CROS

(Custom Raspberry Pi Operating System)

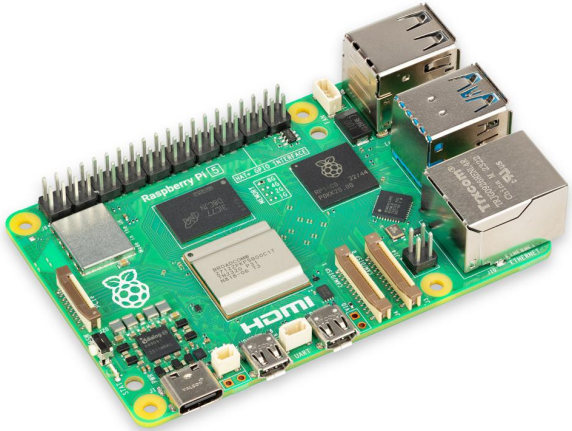
Group 16

Nathan Giddings, Sam Lane, Hunter Overstake
Connor Persels, Kyle Clements

Our Project - Recap

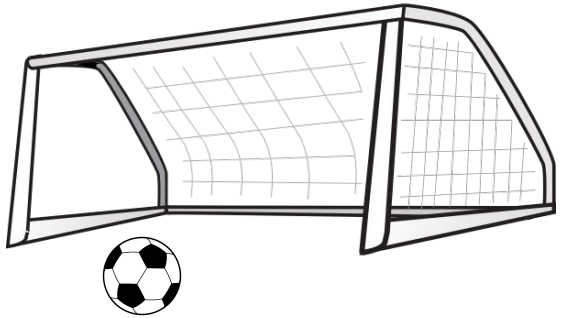
- Custom Operating System:
  - Targeting Raspberry Pi 3/4
  - Terminal based
  - GPIO support

- Components:
  - Shell
  - Kernel
  - Libraries



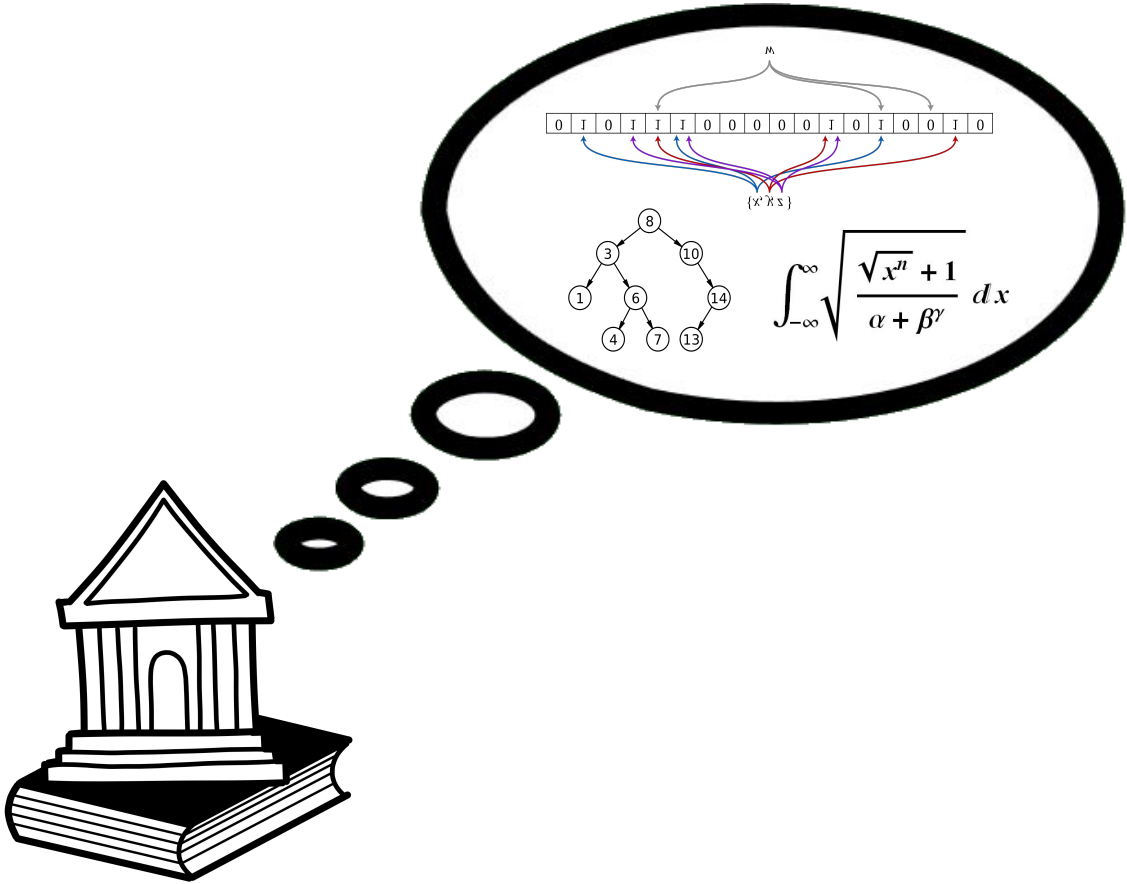
Goals for this Demo

- Shell
  - Added features: piping, executing binaries, etc.
  - Communicate with kernel
- Kernel
  - Ability to run userspace programs
  - Integrate file system into kernel
- OS Library
  - Writing custom standard C library
  - Integrate with kernel system calls
- Container Library
  - Write custom data structure for general use
  - Write custom types/helper tools



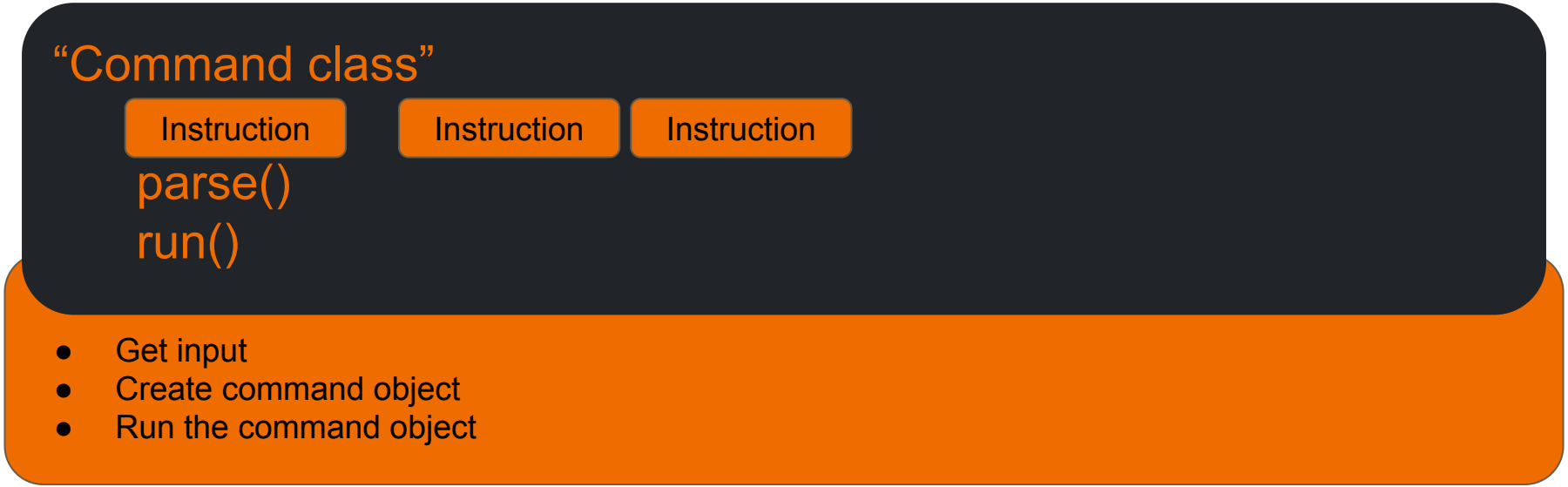
Container Library

- Data structures
  - Vector
  - Linked List
  - Hash Map
  - Binary Search Tree
  - Pair
- Types/other
  - String
  - Math



Shell - Technical Details

- Shell
  - Interpretation
  - Processing



Challenge

- … fork bomb?
- Debugging pipe
- Linker issues with userspace
  - (our C library)
- User space heap

Bring it all Together

Create a directory and install kernel code using make

Install the C library to the same created directory

Compile a sample program from the user

Paste that sample program into the OS install and compile

OS Library - Current Status

- Crt0.s is minimally built
  - Stores parameters given by kernel
  - Loops
- Uinit can be reached by Crt0.s
  - Makes the library active
- Can compile successfully with stubs
- Libraries with partial work:
  - String (functional)
  - Stdlib (functional)
  - Stdio (functional)
  - Time (stub)
  - System call wrapper (stub)

OS Library - Remaining Work

- Filling in stubs
  - Time.h
  - System calls
  - Additional functionality for other libraries
- Making heap for userspace
  - Active history log
  - Dynamic memory allocation
- Link with shell

Kernel - Current Progress

- Memory management
  - Paging
  - Physical memory allocation
  - Kernel heap
- System Call Handler
  - **`mmap`**, **`munmap`** implementation
  - Stubs for other syscalls
- Interrupt Handler Table
  - **`find_irq_source()`**
- Context Loading
- Program Loader

Kernel - Remaining Work

System Call interface

Memory Management	Process	Signal	File System
mmap()	clone()	sigraise()	open()
munmap()	terminate()	sigret()	create()
	exec()	sigwait()	unlink()
	yield()	sigaction()	read()
			write()

Kernel - Remaining Work

- File System
  - Integration with kernel
- Scheduler
  - Save/load process states
  - Round-robin scheduling
- Device Drivers
  - SD Card
  - GPIO
- Signals

Kernel - Challenges

- Hardware/Emulator differences
  - Testing on physical hardware is time consuming
  - Emulator is much more forgiving than hardware
- Example: 4-level paging vs. 3-level paging

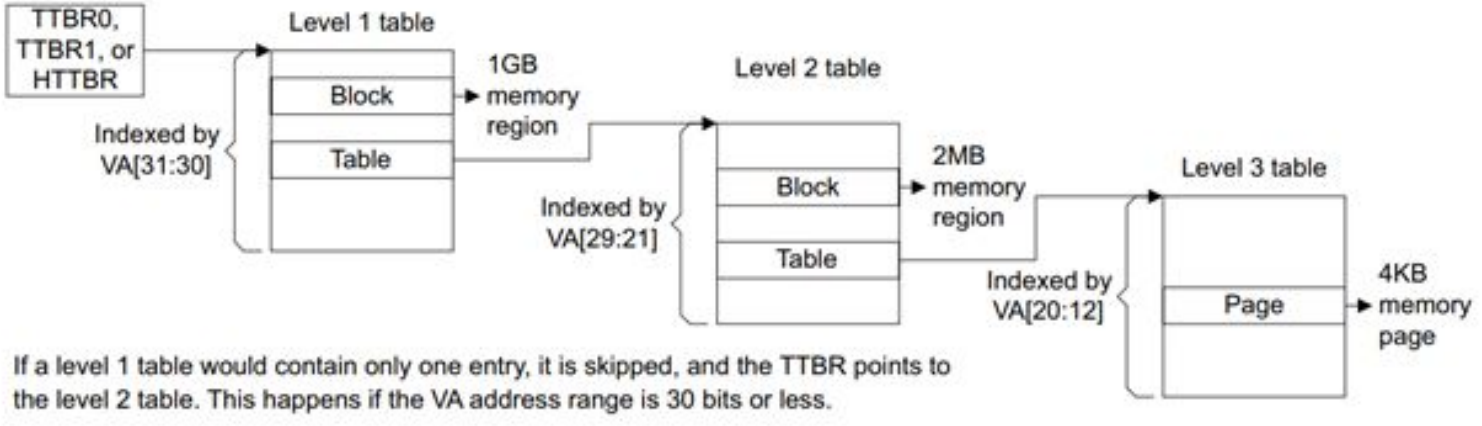
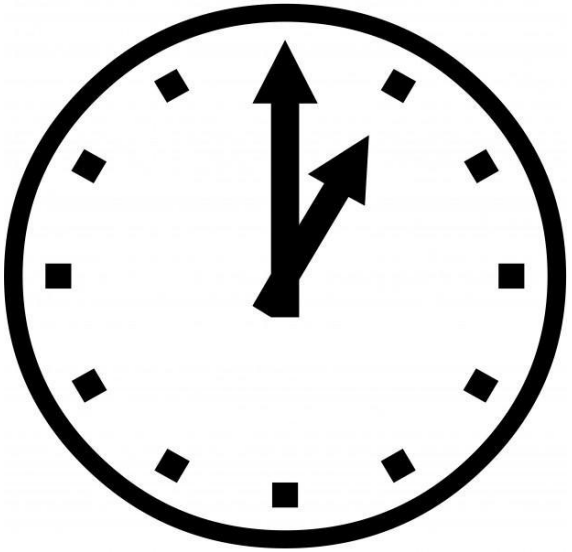


Figure G5-8 General view of VMSAv8-32 stage 1 address translation using Long-descriptor format

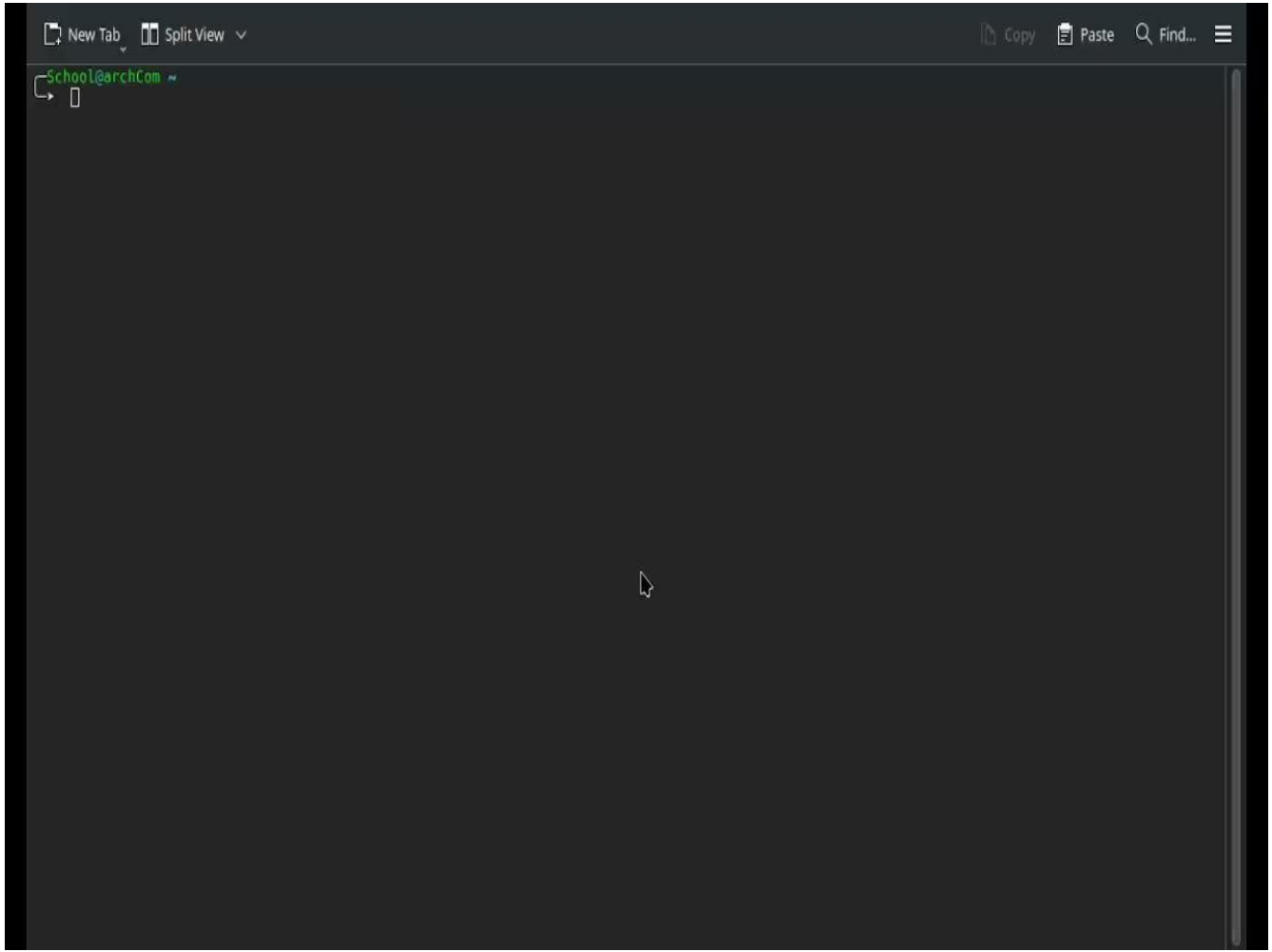
Timeline

Demo 3

- Integration
  - Link shell to kernel
  - File system
- Implementation
  - Scheduler
  - Disk driver
  - Full set of system calls
  - GPIO support



Demo



Questions