What! No GUI? – Teaching A Text Based Command Line Oriented Introduction to Computer Science Course

Ira Goldstein
igoldstein@siena.edu
Department of Computer Science
Siena College
Loudonville, NY 12211 USA

Abstract

Computer Science students need to acquire knowledge about both the hardware and software aspects of computing systems. It is necessary for them to understand how each layer interacts with one another. However, since Graphical User Interfaces have become ubiquitous, the opportunities to interact with the computer via a command prompt as part of their course offerings are few and far between. The result has been that an intuitive understanding of this interplay has been lost. This paper describes an Introduction to Computer Science course that utilized the Raspberry Pi Linux based computer in a text based, command line environment for all programming assignments. The students edited their programs using the Nano text editor. They submitted their programming assignments using SFTP. They configured and managed their Raspberry Pis, including installing and configuring the Apache web server, from the command line.

Keywords: Computer Science Education, Introduction to Computer Science, Raspberry Pi, Linux, Pedagogy, Command Line.

1. INTRODUCTION

The Introduction to Computer Science (CS) course (CSIS110) at Siena College is a blend of CS0 and CS1 topics, with an even split between CS concepts and programming. It is a required course for both CS majors and Information Systems minors, as well as for students majoring in Computational Science and Actuarial Science.

In order to attract students with varying interests, several variations of the course, each with its own focus (flavor), have been offered in recent years. The offerings have included flavors in Alice, graphics and games, multimedia, music, and scientific computing, with the last three being offered using the Python programming language. While all of the sections utilize the Dale and Lewis (2013) text and cover the same CS concepts, each of the flavors utilizes a second textbook appropriate to its focus.

Over the years, as operating systems have evolved, we have moved away from using a command line interface, thereby abstracting how a computer operates. As the desktop Graphical User Interface (GUI) became the de facto standard, we have been graduating CS students who, at most, were vaguely aware of the existence of an operating system's command line interface. This runs contrary to the need for CS students to understand how hardware and software layers interact with one another.

Kendon and Stephenson (2016) report the results of a non-credit course that provided hands-on Linux command line instruction. The course covered file management, text editing, piping and redirection, and compiling and running programs. The authors report that the course was well received, and based upon post-instruction surveys, the students found the hands-on labs

and learning about the command line to be valuable.

While examining CS faculty's perception of the instructional use of Unix, Doyle and Lister (2007) found that faculty believed that it should be part of the CS curriculum since it allows you to "interact with [the computer] more directly than using something like windows which has a GUI on top of it" (p 21). They also found support for the idea that working at the command line provides a more powerful environment than working in Window's GUI. When reporting on the use of a treasure hunt game to motivate learning Unix, Moy (2011) found that the command line forces students to better understand the task at hand.

The Raspberry Pi is a credit card sized affordable single-board computer developed in the United Kingdom by the Raspberry Pi Foundation, and is capable of running a number of different operating systems, including Debian Linux. The foundation's goal is to put computing power into people's hands "so they are capable of understanding and shaping our increasingly digital world, able to solve the problems that matter to them, and equipped for the jobs of the future" (Raspberry Pi Foundation 2018).

Incorporating hands-on activities in an introductory CS course has been shown to augment a student's understanding of the course material (Wu, Hsu, Lee, Wang & Sun 2014). The Raspberry Pi has been used successfully in providing hands-on instruction in a number of fields, from bioinformatics (Barker, Ferrier, Holland, Mitchell, Plaisier, Ritchie, & Smart 2013) to building a microscope as part of a Life Sciences course (Rajani, Markus, Ward, McLean, Gell, & Self 2017) to Chemistry (Geyer 2014), and Physics (Singh & Hedgeland 2015), as well as in CS (Jaokar 2013; Frydenberg 2017; Black & Green 2017).

Having had some experience with the then new Raspberry Pi, I proposed offering a flavor that focused on Linux for the Fall 2014 semester, providing students with a number of command line, text based labs and homeworks. In order to not inflate the textbook cost for the course, the students purchased their own Raspberry Pi as their second "textbook." Open source and on-line material were used for supplemental readings.

2. BACKGROUND

The primary goal of the Linux flavor was for the students to feel comfortable in a command line environment, which, to the uninitiated, can seem intimidating. Being able to use the command line is often more efficient than point and click; can give the user greater control over the computer, especially when performing administrative functions; allows the user to install programs that may not be available as an application; and can automate repetitive tasks.

The course consisted of two one-hour lectures each week, as well as eleven labs. I created five new labs in order to cover the new topics. Using material from the other existing flavors, I modified three existing labs, such as enhancing the operating systems lab, and reused three of the labs that covered topics, such as exploring object oriented programming using ALICE. Labs were run following the paired programing paradigm (Bevan, Werner, & McDowell 2002; Simon & Hanks 2008).

Knowing that I wanted the students to be able to write programs that generated dynamic web pages via Common Gateway Interface (CGI), I selected Perl (Wall 2000) based on how commonly Perl is used for this purpose. While not currently in vogue as a first programming language, Perl seemed like an obvious choice for teaching programming in a strictly text based environment. In addition, given that Perl has weakly (dynamically) typed variables, the students did not need to worry about declaring variable data types.

Following the Dale and Lewis (2013) text, the course covered a breadth of topics. One topic was data representation: binary, octal, hexadecimal, signed magnitude, text compression, colors, images, and audio. Another topic included Boolean expressions, gates, truth tables, and circuits. The computing components topic covered how to calculate disc seek, latency, and transfer times, and von Neumann architecture, which serves as an introduction for assembly and machine language. It also touched on concepts from operating systems, programming languages, and artificial intelligence. While required for CS majors and minors, a wide spectrum of students enroll in CSIS110 since the course can be used to fulfill the college's quantitative analysis graduation requirement.

The students' Perl code needed to follow a set of standards. First, the code needed to follow ***perlstyle*** as described in the Perl Programming documentation (Perldoc 2018). Programs needed to contain the program's name, the author or authors' names, and a short description, each as comments at the top. Each section of code required descriptive comments. Pragmata were

used to control runtime behavior of Perl. The students were required to include two pragmata. The ***strict*** pragma disabled certain Perl constructs that could behave unexpectedly. The ***warnings*** pragma enabled Perl's optional warnings, which would help debugging programs. When writing backend web programs, Perl programs needed to use the CGI core module.

3. ENVIRONMENT AND SETUP

For each offering, we used the most recently released version of the Raspberry Pi model B. Initially, we used the Raspberry Pi 1 B+ that had a single core ARM 32-bit processor running at 700MHz, 512MB memory, 4 USB ports, and 10/100 Ethernet. The Raspberry Pis ran the Raspbian OS, based upon the 3.12 Wheezy release of Debian. In addition to the Raspberry Pi, the students needed to purchase a power supply, keyboard, micro-SD card, case, and a USB wireless Ethernet (Wi-Fi) adapter. More recent offerings have used the Raspberry Pi 3B which has a quad core 1.2GHz processor, 1GB memory, and built-in Wi-Fi (eliminating the need for the students to purchase a USB Wi-Fi adapter). Unfortunately, the campus bookstore was not, and is still not, able to order Raspberry Pis. Therefore, the students were given links to multiple on-line vendors from whom they could purchase either the individual components or kits. The cost for a fully configured Raspberry Pi was less than a typical textbook.

Since each student would have their own Raspberry Pi that they would use in and out of class, they would need to be able to access it not only in lab, but also at other locations. The Information Technology Services (ITS) group is very focused on ensuring that faculty has access to all necessary resources. Working together, we determined that the best way to connect the Raspberry Pis in lab would be via Wi-Fi, and added an HDMI cable to the secondary monitor on each of the lab's Windows PCs. While a bit cramped at a given workstation, this allowed the students to get to their e-mail and other resources while also directly connecting to their Raspberry Pi's console.

By using the college's Wi-Fi, the Raspberry Pis could connect to the network from any location on campus (Figure 3). As students became more comfortable with using their Raspberry Pi via the network, many students opted to leave their Raspberry Pi in their dorm room and connect from the lab using ***PuTTY***[i]. Instructions were also provided on how to configure the Raspberry Pi to work on other Wi-Fi networks for those students who lived off-campus.

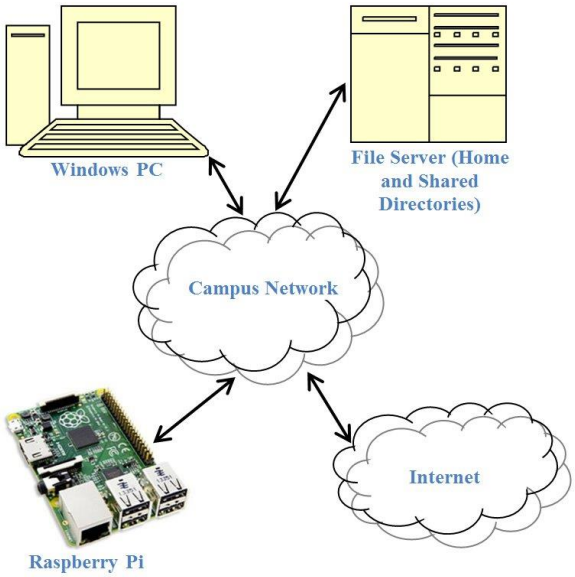

Figure 3 – Campus network environment

In order to get the students up and running as quickly as possible several customizations were made to the base Raspbian operating system, using the then most recent release of Raspbian. The first several customization items related to the wireless network. The college Wi-Fi network was added to the WPA supplicant configuration file. In addition, a shell script was added as part of the boot sequence that automatically sent out an e-mail with the system's network information (***ifconfig***), which included the current IP address. This enabled the student to remotely access their Raspberry Pi even if their IP address changed. In order to enable e-mail, an SMTP relay was configured to use a common Gmail account that was created for the course. Email utilities (***ssmtp***, ***mailutils***, and ***mpack***) and ***Lynx*** (a text based web browser) were installed. The system was then configured to boot to the command line interface, and to use US English. Finally, since there was no way to recover a lost password, a "csprof" account with full root access was added. This account would allow me to log in and perform any administrative tasks, including resetting the student's password. The students were informed of the existence of this account, and they were reassured that the account would not be used to access their system without their explicit consent. The students were then given until the beginning of the second lab to copy the customized version of Raspbian to their micro-SD card.

Students were given read-only access to the materials for each lab via a shared Windows drive. The materials included instructions, sample code, and support files. Students would copy the material to a lab folder on their own Windows home directory. ITS has a secure Linux server that automatically maps a user's home directory upon login. Using ***SFTP***, students would then copy any necessary files from their lab folder under their home directory to their Raspberry Pi. At the end of the lab, the students would use SFTP to copy their work back to their lab folder. This provided two benefits. The first benefit relates to disaster recovery. Since all files that the student modified on the Raspberry Pi were copied to their lab folder, if there was a catastrophic failure of their Raspberry Pi, recovery simply consisted of imaging a new micro-SD card, resetting the system password and name, and copying all of their files back to the Raspberry Pi. The second benefit relates to printing. Rather than having to configure the Raspberry Pis to work with the network printers, students were able to print off their work from the Windows PCs using ***Notepad++***[ii].

4. LABS

The students needed to complete eleven labs over the course of the semester (Table 4). Labs were run with students working in pairs. The lecture prior to each lab provided the students with scaffolding for each of the lab topics. In addition, the students needed to complete a pre-lab for all but the final lab.

Pre-labs (Appendix A) typically consisted of several readings followed by a short on-line multiple choice quiz on the reading material. In preparation for later labs, the pre-lab had the students install software packages, such as the Apache web server. A number of the labs (Appendix B) ended with reflection questions that were meant to make the students think critically and creatively about some aspect of the lab. Three of the labs, von Neumann (lab 7), Python (lab 10), and Artificial Intelligence (lab 11), were common to all flavors of Introduction to CS and were not modified. The discussion that follows and the appendices are limited to the labs, or portions of the labs, where the students used their Raspberry Pis.

Lab Number	Description
1	Linux command line
2	Configure individual Raspberry Pi
3*	Gates and Circuits – Standard input via Perl
4	Loops and conditional logic
5	Arrays and subroutines
6	Apache and dynamic HTML
7**	von Neumann architecture
8*	Alice - ping/traceroute - CGI
9*	Operating Systems - Processes
10**	Python
11**	Artificial Intelligence

Table 4 – Lab Descriptions

* Modified common lab
**Common lab across all sections.

The first lab was run with the students connecting to one of several Raspberry Pis that I had placed on the network. This ensured sufficient time for the students to procure their own Raspberry Pi and to copy the course's version of Raspbian OS to their micro-SD card before they needed to use them in lab. In this lab, the students learned basic Linux commands and about the network environment that they were using. The flow of Lab one is summarized in Table 5. Objectives for this lab included the ability to identify the components of the networking environment, and to demonstrate how shell scripts can be customized to perform specific tasks.

1. Connect to a remote Raspberry Pi via ***PuTTY***
2. Interact with the Linux BASH command line
   a. List the contents of a directory
   b. Display files
   c. Change file permissions
   d. Run shell scripts
3. Use GNU ***Nano***[iii] text editor to modify an existing shell script (Figure 4)
4. Use ***sftp*** to transfer files
5. Use ***man*** to access the Linux on-line reference manuals to discover various options for system commands

Table 5 – Lab 1 Flow

Figure 4 – Nano editor

The second lab began by having the students set up their own Raspberry Pis. Depending upon the number of upgrades issued since I created that semester's course's version of Raspbian, the students then patched their systems with the most recent update using the Advanced Packaging Tool ***apt-get***. If the upgrade would take a significant amount of time, the students were instructed to perform the upgrade before the next lab. The flow of Lab two is summarized in Table 6. Objectives for this lab included having to describe the steps necessary to set up a Raspberry Pi, and to explain how arguments are passed to a shell script.

1. Use the ***raspi-config*** utility to
   a. Change the default password
   b. Set the host name
   c. Expand the filesystem to use all of the space on their micro-SD card
2. Customize a provided shell script to send the system's network information to their e-mail account
3. Register the system on the campus Wi-Fi
4. Use the Lynx text based web browser to perform a Google Search (Figure 5)
5. Patch the system
6. Use BASH pipes and redirection

Table 6 – Lab 2 Flow

During lecture, programming examples were provided in Perl. Starting with the third lab, the students began modifying and writing simple Perl programs on their Raspberry Pis. The fourth lab built on this and introduced loops and conditional expressions. The fifth lab introduced one dimensional arrays and subroutines. Some Perl programming topics, such as string manipulation, were covered in lecture and homework, and were not standalone lab topics.

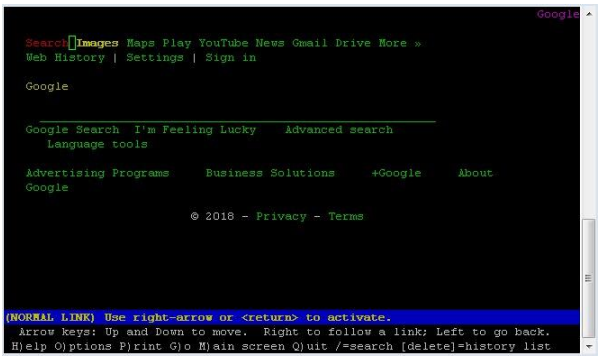

Figure 5 – Text based view of Google

Lab six had the students set their Raspberry Pi up as a web server. So as not to take up excessive lab time, the students installed Apache2 as part of the pre-lab. By the end of the lab they had created their own CGI program that displayed the current date and time as a dynamic web page. The flow of Lab six is summarized in Table 7. Objectives for this lab included the ability to identify the directories used by Apache, and to demonstrate how to manage Apache.

1. Configure Apache
2. Setup directories for
   a. HTML files
   b. Images
   c. CGI programs
3. Edit HTML using nano
4. Create CGI program
5. Monitor Apache's processes

Table 7 – Lab 6 Flow

Lab eight found the students building upon Lab six. The lab had the students look at how packets transverse the network using the ***ping*** and ***traceroute*** commands. They then explored how HTML forms pass data to backend programs.

Building upon Java applet simulations for process management which was common to all flavors of Introduction to CS, Lab nine allowed the students to interactively explore how CPU prioritization of a given process impacts other processes running on a system. Table 8 summarizes the flow of Lab nine.

1. Manage concurrently running jobs with
   a. ***kill***
   b. ***fg***
   c. ***bg***
2. Monitor running processes with
   a. ***ps***
   b. ***top***
3. Adjust process priority with ***nice***

Table 8 – Lab 9 Flow

The primary objective of this lab was for the students to compare and contrast how processes ran under contention and when set with varying priorities. The students were provided with two shell scripts: timehog.sh and longloop.sh. The timehog.sh script repeatedly copied blocks of 1024k zeros to the null device. Left unchecked, this script could utilize all available CPU cycles. The longloop.sh script repeatedly calculated 1000 MD5 checksums. The students noted how long it took longloop.sh to run with and without timehog.sh running in the background, and by changing the priority of the two scripts with the ***nice*** command.

5. HOMEWORK ASSIGNMENTS

While lab assignments were team efforts, all of the homework assignments were individual efforts. There were a total of five homework assignments. In order to emphasize that CS is not just coding, the "programming" portion of the first homework provided the students with specifications for several projects, and they were tasked with developing algorithms for each one. Several of the projects appeared as coding tasks in subsequent homework assignments. Rather than the typical situation where students struggle as they attempt to write code from their heads, the students were able to code from the graded/corrected copy of their algorithms.

The homework assignments reinforced the students' lab work. The second homework assignment had the students write a program to print out a multiplication table using nested loops. The third homework assignment required the students to use a one dimensional array to compare two compound interest scenarios. In the fourth homework assignment students created their own subroutines to manipulate strings.

Their final programming homework assignment was to develop an application that used a simple HTML frontend to pass data to their Perl CGI backend for manipulation, and then displayed the results as an HTML document. The students were given the choice of several scenarios to choose from. These choices included taking a name and producing output based on the lyrics of Shirley Ellis' Name Game song, taking an order for a cookie shop, or translating text into Pig Latin.

6. STUDENT REACTION

Given that the text based environment used in this flavor of the course was drastically different than the GUI environment used by any of the other flavors, I was interested in determining how well the course prepared them for subsequent CS classes. An on-line survey was sent to the 128 students who had taken this flavor of the course more than a year previously in order to find out if they would be interested in participating in a focus group discussion about their experience. Six students participated, all of whom had also taken at least one other CS course. Two of the six were Accounting majors, and the other four were CS majors. Three were male and three were female.

The general consensus was that initially the course was intimidating. For most of the students this was their first formal computer science course. However, they all agreed that it was a worthwhile experience, and its benefits extended beyond the classroom. The following are excerpts of the discussions.

"The Linux portion of it was such a foreign concept to me. It ended up being the most rewarding part because my internship; and every other interview that I've been in on they've asked me if I am comfortable on a Linux terminal and things like that and I've used it a lot. So, although it was the most, you know, it was the most anxious part for me for the course, it pays dividends."

"I actually know and kind of use it (the Raspberry Pi) now. Yeah, I use it for some like home automation stuff, making a home homebridge like certain products that didn't talk to each other."

"I came in with no knowledge and I was a nervous wreck the whole time. But I made it through and it was probably the course that made me decide on what major I wanted to choose which ended up being computer science."

"You know, I've even used the Nano editor again because, you know, working in a terminal you have the VIM or the Nano one, so it's like that part was very helpful."

"I thought it was a good basis because even before going in I heard that it was the hardest 110 actually, just concept wise. So I think going in with that kind of structure of like a harder 110 it ended up helping me with my further courses."

7. REFLECTIONS AND NEXT STEPS

By and large, the Linux flavor of Introduction to CS was well received. As with any journey, there

were some bumps in the road. Thankfully they were all navigable.

One of the first bumps relates to the use of Wi-Fi. By its nature, Wi-Fi is a shared medium. This makes it very difficult to guarantee bandwidth. ITS does an admirable job maintaining the network. However, periodically situations, such as an iOS update or the World Cup, would spike demand and slow down access to the Raspberry Pis. Given that rewiring the lab to double the number of Ethernet drops for this one course is not a practical solution, we have continued to use Wi-Fi. On the rare occasion when the networks slowed down, it provided an opportunity to discuss networking with the class, and the pros and cons of wired and wireless environments.

The students used SFTP to transfer sample code and finished programs between their lab folder on their own Windows home directory and their Raspberry Pi. This worked well once the students understood the difference between the bash shell prompt and SFTP prompt. However, several students in the focus groups mentioned that during job interviews they were asked about their experience with version control. Therefore, while I would still introduce SFTP at some point in the course, it may be beneficial for the students to use GitHub instead of SFTP. I could then treat each lab and homework assignment as its own project.

After the first offering, I was fortunate to be able to have lab assistants who had previously taken the course and were able to assist the current students. These positions were offered to students who had excelled in the class, and had been the "go to" for other students. I used them to run through the labs ahead of time to look for bugs, typos, and for any items that were not clearly explained. While they assisted in answering questions during the lab, they neither gave formal instruction nor graded any of the material.

Several of the other flavors of the course use Finch robots[iv] to teach programming concepts. In these, the students manipulate the color of the Finch's beak and write a program that uses the Finch's sensors to avoid obstacles. Giving students the ability to control real world objects with their programs can be a very powerful learning experience. I am planning to integrate the Finch into several of the existing labs.

8. REFERENCES

Barker, D., Ferrier, D., Holland, P., Mitchell, J., Plaisier, H., Ritchie, M., & Smart, S. (2013). 4273 π: Bioinformatics education on low cost ARM hardware. BMC bioinformatics, 14(1), 243-248.

Black, M., & Green, R. (2017). Server on a USB Port: A custom environment for teaching systems administration using the Raspberry Pi Zero. *Information Systems Education Journa*l, 16(6), 31-38.

Bevan, J. , Werner , L., & McDowell, C. (2002). Guidelines for the Use of Pair Programming in a Freshman Programming Class, Proceedings of the 15th Conference on Software Engineering Education and Training, 100-107.

Dale, N. & Lewis, J. (2013). Computer Science Illuminated. Jones & Bartlett Learning.

Doyle, B., & Lister, R. (2007). Why teach Unix?. Proceedings of the Ninth Australasian Conference on Computing Education, 66, 19-25.

Frydenberg, M. (2017). Ding Dong, You've Got Mail! A Lab Activity for Teaching the Internet of Things. *Information Systems Education Journal*, 15(2), 20-31.

Geyer, M. (2014). Mole Pi: Using New Technology To Teach the Magnitude of a Mole. Journal of Chemical Education, 91(11), 2005-2006.

Jaokar, A. (2013). Using Raspberry Pi to Teach Computing 'Inside Out'. Educational Technology, 53(2), 37–40.

Kendon, T., & Stephenson, B. (2016). Unix Literacy for First-Year Computer Science Students. Proceedings of the 21st Western Canadian Conference on Computing Education (WCCCE '16), 14-17.

Moy, M. (2011). Efficient and playful tools to teach Unix to new students. Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, 93-97.

Perldoc. perlstyle – perldoc.perl.org https://perldoc.perl.org/perlstyle.html, retrieved 7/8/18.

Rajani S., Markus R., Ward, I., McLean D., Gell C., & Self T. (2017). Build your own Raspberry Pi Microscope. infocus Magazine, (46), 46-52.

Raspberry Pi Foundation. About Us. https://www.raspberrypi.org/about/ retrieved 7/9/18.

Simon, B., & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. Journal on Educational Resources in Computing (JERIC), 7(4), 5.

Singh, P., & Hedgeland, H. (2015). Special relativity in the school laboratory: a simple apparatus for cosmic-ray muon detection. Physics Education, 50(3), 317-323.

Wall, L. (2000). Programming Perl (3rd ed.). Mike Loukides (Ed.). O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Wu, H. T., Hsu, P. C., Lee, C. Y., Wang, H. J., & Sun, C. K. (2014). The impact of supplementary hands-on practice on learning in introductory computer science course for freshmen. *Computers & Education*, 70, 1-8.

Appendix

Appendices can be found on-line:

Appendix A – Pre-Lab
https://drive.google.com/open?id=1UjCdDDX82QLUCmKeetHMkvInjzy3liEv

Appendix B - Lab
https://drive.google.com/open?id=1EcrKZo9yLYY5-iVuWAH-ZGNS50XDk97T