Kevin Wong

Dr. Amar Raheja

CS2600

December 8th, 2022

Programming Project 3 Write-up

Phonebook.bash and phonebook.csh were quite similar in terms of code. Of course, they are both shell scripting languages so they would not differ much in the first place, but the only differences came to their syntax. Since C-shell is mainly based off of the actual C language, the C-shell script was a smaller, more concise, and simpler shell script than the bash script, phonebook.bash. From the beginning, I was more familiar with C programming than anything that had to deal with the bash shell, or otherwise known as the Bourne-Again shell. I am by no means an experienced programmer, let alone an experienced C-language programmer, but the small things like the if-loop syntax, while loop syntax, and setting variables equal to the user input was more intuitive for me and led me to less confusion overall.

But on the other hand bash also has its perks as instead of setting variables and using =$<, you can just read the variable and that variable will already be set to the user-specified input. A reason I saw that allowed the C-shell script, phonebook.csh, to have less lines than the Bourne-Again shell script, phonebook.bash, was that in the if-loop syntax, the Bourne-Again shell has you put “then” underneath the if and elif statements. Or if you want the if and elif statements to be in the same line as the “then,” a semi colon, “;”, must be placed before the then. In a C-shell script, the “then” can be placed behind the if and else if statements with no need for a semicolon before the then. That makes C-shell (tcsh) to be a lot more intuitive and less prone to errors in that specific case.

Another thing I saw that differentiated the two scripting languages was the while loop. Since the Bourne-Again shell has you type “done” to end the while loop, and C-shell has you type “end” to signify the end of a while loop, there is not much of a difference in simplicity there. But when you look at the beginning of the while loop’s syntaxes, the Bourne-Again shell has you type “do” after the while loop’s initial condition statement. This results in an extra line of code that C-shell does not have to require you to type out.

If I had learned the basics of C programming after I learned how to script in the Bourne-Again shell, I would likely not have much trouble either since the two do not stray from each other that greatly. Instead of endif like in C-shell, it is fi in the Bourne-Again shell. These two statements to end the if control structure are not terribly different and are not easy to mix up.

A feature that I like about Bourne-Again shell, is that when setting up variables that involve getting a user’s input, all you have to do is use a simple command, which is the “read” command. After you use an echo statement to display a message and tell your user what to input, the read command takes the user-specified input and splits the string of characters into separate fields, assigning each new word to an argument. You type a variable name after the read command and everything the user types in on the screen is accessible with that variable.

An error or problem that I kept running into with the Bourne-Again shell is that I repeatedly saw myself typing () instead of []. After coding in languages like C, Java, C++, and Python, I have been extremely comfortable with parentheses over brackets. In my mind, I was not working with arrays or anything like that, like indexes for example, so I typed () for the if control structure in phonebook.bash.

Since I used commands that were unix utilities like sort, sed, awk, echo, clear, and the > sign for files, everything else in the scripts were pretty much the same. But this is something that I really do like about scripting languages in Unix. Regardless of the different syntaxes between scripting languages like Bourne shell, C-shell, Bourne-Again shell, and others, Unix utilities can still be used to make the scripting easier to do when you know the basic universal utilities.

But even with something relatively simple like the read command, I think that I would still go with the C-shell scripting format over the Bourne-Again shell scripting format. I am just more familiar with the C language, it is a little bit more intuitive for me, and I am just more comfortable with it and its syntax. Sure it is not exactly like the actual C-language, but the C-shell scripting language being based off of it helps a whole bunch in reality.

Outside of this phonebook.bash and phonebook.csh scripting assignment, there is a lot more to consider about both scripting languages as there are definitely a lot of functions that I did not use, and many that I do not even know about. But after highlighting the advantages and disadvantages of each shell, and going through each while typing up the phone book scripts, I would have to choose C-shell over the Bourne-Again shell.

It is not necessarily true that C-shell (tcsh) is better than the Bourne-Again shell by any means. Both shells have their own unique syntax, features, and strengths, so in the end, the choice of which one to use ultimately depends on the specific needs and preferences of the user. In my case, I prefer familiarity and I just so happen to be more familiar with the C programming language.

With the Bourne-Again shell, I had to look over the handout that was posted on Canvas multiple times. But with C-shell, since we did the C-shell scripting assignment after the Bourne-Shell scripting assignment, everything was slightly easier for me. In regard to errors, there were plenty in both scripts as I typed them out. There were a few moments that had me scratching my head, and the error statements were not very helpful for both, as I am just a beginner with scripting. There were times I had to consult classmates and others who could assist me for help and luckily they guided me so I could figure out the problems, which were usually caused by some miniscule syntax error, like a semicolon, quotation marks, or even spaces. A thing that helped me were youtube videos that went into depth about both the Bourne-Again shell and the C-shell. Most of the videos were what we went over in class, but I truly recommend that every student, no matter the area of study of their class, should head towards youtube for some type of aid, because personally it helped me with memory and also with the important idea of familiarity. Just watching a video with all of the commands on the screen, and another explanation is what I needed to progress with my code.

Writing these scripts was actually fun and eye-opening. At the moment, I sure was frustrated, but it helped me in the long run, and most likely helped me with preparation for our class's final exam. In the end, I am proud of what I have accomplished and look forward to more.