

Systems Software Report CA1

TU856

BSc in Computer Science

**Philip Herweling**

**C18470774**

School of Computer Science

TU Dublin – City Campus

**07th of November 2021**

**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 directories for Backup / Transfer* 5

*Feature 7 – Process management and 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) | No |
| F3 | Daemon (Implementation) | Yes |
| F4 | Backup Functionality | Yes |
| F5 | Transfer Functionality | Yes |
| F6 | Lockdown folder for Backup / Transfer | Yes |
| F7 | Process management and IPC | Yes |
| F8 | Logging and Error Logging | Yes |

Have you included a video demo as part of the assignment: Yes

Link to Video: <https://youtu.be/kJxIycxNGsI>

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.

Signed:

A picture containing text, whiteboard

Description automatically generated

Philip Herweling

07/11/2021

# *Feature 1 - System Architecture including makefile*

Detailed description of the system architecture choices made.

How Separation of Concerns (SoC) and Single Responsibility Principle (SRP) was followed.

Separations of concerns I believe I followed very well. My program I believe is split onto distinct sections, each section doing a different thing. I have different files and functions for every part of the assignment. I have a backup.c file which holds my backup function, I have a sync.c file which holds my sync function for the transfer, I have an access.c file which has four function to lock and unlock two directories. Each of these files address a separate concern. myDaemon.c is where all these functions are called. myDaemon is also split into separate parts, each part addressing a separate problem.

In my opinion I followed the SoC and SRP very well in the making of my program.

My make file compiles all the files I have in my program and allows me to run them all at once.

Text

Description automatically generated

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

Detailed description of the daemon setup:

Startup Script and init process

Daemon control options

Couldn’t get it working properly. Did include my attempted code in the hand up.

# *Feature 3 - Daemon (Implementation)*

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

My Daemon file is where I create a background process and call on all the functions I made.

1. So I start off by creating a child process. I do that by using the fork command.
2. After creating the child process I exit/ close the parent process
3. I then elevate the orphan process to the session leader, this allows me to lose the controlling TTY
4. I then make another fork to ensure that the process is not the session leader, if this is successful, we close it down.
5. I then change the working directory from dir to root
6. Then I close the alaureport open file descriptors.

I have now successfully created a daemon and added in a signal handler which allows authorised users to run the transfer and backup function on demand. I also added in a auditing system which audits the Intranet directory. The last thing I added into my daemon was a function which counts down until 1am, when it hits 1am the Intranet and Live directories are locked a transfer and backup happens and the directories are then unlocked. Also when it hits 1am the audit logs are stored in the Logs directory.

*Feature 4 - Backup Functionality*

Detailed description of the backup implementation

For the backup I first made two new files, backup.c and backup.h. In the backup.c file I started off by making a backup method. In this method I first make a variable called destination where I store the location of where I want the backups to be stored and concatenate on the current time. The reason I add the current time stamp is so that when a backup is done, the new folder will be renamed to the current date and time. After that I then make a char pointer array where I store the command name (cp), the path of the folder I want to make a backup of and then I store the path to the backup folder where I want to store the folder, I just copied i.e. the destination variable I made at the start of the program. I then fork the program and get the exec to run a different process in the fork.

I then have a simple if / else statement. If the pid == 0 i.e. if it’s the child process I have a printf displaying that the cp command is executing and then I execute the command.

I then call the backup function in my daemon twice. Once I call it when the time hits 1am and the second is in my signal handler which allows one to make a backup at any time.

# *Feature 5 - Transfer Functionality*

Detailed description of the transfer implementation

For the Transfer I first made two new files, sync.c and sync.h. In the sync.c file I started off by making a sync method. I then make a char pointer array where I store the command name (rsync), the path of the folder I want to make a transfer from and then I store the path to the live folder where I want to store the files I just synced. I then fork the program and get the exec to run a different process in the fork. The reason I use the rsync command instead of the cp command like I did in the backup is because I don’t want to add new files every time a change is made. I just want to sync any changes made or any files added in the Intranet folder and add these new files made to the live directory and any changes made to existing files will also be made to the files in the live directory.

I then have a simple if / else statement. If the pid == 0 i.e. if it’s the child process I have a printf displaying that the rsync command is executing and then I execute the command.

I then call the sync function in my daemon twice. Once I call it when the time hits 1am and the second is in my signal handler which allows one to make a transfer at any time.

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

Detailed description of the lockdown functionality/implementation

For the lockdown functionality I made a file called access.c which contained 4 functions.

The four functions are lockIntranet, unlockIntranet, locklive, unlocklive.

1. lockIntranet()

This function locks the Intranet folder when a transfer to the live happens. In this function I create a mode variable which I use to change the permissions on the folder after. In the mode variable I store “0700”, this means the folder can be read and no one can add anything to the folder or edit files in the folder.

I then create a variable called buf where I store the path to the folder I want to lock i.e. “/var/www/html/Intranet”.

I then created an integer variable called ‘I’. I then set I equal to the mode by using the strtol() function, which converts strings into long integers.

I call this function before calling the sync function in myDaemon.

Text

Description automatically generated

lockLive() function is basically the exact same as the lockIntranet folder except the path is changed to “/var/www/html/Live” because this time the function closes the live folder.

Both unlock functions are also very similar except the paths change depending on the folder the function is unlocking and also the mode variable is set to “0777” which allows authorised users to read, write and add files to the folder.

# 

# *Feature 7 – Process management and IPC*

Detailed description of how child processes communicate success/failure of tack to be completed to parent process etc….

For this I implemented a signal handler in myDaemon file. The signal handler is set up to catch signals in particular the two I picked i.e. SIGIO and SIGPWR. When the signal handler picks up the SIGIO signal it makes a transfer from the Intranet folder to the Live folder. If the signal handler picks up SIGPWR it does a backup from the live folder to the backup folder.

When these signals are run it basically interrupts the software in this case it would interrupt myDaemon file which is counting down till the next transfer and backup at 1am. myDaemon then calls on the signal handler function to deal with this interrupt.

Text

Description automatically generated

# *Feature 8 - Logging and Error Logging*

Detailed description of the error and logging functionality included in the code solution.

For the logging functionality I made a new directory called Logs in my html directory. In Logs I created a text file called Auditlog.txt where I save all the audit logs at 1am every morning.

In myDaemon I make an auditctl call which basically audits every time a user writes, reads, executes, or appends any files in the Intranet folder. After doing this I then log the results in the Auditlog.txt file in the Logs directory but first I tidy up the logs, so I do an ausearch on the logs from the Intranet directory. After that I then use aureport -f -I command, this tidies up my logs and makes them a lot easier to read and understand. After that then I pipe my results into the Auditlog.txt file in my Logs directory.

# *Conclusion*

This was a challenging assignment but enjoyable. I feel like after completing I have gained a lot more knowledge about the module. I think for the most part I was able to implement it well, the only really struggle I had saws implementing the daemon.service. I think everything else I completed to the best of my abilities. I have everything else working and running.