CSCI 460 Operating Systems

Follow-ups From Last Class / Review

Professor Travis Peters
Fall 2019
Questions From Last Class…

Discuss in groups; get up and use whiteboards around the room to write on!
1. What are User Space and Kernel Space?
   - Why do we have different “spaces” (“User Space” and “Kernel Space”)?
   - Why is it expensive to “switch” between them?
   - What is the canonical memory layout for User Space and Kernel Space
1. What are User Space and Kernel Space?
   - Why do we have different “spaces” (“User Space” and “Kernel Space”)?
   - Why is it expensive to “switch” between them?
   - What is the canonical memory layout for User Space and Kernel Space?
Questions From Last Class…

Discuss in groups; get up and use whiteboards around the room to write on!

2. Heaps (user heap vs. kernel heap)
   • How are heaps used in OSs?
   • For that matter, what is the stack? And how is it used?
Questions From Last Class…

*Discuss in groups; get up and use whiteboards around the room to write on!*

2. Heaps (user heap vs. kernel heap)
   - How are heaps used in OSs?
   - For that matter, what is the stack? And how is it used?
Questions From Last Class…
Discuss in groups; get up and use whiteboards around the room to write on!

3. Fork—why would you `fork()`?
   - What does `fork()` do? Why is that useful?
   - Pros and cons of `fork()`?
   - Alternatives to `fork()`?
Questions From Last Class…
Discuss in groups; get up and use whiteboards around the room to write on!

3. Fork—why would you \texttt{fork()}?

- What does \texttt{fork()} do? Why is that useful?
- Pros and cons of \texttt{fork()}?
- Alternatives to \texttt{fork()}?

\textbf{pipes}

- Sometimes you want to send the output of one process to the input of another: `ls | wc -l`
- Processes are like pipes:
  - pipe buffers: `ls` is gonna write a bajillion bytes to `wc`
  - what if your target process dies? `uh no if my buffer is full you have to wait` (\texttt{SIGPIPE})
- You can pipe so many things together: `ls a b c d l e`
- A pipe is a pair of 2 magical file descriptors: `ls` and `wc`
- When is does `write()`, "hi" we can read it! `read()` returns "hi"

\textbf{Julia Evans}
@bdrk
drawings.juns.ca

\textbf{Figure 24-1: Overview of the use of \texttt{fork()}, \texttt{exit()}, \texttt{wait()}, and \texttt{execv()}}
Photos of whiteboards after class
(Unfortunately, some got erased before I could snap a photo...)
- Unprivileged instructions
  - Privileged instructions (e.g., interact with device)
  - User space vs. Kernel space
    - Syscalls

- Why?
  - For control & security purposes

- Expensive to switch
  - Not the same memory space
  - More change of CPU
  - Ensure user space cannot execute/access certain privileged code

- For:
  - Duplicate new process
  - Check if you're a child (return code)
  - PCB
  - Security issue
  - Cons: strange, pro: origin

Heap:
- Information to store in dynamic memory that you can access in any order

Stack:
- Cannot access in any order
- Completed calls
Heap
- Allocates Dynamic Memory
  - malloc() + calloc()
- User heap
- Kernel heap

\[ U \]

\[ K \]

\[ \text{less permissions to hardware-level resources} \]

\[ \text{init} \rightarrow ? \]

\[ P_1, P_2 \]