Operating Systems COMP30640

Project

Password Management System

Student: **John O'Meara**

Student Number: **19200612**

Introduction

This represents an implantation of a comprehensive password management system using bash script files.

Requirements.

A server process running with a pipe to receive instructions

A user interface based around a client script that takes various parameters and sends requests to the server in the appropriate format. The client script also creates a client pipe to receive feedback and communication from the server.

Server requests:

Init	Creates a new user folder and feeds back success or error messages to the client
Insert	Creates a file and subfolder(s) within the user folder recording supplied logon credentials for the relevant service. Again, appropriate success or error message are communicated back to the client
Show	Displays the logon credentials for the service in question. Again, appropriate success or error message are communicated back to the client
ls	Displays the folder and service tree structure for a user or a folder within a user folder. Again, appropriate success or error message are communicated back to the client.
edit	retrieves the logon credentials and allows the user to edit them, then passes the new data back to the server to update the recorded credentials.
Rm	Removes a service for a user
Shutdown	Shuts down the server.

Challenges faced

Section 2

2.1 init.sh

No real issues other than syntax

Enclosed $1 in in weak quotes to allow entry of spaces

2.2 insert.sh

Hard getting grep and regex to cut out part referring to path

Getting grep to read a sample string from another string

implemented mkdir -p to allow multiple subfolders – had to reference the last / in regex for the path

-e parameter on echo to allow interpretation of \n thus putting login and password on separate lines

2.3 show.sh

echo won't accept a pipe from grep, so the output of grep has to be passed as a variable to echo

2.4 alter insert.sh

No issues

2.5 rm.sh

No issues

2.6 ls.sh

Had to use a nested if statement to allow for one or 2 variables – initially threw a "folder doesn't exist" error when only using one variable"

Section 3

I had to use -r modifier to read backslashes rather than interpreting them (this allows the \n in the file info to be passed without parameter expansion)

I had to surround variables in each case with quotes to pass parameters correctly or else the script would split on whitespace

But now the quotes get written into the service file – used sed to remove

I piped the echo to one sed that removed the first char from the first line and then piped the output of that to another sed that removed the last char of the second line and then fed output to the service file

Semaphores – had to specify $1 and $2 as parameters – threw an error with $1 only

Section 4

Had to update parameter errors in each script to account for empty parameters passed to server by the client

insert – getting it to go to a newline for the prompts for password – used command substitution quotes

Had some trouble with how show.sh was outputting 2 lines so changed how it pulled the data from the service file with Grep Sed to output 2 variables in a row and read that into two variables in client

Edit Had to use >> to append rather than overwrite to second line

Extracted password and login from tempfile using Grep + sed

Included extra quotes around password and login string to account for the sed extraction of same in update.sh

Conclusion

As described in the assignment sheet, each component of this system was simple in and of itself, but together they combine to create a complex system. There were a few tricky challenges along the way – primarily due to parameter expansion and data extraction using appropriate Grep, sed and regex statements.

The system would benefit from another frontend script that prompts the user for which function is required and prompts for each parameter required in turn as appropriate.