Canterbury Christ Church University

School of Engineering, Technology and Design

Undergraduate Computing Suite

2019-2020

Operating Systems (MCOMD3PST)

Date Set: 26-Nov-2019

Assignment 2

General guidelines for submission

- This is an individual submission and must be your own work.
- The required date of submission is **08-Jan-2020 before 14:00.** Scripts should be submitted via normal Blackboard, and the written part via Blackboard (Turnitin) – see below for details. **If you experience any problems with these systems, then please contact Andy Hook (**andy.hook@canterbury.ac.uk) **.**
- This assignment has been set by **Andy Hook**
- This assignment has been moderated by Dr Hannan Azhar.
- This assignment is worth **50%** of the individual course mark

General advice

- You are required to back up your work regularly, both on the campus network and on removable storage devices, and in any other suitable way. Always check the date stamp on your files before submission.
- You *must* submit your work using the versions of software currently in use on the University's network. This means that your scripts should run correctly on the student UNIX system, and that your report should be either a PDF file, or prepared using a compatible version of Microsoft Word.
- Please note that a high standard of presentation is expected. Reports should have an appropriate title page and contents page, and pages should be numbered.
- The title page should have the following information:
  - Module code
  - Module name
  - Assignment number and title
  - Lecturer's name
  - Student's name
  - Submission date
- You are required to use the originality checking feature available in Turnitin to help improve your academic report writing skill.
- You are advised to perform the originality check at least 2 days before the hand-in date to give you sufficient time to benefit from the feedback you receive from Turnitin. You can submit your work to Turnitin as many times as you wish, but only the most recent submission will be marked.

Before you start

C is a system programming language, meaning it is ideal for writing software for operating systems, in contrast to application programming where you are creating software for end-users.

You will have gained some experience writing C programs in the lab classes, however there will be tasks where you will need to do further research to meet the criteria. It is suggested that you look at file and directory handling in C (i.e. how to list the contents of a directory, how to create and delete files etc.). You will also need to think about how filing systems store files across blocks and keep this in mind whilst writing your program as it will ensure you have an appreciation for what you are doing. Documentation will need to be comprehensive for this part of the assignment, and you will be expected to explain your use of I/O programming techniques in your C code, along with your use of libraries and the relationship with the software you are writing to explain the advantages of device independence.

Lastly, for the mini-essay, you will need to become familiar with the similarities and differences between filesystems, and consider the technical characteristics of these and I/O hardware.

Learning outcomes tested by this assignment

- Evaluate critically different operating systems in terms of their technical characteristics, performance, reliability and related aspects;
- Describe typical characteristics of I/O hardware and explain the purpose of various I/O programming techniques and explain the advantages of device independence;

The task (overview)

Your task is to develop your own filing system using the C programming language. The idea is that by doing this, you will gain an understanding and appreciation of the challenges faced by filesystem designers.

Your program should provide the following:

1. **Program entry point (wrapper)**
2. **Basic help**
3. **Ability to store simple text files of varying sizes**
4. **Ability to retrieve stored files**
5. **Ability to update a file** (you need to handle files with the same name that are smaller or bigger than when they were originally stored)
6. **Ability to delete a file**
7. **Detailed logging of both successful events (informational) and errors**
8. **Comprehensive code commenting**

It is suggested you create a new directory to contain your work whilst you are developing the solution. As you will be creating a “simulated” filesystem, it is a requirement that your data storage follows this directory structure within your working directory:

DISK

- **BLOCK0**
- **BLOCK1**
- **BLOCK2**
- **etc…**

More detail

1) Program entry point (wrapper) [15 marks]

As with many programs, your filesystem program is required to handle multiple commands. That is, the user should be able to make use of different functionality that your program exposes. For example, when storing a new file in your filesystem, the user should be able to run a command such as **./myfs store data.txt**. You will need to develop an appropriate strategy in order to decode/interpret the command that the user has provided and alter the path of execution through your code accordingly.

2) Basic help [2 marks]

If your program is called without any arguments, it must output a useful help message to standard error, including how the user can correctly use the program. You should also use an appropriate exit code to indicate that the execution ended in error.

3) Ability to store simple text files of varying sizes [20 marks]

Your filing system must be divided into blocks of **512 bytes**. This means that if the user of your program wishes to store files greater than this size, it will need to be divided up and stored across multiple blocks.

It is suggested you create a **"Documents"** directory at the same level as your **DISK** directory so that you can run a command such as **./myfs store Documents/test.txt**.

Clearly, if the first file to be added to your filesystem is less than 512 bytes in size, it will fit into the first block. However, if other files have already been stored, you will need to be able to determine the first available block with enough space to store a part, or the whole file.

4) Ability to retrieve stored files [13 marks]

In the previous section you will have written program logic that is able to split files into sections in order to store them across your simulated disk in blocks (subdirectories). This section is about bringing those sections of files back together so that the file can be returned to the user in its original state.

For example, your program will need to implement a command such as **./myfs retrieve test.txt Output/test.txt**. It should determine where the parts of this file are stored across your filesystem and concatenate them back together in the right order. In this example, the first argument will be your program name, the second is the command, the third is the file to be retrieved and the fourth is where the retrieved file should be written to.

5) Ability to update a file [10 marks]

As mentioned in the task overview, your program will need to handle the situation where a user has already stored a file in your filing system, and they wish to update it. This should not be implemented as a separate command in your program, instead it should form part of the **store** command.

Handling a file that has increased in size should be straightforward, however you will need to think more carefully about how to handle a file that has decreased in size, since it is possible that the file will now not consume space across as many blocks.

6) Ability to delete a file [5 marks]

Your program will need to implement the ability to delete (remove) a file. It should remove all parts of the file stored across as many blocks as were necessary when originally storing the file.

7) Detailed logging [8 marks]

You should provide a daily log file in a separate "logs" directory. This should be at the same level as your main **DISK** directory. You will need to report at least the following:

- File stored successfully and where (informational)
- File not able to be stored (error)
- File retrieved successfully (informational)
- File not retrieved successfully (error)
- File updated successfully (informational)
- File not updated successfully (error)
- File deleted successfully (informational)
- File not deleted successfully (error)
- Program called without any arguments (error)

Each entry in the log file should be timestamped. Do not overwrite any log files, append to them.

8) Documentation [7 marks]

You will need to ensure that your code is thoroughly commented in order that the marker can clearly see what it is you are intending to achieve with each section of your code. Also include comments which show your understanding of the programming techniques you are using, for example why you decided to use dynamic memory allocation or to work with files in blocks as opposed to individual characters etc.

Include a separate README file that provides information on the files contained within your working directory (which you will upload to Blackboard). Also describe your use of libraries and explain the advantages of device independence.

9) Mini essay [20 marks]

Research into the **NTFS** and **ext4** filesystems.

Write a mini essay (target 800 words), comparing and contrasting these two filesystems. You do not need to go into great detail on each (remember the word limit), but the reader should be able to come away with a reasonable insight into the critical differences that would enable them to make an informed decision when selecting a filesystem.

You should investigate the technical characteristics, performance and reliability considerations in relation to each filesystem to form the basis of your mini-essay. As part of your research, remember to consider the typical technical characteristics of I/O hardware, for example hard disks are "block devices". You will have likely worked with "blocks" of data during your C programming lab classes.

Assessment

You will be assessed on your ability to present clear explanations and precise descriptions. The written work must show your understanding of the subject matter and be Harvard referenced throughout.

This report will be marked in accordance with the Level 6 Generic Assessment Criteria for Undergraduates available at:

http://www.canterbury.ac.uk/quality-and-standards-office/assessment-criteria.aspx

Deliverables and submission

There are two separate deliverables for this assessment. You *must* submit both.

1. Upload a **zipped** version of your scripts to Blackboard (*not* the Turnitin part for Assignment 2). Include copies of your C source code, original “Documents” directory, generated “DISK” directory structure, output files as well as logs. Details will be provided in the same location where you downloaded this assignment.
2. Upload your write up onto Blackboard (Turnitin); a link will be provided in the same location where you downloaded this assignment.

A separate document will be provided in the same place as this one to provide assistance and details of file transfer between UNIX and Windows.

Failure to provide an adequate write up will result in SCALING of code marks.

Indicative mark scheme

Task	Mark	Subtotal
1) Program entry point (wrapper)		
Decode and interpret arguments, control flow	4	
Functional Decomposition, Abstraction of code	6	
Error handling	3	
Correct use of return and exit codes throughout where necessary	2	
		15
2) Help		
Useful help message printed to standard error, including guidance	2	
		2
3) Ability to store simple text files of varying sizes		
Simulated filesystem using directories (BLOCK0, BLOCK1 etc.)	5	
Each block to contain <= 512 bytes	5	
Determination of first available block	5	
Correct segmentation of file parts	5	
		20
4) Ability to retrieve stored files		
Correct location of file parts according to passed argument	3	
Correct assembly/concatenation of file parts (in sequence)	7	
Output to path as specified in passed argument	3	
		13
5) Ability to update a file		
Remove unnecessary file parts if file now requires less blocks	5	
Remove unnecessary blocks if they are no longer required	5	
		10
6) Ability to delete a file		
All parts of file to be removed from the simulated filesystem/disk	5	
		5
7) Detailed logging		
New log file to be used each day (format: YYYY-MM-DD)	2	
Event timestamp in the format: [12/10/2019 14:20:00]	2	
Logging of events as specified under "Detailed logging"	2	
Log file should be appended to, not overwritten	2	
		8
8) Documentation		
Comprehensive code commenting	4	
README file with information/instructions	3	
		7
9) Mini Essay		

800 word write up	20	
		20
TOTAL		100