Assessment Pro Forma

1. Module number	CSN08116
2. Module title	Operating Systems
3. Module leader	*Nick Pitropakis*
4. Tutor with responsibility for this Assessment	*Nick Pitropakis*
5. Assessment	*Practical Skills Assessment*
6. Weighting	*40%*
7. Size and/or time limits for assessment	*400 to 600 words for the documentation file*
8. Deadline of submission	*23:59* *21ST November 2021*
9. Arrangements for submission	*Script files and documentation must be submitted via Moodle.*
10. Assessment Regulations **All assessments are subject to the University Regulations.**	*No exemptions.*
11. The requirements for the assessment	*See Attached.*
12. Special instructions	*None.*
13. Return of work	*Feedback is provided via Moodle*
14. Assessment criteria	*See Attached.*

CSN08116 Operating Systems

Operating Systems
Practical Skills Assessment

The purpose of this coursework is to introduce you to the functions performed by an operating system and, in particular, to give you practical experience of the associated command language, the file system facilities and Docker Containers.

On completion of this coursework, you will be able to:

(a) Use the facilities provided by a Linux-based system and particularly the file attributes and directory structures.

(b) Use the command line associated with a Linux-based system, including some commonly used commands and the widely encountered structures.

(c) Use Docker containers and their interaction mechanics.

(d) Appreciate the strategy associated with testing systems software and the need to build robust products.

(e) Understand and implement 'best practice' housekeeping procedures to prevent user/system misuse or mistakes.

Game of Dockers: A song of VMs and Scheduling

The story so far…

As the book sales are going up, your publisher has asked you to deliver the book you promised a couple of years ago, and they are after your next bestseller. There is one problem, you have done nothing so far, and you need to save the day. The solution to your problem will be your knowledge of operating systems and Docker containers.

Instead of writing your bestseller, you have spent the past few years experimenting with Docker containers and scripts. Inside three of your Docker containers, you have stored several text files which were going to be used for your book. Somehow you need to arrange those files and create a small book chapter so you can buy some time for yourself. Mother luck might be able to help you… but how???

Description

You need to use the scheduling algorithms as a tool towards your redemption, and your matriculation number will be the key for the algorithms to work. On Moodle, you will find three folders with text files inside them. The text files have random names. Each folder corresponds to each of the Docker containers you need to create. You have three folders, thus leading you to create three Docker containers and upload the content of those folders to them.

Your task is to assemble your texts in one unified text file. The technique that you are going to follow is Round Robin. To create your book chapter, you need to create your Final Book Chapter text file from all the Docker containers attached to your Linux distribution. Following the Round Robin algorithm that you learned from scheduling, with quantum as two text files, you need to gather two files in each round from each Docker container and attach them to a Final Book Chapter text file. As an illustration, you need to collect the text of the first two text files of your first Docker container, then the text of the first two text files of your second Docker container, then the text of the first two text files of your third Docker container, then the text of the third and fourth text files of your first Docker container, the text of the third and fourth text files of your second Docker container, the text of the third and fourth text files of your third Docker container etc. You should be careful as the number of text files is different for each Docker container.

Taking a closer look at your text files folders, you will notice that their names are randomized, and you need an efficient way to tide them up. To achieve that goal, an efficient way is to rearrange them according to the length (in characters) of their text files. You will use your matriculation number, which you are going to divide by three. If the remainder of the division is:

- 0, then:
    - you will leave the files of your Docker container 1 intact (First Come First Served), and
    - you will rearrange the files of your Docker container 2 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.), and
    - you will rearrange the files of your Docker container 3 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.)
- 1, then:

    - you will rearrange the files of your Docker container 1 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.), and
    - you will rearrange the files of your Docker container 2 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.), and
    - you will leave the files of your Docker container 3 intact (First Come First Served)
- 2, then:
    - you will rearrange the files of your Docker container 1 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.), and
    - you will leave the files of your Docker container 2 intact (First Come First Served), and
    - rearrange the files of your Docker container 3 with Shortest Job Next (first the file with the smallest length, then the file with the second smallest length, etc.)

You should develop scripts that achieve the described functionalities and present to the user what is happening at each time. For example:

Creating Docker containers….

Docker 1 created…

Docker 2 created…

Docker 3 created…

Loaded files to Docker 1…

Loaded files to Docker 2…

Loaded files to Docker 3…

Beginning text creation GAME_OF_DOCKERS.txt…

Loading n^{th} text from z^{th} Docker container…

……..

Finished loading text…

Would you like to read Game of Dockers Chapter? Yes/No

….

Would you like to remove any text from Game of Dockers? Yes/No

Would you like to add any text to Game of Dockers? Yes/No

Would you like to terminate the program? Yes/No

From the example, you can see that you also need to create a terminal user interface so you can help yourself with the chapter writing. Initially, the terminal user interface should reflect all the necessary steps that create the initial text. Soon afterwards, you should implement the option that lets the user read the text through the terminal user interface. The next option is for the user to search and remove text from the book chapter, and the last option lets the user add text to the end of the book chapter.

The communication between the user and the terminal is presented in an abstract way on purpose to give you the flexibility to design your script according to your taste.

Your script should cover all of these functionalities:

- **Load the files to the appropriate Docker containers**
- **Create the final text according to the described algorithm**
- **Allow the text to be removed or added**

Your script should contain comments that explain the functionalities that you have implemented. Along with your script, you should submit a documentation file (a word document). The documentation file will explain your work plan, and if necessary, code parts that you think are not clarified by your comments. The length of your documentation file should be at least 400 words up to 600 words. **Without the documentation file, your coursework will not be marked.**

This is an individual project. You are responsible for your own project. You should code carefully, avoid plagiarism, and provide an accurate description in the documentation file.

You need to submit:

1. **Your script files**
2. **The created final text file of the chapter**
3. **The documentation file**

Submission Deadline: Sunday November 21st 2021 at 23:59

Marking scheme:

Documentation File	5
Embedded #comments	5
Create Docker containers and load text files	5
Copy the text of the files to the Final Book Chapter text file	12
Exception handling of incorrect/missing user input arguments	5
Remove custom text	4
Add custom text	4
Total	40