

Muun Macdonald

40536364  
40536364@live.napier.ac.uk

Game of dockers

Documentation



To run this project, place the folder in your home directory. Using the terminal navigate into the directory where the script is and run the script. Running the script will produce a text file named final\_chapter.txt which can be compared with the provided final\_chapter\_cw.txt contained in the project folder.

1. Coursework Planning

Before starting to code, I broke down the specification to determine the different sections that would need to be worked on. There were five main parts to be developed, namely the creation of the docker containers, the sorting of the files in their respective container, the “round robin” scheduling for the text in the files to be added to the final\_chapter.txt text file, the creation of a terminal user interface and lastly the exception handling of user input. Each part was handled in the aforementioned order, as each part relied on the one before it, to work.

1. Implementation Details

The first part of the script checks whether Docker is installed on the system. If it is not, Docker is installed on the users’ system. The three Docker containers are then created using an Ubuntu image for the convenience of having a good selection of command line tools preinstalled by default. The containers were run in detached to avoid having the user encounter a terminal interface on the container after it is created. The three folders containing the text files were then sent to their respective containers.

Three scripts were created to handle the sorting of files in their respective containers and were sent to their container for execution. The scripts create a directory where renamed files can be sent. The files are renamed according to their length using ‘wc’ and then sent to the renamed directory. There, they are sorted, renamed with an incremented number, then sent to their original directory.

The files are then fetched to the users’ virtual machine where the ‘round robin’ scheduling algorithm appends the appropriate files to final\_chapter.txt. The scheduling algorithm contains of a while loop that terminates when the total number of files have been handled. Nested in the while loop are six ‘if’ statements, two for each collection of files, that check if there are still files in their respective directories. If there are, they are appended to final\_chapter.txt. Variables called FETCH1 and FETCH2 are used to select the appropriate files to append.

With the final\_chapter.txt created, the contents are then printed to the screen and the user can select actions from a menu. The ‘sed’ command was used to select text from final\_chapter.txt and delete it, by replacing the chosen text with an empty string. After each action performed by the user, they are then returned to the main menu to choose another action. To exit the program the user can enter ‘x’.

Exception handling was added last to handle unrecognised user input. From the main menu, if the user enters an unrecognised input, they will be informed that their choice was not recognised and returned to the main menu. If a user wants to delete an item and, instead of pressing enter to delete the text item, they enter some text, they are then informed of their error and sent back to the main menu.

That concludes my coursework. To run the script, open the terminal, navigate to the directory where the script is, enter ‘./SCRIPT.sh’ and enjoy playing Game of Dockers.

*(564 words.)*