Discovering Raspberry Pi

Bruce Chittenden
Department of Computer Science
Hampton University
100 E. Queen Street
Hampton, Virginia 23668
01 757-412-8746
bruce.chittenden@hamptonu.edu

ABSTRACT
The Raspberry Pi is a low cost, credit-card sized single-board computer that plugs into a HDMI monitor and uses a standard USB keyboard and mouse [1]. This paper outlines how the Raspberry Pi was introduced and incorporated into Operating Systems at Hampton University. A series of program assignments are developed utilizing the Raspberry Pi to enhance students' learning in Operating Systems concepts are discussed in this paper.[1] The feedback from the students was incredibly positive as they really enjoyed the hands-on nature of working with the Raspberry Pi and the actual source code for the Linux Operating System.

CSS Concepts
CSS → Software and its engineering → Software organization and properties → Contextual software domains → Operating systems

Keywords
Raspberry Pi; Operating Systems; Linux

1. INTRODUCTION
Typically a course on Operating System would include several programming assignments in Java or C++ that emulated different features of the Operating System to give the student insight into these functions. By introducing the Raspberry Pi the emulation of Operating System functions is no longer necessary. The student is actually working in a real Operating System development environment using the source code for the Linux Operating System.

The Raspberry Pi design is based around a Broadcom BCM2835 SoC [2], which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU, and 512 MBs of RAM. The design does not include a built-in hard disk or solid-state drive, instead relying on an SDHC card for booting and long-term storage. Currently there are several versions of Linux that runs on the Raspberry Pi. The most popular is Raspbian which is based on a version of Debian Linux. Several Programming Languages are also available for the Raspberry Pi including C, C++, FORTRAN, Java, Python, and Scratch [1].

Over the summer of 2014, I worked with a Lauren Patterson [3], a student from Hampton University, on an Externship with the National Center for Atmospheric Research, NCAR. We created a cluster using four Raspberry Pi Model B (Figure 1) computers and a Western Digital Router. We ported Message Passing Interface mpich2 to interconnect the Raspberry Pi computers. This configuration is now referred to as Raspberry Pi Bramble.

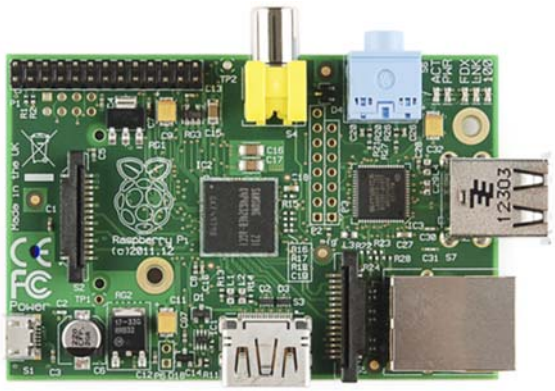

Figure 1. Raspberry Pi Model B 512 MB

Over the summer of 2015, I again worked with another student from Hampton University, Whitney Nelson [4], on an Internship with the National Center for Atmospheric Research, NCAR. We created a cluster this time using four Raspberry Pi 2 Model B computers and a Western Digital Router. The Raspberry Pi 2 Model B (Figure 2) is an upgrade for the Raspberry Pi Model B based around the Broadcom BCM2836 SoC, which includes an ARM Cortex A7 quad-core 900 MHz processor and 1 GB of RAM [5]. We ported Message Passing Interface mpcich2 to interconnect the Raspberry Pi computers. Then we ported OwnCloud to the cluster to create a cloud computing platform to collect weather data.

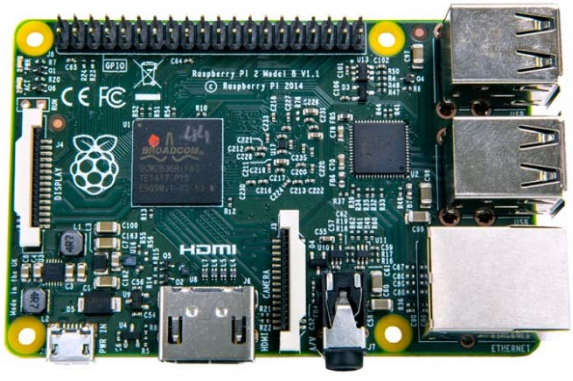

Figure 2. Raspberry Pi Model 2 Quad-Core 1024 MB

With the experience I gained over the past two years, I introduced Raspberry Pi and the Linux Operating System into our course on

[1] The developed materials for this course (PowerPoint Lectures and Programming Assignments) are available for any professor teaching a similar course upon request.

Operating Systems, CSC 301 - Operating Systems. There are a total of seven assignments related to the Linux Operating System and the Raspberry Pi, which included building a Linux kernel from source code, adding a system call to the Linux kernel, writing a device driver for the Linux Operating System, and a team project of building a Raspberry Pi Bramble. This paper discusses in detail these assignments.

2. HAMPTON UNIVERSITY

Most classes on Operating Systems are usually build around textbook [6] assignments and a few programming assignments in either Java or C++ simulating Operating System functions. It would be rare for a student in an undergraduate course in Operating Systems to actually have Operating System development experience, but with the knowledge I gained from the Summer of 2014 "Raspberry Pi Hadoop Cluster" and the Summer of 2015 "Pi in the Sky" and the low cost of $35 for the Raspberry Pi Computers it was possible at Hampton University. The Linux Operating System [7] was chosen because of all the Operating Systems available, it is the only one that is Open Source, and the source code to the kernel (9,868,933 lines of code, 12,020,528 lines with comments spread over 36,595 unique files) is available on the Internet.

Using the Raspberry Pi Computer and Debian, a flavor of Linux that is the primary version of Linux for the Raspberry Pi, I designed a curriculum with real hands-on Operating System experience.

3. CSC 301 - OPERATING SYSTEMS

The curriculum consists of seven programming assignments spread across fifteen weeks of the course. The assignments are:

Program 1 - Install VirtualBox and Debian
Program 2 - Linux Commands
Program 3 - Install Raspberry Pi Raspbian Debian Wheezy
Program 4 - Build Raspberry Pi Kernel
Program 5 - Add System Call to Raspberry Pi Kernel
Program 6 - Write a Device Driver for Raspberry Pi Kernel
Program 7 - High Performance Computing and Message Passing Interface

3.1 Program 1 - Install VirtualBox and Debian

This assignment has two purposes. The first is to familiarize the student with the concept of a virtual machine, in this case VirtualBox [8], and how it is possible to run a guest Operating System on your Personal Computer. However, the real purpose of this assignment is to allow the students to become familiar with Linux in an environment that is familiar to the student. The students will learn how to install Linux Debian [9], how to login, and how to use some of the basic commands and utilities. Debian was chosen because that is the preferred version of Linux on the Raspberry Pi. Debian can be found at https://www.debian.org/.

3.2 Program 2 - Linux Commands

The purpose of this assignment is to learn the basic set of Linux Commands [10] that the student will use during this course. The student will go through the commands located in the /bin and /usr/bin directories and write a brief description of the commands along with what the typical arguments might be for their usage. The end result of this assignment is a Quick Reference Guide that will be useful throughout this course.

3.3 Program 3 - Install Raspberry Pi Raspbian Debian Wheezy

Each student is given a Raspberry Pi Kit (Figure 3 and Figure 4) and a USB Keyboard. For a Monitor the students are expected to use the Television in their dorm room or in their apartment. The Raspberry Pi Kit, which the cost is less than $100 (Table 1), contains everything necessary for the Raspberry Pi programming assignments.

Figure 3. Raspberry Pi Kit

Each student is expected to set up the Raspberry Pi Computer development environment, download the image of Raspbian Linux (Debian Wheezy) [11] from the Raspberry Pi website https://www.raspberrypi.org/downloads/ to their Personal Computer, copy that image to the SDHC memory card, place the SDHC card in their Raspberry Pi Computer, and boot up Linux. On the initial boot there are several configuration questions and settings that need to be established. Once configured the student logs in as user pi.

Figure 4. Raspberry Pi Kit

Table 1. Contents of the Raspberry Pi Kit

Description	Unit Price
Raspberry Pi Model B	35.00
Raspberry Pi Case	7.95
USB Power Supply	5.95
Sandisk SDHC Card	18.95
Sandisk MicroMate SD / SDHC Memory Card Reader	20.79
GPIO Ribbon Cable for Raspberry Pi Model A and B - 26 Pin	2.99
HDMI Cable 6'	5.95
Red LED	0.99
	98.57

3.4 Program 4 - Build Raspberry Pi Kernel

This assignment is to build a kernel for the Raspberry Pi Computer from source code [7]. The source code for the kernel contains 9,868.933 lines of code spread across 36,595 files. The size on disk of the source code is approximately 2 GB. The source code can be downloaded from GitHub [12] https://github.com/raspberrypi/linux. Once downloaded the student makes a default configuration for the kernel that is currently running and then does a make to build a kernel from source. To build a kernel from source code takes approximately 15 hours on the Raspberry Pi. When the build is complete the new kernel is copied to the boot partition and the Raspberry Pi is rebooted.

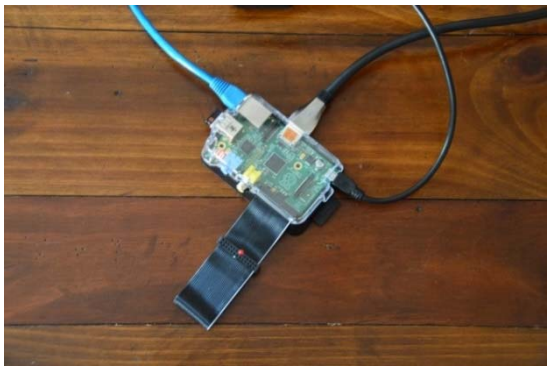

Figure 5. Raspberry Pi with GPIO Cable and LED

3.5 Program 5 - Add System Call to Raspberry Pi Kernel

This assignment is to implement a new System Call [13] [14] [15] in the Linux Operating System [7]. Normally you would add a system call at the end of the System Call Table, but for this assignment we have selected a system call that is no longer in use, System Call 59. There are four kernel source files that must be modified

/arch/arm/include/uapi/asm/unistd.h,
/arch/arm/kernel/calls.S [16],
/include/linux/syscall.h, and
/arch/arm/kernel/sys_arm.c.

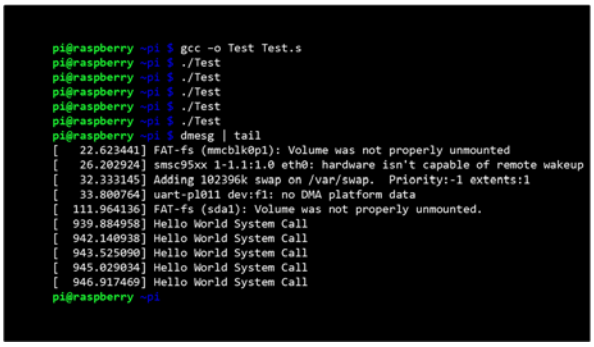

Figure 6. Compiling and Executing Hello World System Call

The last file, sys_arm.c, is the actual code to implement your new system call. This assignment does require a kernel recompilation, so this is another 15 hour build process. What our new System Call does is to write "Hello World System Call" (Figure 6) to the dmesg log using the kernel method printk when the system call is invoked.

There are several papers on the Internet that describe how to add a system call, http://blogsmayan.blogspot.com/p/adding-simple-system-call.html.

3.6 Program 6 - Write a Device Driver for Raspberry Pi Kernel

This assignment is to write a Linux Device Driver [17] for the Raspberry Pi that blinks an LED at a specified frequency. We will be controlling the Broadcom BCM2835 ARM Peripherals [2] https://www.google.com/search?sourceid=navclient&ie=UTF-8&rlz=1T4GUEA_enUS592US592&q=Broadcon+BCM2835+manual by accessing hardware registers. We want to connect an LED to one of the General Purpose I/O (GPIO) pins.

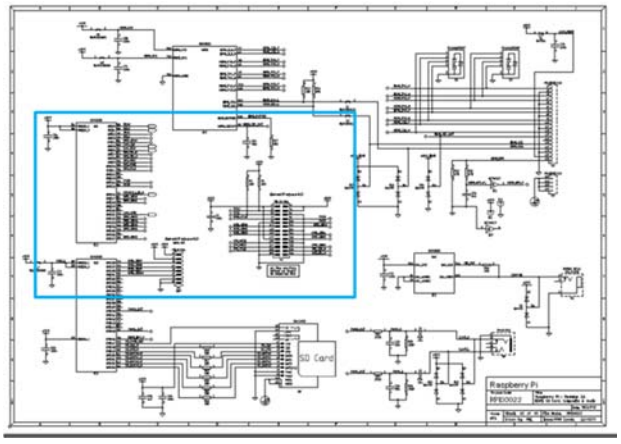

Figure 7. Raspberry Pi Schematics

In this example we will select GPIO_GEN1 signal that corresponds to pin 12 on P1 and GPIO18 (Figure 8).

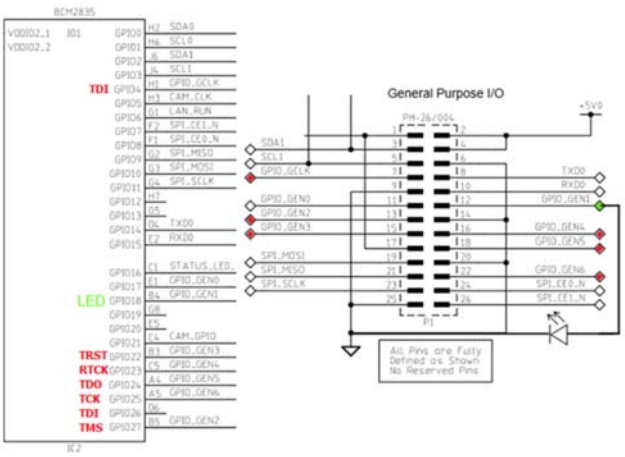

Figure 8. General Purpose I/O

An example of this driver can be found at http://sysprogs.com/VisualKernel/tutorials/raspberry/leddriver/. There is a textbook that is available on how to write device drivers for Linux, Linux Device Drivers, Third Edition, that is available on the Internet https://lwn.net/Kernel/LDD3/.

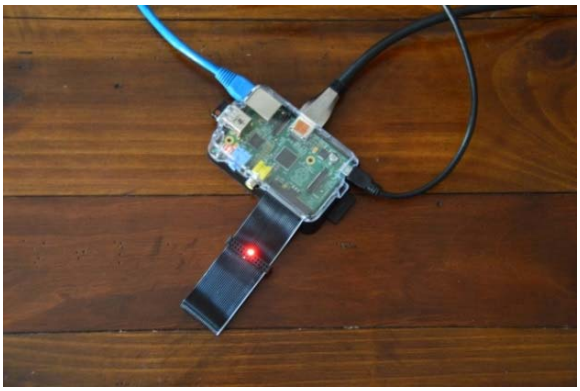

Figure 9. Device Driver Flashes the LED

Figure 9 shows the Raspberry Pi that blinks an LED.

3.7 Program 7 - High Performance Computing and Message Passing Interface

This assignment is to build a four node Raspberry Pi Cluster (Bramble) [18] (Figure 11) and compile and run the program mpi_hello_world, which will report with Hello World which processor it is executing on.

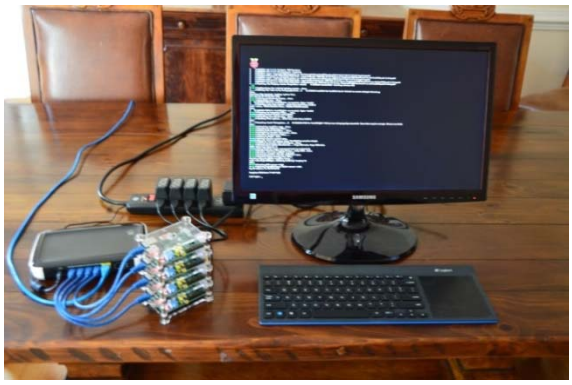

Figure 10. Raspberry Pi Development Environment

This is a several step assignment. The student already has a Raspberry Pi running Linux, which is the First Step. The second step is to implement the Message Passing Interface (MPI) [19] http://www.mpich.org/downloads/. This allows the four processors to communicate between one another. There are several configuration files involved in this step; /etc/hostname,/etc/hosts, /etc/network/interfaces, /etc/resolv.conf, and ~/mpi_testing/machinefile. The Third Step is using RSA to setup quick logins between the processors. And the Last Step is to compile and execute mpi_hello_world.c. (Figure 12)

Figure 11. Raspberry Pi High-Performance Computer

```
pi@hu0 ~mpi_testing $ ls -l
total 40
-rw-r--r-- 1 pi pi     40 Jul  9 08:44 machinefile
-rwxr-xr-x 1 pi pi  30090 Aug 11 08:28 mpi.h
-rwxr-xr-x 1 pi pi    494 Aug 11 08:28 mpi_hello_world.c
pi@hu0 ~mpi_testing $ mpicc -o mpi_hello_world mpi_hello_world.c
pi@hu0 ~mpi_testing $ ls -l
total 792
-rw-r--r-- 1 pi pi     40 Jul  9 08:44 machinefile
-rwxr-xr-x 1 pi pi  30090 Aug 11 08:28 mpi.h
-rwxr-xr-x 1 pi pi 767270 Aug 11 08:29 mpi_hello_world
-rwxr-xr-x 1 pi pi    494 Aug 11 08:28 mpi_hello_world.c
pi@hu0 ~mpi_testing $ mpirun -n 4 ./mpi_hello_world
Hello World from Process 3 of 4
Hello World from Process 1 of 4
Hello World from Process 2 of 4
Hello World from Process 0 of 4
pi@hu0 ~mpi_testing $
```

Figure 12. Compiling and Executing Hello World MPI

4. FEEDBACK FROM STUDENTS

This is some of the feedback from students given at the end of the semester in the Course Survey.

"I like the Raspberry Pi assignments we were given during the Computer Science 301 Course. I feel like it gave us more hands on experience with programming than other computer classes."

"Instead of giving us the traditional program and asking us to figure it out ourselves, we were provided with detailed instructions and methods of implementing our programs. I feel that it was more interesting than the traditional programming assignments."

"I really liked the idea and execution of the hands-on learning we experienced. Professor Chittenden gave us a reasonable amount of time and resources to complete the projects."

"In the future, the class could focus more on different features of the Operating System such as file system management, etc."

"I really enjoyed using the raspberry pi in class because it gave us a way to actually apply the skills we were learning. It also added an element to the class, unlike any other course I have taken in the past, where we were able to take the notes from class and actually see a result."

"I really enjoyed using the Raspberry Pi to learn LINUX/UNIX in 301. I liked that they were small and portable, making it easy to use in different locations. The only thing I would change would be to maybe have the assignment when you pull the kernel from the GitHub repository as a class assignment, as it was a little confusing trying to do it on your own."

5. FUTURE PLANS

The feedback about focusing of the different features within the Operating System will be taking into account for next offering, e.g. adding the development of a command that originated in Multics [20], but did not make it into UNIX called Walk Subtree, ws. The syntax for Walk Subtree is "ws path command line {-control_args}". Walk Subtree starts at the specified path and walks the entire subtree from that starting point executing the command specified. For example, the command "ws / ls –l" would list the entire file system directory structure.

6. CONCLUSION

The Raspberry Pi has many uses besides courses in Operating Systems. Because of the many Programming Languages that are available on the Raspberry Pi it could be used in several different classes in either Computer Science or Computer Information Systems. With the ability to interconnect several Raspberry Pi computers using Message Passing Interface, courses in distributed file systems, like Hadoop, or parallel computing can easily be created again giving the students hands-on experience in these environments. For any Operating System course where students are rebuilding the Kernel, it is essential that students had a dedicated machine. The Raspberry Pi System provides each student with a dedicated machine with a cost under $100.

7. REFERENCES

[1] The Raspberry Pi Foundation. Website is accessible at https://www.raspberrypi.org/.

[2] BROADCOM BCM2835 ARM Peripherals.

[3] National Center for Atmospheric Research, Summer 2014 Extern Lauren Patterson, Website is accessible at https://www2.cisl.ucar.edu/siparcs/calendar/raspberry-pi-hadoop-cluster.

[4] National Center for Atmospheric Research, Summer 2015 Intern Whitney Nelson, Website is accessible https://www2.cisl.ucar.edu/siparcs2015presentations/nelson.

[5] Cortex-A7 MP Core Technical Reference Manual.

[6] Operating System Concepts with Java, Eight Edition, Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne

[7] Understanding the LINUX KERNEL, 3rd Edition

[8] Oracle Corporation VirtualBox. Website is accessible at https://www.virtualbox.org/.

[9] Debian The Universal Operating System. Website is accessible https://www.debian.org/.

[10] LINUX IN A NUTSHELL, 6th Edition.

[11] The Raspberry Pi Foundation. Website is accessible at https://www.raspberrypi.org/.

[12] GitHub Guides. Website is accessible at https://guides.github.com/.

[13] Adding a Simple System Call to the Raspberry Pi. Website is accessible http://blogsmayan.blogspot.com/p/adding-simple-system-call.html.

[14] Anatomy of a system call, part 1. Website is accessible https://lwn.net/Articles/604287/.

[15] Anatomy of a system call, part 2. Website is accessible https://lwn.net/Articles/635604/.

[16] ARM Compiler toolchain Version 5.0 Assembler Reference

[17] Linux Device Drivers, Third Edition.

[18] Steps to Building a Raspberry Pi Supercomputer, University of SouthHampton. Website is accessible at http://www.southampton.ac.uk/~sjc/raspberrypi/

[19] Message Passing Interface, MPI-3.1 Standard, The Argone National Laboratory. Website is accessible at https://www.mpich.org/.

[20] Series 60 (LEVEL 68) MULTICS PROGRAMMER'S MANUAL COMMANDS AND ACTIVE FUNCTIONS, Honeywell Information Systems.