

Systems Software Report CA1

DT228

BSc in Computer Science

**Gary Kelly**

**C16380531**

School of Computer Science

TU Dublin – City Campus

**13/03/2020**

**Table of Contents**

*Functionality Checklist* 3

*Feature 1 - System Architecture including makefile* 4

*Feature 2 - Daemon (Setup/ Initialisation/ Management)* 4

*Feature 3 - Daemon (Implementation)* 4

*Feature 4 - Backup Functionality* 5

*Feature 5 - Transfer Functionality* 5

*Feature 6 - Lockdown folder for Backup / Transfer* 5

*Feature 7 - Reporting (IPC)* 5

*Feature 8 - Logging and Error Logging* 5

*Conclusion* 6

# *Functionality Checklist*

|  |  |  |
| --- | --- | --- |
| ***Feature*** | ***Description*** | ***Implemented*** |
| F1 | System Architecture including makefile | Yes |
| F2 | Daemon (Setup/Initialisation/Management) | Yes |
| F3 | Daemon (Implementation) | Yes |
| F4 | Backup Functionality | Yes |
| F5 | Transfer Functionality | Yes |
| F6 | Lockdown folder for Backup / Transfer | Yes |
| F7 | Reporting (IPC) | Yes |
| F8 | Logging and Error Logging | Yes |

Have you included a video demo as part of the assignment: No, arranged a video call demo as I was having technical difficulties with OBS.

Link to Video: please paste link here

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

A close up of a logo

Description automatically generatedSigned:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gary Kelly

13/03/2020

# *Feature 1 - System Architecture including makefile*

A screenshot of a cell phone

Description automatically generated

Figure 1 System Architecture

Shown above is a diagram showing the design choices made for the architecture of this system. This architecture was designed with the single responsibility principle and separation of concerns in mind. Each function is written for a sole purpose and is called by the daemon when needed. A makefile was used to compile the program for ease of use. In the following sections I will go through each of the functions shown above and outline how they operate.

# *Feature 2 - Daemon (Setup/ Initialisation/ Management)*

In order to setup the daemon service I followed 4 steps:

1. Save the daemon in the /usr/sbin directory using the *cp daemon /usr/sbin/daemon* command.
2. Write a basic startup script in the /etc/init.d for the daemon. This script includes the ability to start and stop the daemon on request or get the status of the daemon by typing *service daemon start/stop/status.*
3. Change the permissions of this script so it can execute using the *chmod 755 daemon* command.
4. Include the script in the startup list with low startup priority using the *update-rc.d daemon defaults 97 03* command

# *Feature 3 - Daemon (Implementation)*

Detailed description of the process followed to create the background process.

In order to create the daemon, I followed five steps:

1. I first created an orphan process by forking and killing the parent process.
2. Then I elevated the orphan process to session leader using the setsid() command so that it is no longer associated with a controlling terminal.
3. After that I called umask(0) to give the daemon the necessary permissions to read and write files.
4. Then I changed the current working directory to root using the chdir(”/”) command.
5. Lastly I closed all file descriptors by looping through them and using the close() command to release all the IO connections that the child inherited from the parent.

# *Feature 4 - Backup Functionality*

The backup functionality works using the cp -r command which recursively copies files from the Live\_site directory to the Backup directory. All the backup functionality is contained in the backup.c file which is linked to the main daemon script through the backup.h header file. Backups are called at midnight every night before a transfer is completed and is also available on request by running the client.c file and sending the backup message to the message queue.

# *Feature 5 - Transfer Functionality*

The transfer functionality works using the rsync -r command which which recursively copies files from the Intranet directory to the Live\_site directory. All the transfer functionality is contained in the transfer.c file which is linked to the main daemon script through the transfer.h header file. Transfers happen every night at midnight after the backup is completed and is also available on request by running the client.c file and sending the update message to the message queue.

# *Feature 6 - Lockdown folder for Backup / Transfer*

Before a backup or transfer the Intranet and Live\_site directories permission is changed to 000 for all groups. This ensures no user can access any files while the transfer is happening. After the backup or transfer is completed permissions are reverted back to read and write for Intranet and read only for the Live\_site. This is done using the chmod function and all lockdown functionality is contained in the allowAccess.c and denyAccess.c files.

# *Feature 7 - Reporting (IPC)*

A message queue was used in order to request a backup or update from the daemon at any time. A user can communicate with the daemon by running the client program which opens the message queue and continues to get messages from standard input and send them to the server using the mq\_send() command. The server code is constantly running on the daemon ready to receive messages and comparing them with the chosen keywords backup and update to see which functionality to implement.

# *Feature 8 - Logging and Error Logging*

All logging is done using syslog. After each function is called, we log:

* Process ID
* User
* Whether the function was successful / unsuccessful
* If unsuccessful the error code returned

Each file for the various functions uses error logging messages which clearly identify what aspect of the program failed and includes the file name to clearly identify which file the error occurred in.

# *Conclusion*

In conclusion, the above architecture fits the requirements in full. All the functional requirements listed by the CTO have been completed. I am very pleased with my solution to this task and I learned a lot in doing it about working in a linux based environment.