PegasOS

Group 23

PegasOS Team

Hardware

Jacqueline Godier
- Drivers
Jacob Thomas
- MMU

Project Management/Sponsor

Christopher Walen

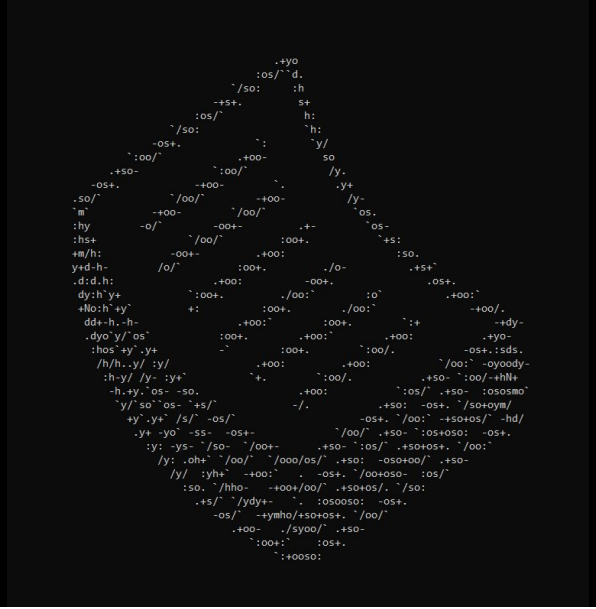
Software

Kenny Alvarez
- File System, Shell
Jacqueline Godier
- Scheduler, System Tasks
Giancarlo Guillen
- Command Shell, User System
Jacob Thomas
- Memory Management, Page Table
Christopher Walen
- Shell, Scheduler, Dependencies

Special Thanks

Raspberry Pi Foundation

Circle Library and Rene Stange (rsta2)



About PegasOS

What is PegasOS?

PegasOS is a new operating system for the Raspberry Pi 4. The primary focus is for 64-bit support, but it can also be compiled into a 32-bit kernel image.

Previously known as KnightOS, the name had to be changed to avoid conflict with an existing open-source OS for calculators.

Who is PegasOS for? Why?

PegasOS is for those that are interested in working with the bare-metal of the Raspberry Pi 4. Every aspect of the OS is documented, making it as transparent as possible for users who want more information about the system they're using.

PegasOS provides an entry point into the lower end of the Raspberry Pi 4, which has previously been a more involved and obscure process.

Project Budget

Project Budget			
Item	**Price**	**Quantity**	**Total Price**
Raspberry Pi			
Micro HDMI cable	$7	1	$7
Micro SD card	$9	1	$9
Raspberry Pi Kit	$75	3	$225
Total			$241

Project Licensing

The project is open source under the GNU General Public License, v3 (GPL 3) and newer. All repositories and documentation are currently open to the public and will remain so.

The Circle Library is open source under GPL 3 as well.

The Raspberry Pi itself does not have any licensing requirements, as per the Foundation’s response in their FAQ.

https://www.raspberrypi.org/documentation/faqs/#commercial

Features

Hardware Compatibility
Uses the Circle Driver Library, which includes support for USB devices, Raspberry Pi GPIO and System Components, Networking, and Graphics.

Simple Command-Line Interface
Commands designed with ease-of-use syntax and naming, to reduce the number of times Google is pulled up.

No Bloatware
Any program or service on the system is solely for running the system, and no data is collected and sent to a database somewhere.

Research

To get PegasOS off the ground, we had to become very familiar with the Pi 4 and its changes over previous models. A lot changed between just the Pi 3 and Pi 4.

The research behind PegasOS can be used to get developers up to speed with putting their own projects on the Pi hardware, using the Circle Library, or developing for PegasOS.

Research

Our source-code repository is open to the public. All of our designs, guides, and documentation are available online in a separate GitHub repository.

The work put in to making the system usable for us is as much a resource for ourselves as it is for others wanting to work with the bare-metal of the Pi.

https://github.com/MrJellimann/PegasOS
https://github.com/MrJellimann/PegasOSDocumentation

Challenges

- Raspberry Pi 4’s quirks meant we couldn’t use old or simple drivers for previous Pi’s, and finding non-Linux drivers for the Pi 4 took time.
- Booting off of the Pi previously was an effort in C, converting to Circle meant the entire project had to be rebased in C++.
- Circle is a very large driver library, making sense of it has taken a lot of effort and time away from actual development.

PegasOS Hardware

The Raspberry Pi 4

Specs:

- CPU: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC, 1.5Ghz
- RAM: 2GB/4GB/8GB
- Connectivity: 2.4GHz/5.0GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Codecs/Other: H.265, H264, OpenGL ES 3.0

Ports:

- 2x USB 3.0, 2x USB 2.0
- GPIO 40x pin Header
- 2x Micro-HDMI ports
- 2-lane MIPI DSI display port,
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- Micro-SD card slot for OS and Storage
- USB-C Power (5v, 3A minimum)
- Power-over-Ethernet

Why the Raspberry Pi 4?

Previous Raspberry Pi's were only capable of running 32-bit operating systems, with the exception of the Pi 3.

With the Pi 4 becoming more popular, we want to use this opportunity to increase the 64-bit OS market.

We also wanted to take full advantage of the SBC's hardware capabilities. Instead of just running a 32-bit system in 64-bit space.

64-bit

64-bit systems are not going away any time soon, and for good reason. It can handle more data, both in cache and in memory, and old 32-bit software because it can occupy the same space as before on the processor.

Because PegasOS can be compiled by the user, it can also be compiled for 32-bit if they want a 32-bit system.

Cross-Compiling to ARM

Setting up a cross compiler came with several challenges.

Since there was no clear documentation on the process, it was trial and error to get everyone compiling on their computers.

Most team members ran the cross-compiler through Ubuntu on Windows OS. The process of setting up is very long and confusing, and we have a written guide explaining each step of the process.

The Challenges of New Hardware

When researching the hardware design of the Pi 4, we found it to be not just an upgrade over the previous generation, but an entire redesign as well.

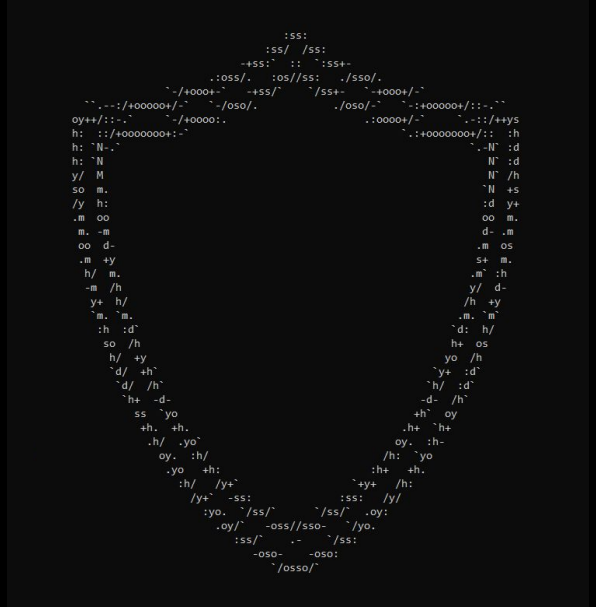
To start, the boot is no longer from bootcode.bin but from files already on the internal EEPROM. Any customization to it must be exclusively done through config.txt.

Furthermore, there is little to no online documentation for Pi 4 hardware due to how new it is.

Hardware Drivers

Whereas the Pi 3’s port could be handled with a simple USB driver, the Pi 4 needed PCI bus, host controller, and several types of usb drivers to communicate in tandem.

As such we decided to source the Circle Library, which provides bare-metal environments for programs to run on. The decision to do this was mostly based on time constraints, as building just one driver on the Pi 4 from scratch is a Senior Design project in of itself.



PegasOS Software

Circle Driver Library

The Circle Library is an open-source C++ driver library for the Raspberry Pi, that comes with a bare-metal environment to interact with those drivers.

It allowed us to quickly get above ground level and start implementing substantial OS code.

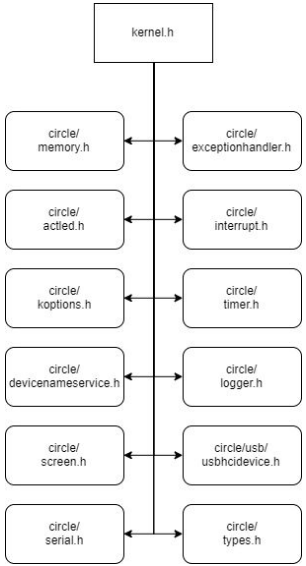
Circle has been maintained for 6 years and we are actively in contact with the creator.

Kernel Design

Our original designs had to be put to the side to work with the framework provided by Circle.

The kernel contains:

- Task Management and Interrupts
- Memory Management through the MMU
- File Management through FAT
- USB Keyboard Drivers



Task Management

As of now, the scheduler is based on a FIFO queue. Each tasks can be programmed to given up control after a certain amount of time.

Every task class is encapsulated by a CTask class, so new types of tasks can be scripted and fed into the scheduler as long as it is a child of the CTask class.

We have also implemented weights to each task. The higher the weight, the farther to the front the task will be added to the queue.

Memory Management

An important part of this OS is for it to communicate with the Memory Management Unit (MMU).

A MMU is a hardware component that is responsible for handling memory and caching operations. This MMU is usually located within the CPU.

MMUs will use a translation table to allow us to translate virtual memory to physical memory.

Memory Management

Utilizing the MMU for our OS will help solve several important issues when a program is running.

1. Required in order for the Raspberry Pi to boot.
2. Allows for programs to use their own private virtual memory space.
3. Lets the OS manage the memory, not the programmer or their program.

File Management

Our original plan for PegasOS was to get Ext2 working as our file system. Some of the benefits it offers for our system includes:

- Its open source (under GNU GPL v2).
- Support of large file sizes (up to 2TB).
- More efficient and less likely to encounter errors when compared to a FAT file system.

However the code-base for Ext2 was more complex than we had anticipated.

File Management

We decided on a FAT variant for our system for its simplicity and compatibility with the storage mediums for the Pi (SD, USB).

We did not make our own implementation for the sake of time, and instead are extending a basic implementation found in the Circle Library.

File Management

The Circle Library also contains exFAT but the Pi 4 cannot boot off of exFAT as it can only boot from FAT32/16 partitions and our system can only use the partition it booted from.

Currently PegasOS uses FAT32 on a single partition of the micro SD card.

PegasOS File System Features

- Open/create files
- Read data from files
- Write data to files
- Expand file sizes
- Shorten file sizes
- Remove files
- Move files to a different directory
- Rename files
- Get size of a file
- Test for errors on a file

- Open/create directories
- Remove directories
- Read items in a directory
- Change current directory
- Check existence of a file or directory
- Change attribute of a file or directory(ex. set to read-only)

Command Line Interface

While GUIs are ubiquitous and intuitive, they are a project in their own right. We needed something that would be feasible given our scope and timeframe.

The traditional Command-Line has a great aesthetic that we like, and it offers a ton of control over the system that can be hard to find even in well-made GUIs.

PegasOS Visual Shell Design

For the visual design of the shell, we stuck with the tried and true look, hence the black screen and white letters on the screen. We also have commands that allow the user to change their user name and directory color to add some customization.

Our stretch goal was to add more features to the CLI to give PegasOS some more flair but due to time constraint we had to omit them.

PegasOS Internal Shell Design

The way we planned on implementing the shell commands was similar to that of Linux. They have separate files for each of their commands and have one main file that calls them.

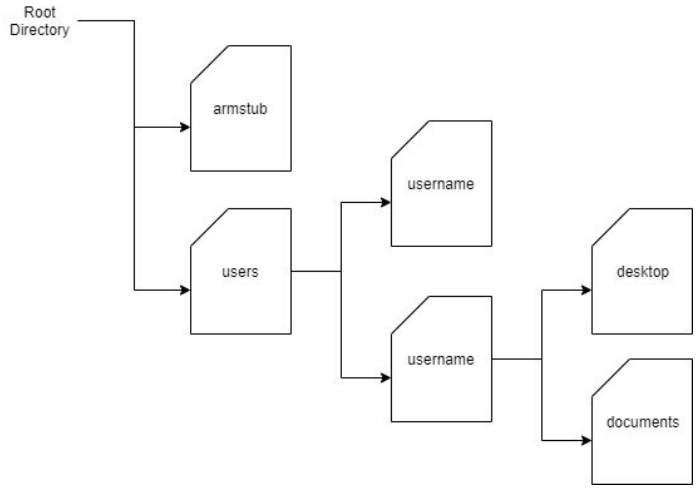
We wanted to do this approach because it would future-proof the shell. Expanding or modifying the available commands would be as simple as adding new binaries to the shell’s command directory.

Unfortunately, we were not able to do so due to certain limitations of Circle.

PegasOS Internal Shell Design

For usage of the system, the user will be interacting with the root directory on the initial boot. The user will then be prompted to sign in via entering their username and password.

If no user is found, they will be asked if they wish to create one, if not then they re-enter their username.



PegasOS Shell Commands

For PegasOS we will be implementing 23 commands that will enable the user to navigate through directories, use the file system, and execute tasks.

The supported commands are:

- changedir
- clear
- copy
- createdir
- createfile
- currenttasks
- delete
- deletedir
- dirtext
- displaytasks
- echo
- head
- hello
- help
- listdir
- memorystats
- move
- power
- reboot
- systeminfo
- tail
- usertext
- writeto

PegasOS Shell Commands

```
chris@RasberryPI:SD:/users/chris$ changedir desktop
chris@RasberryPI:SD:/users/chris/desktop$ createdir myfiles
chris@RasberryPI:SD:/users/chris/desktop$ listdir
Current Working Directory is: SD:/users/chris/desktop
myfiles
chris@RasberryPI:SD:/users/chris/desktop$ _
```

```
chris@RasberryPI:SD:/users/chris$ dirtext 5,5,25
chris@RasberryPI:SD:/users/chris$ help
This is a list of the Commands for PegasOS:
        changedir
        clear
        concat
        copy
        createdir
        createfile
        delete
        deletedir
        dirtext
        displaytasks
        echo
        head
        hello
        help
        listdir
        move
        power
        systeminfo
        tail
        tasklist
        terminatetask
        usertext
chris@RasberryPI:SD:/users/chris$ createfile test.txt
chris@RasberryPI:SD:/users/chris$ listdir
Current Working Directory is: SD:/users/chris
chris.txt         test.txt          documents         desktop

chris@RasberryPI:SD:/users/chris$ _
```

The Future of PegasOS

What's next for PegasOS?

Our scope had to change as R&D progressed, so not everything that we hoped for is in the current system.

However this means that PegasOS has lots of room for expansion and extra features. With a solid foundation we hope that it can serve as a great jumping off point for future projects, both in and outside of PegasOS.

What’s next for PegasOS?

Future Feature Ideas:

- A true GUI interface
- System Encryption
- System Networking
- PegasOS Drivers

PegasOS