PegasOS Design Document

Fall 2020

PegasOS Team

Group 23

File System, Software
Kenny Alvarez

Task System, Scheduler, Hardware, Software
Jacqueline Godier

Command Line/Shell, Software
Giancarlo Guillen

Memory Management, Hardware, Software
Jacob Thomas

Project Lead/Sponsor, Software, Circle Documentation
Christopher Walen

0.1.0 Table of Contents

0.1.1 Preface

The purpose of separating the design document into two separate documents is two-fold: one is to preserve the nature of the original design document and its intentions for PegasOS, and two is to contain and manage the changes that have been made in design and development since the creation of the original design document.

This document will serve as both an addendum to the original document, as well as its own document regarding the current state of the system as of the end of the Fall 2020 semester. As we are sure you can imagine, the scope of the original project was… large. Because it was rather large, we have not been able to achieve some of the finer details that were discussed in the Spring 2020 design document. However we feel that despite that, we've achieved our goal in creating a new operating system for the Raspberry Pi.

For the original design document as well as all other documentation and research created over the course of this project, go here:

https://github.com/MrJellimann/PegasOSDocumentation

For the project source code and release, go here:

https://github.com/MrJellimann/PegasOS

0.1.2 Overview

This document describes the changes and new decisions that affect the design and implementation of the final system. Here, we'll briefly discuss these changes by breaking them down into more manageable sections that can be explored at will. You can also use the Table of Contents to jump to the section of your choosing.

By far the biggest change was the adoption of the Circle Driver Library, made by rsta2 on GitHub, for interfacing with the hardware components on the Raspberry Pi. At the end of the Spring 2020 semester we had achieved a bare-metal boot environment on the Pi 4, and we then proceeded to tackle our next problem which would be keyboard input. This was not a problem that we could solve ourselves - not in the given time frame at the very least. In our search for open-source drivers, we found the Circle library which contains not just keyboard drivers, but gamepad drivers, sound drivers, other USB device drivers, and much more. Adopting Circle early-on in development saved us a lot of time in the later months, but it has changed the nature of the project quite substantially. We will discuss the features and capabilities of Circle, as well as how Circle is being used in PegasOS in the first major section: Circle x PegasOS.

We have underestimated the time that components on the project would take to develop, and as such we've fallen behind when it comes to the more ambitious features of PegasOS. While not all features have been cut, some have been reworked to better accommodate the new structure of the system. Though as development will be continuing until the deadline, assume that the cut and modified feature list is not final unless stated otherwise. We will be discussing these cut and modified features, and what the plans are for dealing with these features in the second major section: PegasOS Cuts and Edits.

After discussing the cut and modified features and components of PegasOS, we can discuss the features that were preserved and have been implemented in the final system. Some details will be sacrificed here, though if you wish to have more detail on a particular component that information can be found in the Spring 2020 Design Document for PegasOS. This section will outline the implemented features of PegasOS, and describe the current capabilities of the system as of Fall 2020 in the third major section: PegasOS First Release.

As the project of PegasOS has become part-development and part-research, it also has potential for expansion for future senior design projects as well as for educational purposes in systems development (which was an additional benefit discussed in the original document). This section will discuss the research-side of the project and the

project's utility and potential in other projects in the fourth major section: PegasOS Future.

The final section will contain some administrative information, such as the final budget of the project and legal information. This section will also contain the closing remarks from the members of the team. This section will be called: PegasOS Team.

1.0.0 Circle x PegasOS

As was alluded to in the Overview, the adoption of Circle for the PegasOS project resulted in a number of changes that affected the structure of the entire project, forcing us to rebase the project's code from C to C++. Most, if not all, of the original code was scrapped in the process. The specific nature of the changes will be discussed later in this section, but the beginning of this section will be used to describe what Circle is, what Circle does for PegasOS, and a brief overview of its features.

1.1.0 Circle

"Circle is a C++ bare metal programming environment for the Raspberry Pi. It should be usable on all existing models (tested on model A+, B, B+, on Raspberry Pi 2, 3, 4 and on Raspberry Pi Zero). It provides several ready-tested C++ classes and add-on libraries, which can be used to control different hardware features of the Raspberry Pi. Together with Circle there are delivered several sample programs, which demonstrate the use of its classes. Circle can be used to create 32-bit or 64-bit bare metal applications." - Rene Stange, Circle GitHub

Most of the discussion about Circle will revolve around Release 42 (aka Step 42) which is the release of Circle that PegasOS is being developed with. Since the adoption of Circle by PegasOS, Circle has moved on to Release 43.1.

As the author stated, Circle is a bare metal programming environment on the Raspberry Pi, though this really sells it short. Circle's programming environment contains a variety of components that make up a basic real-time system on the Pi, that is not reliant on another kernel or other drivers. This is quite significant, as it is not beholden to the Linux drivers or kernel, though this means that working with the Circle environment means doing it the "Circle way".

Circle contains its own kernel, and the Circle system can have a scheduler component, memory management, file system, and any combination of hardware drivers for both USB and on-board hardware. Some of these systems have limitations that we will discuss shortly.

1.1.1 Circle File Systems

Circle contains support for the FAT-family of file systems, from FAT12 all the way up to exFAT in an addon. We have attempted to get the exFAT addon to work with PegasOS, however this has proven to be extremely difficult and has not cooperated well so far. As such we are using Circle's implementation of FAT32, which means it comes with all of

the standard operations you come to expect from a file system, with the traditional memory limits of FAT32.

1.1.2 Circle Memory Management

Circle contains definitions and infrastructure for memory paging like in a real operating system, however after confirming with the author we know that Circle does not contain true virtualization of the addressable space. Instead, because the ARM CPU requires a memory management unit to be instantiated and running on the chip in order to continue booting, Circle fulfils the requirement but then allows the system and its tasks to use a 1-to-1 mapping of the given page, and allocates the page accordingly.

As far as we can tell, a single page is allocated(?) for the memory management system. For a traditional or even a modern OS this would be incredibly small, however Circle is using 64KB pages which makes sense given the original intent is to provide an environment for bare metal and by extension real-time applications. These are not applications that are typically bogged down with the frequent swapping in and out of hard disk space that OS's must put up with, and these applications do not typically have entire programs or at the very least not large programs that require multiple pages of memory to run. This is something that we are obviously addressing in PegasOS, as we need a proper page-swapping system in place with ideally typical OS page sizes of around 4KB to make the OS more standardized.

On that note however, Circle does include different functionality for its memory system based on whether you are building for a 32-bit application or a 64-bit application. 64-bit applications unlock the 3rd level of the page tables, whereas 32-bit applications may only utilize up to the 2nd level of page tables.

1.1.3 Circle Scheduler

Circle's scheduler is a simple First-In-First-Out Queue that uses the Circle CTask class to execute tasks sent to the scheduler for execution. The CTask class is quite powerful, in that any class can be run as a task on the system if it is a child of the CTask class. As such, CTask dictates how tasks are actually run on the system itself.

Since the scheduler has no weighting system or any real methods to prevent tasks from being starved in extreme conditions, the scheduler is lacking 'modern sensibility' though as we said for the Memory Management system, this sophistication is not really required for Circle's intended purposes.

This is one of the parts of Circle that we are changing drastically for PegasOS, as for PegasOS to be closer to a true OS we need to have scheduling that can handle a greater variety of tasks and task volume.

1.1.4 Circle USB Drivers

Circle includes a large number of USB drivers for various devices, ensuring that Circle has great compatibility with many generic USB device types. Some of these types include:

- USB Bluetooth
- USB Ethernet
- USB Gamepads
- USB Mass Storage Devices
- USB Midi Devices
- USB Mouse
- USB Printer

Using these drivers is fairly straightforward; include the desired USB device header in the kernel and begin using the functions of the device driver.

1.1.5 Other Circle Features and Components

Due to the nature of Circle being a standalone bare-metal environment, some of the benefits of things like the Standard Library are harder to come by without the appropriate system calls that simply do not exist on a bare-metal system. As such, Circle actually includes definitions for additional data types, as well as a number of string manipulation functions that we've come to expect to be able to use like *strcpy()* or *strcmp()* for example. Circle also has its own implementation of *malloc()* which allows for the typical dynamic memory allocation seen in C/C++. Circle also supports variable arguments for function calls, spinlocking cores, and much more.

Circle also includes additional drivers and system components in its included */addon* directory.

1.2.0 Circle in PegasOS

Circle provided the groundwork for us to be able to work on the system at large, instead of spending a long amount of time on developing our own hardware drivers for the given devices. While Circle does include a basic environment to get code running on the Pi, the environment itself is as minimal as possible to ensure that it contributes as little overhead as possible. Circle even includes a template for a blank kernel so that the

developer(s) building off of Circle can take exactly what they want for their needs. Our needs are much larger in scope than Circle anticipates, as such we have replaced components where necessary and extended others to increase their capability.

1.2.1 PegasOS Usage of Circle

PegasOS only takes a few things of importance from Circle, apart from the drivers needed to run the hardware on the system. These are the major components, as small functions and other pieces of code or structure are taken from classes, components, or paradigms from within Circle.

The first of these is the *kernel structure*. Using the structure of the sample kernels that Circle has given us, we have based our kernel design around the main functionality that Circle expects from its kernels, and have extended the capabilities and functionality of the kernel from there.

The second being the Circle *Scheduler*. Circle's scheduler on its own is rather basic - it operates on a simple queue that switches tasks every few seconds to the next one. We have taken this scheduler as the base and extended it with weights to allow certain tasks to have priority over others. Current tasks can also be viewed via command line.

The third being Circle's *Memory Management Unit*. Circle's MMU is quite bare-bones on its own - its purpose within Circle is quite literally to appease the ARM CPU at the heart of the Pi, and from there Circle effectively uses a one-to-one memory mapping onto the actual memory space. PegasOS's modifications allow for this MMU to use actual virtualization on this memory, so that multiple programs can run on the system and have access to their own space in memory - much like any other modern OS.

The fourth being Circle's implementation of *FAT32*. Circle includes an implementation of FAT32 that works on the SD card that Circle is on, and allows for real-time access of files as one would expect from FAT32. PegasOS leaves the implementation of Circle's FAT32 basically unchanged, however the user is able to interact with FAT32 through the shell or CLI.

The fifth, and perhaps one of the most important was the USB keyboard driver. Without that, it would be very hard to use anything on the system, for obvious reasons! As such we've allowed the USB keyboard drivers to act exactly how they would expect to within Circle's environment, taking care not to edit the driver code itself nor modify any internal settings to the driver (that cannot obviously be reverted or damage the system's integrity at run time).

1.2.2 PegasOS Original Components

The major component that was built from scratch was PegasOS's shell, or its Command Line Interface (CLI). This allows the user to interact with the system through a number of built-in commands, of which are discussed in Section 3.1.0 of this document. This is also the primary method in which the user is able to interact with data on the system through the file system, FAT32.

PegasOS's shell connects with Circle through a number of internal classes and functions from within the kernel and its dependencies to ensure that writing to the screen is done as intended, and that there is minimal delay between writing and reading input from the user's keyboard. PegasOS also makes occasional use of Circle's logger for some shell commands.

2.0.0 PegasOS Cuts and Edits

A number of aspects in PegasOS had to be changed due to time constraints, largely due to the ambitious nature of the original requirements. As this is an entirely student-driven project, none of us were fully aware of the exact time frames each component within the system would require - and neither could we know.

Time was spent in getting a completely bare-bones, driverless system off the ground as our original intention was to create the keyboard driver ourselves so that the system was all new code purpose-made for our system. Quickly this was abandoned, and we searched for open-source keyboard drivers that worked with the desired platform - the Pi 4 - that were not dependent on other systems or architectures. This again had to be abandoned, as no drivers that fit this bill existed. It was through this process that Circle was discovered, and the decision was made to rebase the project in C++ and around the structure of Circle.

This final restructuring has resulted in a number of planned features being either modified or removed due to time constraints. As such, we will discuss the cut features first, and the modified features second.

2.1.0 PegasOS Component Cuts

The cut features are any requirements and/or stretch goals mentioned in the original document that have been removed in scope and/or requirement and/or stretch goals from the final project. The following features have been removed entirely from the project:

2.1.1. PegasOS Miscellaneous Programs

Pretty much every single miscellaneous program was cut from the project. These were all of the ideas under *Christopher's Ideas, Kenny's Ideas, Jacqueline's Ideas, Jacob's Ideas* and *Giancarlo's Ideas*. If a program was saved and made its way into the shell or otherwise into the first release of PegasOS, it will appear in the *PegasOS First Release* section.

2.1.2 PegasOS System Configuration

By System Configuration, we namely mean System Localization and Location. While some of this can be edited through Circle's keyboard configuration settings and timezone settings, PegasOS itself does not have the previously described Language packs or Location settings for time zones.

2.1.3 PegasOS Program Execution

Unfortunately, we were not able to determine a way to load and execute binaries from within PegasOS with the time we had. Based on what we know about Circle and how PegasOS has developed, we believe that a system rework of the task system and scheduler would be necessary in order to facilitate the execution of program binaries.

However, there is still a way to get custom programs to execute on the system. This consists of four steps:

1. Place the program's header file in the /include/pegasos directory of the project.
2. Place the program's source code file in the /lib/pegasos directory, and add its name followed by the extension *.o* to the Makefile in that directory. Like so:

lib > pegasos > M Makefile

```
#
# Makefile
#
# Here's how to use the makefile:
#     You're going to put the names of any .cpp files on the OBJS line, but with the .o extension
#     If the line gets super long, you break it with a '\' then continue on the next line.
#
# Uncomment the rest of the contents once you're ready to start having your stuff compiled.
# And please... once you have something useful, get rid of 'something.cpp' and 'something.h'.
# On that note, kernel.cpp does have an include for 'something.h' to prove that the convention works.

CIRCLEHOME = ../..

OBJS    = something.o shell.o   # your addition here!

libpegasos.a: $(OBJS)
    @echo "  AR    $@"
    @rm -f $@
    @$(AR) cr $@ $(OBJS)

include $(CIRCLEHOME)/Rules.mk

-include $(DEPS)

```

3. Create a command to execute the program within the PegasOS shell, inside of shell.cpp in /lib/pegasos
4. Compile PegasOS and run on your device.

2.1.4 PegasOS Drive Partitions

While this was not originally a design choice, after achieving a working file system with FAT32 we wanted to allow the system to partition the SD card and still be able to boot off of one of those partitions. This would have also allowed us to use a FAT32 boot partition then have a exFAT data partition on the drive. While we still believe this would be possible on the current system, this is something that we had to cut due to time constraints where the effort was better spent elsewhere.

2.2.0 PegasOS Component Edits

Edited features are any requirements and/or stretch goals and/or components of PegasOS that were present in the original design, that have since been changed and no longer reflect in whole or in part their original design. The following features have changed in function and/or scope in the project:

2.2.1 PegasOS Scheduler

While PegasOS still contains a scheduler, it is not reflective of the original design pattern, namely in its algorithm. A multi-level feedback queue was not feasible in the time frame after the Circle rebase occurred, and as such we began with a much simpler, straightforward scheduler. We increased its scheduling complexity as time allowed through adding task weights and insertion into the task queue by those weights, but we were not able to achieve the complexity and flexibility of the original design.

Related to the scheduler, we were also not able to have custom binaries executable from within PegasOS. We are fairly certain that this would require extensive reworking of the system due to the reliance on the CTask class for executing tasks through the scheduler. It is possible however, to include custom programs as a part of PegasOS and compile them with the system and the kernel. For more information on this process, refer to the guide on the Documentation repository here.

2.2.2 PegasOS Memory

While PegasOS still contains a memory system and through Circle we do have paging support, the pages are not ‘traditionally OS-like’ nor does it provide any sort of real virtualization of memory. This is elaborated on more in Section 4.2.5, and provides suggestions for future groups that tackle the problem.

3.0.0 PegasOS First Release

The first public release of PegasOS will be released at the end of the Fall 2020 Semester, which means that the first public release of PegasOS will be available on Wednesday, December 2nd, 2020. The link to this release is here. The original PegasOS Team may or may not continue work on PegasOS past this point, though this is not guaranteed. This point will be discussed further in *PegasOS Future*.

We will break down the features of this release into four main components: the Command Line Interface or Shell, the File System, the Task System, and the Memory Management System.

3.1.0 PegasOS Shell

PegasOS comes with a Command Line Interface, which we will call the Shell from here on. It is made up of two main components: the commands that can be executed on the system, and the user system that allows for users to have their own space on the system.

3.1.1 PegasOS Shell Commands

The PegasOS Shell on release contains 24 commands, as given here including the appropriate arguments for each one:

changedir
changedir <path>
Will change the current directory to the specified parameter given.

clear
clear
Clears the screen of all previous input/output.

copy
copy <file> <directory>
Creates an identical copy of the specified file to the specified location given.

createdir
createdir <directoryname>
Creates a subdirectory within the current directory.

createfile
createfile <filename>

Creates a file with the given name and extension in the current directory.

currenttasks
cutrenttasks
Shows a list of all currently active tasks in the scheduler.

delete
delete <file>
Deletes the specified file given.

deletedir
deletedir <directory>
Deletes the specified subdirectory given.

displaytasks
displaytasks
Displays the scheduler demo. To stop displaying tasks enter ‘s’ and return.

dirtext
dirtext <num1,num2,num3>
Will change the directory color to the specified (x,y,z) values between 0-31 for Red, Green, and Blue respectively. E.g. *dirtext 9,13,24*

echo
echo <text>
Repeats the inputted text, surrounding it with “Echo” and “Echo!”

head
head <file>
head <file> <n>
Prints out the first ‘n’ given lines of the specified file within the current directory. If no value for ‘n’ is not given, then it will simply print out the first 5 lines.

hello
hello
The system responds to the user and says hello.

help
help

Lists all available commands in the system. (With a more mature system this will not be feasible).

listdir
listdir

Displays the path to the current and the current working directory.

memorystats
memorystats

Displays the amount of free space on the low, high, and combined heap and how many memory blocks the system has.

move
move <file> <directory>

Moves the specified file given to the directory specified.

power
power

Powers off the system. This will occur immediately after this command is entered.

reboot
reboot

Reboots the system immediately after this command is entered.

systeminfo
systeminfo

Displays the information for the separate components of the Pi.

tail
tail <file>
tail <file> <n>

Prints out the last ‘n’ given lines of the specified file within the current directory. If no value for ‘n’ is not given, then it will simply print out the last 5 lines.

usertext
usertext <num1,num2,num3>

Will change the user color to the specified (x,y,z) values between 0-31 for Red, Green, and Blue respectively. E.g. *usertext 12,22,30*

writeto
writeto <file> <string>

Will add the given string to the *end* of the given file.

3.1.2 PegasOS Users

A user account is required to use the system. When PegasOS boots up, they will be prompted with the log-in screen. Here they will enter their username and password. Once the password is entered, it will be compared to the password stored in the user file corresponding to the user name. If they match, the user will be granted access. Otherwise, they will have to try again.

Welcome to PegasOS!
Please log in to continue…
Username:
Password:

If a username is entered for a user that does not exist, they will be prompted with the following text. This prompt will also be displayed if there are no users on the system at all:

User does not exist. Would you like to create one? [Y]es/[N]o
Enter desired Username:
Username was created. Now please enter a password:

If the password was not entered correctly, the user will be able to attempt to enter their password again.

Password: Password mismatch.
Re-enter Password:

Once the user has entered their username and password correctly, or created their user account, they will be shown the following prompt and may begin using the system.

Successfully logged in!
user@RaspberryPi:SD:$ _

3.2.0 PegasOS File System

PegasOS uses a FAT32 system, as provided in Circle. While FAT32 is basic by today's standards, it is not lacking in any of the typical features one would expect from a modern file system. PegasOS's file system supports the following features:

Create Files

The user can create files of any type, the type being the file extension. A user may even create a file without an extension. The files can then be opened and written to or read from.

Create Directories

The user may create directories wherever they wish, as long as the device has enough storage.

Move Files

The user may move files between directories, wherever they wish (with the appropriate permissions).

Move Directories

The user can move a directory from one location on the disk to another (i.e. move a subdirectory from one directory to another).

Open Files

The user can open files for reading or writing purposes.

Navigate Directories

The user may change the active or working directory, so that new commands are begun from the new active or working directory.

Delete Files

The user can delete files (with the appropriate permissions).

Delete Directories

The user can delete directories (with the appropriate permissions).

Mount Storage Devices*

The user can mount different/additional storage devices.

*This is technically true, though this is primarily the SD card that the system is contained on.

3.3.0 PegasOS Task System

The task system for PegasOS is fairly straightforward. There is a main task within the kernel that starts the tasks, and each task runs through its allotted time as long as it is instantiated and a member of the CTask class. The main task running is the shell. It is made in CKernel::Run() and sets the speed at which tasks have control of the system. Each task is instantiated within CKernel::Run() as well, but the management of the tasks goes to scheduler.cpp.

CTask is the parent class to all tasks in the system. It allows for quite a bit of customization, as you can create tasks with different arguments (LED tasks, screen tasks, etc.). Any task created from CTask gets fed into an array in the scheduler, which gets processed through any function the scheduler may need. Each task also has an initial weight of zero, which is the lowest priority. Tasks that have higher weights will get inserted farther ahead in the array. Currently there is no ceiling limit to the weight of a task.

As of now tasks cannot be removed via the command line, as this proved to be catastrophic to the system (though this could be addressed by another team in the future). However you can view all running tasks in the system and display them in real time. The task demo command *displaytasks* also showcases four screen tasks running on the system to demonstrate its multi-tasking abilities.

Creating new tasks on the system is as simple as creating a class that is a child of CTask, and instantiating it. This could then be tied to commands on the shell for specific system tasks and/or components.

3.4.0 PegasOS Memory Management System

PegasOS essentially uses the memory system given through Circle, which consists of several key features one would want in a memory system for an OS. The memory system has support for paging on a flat memory space through a two-level page table for 32-bit systems and a three-level-max page table for 64-bit system. Page sizes are configurable through the header files for the memory system. One drawback of the system however is that it provides no virtualization for programs - it is essentially a flat memory space of literal addresses within the page. The default page size of Circle (and therefore PegasOS) is 64KB. The memory system also supports dynamic heap allocation on a flat memory space, using heap-low for all Pi models and heap-high for Pi 4 and newer models.

3.5.0 First Release Known Issues

- The ‘changedir’ command cannot move up and down levels in one command. For example, if you are in your /desktop directory and wish to move to your /documents directory, you must enter two commands: *changedir ../* and *changedir documents*. However, if you try to enter *changedir ../documents* you will not move any directories at all. This is due to a parsing issue with the directory string that could not be resolved in time. Interestingly, this issue does not apply to the other directory commands, ‘createdir’ and ‘deletedir’.

- The ‘changedir’ command will not remove instances of ‘./’ in the directory path. If the user enters the command *changedir .* then their directory path will look like the following:

  user@RaspberryPI:SD:/users/user/desktop/./

  This does not have any effect on moving up directory levels, for example if the user was in their root directory */users/user/* and ended up with a ‘./’ in their directory path, they would still be able to enter their /desktop and /documents directories correctly. It will however take multiple entries to move down directories with ‘..’ to clear out every occurrence of ‘./’.

- When logging in to the system, if you enter a correct username the system will not let you continue until you enter their password correctly. The system will repeatedly prompt you to re-enter their password until it is entered correctly or the machine is rebooted. This is due to the lack of a maximum attempts catch on failed log-ins. To switch accounts or create a new one from this state, you must reboot the machine.

- Backspacing after reaching the beginning of the command input will erase characters from the screen. This does not have an actual effect on the input string for commands, it is simply a visual bug that is partially due to the Screen device’s separation from the command shell.

- Shell commands can only accept two parameters, internally known as *_commandParameterOne* and *_commandParameterTwo*. Any delimiters after the second will count towards *_commandParameterTwo*. For example:

  writeto myfile.txt Some text for my file.

  Will be a valid command and write ‘Some text for my file.’ to ‘myfile.txt’. However, this:

  move myfile.txt .. otherparam

Will be an invalid command, not because it *should* have three parameters, but because '.. otherparam' is the second parameter and is not a valid directory to move 'myfile.txt' into.

- The command can be overflowed and leave the system unresponsive. The typical layout of the input string is as follows:

  <*command*> <*param1*> <*param2*>

  However, if the <command> section of the input string, i.e. if everything before the first space exceeds the buffer for the command string, the system will freeze and require a reboot.

- Overflowing the string parameter of the 'writeto' command can overwrite parts of the file extension for a new file creation, and the string written to the file will not be the entirety of what was entered. Furthermore, the end of the inputted string will have *writeto* appended to it.

  If this is attempted with a pre-existing file, the name and extension of the file will not be affected. However the string written to the file will be the same as in the case above.

- After entering 'displaytasks' and running the command, other commands can be run before the cancel command 's' has been entered. This means that the screen tasks in the scheduler demo will continue to print to the screen while the user attempts to use the system. This may be problematic as it will not take long for the prompt about the 's' command to disappear off the screen.

4.0.0 PegasOS Future

Now that our time with the project is done, and the semester comes to a close and we all graduate and move on - what's going to happen to PegasOS? Well, we'd certainly not like the project to die, as we believe that the project itself has a lot of potential now that we've "taken care of the hard part". We have a bare-bones system that boots off of the Pi, and has a number of drivers that allows keyboards and the like to be used by the system. This means that where the system goes from here is potentially limitless. We'd like to use this section to suggest some of the places that PegasOS could go from here, that might spawn future Senior Design projects, or that might even motivate non-UCF students to contribute to the project and bring it that much closer to being a fully fledged operating system.

4.1.0 UCF Projects

Because this project was a UCF CS Senior Design project, naturally this project can be used for future projects or over multiple semesters and classes of students. In this regard, there are many different aspects of the system to focus on, however these could apply to non-UCF students as well if they wanted to extend the system on their own time. For UCF students, note that the following suggestions apply to you too!

4.1.1 Operating Systems Classes

Whether at UCF or beyond, we think that this project could serve as an excellent basis for teaching students new to the world of operating systems how a real operating system works, and what components are inside even a bare-bones operating system. Much like how *minix* works for education purposes, we believe that with all of the documentation that we have for the project and the hardware, PegasOS will make a great resource for those learning about operating systems.

Much like what we told those looking at the UCF Projects section before, the following suggestions could (in smaller scopes/assignments) be used for operating system curriculum, to give students hands-on experience modifying and/or extending a system's components that are crucial to running an operating system.

4.2.0 PegasOS Drivers

This was originally intended, however for reasons we've discussed before this was not possible. To make PegasOS truly something unique, it could bring brand-new open-source drivers to the table. This would serve two purposes - one to bring those drivers to PegasOS so that PegasOS can either extend its Circle base or stand on its own entirely.

The first option would simply need to select a number of devices, preferably a family of devices such that there's sufficient work to be done, then create the appropriate interfaces for those devices to be used with Circle and PegasOS. The important thing here would be to follow the existing structure of code so that the devices can be added and removed easily, much like what Circle does with its */addon* folder. We would suggest a new directory just for these drivers, which can then be included or removed as a *libdrivers.a* file in the compilation process.

The latter option would require porting/restructuring the system to work without Circle, which it is currently based around both in code and in structure. A careful examination of how the system works would need to be performed before pursuing a standalone PegasOS system. All that PegasOS really *needs* is a keyboard driver. However, on the Pi 4 there are other components of the hardware to consider, such as the USB and PCI busses, the Host Controller, the ARM-specific quirks of the chip (i.e. how will you spin up the MMU without Circle's aid?) It is safe to say this is the harder of the two options.

4.2.1 PegasOS Security

While the system should be secure by its isolated nature, some may still be concerned that it is lacking in the security department. While we have done our best to ensure that users have at least a minimum level of security (such that users cannot find another's password and login as that user), there are other measures that could be taken to ensure that the system is secure. There are at least two ways that system security can be increased: one is for user security to be robust so as to protect user data and accounts and be fast in doing so, and two is for the system to be as stable and resistant to attacks on its integrity as possible.

The first aspect of this is perhaps the easiest of the two; ensure that encryption of user files is secure and cannot easily or realistically be undone - unless done so by the owner of those files. User passwords should never be able to be unencrypted - only strengthen the encryption algorithm for user passwords (i.e. it should remain a one-way encryption). In addition to this, users should be able to encrypt and/or password protect files as they wish. This may be limited in functionality by the FAT32 system that is in [this release of] PegasOS.

The second aspect will require deep analysis of potential weak points not just in PegasOS, but Circle as well. We encountered a bug early-on in the bare-metal development of the shell that showed us that positions in memory could be overwritten with long-enough strings in certain commands - in other words a buffer overflow was present in the code. If a weakness like this existed in PegasOS, there must have been

other weaknesses that we did not have time to discover and deal with as our primary focus was on simply finishing the system. But, to leave you with some questions to help you on your way:

> How could a user take over this system? As a legitimate user? As a non-user?
> Can I escalate my privileges to become an admin on the system?
> Can I edit files that do not belong to me when I am not the admin?
> Can I edit permissions of files that do not belong to me when I am not the admin?
> Can I destroy the system in any way? Can I make it unstable and require a restart or reinstall?

4.2.2 PegasOS Networking

PegasOS on its own does not support networking… however Circle *does* in fact contain some networking and bluetooth drivers. While we have not explored either of these avenues, there is potential for an involved project in getting functional networking working with PegasOS. However, this opens cans of worms that have to be dealt with in terms of security. There is some benefit in knowing that most malicious agents will not know about PegasOS and would not know how to perform attacks on a new system such as PegasOS - however that does not mean that PegasOS is completely secure or invulnerable. In addition to networking PegasOS, the system's security would have to be increased to a degree, though potentially it would not have to be involved as we've suggested in the previous subsection.

4.2.3 PegasOS GUI

While this was never a possibility for us (it became a bit of a meme, really) there is potential for another team to create a functional GUI for PegasOS. As we have not explored this avenue at all, we cannot really offer you any support, only good luck.

4.2.4 PegasOS Modern File System

PegasOS does have a working file system, however it's fairly ancient by modern standards. We think there are two ways to go about this:

The first would be to ditch modern file systems altogether and create a new file system specifically for PegasOS that has all of those modern features that exFAT and Ext4 have. To make something as robust as exFAT or even NTFS may be quite involved, though we cannot really offer any insight into this. Creating something as basic as FAT32 would not be terribly involved, our minimal research into custom file systems revealed as much. However, ditching standard or modern file systems means that there will need to be translation between the formatting of the SD card, or the SD card will

need to be formatted in such a way that PegasOS can read it, and non-PegasOS computers can read from it/write to it.

The second would be to get exFAT or another file system (newer and more expansive than FAT32) working with PegasOS. In addition to this, giving PegasOS proper partition support would increase the utility of the system and modernize it that much more.

4.2.5 PegasOS Memory Virtualization and Page Swapping

While PegasOS does have a memory management system, it is effectively just using the systems provided by Circle as is. We attempted to get virtualization and proper OS page usage working with the system, however we were not able to achieve this with the time that we had left. The next steps for the system would be:

1. Using a translation table, virtualize page addresses so that programs can use each page as standard memory that they would expect.
2. Creating functionality to swap pages on the fly (as we believe that Circle does not have this capability) for programs to be able to load the pages that they have allocated to them when needed.
3. After steps 1 and 2 are complete, figure out a smaller page size to use for the system that is still effective and allows the system to work as intended. Our initial tests with lower page sizes had some unexpected behavior, and we believe that these first two steps will help iron out those problems.

5.0.0 PegasOS Team

Apart from wrapping up final administrative content, we'd like to use this section for a few things: for one, we'd like to use it as a thank-you wall for the people in and outside of this project that made it all possible. For two, we'd like to use it to tell you a little bit about ourselves, discuss how we felt about the project while developing and researching it, how we feel about the final state of the project, and what our plans are for the future.

As the project lead and sponsor, I can say that it's been a real pleasure working with all of you on the PegasOS team. This certainly wasn't an easy project, and even more so was this not an easy year! I'm sure that none of us will be able to forget 2020, and I know that for me at least, the time spent on this project with you all - through all the meetings and restarting and the endless amounts of documentation that we've been writing - I've enjoyed all of it and you all have been great to work with. I wish you the best in your future endeavors, and I hope that this project gave you valuable experience that you can take with you wherever you go. And I hope I wasn't too bad of a project lead, hehe. Thanks again you guys, you're awesome.

- Christopher Walen

5.1.0 Administrative Content

There is not much to discuss here, so this section will remain short.

The total budget for PegasOS did not change from last semester, which has been included here for completeness.

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

PegasOS still uses the GNU General Public License version 3 and later. To save space on this document, this license has been included in its entirety in the Spring 2020 Design Document in the *Appendices* section. A link to this license is also included here.

As the project has become half-development and half-research, we have made an effort to make as much documentation available and that it is easily navigated on our documentation GitHub, which is linked here. The source code for the OS is included here. The release of the project that coincides with this version of the documentation and is the submission of the project for Fall 2020 is here.

The documentation includes all original designs, plans, and research from the Spring semester, summer, and Fall semester. As it should be plain to see, these original designs do not reflect the entirety of the system in its current state, which is part of the reason why this is a separate document (as we discussed in Sections 0.1.1 and 0.1.2).

5.2.0 Closing Remarks

The following sections are for each member of the PegasOS Team to write a little bit about themselves, their time with the project, what their plans are for the future, and if they'd like to thank anybody that helped them along the way.

5.2.1 Kenny Alvarez

Thank You's

I'd like to thank our leader Chris for pitching this project, for me it was the only interesting project that I saw as it was mostly web development and simulations on unity that were being presented. I think I learned a lot about system software and would like to get into more of it after I graduate. I would also like to thank my parents for supporting me for the last 2 years at UCF and the rest of the group for just being a great group of people to work with on this project.

Who Am I?

I'm Kenny Alvarez, I'm about to graduate in CS in December 2020. I was always into computers and creating games ever since I was a little kid but my parents at the time couldn't afford a computer for me so I couldn't get into it as much as I wished to.

What did you think of PegasOS? As a project? The final result?

I think that PegasOS was a difficult project that I was glad I was a part of and that it was great that we had a working OS. I knew nothing about making an OS so I thought it wouldn't be that difficult to learn how to make one but I was wrong, just trying to know what to do was already a difficult enough task so the fact we have a working OS was a great feeling.

What's next for you?
I don't exactly have that much planned for after I graduate. I original plan was to get into game development and try to work in a studio but after reading about that the job market is on the bad side, it is very competitive compared to other industries, and the working conditions for many studios are not good (seems like working 60+ hours a week is expected for most studios) so that turned me away from doing full-time game development so I'll make games as side projects. After I'm done with classes I will be doing some personal projects to learn more skills while applying for software development jobs.

5.2.2 Jacqueline Godier

Thank You's
I'd like to thank Chris Walen for pitching such an interesting topic. I feel like I've learned a lot from this experience that I can apply to future endeavors.

Who Am I?
I'm Jacqui, a Computer Science major at UCF. I have a lot of experience in game development, but have really wanted to branch into more low level topics. I also have an interest in Cyber Security, which I hope to learn more about after I commission into the USAF.

What did you think of PegasOS? As a project? The final result?
I thought PegasOS was a great way to challenge myself in a subject I had been interested in for a long time. It was always a huge undertaking, but communicating with the team was always very easy and the progress we made always felt rewarding.

What's next for you?
I will be working fulltime as a software developer as I wait to enter the USAF.

5.2.3 Giancarlo Guillen

Thank You's
I'd like to first say thank you to Chris. Chris pitched the perfect project that showed that modern CS projects are not all about Web Development, so when he presented his idea to the class I knew I wanted to be a part of the team. I would also like to thank him for allowing us to work on this project and for him being open to our ideas. I'd also like to say thanks to my friends and family for helping me with support over these past four and a half years.

Who Am I?

My name's Giancarlo, I am a graduating computer science major. I've always been intrigued by the background work of software and programs. I came to UCF to learn the skills and concepts to one day create innovative projects.

What did you think of PegasOS? As a project? The final result?
I thought this was an amazing opportunity to be a part of creating the groundwork for future projects to come. I hope other senior design groups are able to contribute and make it a sound operating system in the future. I enjoyed the ups and downs of this project, because it gave me experience in the low levels of computer science. It really opened my eyes to how something can seem so simple but actually takes a lot of hard work to get it up and running.
I am very proud of my team and I for pouring our blood, sweat and tears to create PegasOS, I am overjoyed with the end result.

What's next for you?
As a graduating student I hope that I can find opportunities that will help me expand what I have learned thus far. I'm currently applying to internships and jobs in the field.

5.2.4 Jacob Thomas

Thank You's
I would like to take this opportunity to thank everyone on the team. You guys are definitely one of the best groups I ever had. It's been a pleasure working with all of you and I wish you guys the best of luck.

Who Am I?
Hello! I'm Jacob Thomas and I'm a student who is graduating this December with a bachelor's in computer science. Ever since I graduated high school, I always had an interest in computer programming. It is one of the reasons why I came to UCF. To help develop my skills further and to figure out my career.

What did you think of PegasOS? As a project? The final result?
When starting this project, I knew it was going to be a challenging project. When we were starting to prepare for this project did it make me realize how enormous of a project this really is. I definitely feel like we bit more than we can chew.

That being said though, PegasOS has definitely provided me new incitement on how operating systems worked. Even though we had to cut some stuff because of time constraints or lack of knowledge, I feel like we definitely showed something we can be proud of.

What's next for you?
I don't have much planned out honestly. I know I'm a little on that, but hey, got to start somewhere. Right now, I'm trying to find a job to help jump start my career. I like working in software development or low-level stuff so I hope I find something to fill that need. I'm also interested in maybe doing game development as well. Hopefully I can release my own video game one day.

5.2.5 Christopher Walen

Thank You's
I'd like to thank the team again, without you guys this project wouldn't have happened in the first place. It's been such a great learning experience, and it's been a lot of fun getting to work with you guys and make this a reality.

I'd also like to thank my girlfriend, Hope. She's helped me stay sane throughout college and especially my last two semesters at UCF, and her editor skills certainly helped us in the past! Thanks for all the late nights helping me and keeping me company, and all of the encouraging words you've given me over the years. I look forward to all the things that are coming in our life together.

Who Am I?
I'm Chris, I'm graduating this December with a Bachelor's degree in Computer Science. It's been a wild and crazy ride getting here, but it's all been a formative and greatly enlightening experience for me.

In the beginning I wasn't quite sure what I wanted to do with the degree, all I knew was that I enjoyed programming. I came from Valencia with my Associate's in Computer Science, and got to start right off the bat with classes that used everything I knew to the fullest. I struggled a lot and repeated a few classes, but I eventually started to 'get it'. I attribute my CS enlightenment to Computer Science 2 with Szumlanski. That class challenged me like no other, but at the same time it was all making sense to me where other classes in the major just weren't quite sticking yet. Ever since then I've actually felt like a real Computer Scientist and Software Developer, and it gave me a lot of confidence in knowing that I can actually do this.

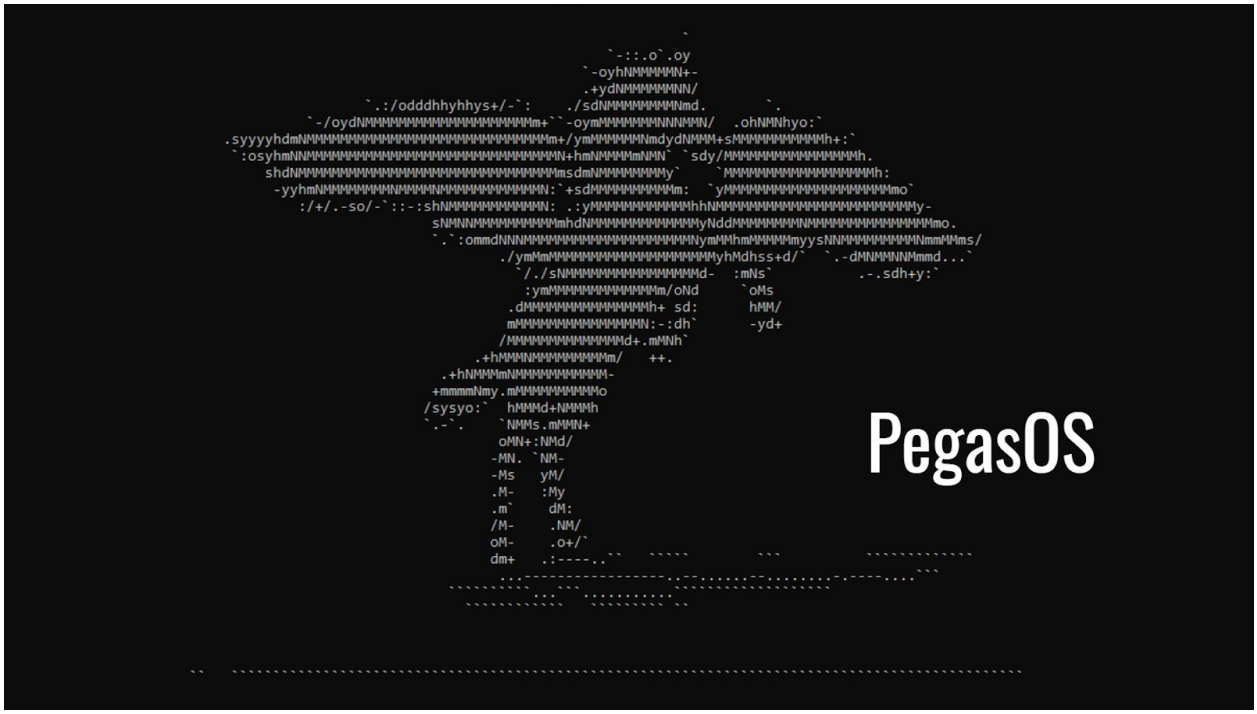
As I've gotten further into my degree at UCF, I've gravitated more and more towards the systems side of Computer Science. There haven't always been opportunities for me to explore this, but whether its been systems in Game Design and Development or a project as crazy as PegasOS, I've really been trying to get my feet wet in the world of Embedded and System development and I've found it really interesting, if not difficult! Now that I'm at the end of my degree, part of me is sad that it's over and that this project

is over, but the other part of me is really excited to see where my life takes me and what cool projects I get to work on next. As long as it's something that challenges me to do my best and make interesting decisions, I'll have a great time.

What did you think of PegasOS? As a project? The final result?
It's been a bumpy road, that's for sure! Though ambitious projects like this always seem to have bumpy roads, so what's new? But I think PegasOS was a great opportunity for us to do something that was really different from what everyone else in the class was doing, and I think that we've come a long way and achieved a lot of what we set out to do. Sure it's sad that we couldn't do *everything* that we had originally planned, but it's honestly amazing that we even have this much to show for it. OS development is hard, and I think it's safe to say we all learned that the hard way!

But PegasOS has been a great experience, for me at least, and I'm glad I got to meet these great people that all brought their A game to this project. Dealing with other classes on top of this wasn't easy, but we pushed through and it's all been worth it in the end. I'm very happy with the state of the project, and I'm really hopeful that Senior Design or Operating System students after us can take this project and really make it their own. I'm glad I got up and pitched it that day back in January.

What's next for you?
At the moment I'm not so sure; right now I'm looking for entry-level embedded system and system engineering jobs. I'm not the pickiest person in the world, so I'm not sure where I'll end up quite yet. I'm applying for some positions at Lockheed Martin currently, so we'll see how that goes! When I'm not working you'd probably find me either playing games or developing my own games - its a goal of mine to get a game I've made onto Steam some day! It could also be cool to work as a game developer, though that market is super competitive and I like the freedom of making whatever you want as an indie dev.



Thank you!