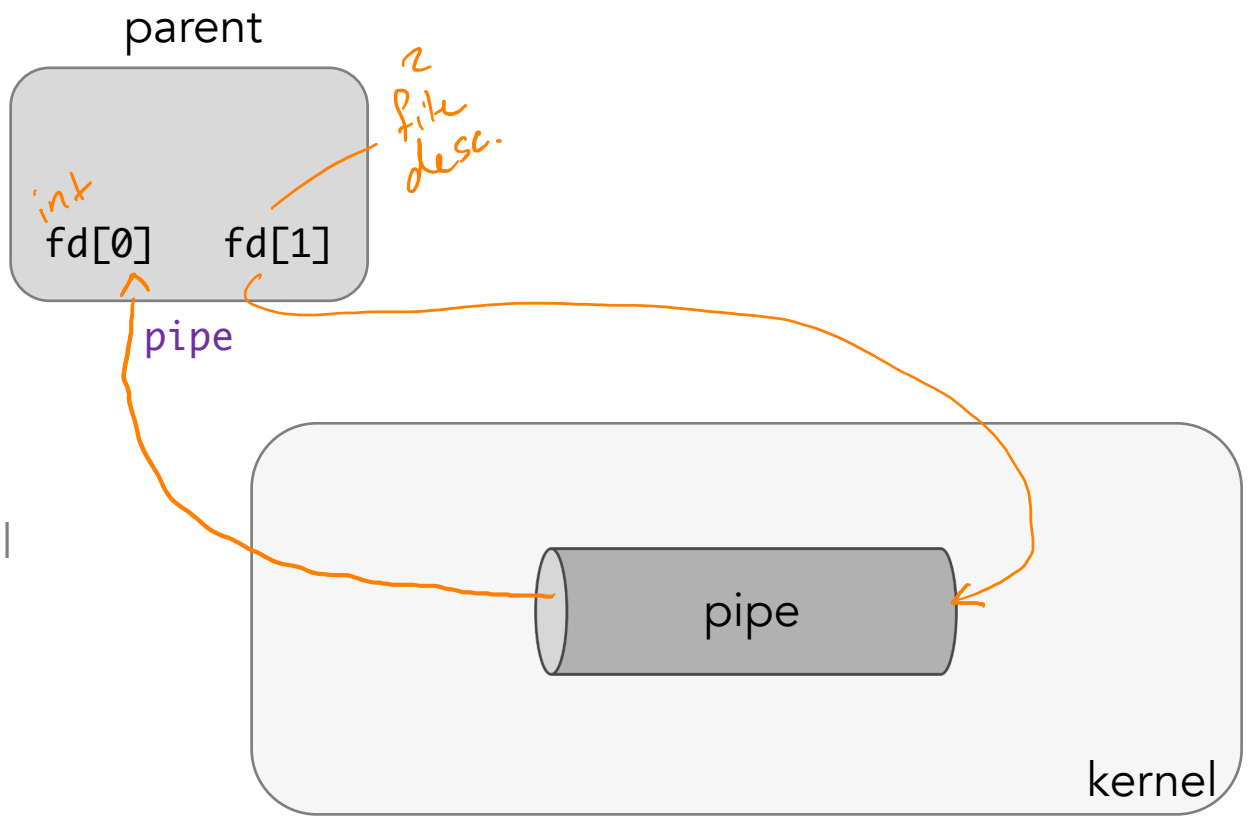
- https://www.facesofopensource.com/doug-mcilroy-2/



Inter-Process Communication (IPC)

Enable multiple processes to communicate & share data

- **Pipes** are used for communication between related processes *(e.g., parent-child)*

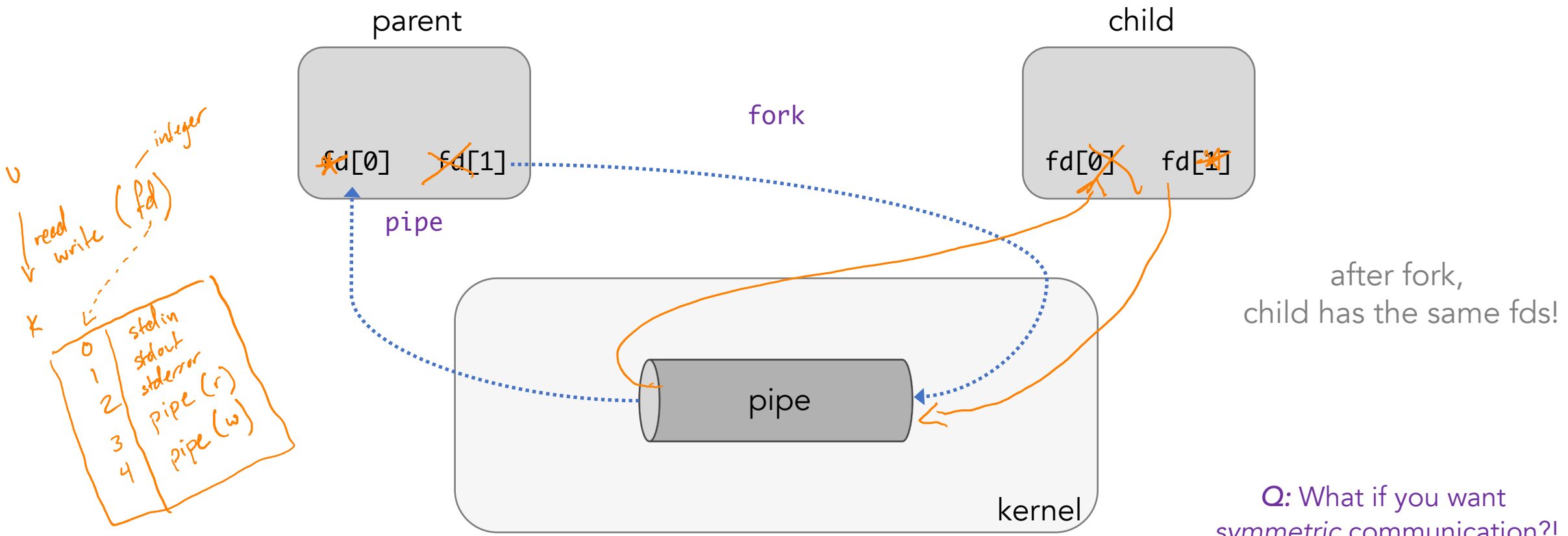


call "pipe" to ask kernel to create a pipe and return r/w fds

Inter-Process Communication (IPC)

Enable multiple processes to communicate & share data

- **Pipes** are used for communication between related processes *(e.g., parent-child)*



after fork,
child has the same fds!

Q: What if you want *symmetric* communication?!
(pipe.c)

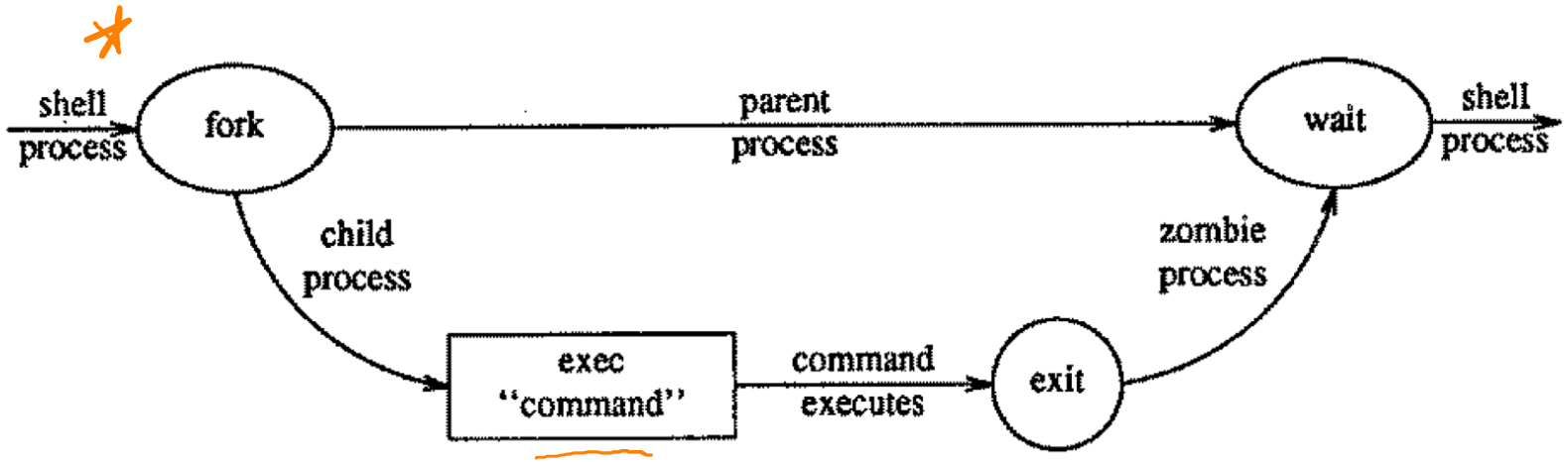
What is a shell?

How might we implement a shell?

Shells

Interacting with the OS

- In a loop...
  - read commands from stdin
  - fork a child to run commands
  - parent waits for child to finish
  - *+ input parsing, error checking, ...*

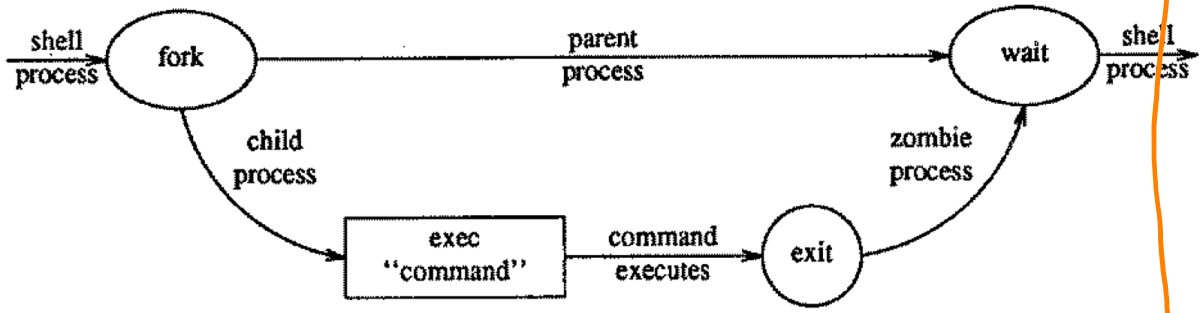


—John S. Quarterman, Abraham Silberschatz, and James L. Peterson, ``4.2BSD and 4.3BSD as Examples of the UNIX System'', Computing Surveys, Volume 17, Number 4, (December 1985), pages 379-418; translated to Japanese, Computer Science (BIT), Volume 18, Number 3, (1986), pages 175-213.

Shells

Interacting with the OS

- In a loop...
  - read commands from stdin
  - fork a child to run commands
  - parent waits for child to finish
  - *+ input parsing, error checking, ...*




```c
/* Start a new process to do the job. */
cpid = fork();
// printf("process id is %d\n", cpid);

if(cpid < 0) {
    perror("fork");
    free(main_ptr); // clean-up
    return;
}

/* Check for who we are! */
if(cpid == 0) {
    /* We are the child! */
    execvp(main_ptr[0], main_ptr);
    perror("exec");
    free(main_ptr); // clean-up
    exit(127);
}

/* Have the parent wait for child to complete */
if(wait(&status) < 0)
    perror("wait");
// printf("wait result for process id %d is %d\n", cpid, status);
```

—fork() and exec() code snippet that Travis wrote back when he was an undergrad...

What if I create LOTS of processes...?

The Fork Bomb

Run at your own risk... probably smartest to run on a VM...

```c
#include <sys/types.h>
#include <unistd.h>

int main()
{
    while(1) {
      fork();
    }
    return 0;
}
```

—https://en.wikipedia.org/wiki/Fork_bomb

DEMO DAY!

Well... the rest of the day... ☺