CROS

(Custom Raspberry Pi Operating System)

Group 16

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

Our Project

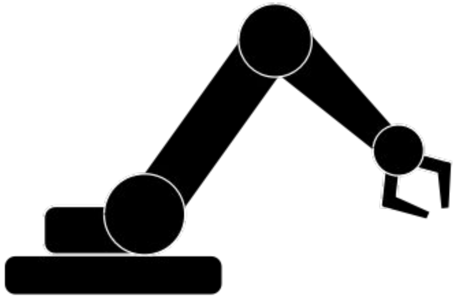
- Custom Embedded Operating System
    - Experience with OS development
    - Understanding System Architecture
    - Systems-level programming practice
    - Kernel development exposure
    - Enables customization

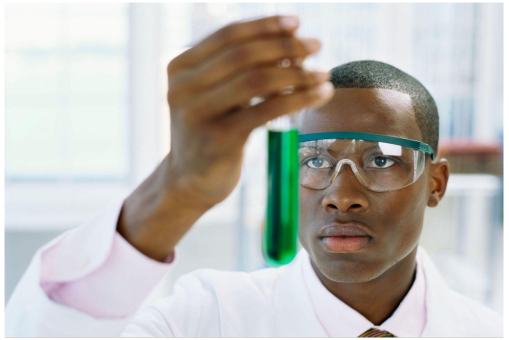
Requirements

- Minimalist operating system for the Raspberry Pi
  - Terminal based
  - GPIO support
  - File system usage and navigation
  - Write and exec user programs
- Customizable for a client's needs
  - Custom shell commands
  - Tailored kernel performance
  - Portability between different Pi versions

Use Cases

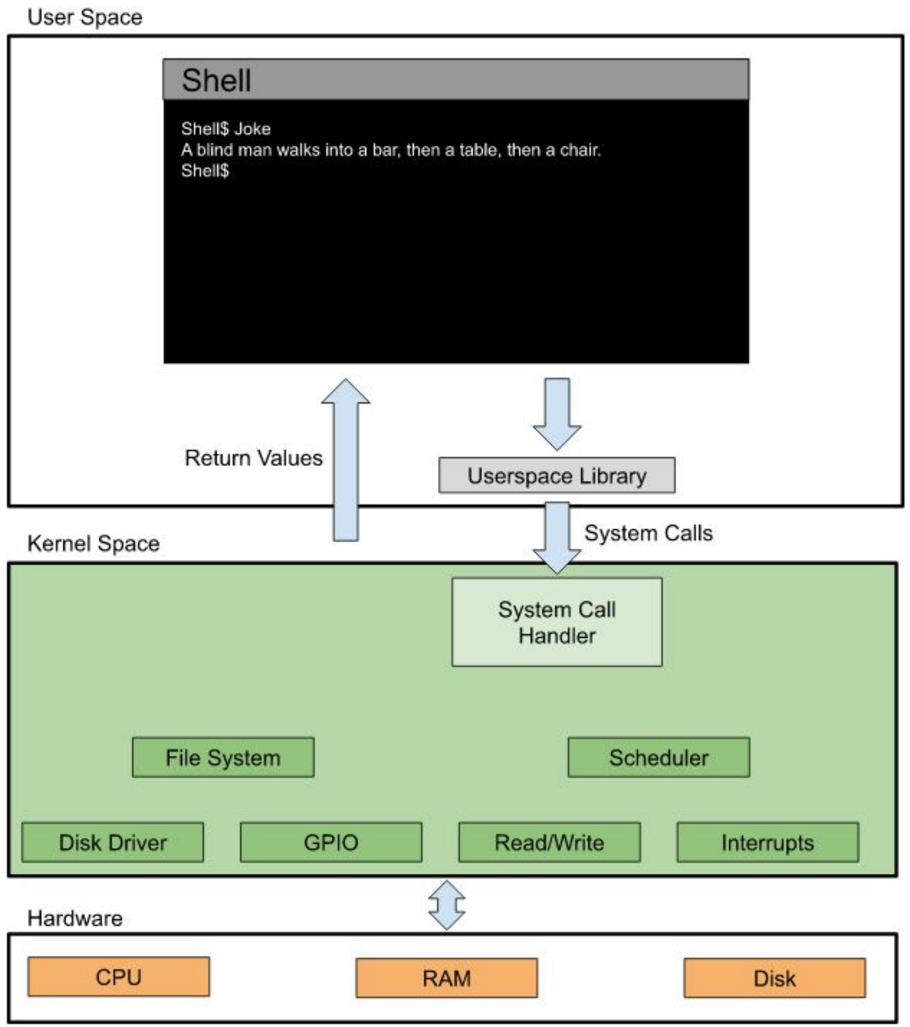
- Embedded Systems
  - Microcontrollers
  - Low-power applications
- Remote Servers
  - Web servers
  - Database servers
- Education/Research
  - Teaching OS development
  - Exploring OS concepts





OS Structure Overview

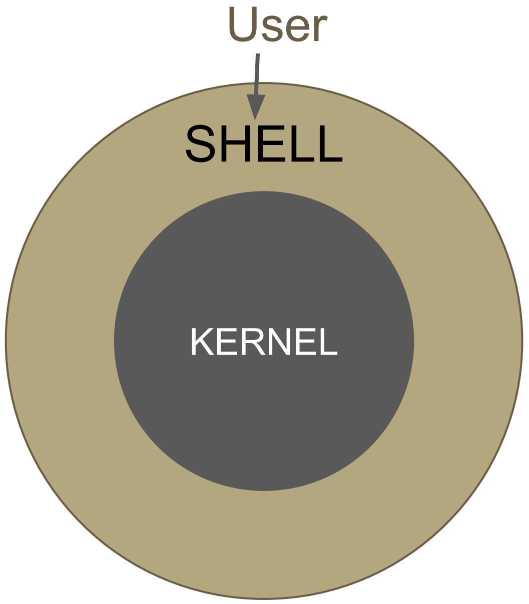
- Shell (Frontend)
  - User command interpreter
  - Processes calling & management
    - foreground & background
    - Stopping & starting
  - File system navigation
  - Userspace library
    - Interfacing w/ kernel for the User
- Kernel (Backend)
  - File System
  - Scheduler
  - Memory Manager
  - Interrupt Handler
  - System Call Interface
  - Device Drivers

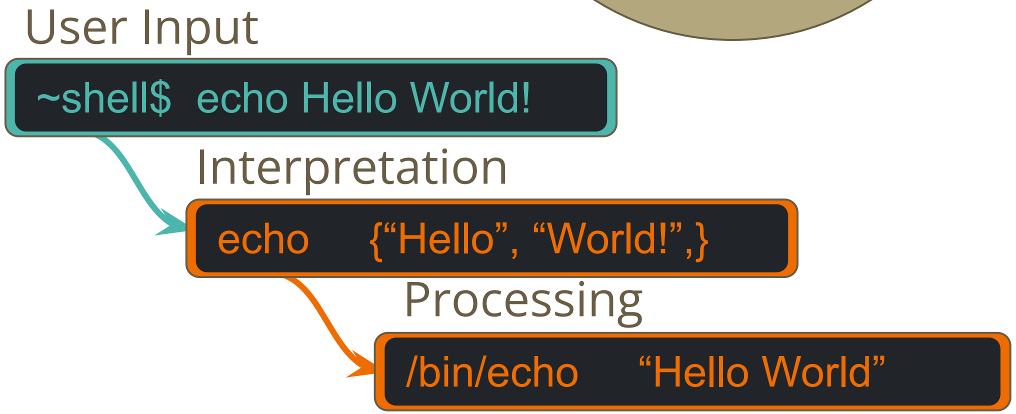


```
Shell$ Joke
A blind man walks into a bar, then a table, then a chair.
Shell$
```

Shell - Design Overview

- Outer layer of the OS
- User interface w/ kernel
- Independent processes handling
- Robust error handling





```
~shell$  echo Hello World!
```

```
echo     {"Hello", "World!",}
```

```
/bin/echo     "Hello World"
```

Shell - Current Progress

- Developing separately in a Linux environment
- (some) Input interpretation
  - Line parsing
  - Parameters and rudimentary piping
- (some) Command processing
  - Containerized process calling
  - Input error handling
- Basic programming
  - date
  - joke

```
~shell$  /
```

Shell - Remaining Work

- Completing command piping
- File navigation
- User input context
  - Input storing
  - Command completion
- Userspace library

- Background vs foreground
- Process management
- Terminal operators
  - &/bg, &&,
- Other commands
  - less, grep,

Shell $\longrightarrow$ ~~Linux environment~~ $\longrightarrow$ Kernel

Shell - Challenges

Connecting the shell and kernel puzzle

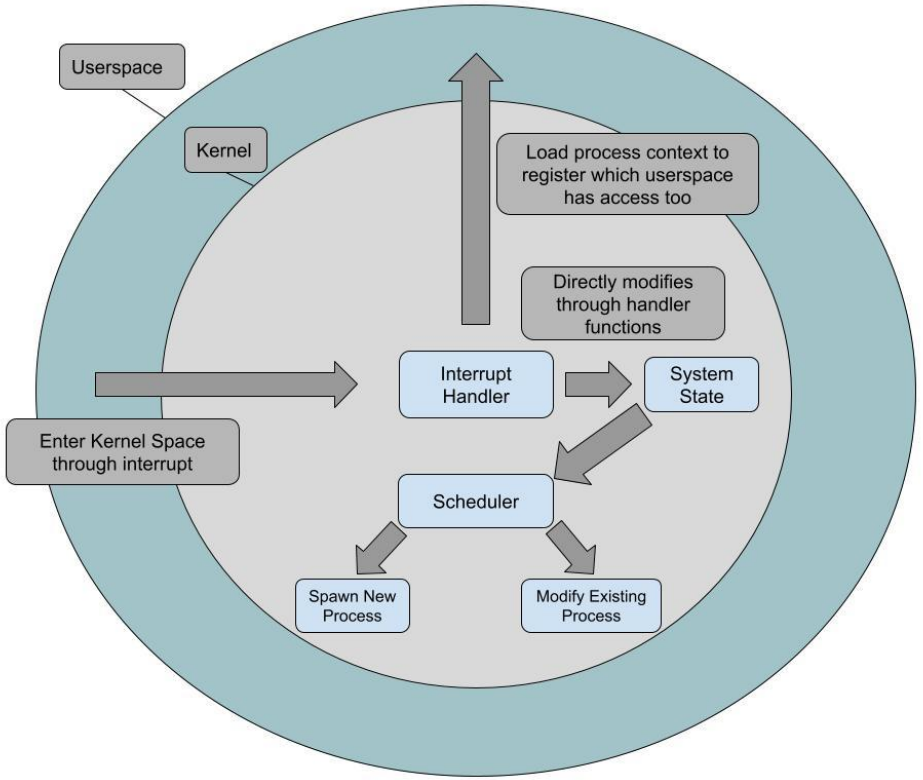
- Userspace library
    - Research required

Internal command parsing & interpretation

- Functionality scales with complexity

Limited library tooling

- C vs. C++ Libraries


Userspace
Kernel
Load process context to register which userspace has access too
Directly modifies through handler functions
Interrupt Handler
System State
Enter Kernel Space through interrupt
Scheduler
Spawn New Process
Modify Existing Process

Kernel - Design Overview

Kernel - Current Progress

- File system: FAT32
  - Parse existing file system
  - File reading/writing
  - Directory listing
- Interrupts
  - Exception handlers in ASM
- Memory Management
  - Paging
  - Heap

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
  - New file/directory creation
  - Asynchronous I/O
- Scheduler
  - Save/load process states
  - Round-robin scheduling
- Signals
- Device Drivers
  - SD Card
  - GPIO
  - Timers

Kernel - Challenges

- File system
  - Asynchronous disk read/write calls
  - Full integration into kernel
  - Data fragmentation
- Scheduling
  - Task switching
  - Signals
  - Process creation/termination
- Device Drivers
  - Will require additional research on components
  - Emulator may not always match real hardware

Tools

- Raspberry Pi 3b/4b
- QEMU
- Vscode
  - C++
  - Assembly
- Cross Compiler (GCC)
  - ARM
  - Baremetal
- GDB (ARM)

Timeline

Demo2

- Shell and Kernel Communication
  - File system is functional and integrated into kernel.
  - Basic kernel system calls working
  - Userspace library

Demo3

- Integration Finished
  - Full set of system calls implemented
  - Custom Text Editor
  - Program execution
  - GPIO support

Demo

- Show kernel booting
- Show filesystem parse disk img
- Show shell taking commands

Questions