Systems Software

DT211/DT228/DT282 Year 4 - Continuous Assessment 1 (15%)

Due Date: 16th March 2018 @ 5pm

Introduction:

A e-commerce company needs to track changes made to its static html website. Problems have occurred in the past where changes were made to the website incorrectly and it wasn't possible to track who made the changes. The company's CTO has listed the functionality they would like to include in their new business model to offer transparency and accountability for all changes made to the website.

How the website management currently works:
Users have an account on a Debian server, they can login and make changes to the website. All changes will be made under their user accounts. The Debian server is running Apache as the webserver. All changes made to the site /var/www/html appear on the website instantly.

What the CTO wants:

The CTO has offered a list of desired functionality for the new website management model:

1. The company will have an internal Intranet site that is a duplicate copy of the live website. Staff can make changes to the Intranet version of the site and see the changes before it goes live. (This will help prevent content issues and page availability issues for users of the site).
2. The website content should be backed up every night.
3. The changes made to the Intranet version of the site needs to documented. The username of the user, the page they modified and the timestamp should be recorded.
4. The live site needs to be updated based on the changes made to the Intranet site. This should happen during the night. There are a large number of files on the website (5000+), only the files that have changed should be copied to the live site folder.
5. No changes should be allowed to be made to the site while the backup/transfer is happening.
6. If a change needs to be urgently made to the live site, it should be possible to make the changes. (Users shouldn't have write access to the new website folder)

Project Requirements:

a. Create a daemon to continually manage the operation of the requirements listed by the CTO above.
b. Identify new or modified site content and log details of who made the changes, this should be generated as a text file report and stored on the server.
c. Maintain a list of site updates (file)
d. Update the live site every night (After the backup completes).
e. When the backup/transfer begins no user should be able to modify and site content.
f. It must be possible to ask the daemon to backup and transfer at any time.
g. A message queue should be setup to allow all processes to report in on completion of a task. (success or failure)
h. Error logging and reporting should be included (log to file)
i. Create a makefile to manage the creation of the executables.

General Assumptions:

1. The company only has one server
2. Backups can be made to a different folder on the server
3. The Intranet and Live site can be two sub directories of /var/www/html

Deliverables:

20%	Project Report (explaining the approach taken and the operation of the application etc.)
65%	C Program Solution (all code and supporting docs uploaded to WebCourses)
5%	5 minute video screen recording showing and verbally describing the operation of your solution. The video must address all the project requirements listed above.

Note: Students may be required to demonstrate their project operating in one of the labs sessions. Non-compliance with this request will result in a zero grade.

Grading Rubric:

	70 +	69 – 60	59 – 50	49 – 40	39 - 0
System Architecture including makefile (10%)	Program follows excellent design principles and demonstrates the proper implementation in the application.	Program good excellent design principles and demonstrates the proper implementation in the application.	Program follows good design principles and demonstrates the proper implementation in the application. Solution contains architectural design flaws and/or demonstrates some principles separate from the application.	Program demonstrates a minimal implementation of design principles. Solution contains architectural design flaws, tight coupling and/or demonstrates principles separate from the application.	Program does not demonstrate the operation of the design principles. Major omissions and lack of understanding of design in the Linux environment.
Daemon (10%)	Background process created and completely decoupled from Terminal and IO. Appropriate error checking and signals used.	Background process created and completely decoupled from Terminal and IO.	Process created. Some issues with the process running in background or with decoupling from Terminal and IO.	Process operates, some of the following not working: not in background, still linked to terminal, issues with file permissions and IO.	Damon process does not operate or does not operate as expected.
Backup and Transfer (20%)	Backups architected and implemented correctly and fit for purpose in a commercial environment.	Backups implemented and working correctly. Some additions needed for a real world solution.	Backups working correctly. Some features omitted or not working as expected.	Some aspects of the backups operates, minimal attempt.	No backups or transfers implemented correctly or not fit for purpose.
Reporting (10%)	Reporting implemented correctly for all aspects of the program.	Reporting implemented correctly for all most of the program.	Reporting implemented correctly for some aspects of the program.	Minimal Reporting implemented in the program.	No Reporting implemented correctly for all aspects of the program or not fit for purpose.
Error Logging and messages (10%)	Error logging and IPC implemented correctly for all aspects of the program.	Error logging and IPC implemented correctly for all most of the program.	Error logging and IPC implemented correctly for some aspects of the program.	Minimal Error logging and IPC implemented in the program.	No error logging and IPC implemented correctly for all aspects of the program or not fit for purpose.
Functionality and Operation (15%)	Application operates as per the requirements and has advanced functionality included in the application.	Application operates as per the requirements.	Adequate functionality, application does not achieve all the functionality in the requirements.	Minimal functionality, application does not achieve the functionality in the requirements.	Application does not perform the required functionality.
Documentation (20%)	The documentation is well written and clearly explains all architectural choices and functionality of the system	The documentation is well written. Could have explained the code and the principle in more detail.	The documentation is acceptable. Could have explained the code and the principle in more detail. Omissions of content or misinterpretation of the principle demonstrated.	The documentation is minimal or not focused on the problem description. Could have explained the code and the principle in more detail. Omissions of content or misinterpretation of the principles demonstrated.	The documentation is simply comments embedded in the code and does explain the code or the principle. Minimal attempt in all aspects.
Exemplar Video (5%)	Video is well prepared and shows and describes the exact operation of the solution. Complex aspects of the solution have been described in good detail.	Video is well prepared and shows and describes the exact operation of the solution.	The video is acceptable. Could have explained the code and the solution in more detail. Minor omissions of content or detail in the video demonstration.	The video is minimal or not focused on the problem description. Could have explained the code and the solution offered in more detail. Omissions of content or detail in the video demonstration.	Video doesn't capture the operation of the solution and/or doesn't offer a verbal description of the functionality of the system from a code perspective.

Notes: