**MOD: 003218**

**Operating Systems Assignment**

**SID 1503126**

**Contents Page**

**Introduction**………………………………………………………………………………………………………Page 3  
**Using the Back-Up Utility**…………………………………………………………………………………..Page 3  
Menu Mode……………………………………………………………………………………………………….Page 3  
Performing a Menu Mode Back-Up…………………………………………………………………….Page 3  
Creating a Back-Up script……………………………………………………………………………………Page 5  
Help!.......................................................................................................................Page 6  
Using the Quick Copy…………………………………………………………………………………………Page 6  
**Testing**……………………………………………………………………………………………………………….Page 7  
**The Code**……………………………………………………………………………………………………………Page 8  
The MS Dos Code……………………………………………………………………………………………….Page 8  
The BASH Shell Code………………………………………………………………………………………….Page 12

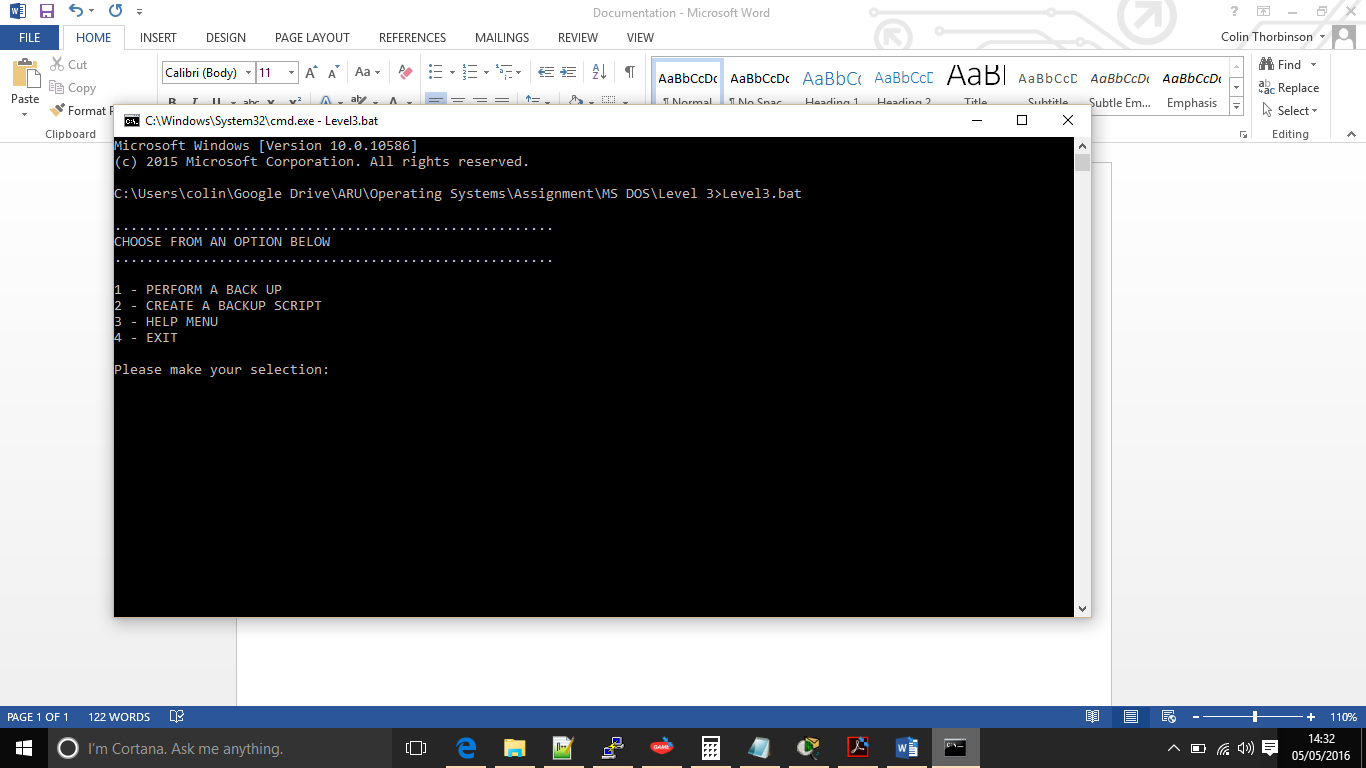
**Back-Up Utility Introduction**

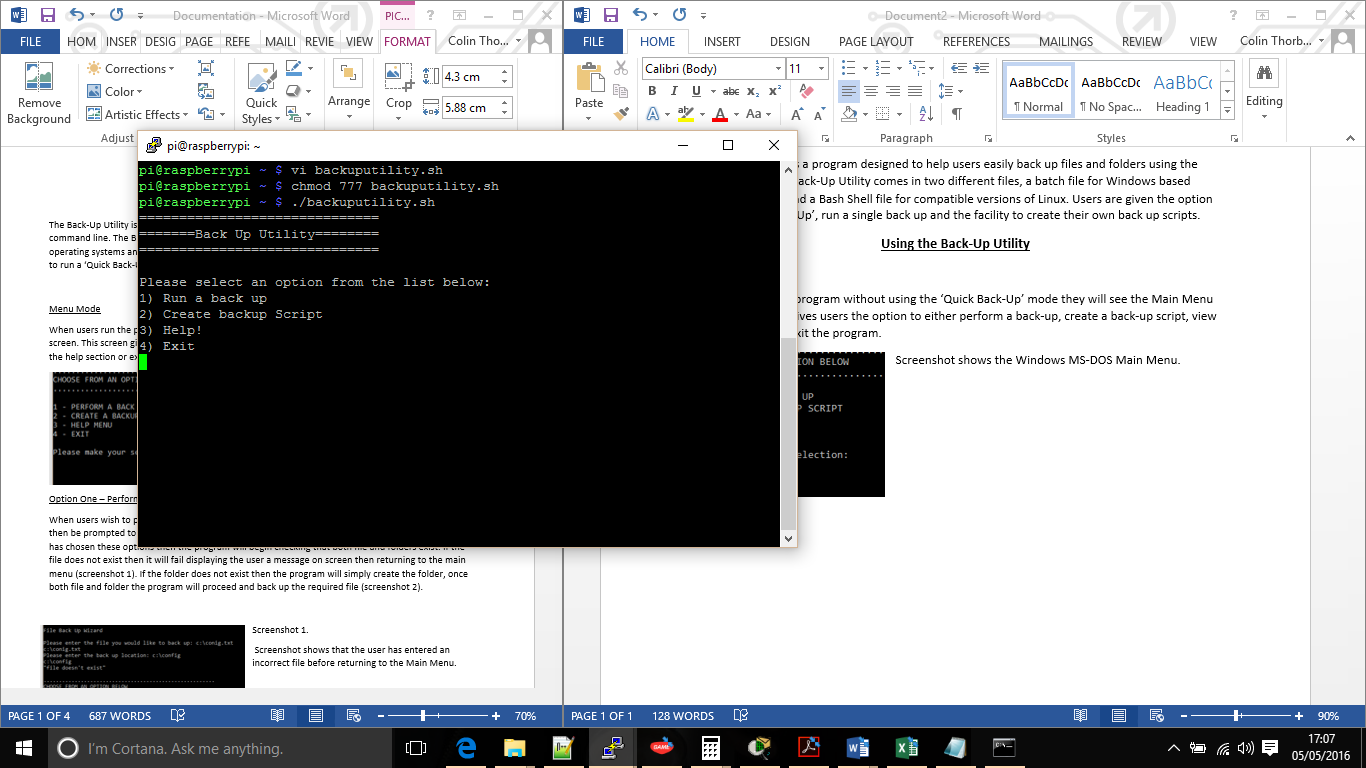
The Back-Up Utility is a program designed to help users easily back up files and folders using the command line. The Back-Up Utility comes in two different files, a batch file for Windows based operating systems and a Bash Shell file for compatible versions of Linux. Users are given the option to run a ‘Quick Back-Up’, run a single back up and the facility to create their own back up scripts.

**Using the Back-Up Utility**

Menu Mode

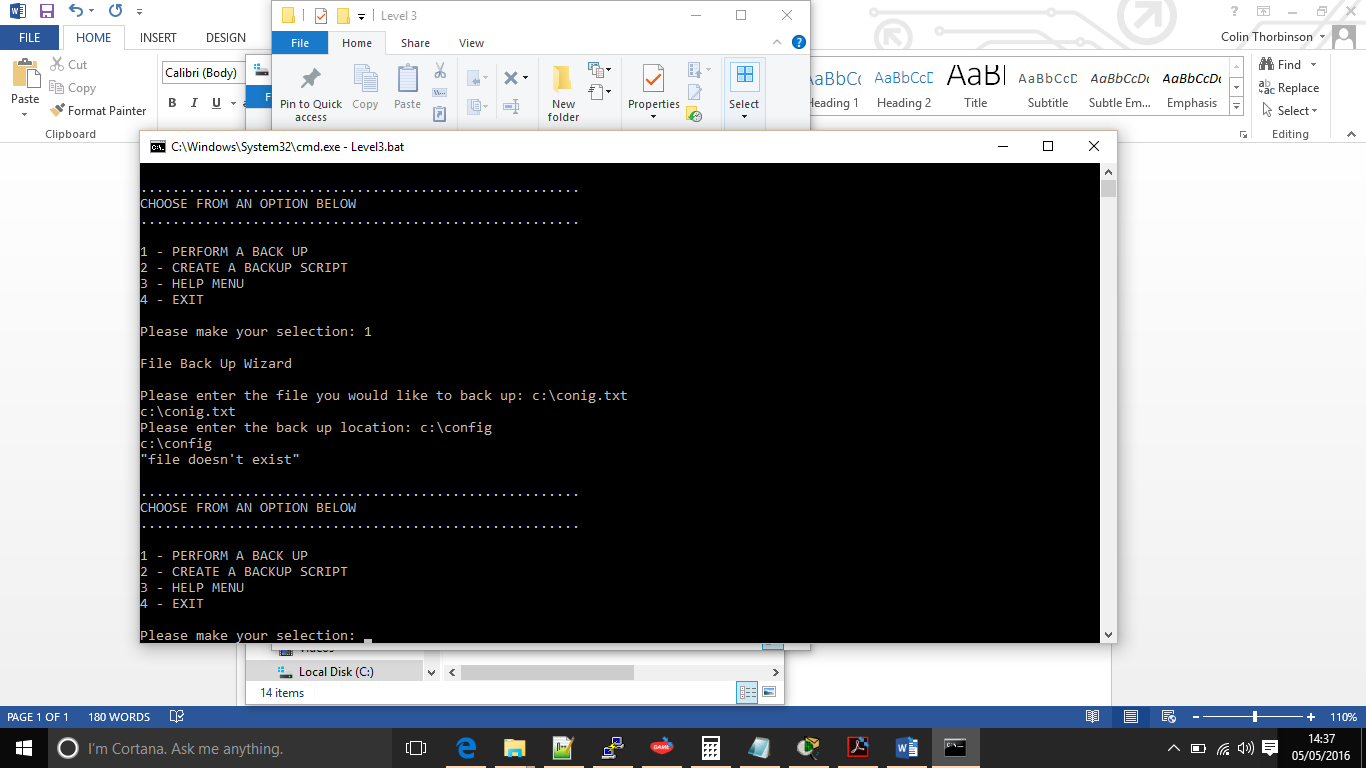
When users run the program without using the ‘Quick Back-Up’ mode they will see the Main Menu screen. This screen gives users the option to either perform a back-up, create a back-up script, view the help section or exit the program.

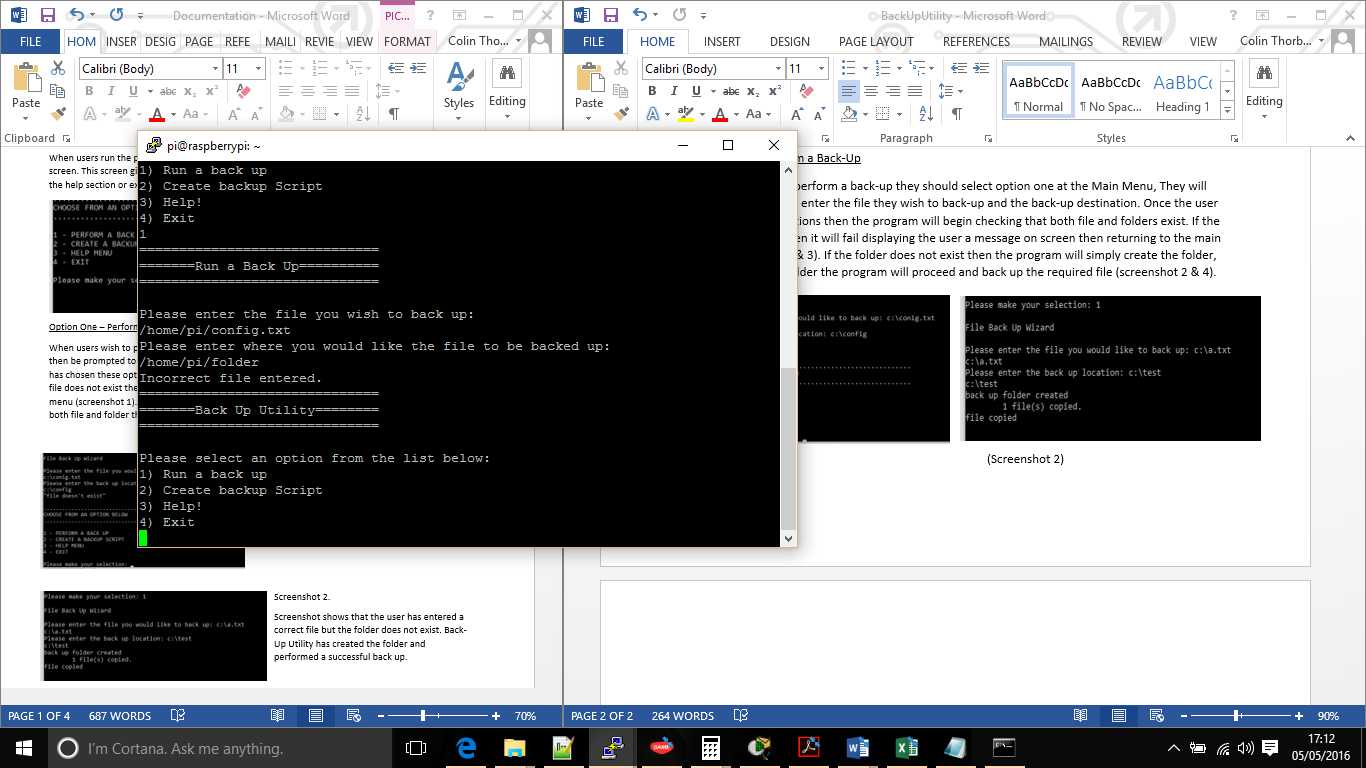
Screenshot show both versions of the Main Menu, (left) shows the Windows MS-DOS version while (below) is the Linux Bash Shell version.

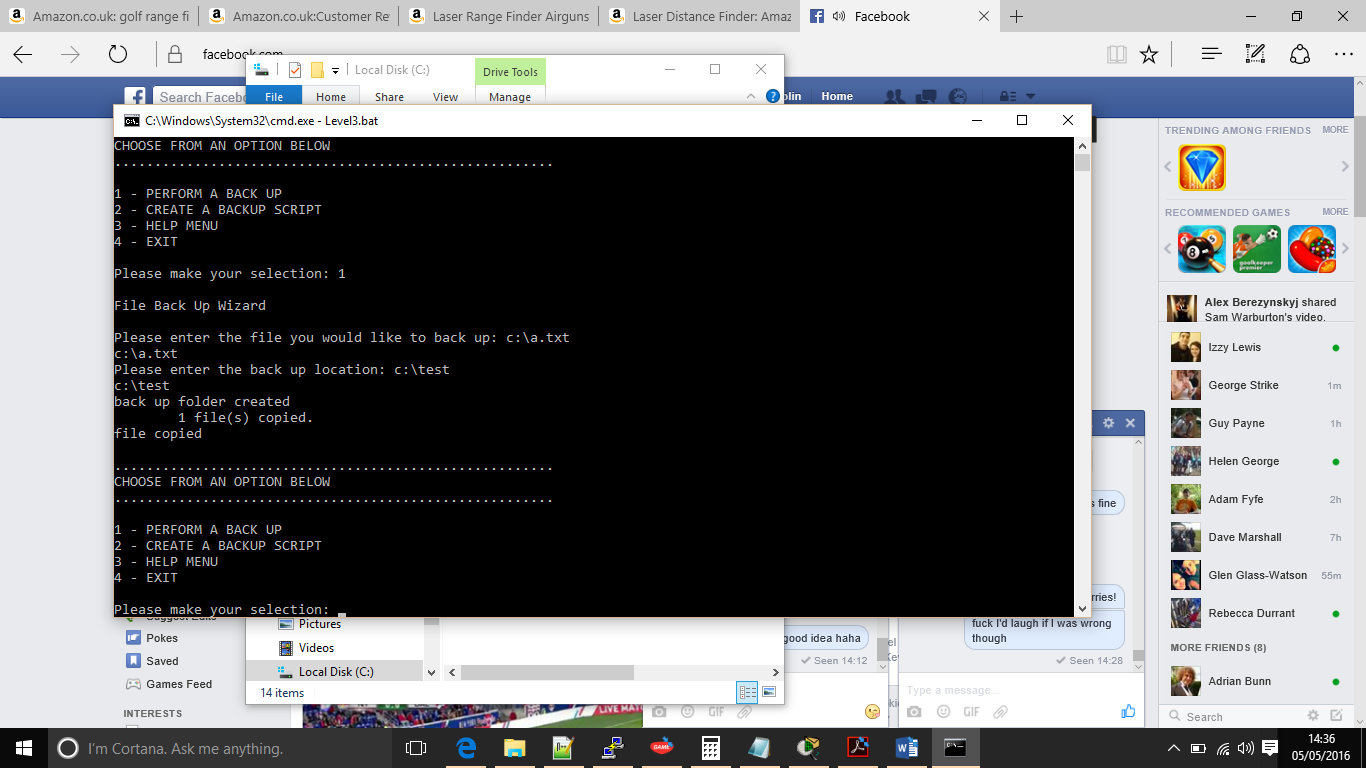


Option One – Perform a Back-Up

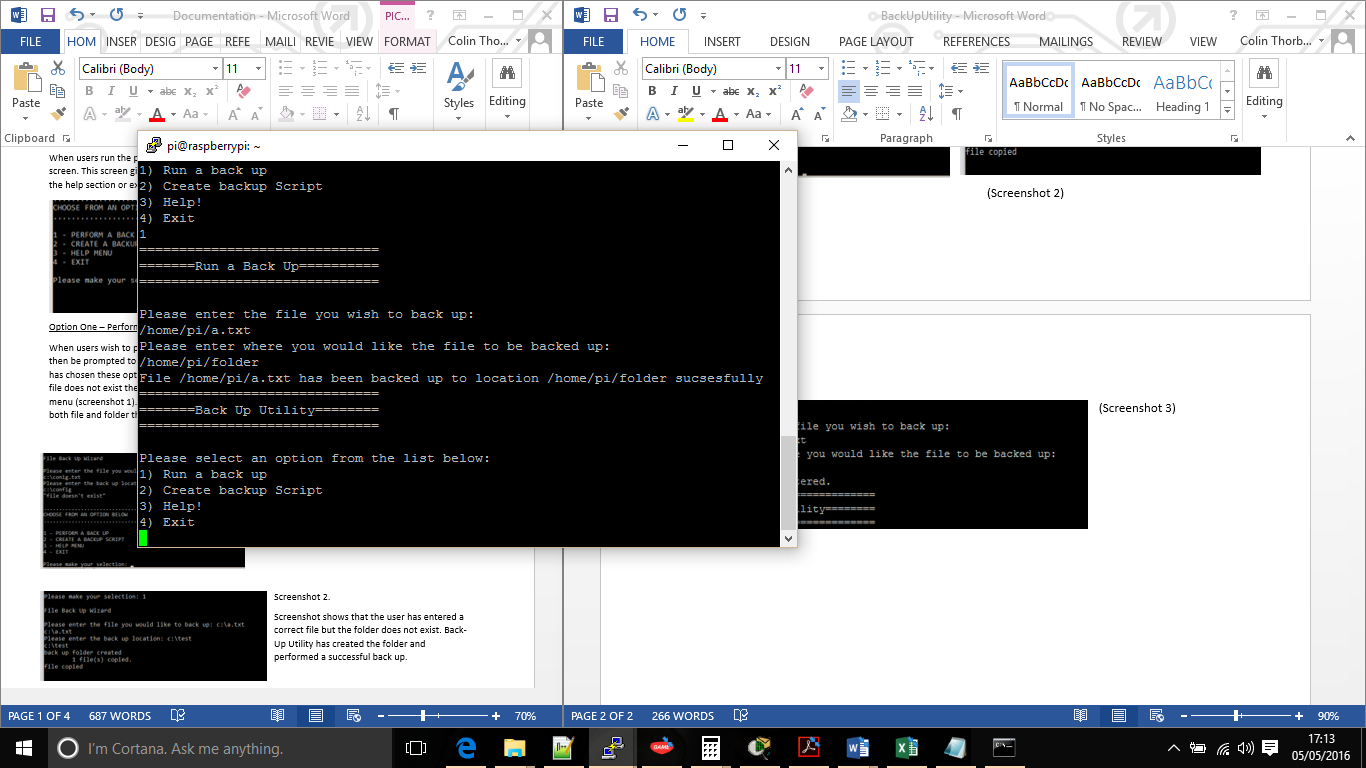
When users wish to perform a back-up they should select option one at the Main Menu, They will then be prompted to enter the file they wish to back-up and the back-up destination. Once the user has chosen these options then the program will begin checking that both file and folders exist. If the file does not exist then it will fail displaying the user a message on screen then returning to the main menu (screenshot 1 & 2). If the folder does not exist then the program will simply create the folder, once both file and folder the program will proceed and back up the required file (screenshot 3 & 4). (Screenshot’s below)

(Screenshot 1)

(Screenshot 2)

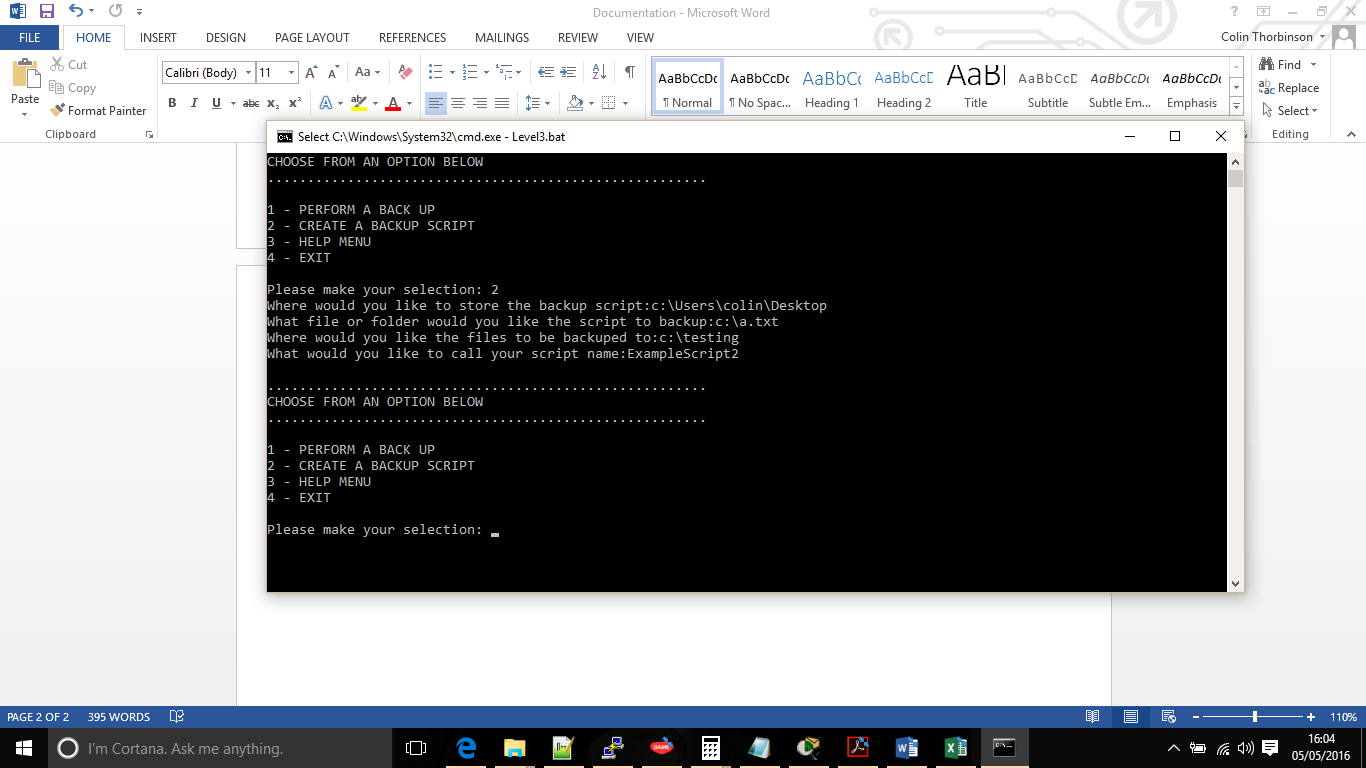


(Screenshot 3)

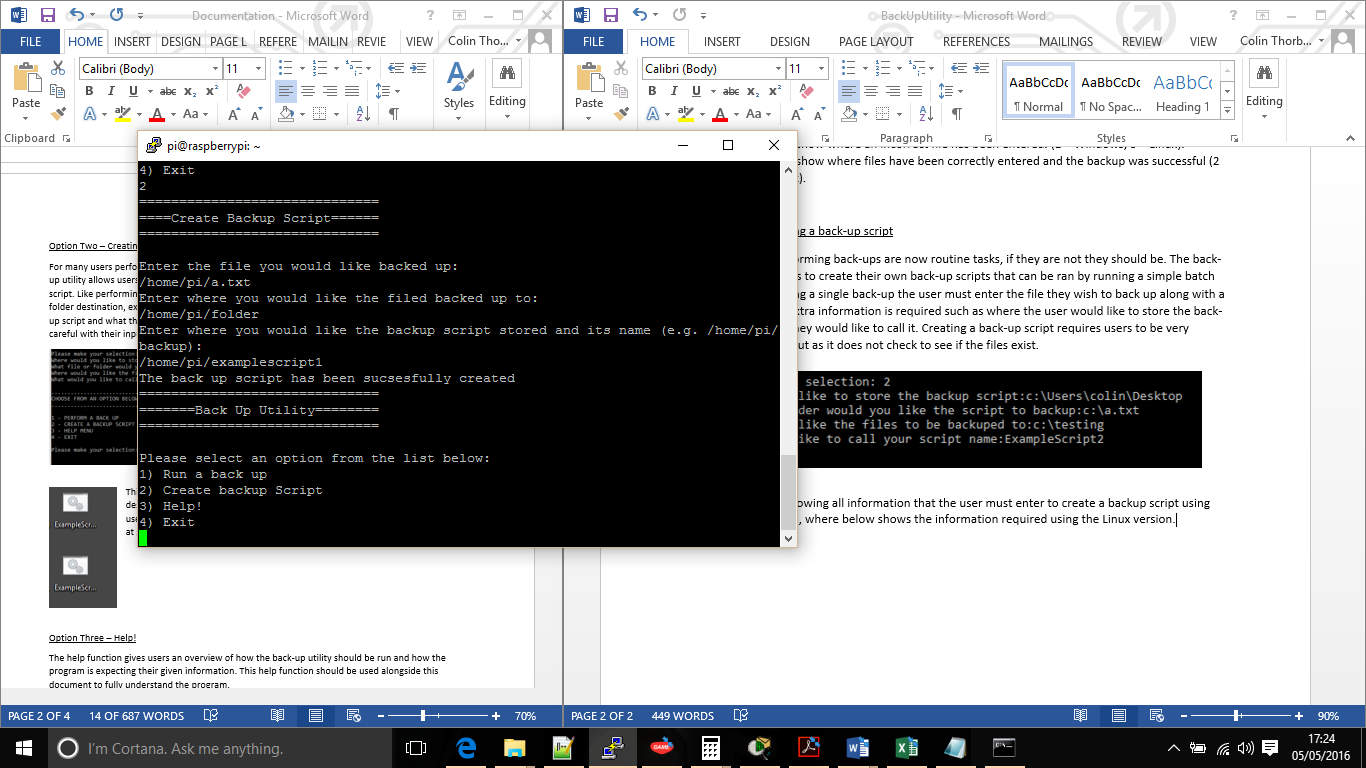
(Screenshot 4)

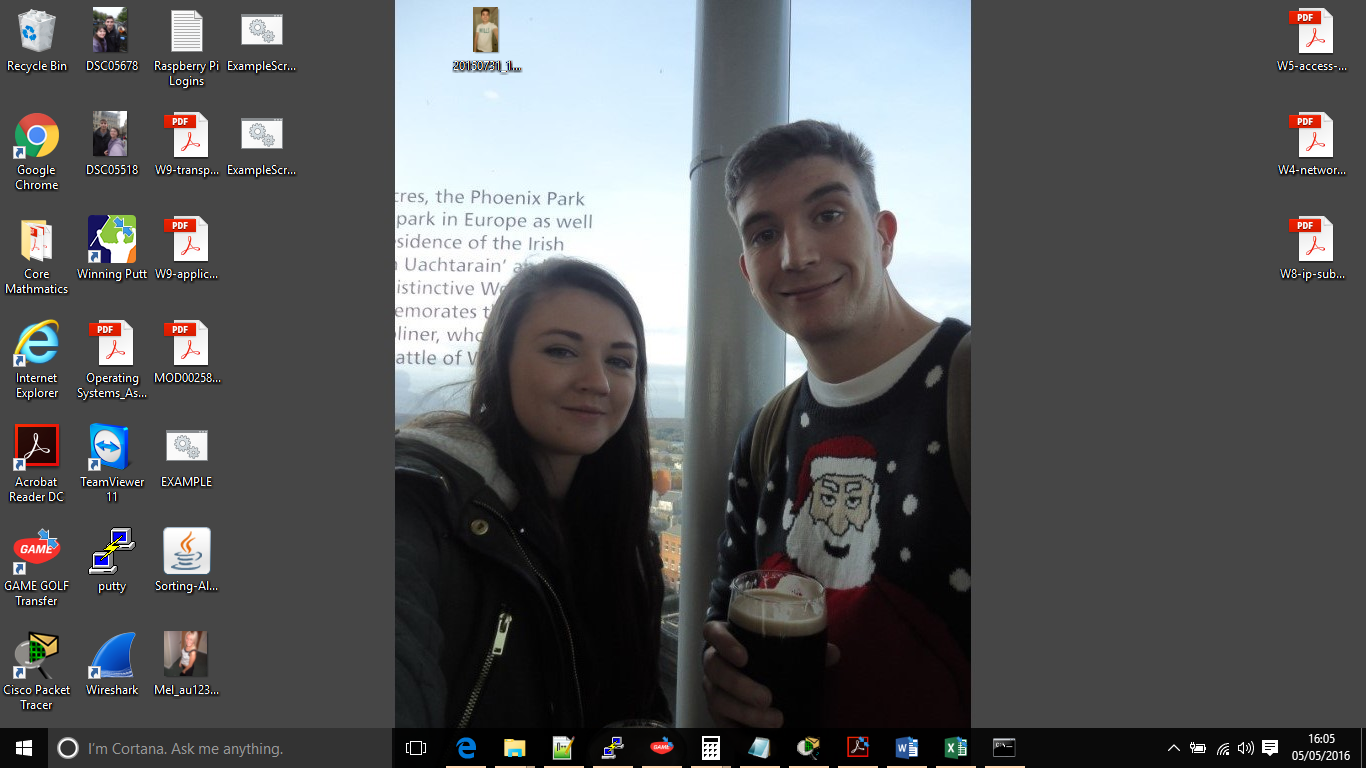
Screenshots 1 and 2 show where an incorrect file has been entered. (1 = Windows, 2 = Linux).  
Screenshots 3 and 4 show where files have been correctly entered and the backup was successful (2 = Windows, 4 = Linux).

Option Two – Creating a back-up script

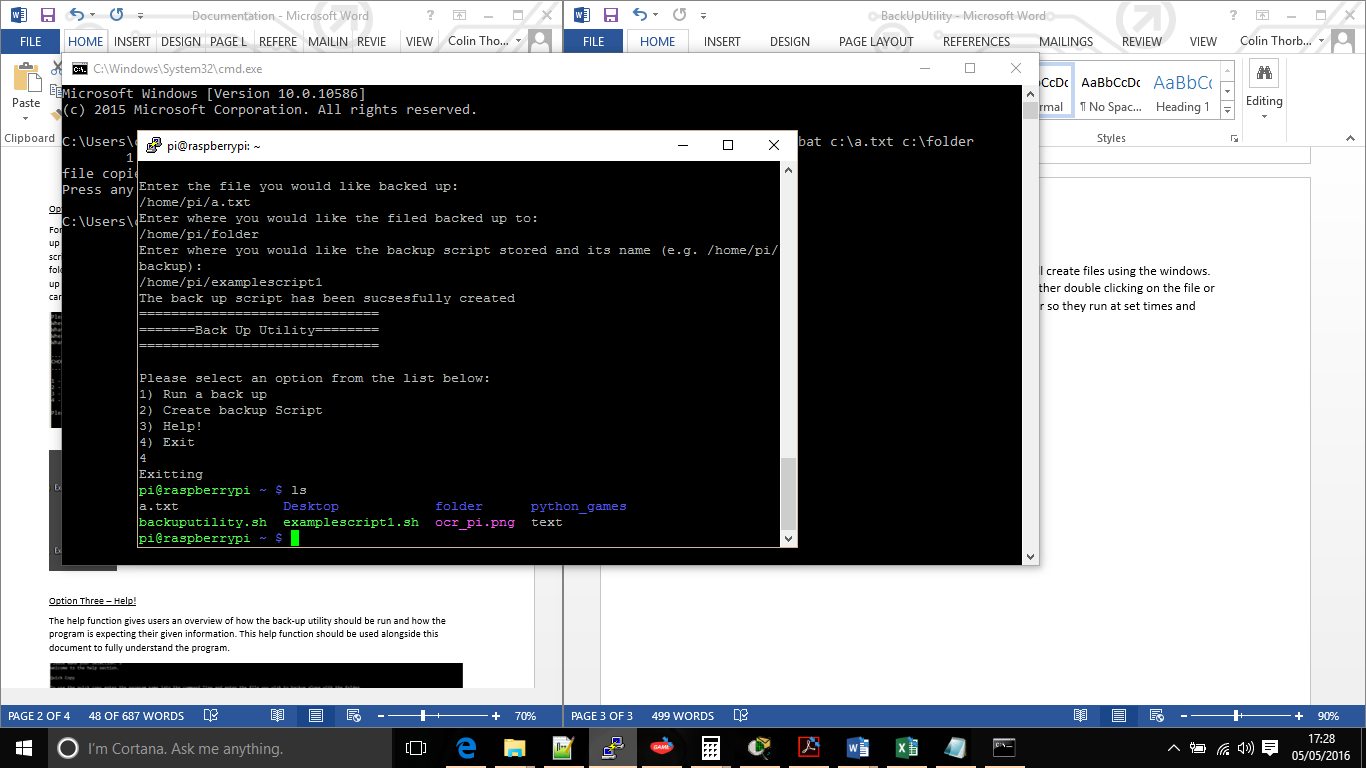
For many users performing back-ups are now routine tasks, if they are not they should be. The back-up utility allows users to create their own back-up scripts that can be ran by running a simple batch script. Like performing a single back-up the user must enter the file they wish to back up along with a folder destination, extra information is required such as where the user would like to store the back-up script and what they would like to call it. Creating a back-up script requires users to be very careful with their input as it does not check to see if the files exist.

Above screenshot showing all information that the user must enter to create a backup script using the Windows version, where below shows the information required using the Linux version.



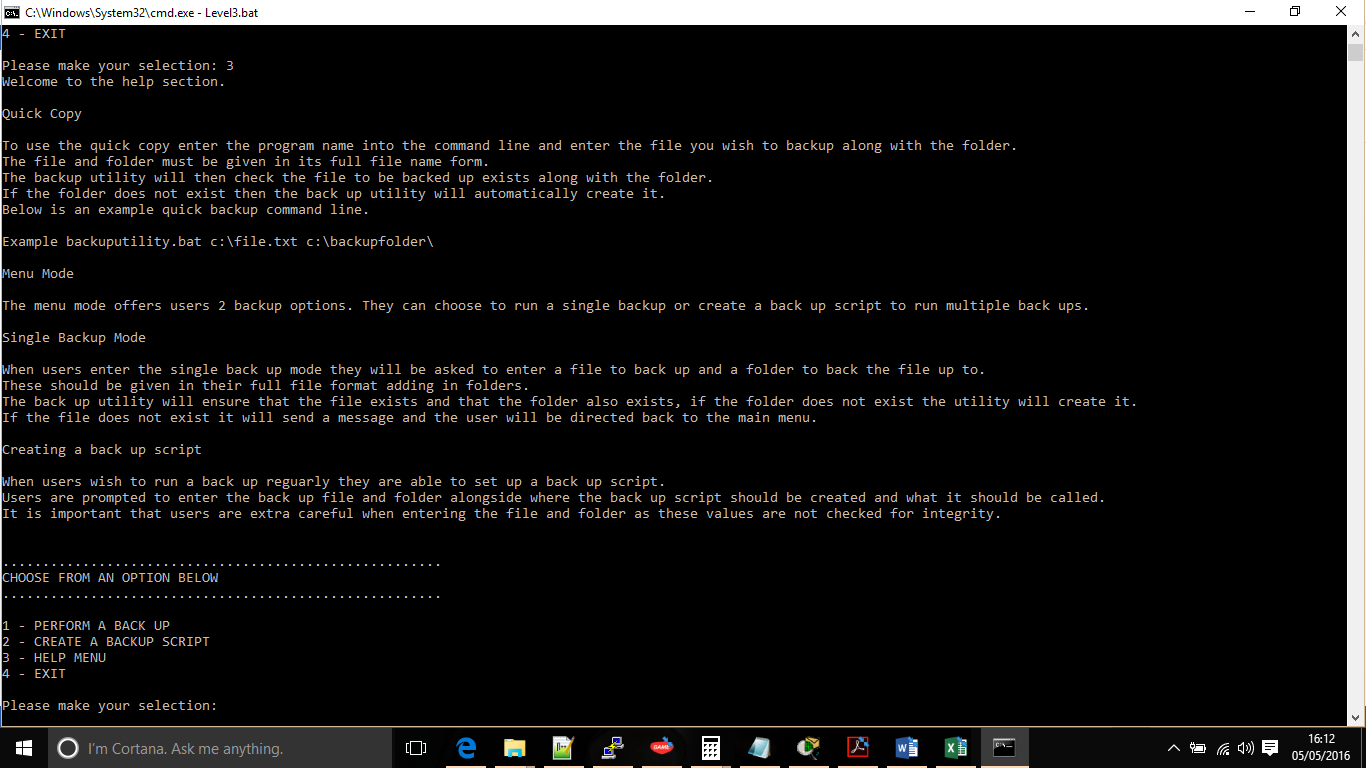
This is an example of how the back-up utility will create files using the windows. From here users can choose to run the file by either double clicking on the file or they can choose to add these to a task scheduler so they run at set times and dates. 

Below is how the script will be saved in the Linux version, since this version of Linux is a command line only operating system users cannot simply double click to run it but on a distribution of Linux running a graphical interface they would.



Option Three – Help!

The help function gives users an overview of how the back-up utility should be run and how the program is expecting their given information. This help function should be used alongside this document to fully understand the program. This screenshot shows the help function running in the Windows version, the Linux version is practically identical.

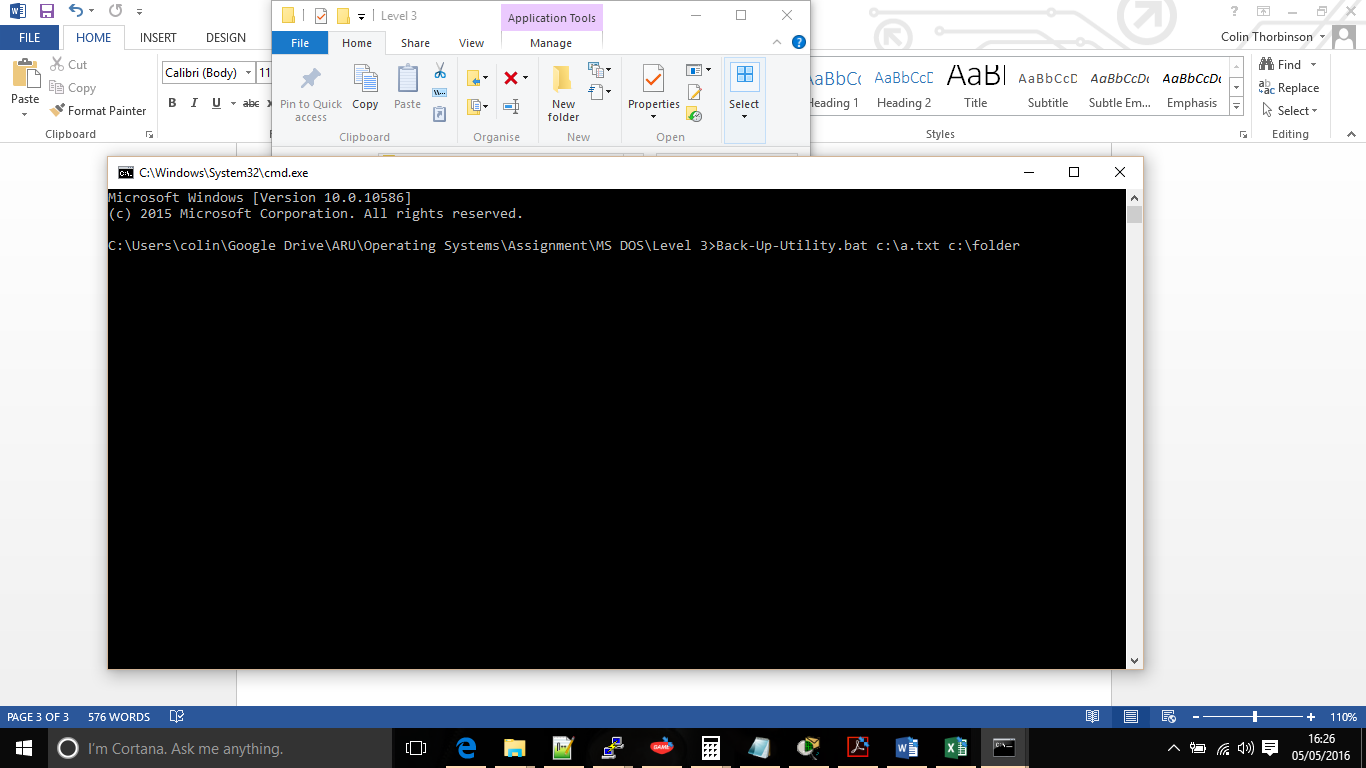


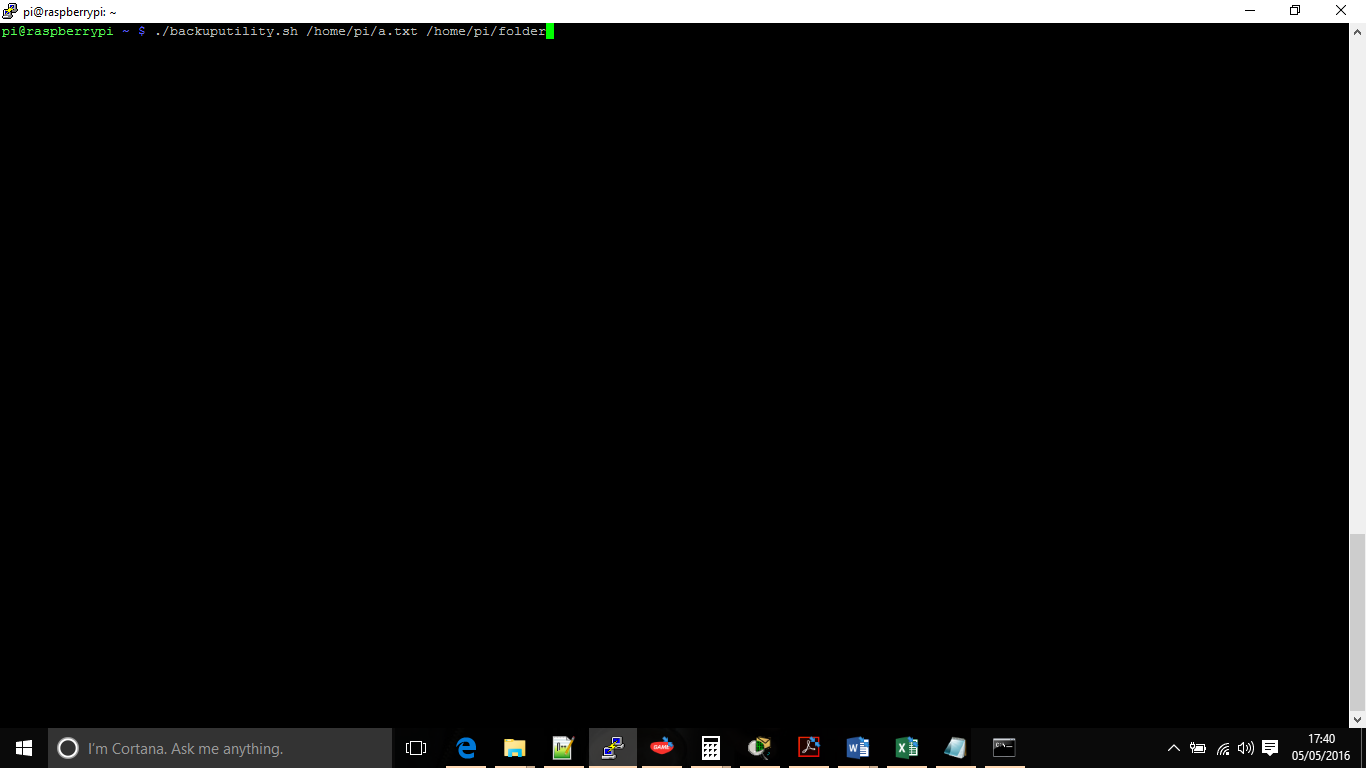
Option Four – Exit

Once users have completed their back-up’s they should select option 4 to exit.

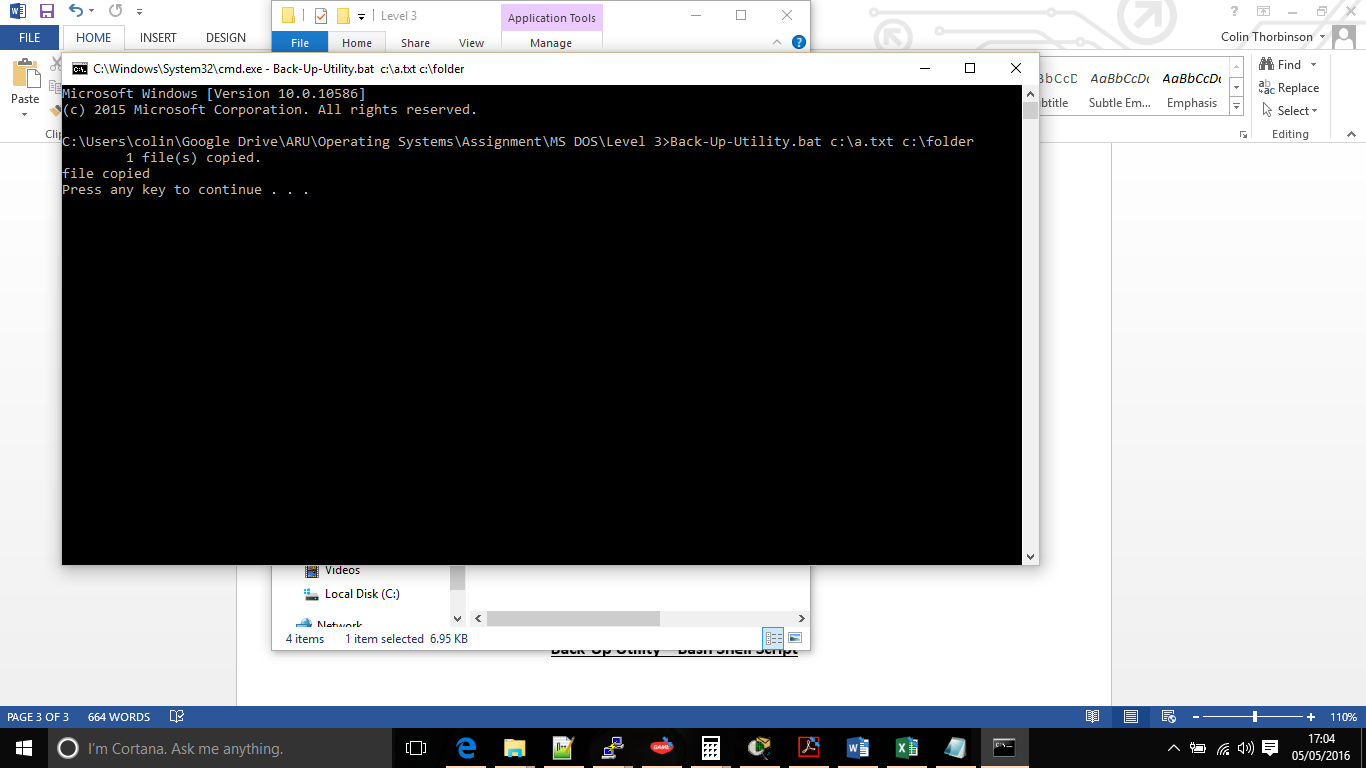
Using Quick Copy

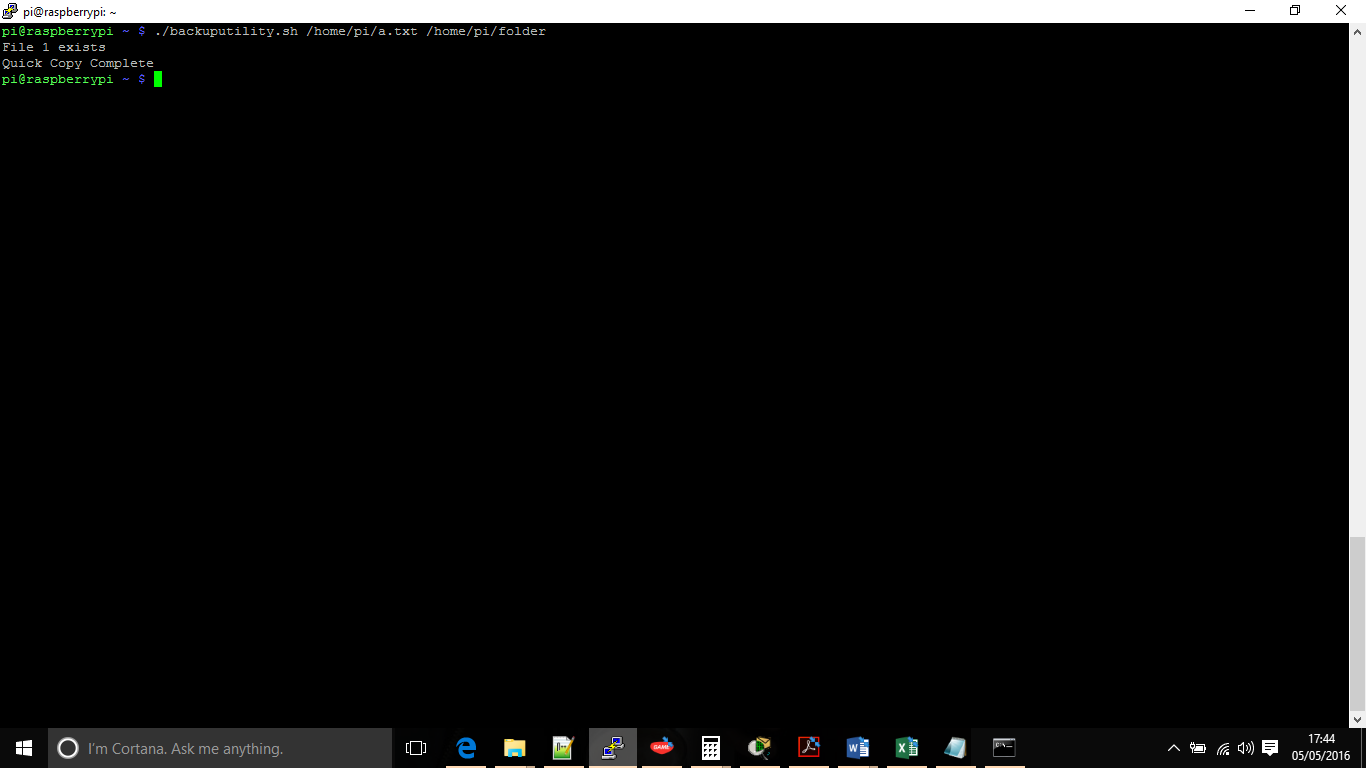
Users have the option of running a back-up without having to use the menu via quick copy. In the command line users must simply add the file they wish to back-up along with the back-up location next to the program name to perform a quick back-up.

These screenshots show both Windows and Linux versions and the correct syntax required to perform a quick copy.



A quick back-up will check that the back-up file exists, if the file does not exist then the back-up utility will default back to the main menu. The back-up folder will also be checked and if the back-up folder does not exist it will be created. Once these checks are complete the copy will take place, a message will be displayed to the user and the back-up utility will close.





**Testing**

Testing was done in basic stages while writing both versions of the program. I decided to write and test each program to each specification level.

The first level testing mainly consisted of testing that the if file exists statement was working where it would produce an error for the user if the file did not exist or create the folder if required. Using the do while loop in Linux caused a few issues in the syntax rather than the ease of the goto command in Windows and I found Linux to be a lot more sensitive with its syntax spacing which must be absolutely correct. Around 10 to 15 different tests took place at level 1, all tests were successful apart from one folder in Linux that had been set with peculiar permissions. During this testing the wording was changed in an effort to make it more user friendly.

Level 2 testing was slightly trickier as it involved some slightly more involved syntax. Before adding the syntax to the level 1 aspect of the program a smaller program was developed where if two parameters were entered they would be outputted to the screen otherwise another message would be displayed to the user. Then the smaller program was upgraded so a file and folder could be inputted and the program would perform checks on if they existed, as per the specification if the file did not exist the user would be presented with a message and if the folder did not exist it would be created. Once this was developed it was integrated into the level 1 program, with the use of the goto command this was a simple integration whereas with Linux it took slightly more work ensuring that the if statement was ended at the correct point in the program. Another 10 – 15 tests were ran, testing both quick copy and menu driven copy functions, once again steps were made to make the wording easier and more user friendly.

Level 3 was a slightly easier level to complete, after gathering the information required from the user it was simply a case of outputting this to a file rather than to the screen and ensuring that it had the correct file extensions. Linux had the added complication of ensuring that the file had the correct run privileges but this was a very simple line of code.

Once I had built both programs to the level 3 specification I ran several tests ensuring that they would fail and succeed, I found that due to the simplicity of the programs that they rand and performed as they should. I found that the sleep time on the exit for Linux was slightly too long at 5 seconds so this was shortened to 2 seconds.

**The Code**

Listed below is the code used for both versions of the back-up utility. As well as being found here they can both be found in text (txt) files on the memory stick provided as well as in the form of a batch file (.bat) and shell file (.sh) file. Due to the formatting in this department some of the ‘#’s have been removed.

Windows MS-DOS

@echo off

REM #####################################################################################

REM ########Checks to see if any parameters have been entered into the command line ################

REM #####################################################################################

REM If %1 is a null value go to the Main MENU

REM Otherwise user wishes to perform quick copy and script will go to Command.

if "%1"=="" (

If "%2"=="" (

goto :Menu)

) Else (

goto :Command

)

REM #####################################################################################

REM ################################## Main Menu #################################

REM #####################################################################################

REM Main Menu will be displayed to user if they decide not to use quick copy REM User has choice to either perform back up or to exit program.

REM Users selection will be placed in select variable which will then be used in an IF statement to determine which action the users requested.

:MENU

ECHO.

ECHO .......................................................

ECHO CHOOSE FROM AN OPTION BELOW

ECHO .......................................................

ECHO.

ECHO 1 - PERFORM A BACK UP

ECHO 2 - CREATE A BACKUP SCRIPT

ECHO 3 - HELP MENU

ECHO 4 - EXIT

ECHO.

set /p select=Please make your selection:

IF %select%==1 goto BACKUP

IF %select%==2 goto SCRIPT

IF %select%==3 goto HELP

IF %select%==4 exit

REM #####################################################################################

REM ################################## Running a back up ##############################

REM #####################################################################################

REM Back up takes back up information from the user and stores it in variables.

REM Then the program checks that the file exists, if the file exists if not it outputs a message on the screen and REM goes back to Main Menu

REM The folder is then checked, if the folder does not exist it creates it.

REM Once file and folder have been checked copy takes place.

:BACKUP

echo.

echo File Back Up Wizard

echo.

set /p BackUpFile=Please enter the file you would like to back up:

echo %BackUpFile%

set /p BackUpLocation=Please enter the back up location:

echo %BackUpLocation%

IF NOT EXIST "%BackUpFile%" ( echo "file doesn't exist" & goto :MENU )

IF NOT EXIST "%BackUpLocation%" ( md %BackUpLocation% )

echo back up folder created

copy %BackUpFile% %BackUpLocation%

echo file copied

goto :MENU

REM #####################################################################################

REM ################################## The Help Section ###############################

REM #####################################################################################

:HELP

echo Welcome to the help section.

echo.

echo Quick Copy

echo.

echo To use the quick copy enter the program name into the command line and enter the file you wish to backup along with the folder.

echo The file and folder must be given in its full file name form.

echo The backup utility will then check the file to be backed up exists along with the folder.

echo If the folder does not exist then the back up utility will automatically create it.

echo Below is an example quick backup command line.

echo.

echo Example backuputility.bat c:\file.txt c:\backupfolder\

echo.

echo Menu Mode

echo.

echo The menu mode offers users 2 backup options. They can choose to run a single backup or create a back up script to run multiple back ups.

echo.

echo Single Backup Mode

echo.

echo When users enter the single back up mode they will be asked to enter a file to back up and a folder to back the file up to.

echo These should be given in their full file format adding in folders.

echo The back up utility will ensure that the file exists and that the folder also exists, if the folder does not exist the utility will create it.

echo If the file does not exist it will send a message and the user will be directed back to the main menu.

echo.

echo Creating a back up script

echo.

echo When users wish to run a back up reguarly they are able to set up a back up script.

echo Users are prompted to enter the back up file and folder alongside where the back up script should be created and what it should be called.

echo It is important that users are extra careful when entering the file and folder as these values are not checked for integrity.

echo.

goto MENU

REM #####################################################################################

REM ################################## Creating a Back Up Script #########################

REM #####################################################################################

:SCRIPT

REM get where the user wants to store the file

set /p store=Where would you like to store the backup script:

REM get the backup files or folders

set /p backupfiles=What file or folder would you like the script to backup:

REM get where the user wants to backup the files too

set /p backuplocation=Where would you like the files to be backuped to:

REM get what the user wants to call the backup file.

set /p scriptname=What would you like to call your script name:

REM Create file

echo copy %backupfiles% %backuplocation%>%store%\%scriptname%.bat

goto MENU

REM #####################################################################################

REM ################################## Running a Quick Copy ###########################

REM #####################################################################################

REM Quick copy where user must input backup file and location into command line.

REM Script will then check that they exist, if file does not exist it will error and user will be presented with main menu.

REM If folder does not exist the program will create the requested directory.

REM Once the copy has take place a message will be displayed to the user and the program will close

:Command

IF NOT EXIST "%1" ( echo "file doesn't exist" & goto :MENU )

IF NOT EXIST "%2" ( md %2 )

copy %1 %2

echo file copied

pause

Linux Bash Shell

#!/bin/bash

#First IF statement checks if users have entered 2 extra parameters to command line.

if [ $# == "2" ]; then

#Initiates the variable quick copy that will be used to ensure both file and folder exists.

quickcopy=0

#Checks that the file exists it will increment quick copy by 1.

if [ -r $1 ]; then

echo "File 1 exists"

let quickcopy++

fi

#Checks that the folder entered exists and will increment quick copy by 1.

#If the folder does not exist it will create the folder using the entered parameter and then increment #quick copy by 1.

if [ -r $2 ]; then

let quickcopy++

else

mkdir $2

let quickcopy++

fi

#If both file and folder exists then quick copy will be equal to 2 and the program will perform a copy # command.

#Once the copy is complete the program will message out a success message to the user and # terminate.

#If quick copy does not equal to 2 then an incorrect parameter has been entered and a message will #be displayed to the user.

if [ "$quickcopy" = "2" ]; then

cp $1 $2

echo "Quick Copy Complete"

else

echo "Incorrect file parameters entered"

fi

#Otherwise continue to Main Menu function.

else

#loop is initated to begin a while loop to allow the program to return to the main menu once an #operation has been completed.

#While the vaiable loop is equal to 0 the program will continue to display the menu.

loop=0

while [ "$loop" = "0" ]; do

#Displays the Main Menu to the user and asks them to make a choice, their choice will be stored in the variable input.

echo "=============================="

echo "=======Back Up Utility========"

echo "=============================="

echo ""

echo "Please select an option from the list below:"

echo "1) Run a back up"

echo "2) Create backup Script"

echo "3) Help!"

echo "4) Exit"

read input

#The result of input is fed into the case statement which will determine which part of the program the users wishes to use.

case $input in

1)

#The backup section displaying a menu to the user prompting them to input a file to backup and the backup location.

echo "=============================="

echo "=======Run a Back Up=========="

echo "=============================="

echo ""

echo "Please enter the file you wish to back up:"

read backupfile

echo "Please enter where you would like the file to be backed up:"

read backuplocation

#Initiates the variable value that will be used to ensure both file and folder exists.

value=0

#If statment to check that the file exists.

#If the file exists it will incriment value by 1.

#If the file does not exist it will display a message to the user that the file does not exist.

if [ -e $backupfile ]; then

let value++

fi

#Checks that the backup location exists.

#If the file exists it will add 1 to value.

#If not it will create the back up folder and add 1 to value.

if [ -e $backuplocation ]; then

let value++

else

if [ "$value" = "1" ]; then

mkdir $backuplocation

let value++

fi

fi

#Checks that value is equal to 2 so both file and folder exist.

#When value is equal to 2 the program will copy the file to the backup #location.

#Once copied the program will display a message to the user.

#If the value is not equal to 2 the program will display a message to the user #and go back to the Main Menu.

if [ "$value" = "2" ]; then

cp $backupfile $backuplocation

echo "File" $backupfile "has been backed up to location" $backuplocation "sucsesfully"

else

echo "Incorrect file entered."

fi

;;

2)

#Program asks user to enter several values such as file to back up, back up location,

#where they would like to store their backup script and name of the backup script.

#Those variables will then be placed into variables.

echo "=============================="

echo "====Create Backup Script======"

echo "=============================="

echo ""

echo "Enter the file you would like backed up:"

read backupscriptfile

echo "Enter where you would like the filed backed up to:"

read backupscriptlocation

echo "Enter where you would like the backup script stored and its name (e.g. /home/pi/backup):"

read backupscriptstore

#This section of code will output the back up details into a scipt file and will change the #permissions so it can be read by any user.

#Once sucsesful a message will be displayed to the user.

echo cp $backupscriptfile $backupscriptlocation > $backupscriptstore.sh

chmod 777 $backupscriptstore.sh

echo "The back up script has been sucsesfully created"

;;

3)

echo "=============================="

echo "============Help=============="

echo "=============================="

echo ""

echo "Welcome to the help section."

echo ""

echo "Quick Copy"

echo ""

echo "To use the quick copy enter the program name into the command line and enter the file you wish to backup along with the folder."

echo "The file and folder must be given in its full file name form."

echo "The backup utility will then check the file to be backed up exists along with the folder."

echo "If the folder does not exist then the back up utility will automatically create it."

echo "Below is an example quick backup command line."

echo ""

echo "Example ./backuputility /home/user1/file.txt /home.user1/backupfolder"

echo ""

echo "Menu Mode"

echo ""

echo "The menu mode offers users 2 backup options. They can choose to run a single backup or create a back up script to run multiple back ups."

echo ""

echo "Single Backup Mode"

echo ""

echo "When users enter the single back up mode they will be asked to enter a file to back up and a folder to back the file up to."

echo "These should be given in their full file format adding in folders."

echo "The back up utility will ensure that the file exists and that the folder also exists, if the folder does not exist the utility will create it."

echo "If the file does not exist it will send a message and the user will be directed back to the main menu."

echo ""

echo "Creating a back up script"

echo ""

echo "When users wish to run a back up reguarly they are able to set up a back up script."

echo "Users are prompted to enter the back up file and folder alongside where the back up script should be created and what it should be called."

echo "It is important that users are extra careful when entering the file and folder as these values are not checked for integrity."

echo ""

;;

4) echo "Exiting"; sleep 2; loop=1 ;;

#The exit option will display a message to the user, sleep the message so it appears on screen for a few seconds then change the value.

#of loop to 1 so it will exit the loop.

esac

done

fi