Choose Your Own Adventure Project Report

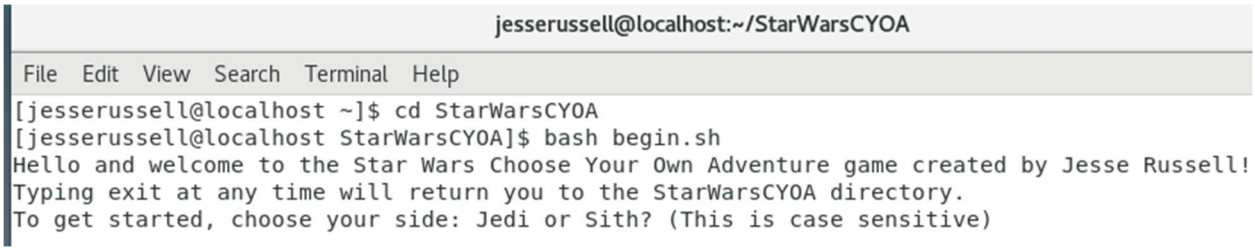
Jesse Russell

Abstract

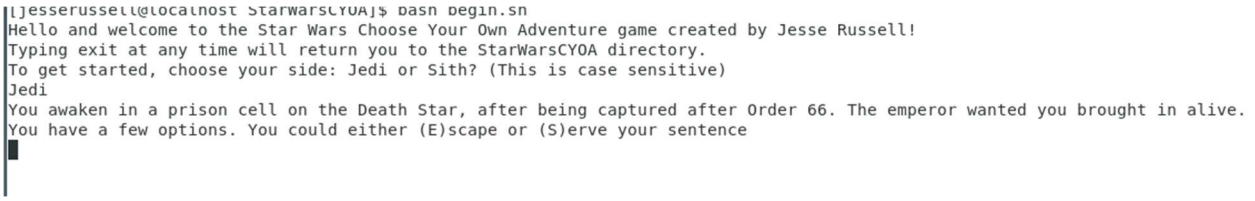
For my project, I decided to create a text based choose your own adventure game, based on Star Wars, a series I've loved since I was a child. I learned a lot of new things, and now have a better understanding of how scripting in bash works. Throughout this report, I will explain how the program works, as well as what I learned and what problems I ran into. I will also include a step-by-step guide so that others can learn from my work and code their own CYOA games with much more in depth paths and original concepts than mine. I hope that others may learn and take this idea to the next level. I hope this is as fun for you to read/play as it was for me to make.

The User-End

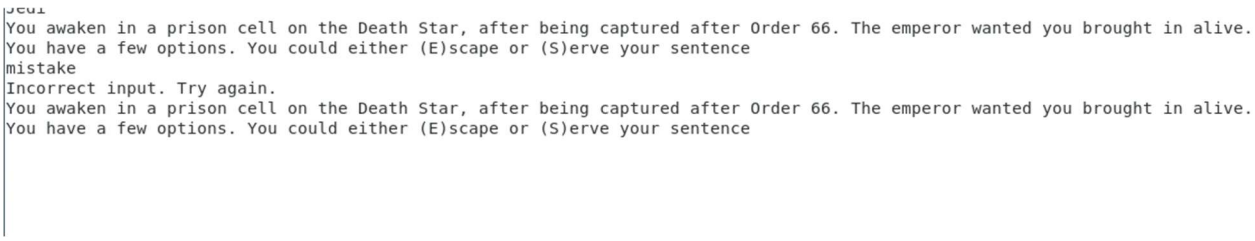
The main concept behind this idea is very simple on the user-end. To begin the program, you must navigate to the StarWarsCYOA directory. Then you must run the begin.sh script by typing in bash begin.sh. You are given a prompt with 2 choices. Type in the letter that is in parentheses (with the exception of the first main decision), and the computer will give you a new prompt. You keep going through the prompts until you reach the end and are given the option to restart. I kept this portion of the project intuitive, because it's meant to be fun. The user does not need any computer experience, other than navigating to the directory and starting the script. After this, the computer does the rest for you.



Above is the beginning of the program. It shows the process of navigating to the directory and starting the program. The user would then input either Jedi or Sith to choose what path they will go down.



Say the user chooses the Jedi path, the computer will prompt the user again to make a decision.



If the user inputs something other than the given options (in this case, E, S, or exit), the script will say that the input was incorrect and will prompt the user again until they give an input the script recognizes.

This is the whole concept of the program. Simple, yet it gets the job done. In the next section, I will go over the script side of it and how the program functions internally.

The Script

```
  GNU nano 2.3.1                                File: begin.sh

#!/bin/bash
# This is the initial script to determine which path the game will take
echo "Hello and welcome to the Star Wars Choose Your Own Adventure game created by Jesse Russell!"
echo "Typing exit at any time will return you to the StarWarsCYOA directory."
echo "To get started, choose your side: Jedi or Sith? (This is case sensitive)" ; read input
case $input in
        Jedi) cd /home/jesserussell/StarWarsCYOA/Jedi ; bash JediDecision1.sh ;;
        Sith) cd /home/jesserussell/StarWarsCYOA/Sith ; bash Decision.sh ;;
        exit) cd /home/jesserussell/StarWarsCYOA ;;
        *) echo "Incorrect input. Try again." ; bash begin.sh
esac
exec bash
```

The script side is a little more complicated. I used the echo command to show the prompts that the computer asks. After those prompts, the script reads the input and decides what to do next. In this case, if the input is Jedi, it will change directories to the Jedi directory and runs the first decision prompt. The "JediDecision1.sh" script is the decision prompt that I mentioned earlier. I could have just had it run the scripts instead of changing directories, but I felt putting each decision in a different subdirectory would be much more organized.

```
  GNU nano 2.3.1                          File: JediDecision1.sh

#!/bin/bash
echo "You awaken in a prison cell on the Death Star, after being captured after Order 66. The emperor wanted you brought in $
echo "You have a few options. You could either (E)scape or (S)erve your sentence" ; read input
case $input in
        E) cd /home/jesserussell/StarWarsCYOA/Jedi/Escape ; bash JediDecision2.sh ;;
        S) bash DeadDeathStar.sh ;;
        exit) cd /home/jesserussell/StarWarsCYOA ;;
        *) echo "Incorrect input. Try again." ; bash JediDecision1.sh
esac
exec bash
```

If you view the contents of the JediDecision1.sh file, you should see that it's very similar to the begin.sh file. Each prompt has a similar script, just with different wording and options. If you look at the E) option, you can see that the script goes into another subdirectory, called Escape. For the sake of time, each decision only has 1 right answer. As you can see, the other leads to death. Theoretically however, one could script an entirely different path for that second option if they have the time to draw out each possibility.

```
  GNU nano 2.3.1                          File: DeadDeathStar.sh

#!/bash/bin
echo "You decide to serve your sentence like the rebel coward you are."
echo "While serving your sentence, you feel a strong presence in the force."
echo "As it turns out, this presence is none other than Luke Skywalker, coming to blow up the Death Star"
echo "The Death Star Blows up and you are dead."
echo "Try again"
bash JediDecision1.sh
```

If you die, however, you will be given the same prompt to choose the correct answer, so you don't have to start over. The script runs the same prompt again.

Trials and Tribulations

This project wasn't all sunshine and rainbows. I did run into quite a few problems that I had to fix. I knew the outcome was going to be a text-based adventure, and I knew that I wanted to organize it by directory, rather than files. I also knew that I would have to accept input and tell the script to react accordingly to those inputs. I was unsure, however, how to use the correct syntax to get the results I wanted. This is where my research came in. I learned some of it in class lectures, but other points I had to search the internet for. This is where I learned that I need to use the 'read input' command to accept the user's input. (Chadwick) Afterwards, I could not get the script to change the directory. It ran without giving me an error code, but the directory remained the same. This is when I researched and found that I need to end it with the 'exec bash' command to properly execute the change directory part of the script. (Stroodbandt, 2016) On multiple occasions, I received an error that told me the script ended unexpectedly. I fixed this by searching for errors in my code and sure enough, I found echo commands without quotations, commands missing the double semicolon at the end, and commands missing the single semicolon to run one command after another. Once I fixed these syntax errors, my program ran smoothly. Lastly, I found that retyping the script constantly was repetitive and boring. To make it easier on myself, I used to 'cp' command to copy the script to the next directory, then changed the contents of that script. This kept the formatting uniform for the most part for every script in this program.

Step by Step

Finally, here is a step by step guide to how I achieved the end result. Hopefully, this guide is intuitive enough to help others follow in my footsteps and avoid the mistakes I made.

Section 1: Create the home directory and begin script

1. Type in '*mkdir DirectoryName'* (Use your own for DirectoryName) and hit enter.
2. Type in '*cd DirectoryName'* and hit enter.
3. Type in '*touch begin.sh'* and hit enter.
4. Type in '*nano begin.sh'* and hit enter. I personally prefer nano, but any text file editor will work.
5. Your script should look similar to this (change the script to match your story, but keep a similar structure and commands):

```
  GNU nano 2.3.1                          File: begin.sh

#!/bin/bash
# This is the initial script to determine which path the game will take
echo "Hello and welcome to the Star Wars Choose Your Own Adventure game created by Jesse Russell!"
echo "Typing exit at any time will return you to the StarWarsCYOA directory."
echo "To get started, choose your side: Jedi or Sith? (This is case sensitive)" ; read input
case $input in
        Jedi) cd /home/jesserussell/StarWarsCYOA/Jedi ; bash JediDecision1.sh ;;
        Sith) cd /home/jesserussell/StarWarsCYOA/Sith ; bash Decision.sh ;;
        exit) cd /home/jesserussell/StarWarsCYOA ;;
        *) echo "Incorrect input. Try again." ; bash begin.sh
esac
exec bash
```

Section 2: Create first path

1. In the directory you have just created, type '*mkdir NewDirectory*' (substitute NewDirectory for your own subdirectory. It might be a good idea to make it related to the decision so it's easy to keep track of.) and hit enter.
2. Type '*cp begin.sh NewDirectory/Decision.sh'* to copy the begin.sh script to the new directory. (In this example, the Decision.sh file is called JediDecision1.sh)
3. Navigate to the new directory by typing '*cd NewDirectory'*

4. Edit the file you just copied by typing '*nano Decision.sh'.* Change the script so that is looks similar to this:

```
  GNU nano 2.3.1                          File: JediDecision1.sh

#!/bin/bash
echo "You awaken in a prison cell on the Death Star, after being captured after Order 66. The emperor wanted you brought in $
echo "You have a few options. You could either (E)scape or (S)erve your sentence" ; read input
case $input in
        E) cd /home/jesserussell/StarWarsCYOA/Jedi/Escape ; bash JediDecision2.sh ;;
        S) bash DeadDeathStar.sh ;;
        exit) cd /home/jesserussell/StarWarsCYOA ;;
        *) echo "Incorrect input. Try again." ; bash JediDecision1.sh
esac
exec bash
```

5. For this example, each decision either leads to progression of the story, or immediate death. If both of your decisions lead to story progression, just copy the script from before again and adjust it accordingly. To make a script that runs when the user dies on this decision, Type '*touch Dead.sh'*. (In the example, this is called DeadDeathStar.sh)
6. Edit the Dead.sh file by typing '*nano Dead.sh'.* This will run the same script again and allow the user to try the prompt again. It should look similar to this:

```
  GNU nano 2.3.1                          File: DeadDeathStar.sh

#!/bash/bin
echo "You decide to serve your sentence like the rebel coward you are."
echo "While serving your sentence, you feel a strong presence in the force."
echo "As it turns out, this presence is none other than Luke Skywalker, coming to blow up the Death Star"
echo "The Death Star Blows up and you are dead."
echo "Try again"
bash JediDecision1.sh
```

7. Repeat steps 1-6 for each decision. You do not have to call the files something different, since they are in different subdirectories. Calling each of the files Dead.sh and Decision.sh will work.

Section 3: Create the second path

1. Navigate back to the directory you created earlier with '*cd /home/username/DirectoryName*'
2. Repeat Section 2 for the second main path.

Conclusion

The possibilities with this script format are endless. Someone can create however many paths they want and get as creative as their mind allows. They can also as in depth as they need to tell the story they want to tell. This project is a lot like Legos. I've given the tools needed and an instruction guide to making your own story. Use your imagination and I believe you could create some interesting, complicated, and efficient concepts.

Works Cited

Chadwick, R. (n.d.). Bash Scripting Tutorial - 3. User Input. Retrieved from https://ryanstutorials.net/bash-scripting-tutorial/bash-input.php.

Stroodbandt, S. (2016, April 21). Script to change current directory (cd, pwd). Retrieved December 6, 2019, from https://unix.stackexchange.com/questions/27139/script-to-change-current-directory-cd-pwd.