TR
Technical Report

Cloud Storage Mounting on Android OS

The goal is to mount a cloud storage directory to the internal storage of an Android device, at the Operating System level.

Hanen B. Alkhafaji

CSMAOS
Cloud Storage Mounting on Android OS

College of Engineering and Computer Science
Wright State University
State of Ohio
City of Fairborn

Hanen Alkhafaji
Wright State University
3640 Colonel Glenn Hwy.
Dayton, OH 45435 USA
Phone (937) 775-1000
alkhafaji.2@wright.edu
http://www.wright.edu

Summary

To be decided...

Preface

To be decided...

Acknowledgements

This template is based on the Laursen's DTU Thesis template.

Contents

List of Figures

CHAPTER 1

Introduction

To be decided...

CHAPTER 2

OCaml Programming Language

2.0.1 Task 1: Online OCaml Lessons

2.0.1.1 Simple Expressions

2.0.1.2 Imperative Programming

2.0.1.3 Functions

2.0.1.4 New Examples by Hanen

2.0.1.5 Critique by Hanen

1. Lesson 1 - Simple Expressions This covered computing numeric values, defining strings, working with arrays, string manipulation, and defining and operating on tuples. Tuples can be made up of different data types. There are some functions built-in for tuples that have two elements. The functions presented in the tutorial were `fst` for getting the first element and `snd` for getting the second.

2. Lesson 2 - Imperative Programming `let` is the keyword used to set the results of some computation to a named variable. However, once a variable is set to a particular value using the `let` keyword, it cannot be modified to a different value. A compilation error results. The way to get around this constraint is to use the keyword `ref` on the right side of the `let` statement. That reference can then be modified. (`let x = ref 42;;`) `printf` is used similar to C to print formatted text to the terminal. Looping syntax is very similar to many other program languages, except when looping through a series of numbers backwards the word `downto` is used as opposed to `to` in the ascending direction. For comparison of values, the output is a boolean of either `true` or `false`. These greater than, less than, equal, or not equal to comparisons are not limited to only numeric values. The equal and not equal comparison symbols are similar to VB where a single equal sign represents an equivalence comparison while a less than sign

followed by a greater than sign represents a non-equivalence comparison. The only limitation is that there isn't support for comparing values of different types. However, functions like `string_of_int` can be used to convert an integer to a string so that it can be safely compared to another string. `if then else` logic is very straight forward. `while` loops logic is also easy to understand and uses a `while do done` structure.

3. Lesson 3 - Functions Functions can be defined in one line pmateti: Huh? using the `let` keyword very similar to defining a variable, but it takes arguments. One argument can be provided or several arguments in a tuple. Calling these functions is exactly the same as all other programming languages. Multiple values can be returned from a function by returning a tuple. Defined functions can also be called in a partial manner. This almost seems like extension methods from the C# world. A function that takes two arguments can be called with only one argument. However, it takes into account the value present at the time in which its called and uses that as the second parameter. `let mul x y = x * y`
`let double = mul 2`
`double 8` Anonymous functions are lambda expressions. They are functions defined without a name. These are useful for generating inline functions to be passed as a parameter to a function. In this case, they do not need to be assigned an identifier. Functions such as `List.map` and `List.fold_left` are useful for combining the power of anonymous functions and iterators to get a task done efficiently by iterating over a list and performing an operation on each value as the iterator visits each element.

4. pmateti: A couple of new examples of OCaml by Hanen?

pmateti: Give a brief opinion on the tutorial.

2.0.2 Task 2: Adjust OCaml Example Projects

2.0.2.1 Go Fish

1. Original Source Code - I found the code for this game on Rosetta Code under the OCaml implementation. The game play is based on one player and an AI player who is automated through the back-end randomization functionality. The user chooses a card and asks the AI player if it has that card. If the AI player does, then it must give it. If the AI player does not, then the user must pick up a card from the deck. The player who loses their entire hand of cards first wins. As the code is currently written, the players are named "a" and "b". There is not a way to change that. Also, the user picks the card to ask about by typing in a number between a provided range which corresponds to the cards left in the player's hand.

2. Allowing For Choosing Players' Names - After wrestling with the code and trying to better familiarize myself with OCaml, this ended up being a pretty straight forward task. It took much longer since I made several failed attempts along the way trying to understand how OCaml projects are structured. The following code ended up being the only piece I needed to change.

   From this:

   ```
   (try
     if Random.bool()
         then make_turn "a" "b" player_a player_b
         else make_turn "b" "a" player_b player_a;
   with Exit -> ());
   ```

   To this:

   ```
   (try
     if Random.bool()
         then make_turn Sys.argv.(1) Sys.argv.(2) player_a player_b
         else make_turn Sys.argv.(2) Sys.argv.(1) player_b player_a;
   with Exit -> ());
   ```

   This is how it was executed through the terminal:

   ```
   hanen@hanen:~/Desktop/gofish$ ocamlc -g -o gofish gofish.ml
   File "gofish.ml", line 153, characters 21-36:
   Warning 52: Code should not depend on the actual values of
   this constructor's arguments. They are only for information
   and may change in future versions. (See manual section 9.5)
   ```

   This is a snippet of the game-play:

   ```
   hanen@hanen:~/Desktop/gofish$ ./gofish "dr. mateti" "hanen"

   player hanen asked for Sixs
   player dr. mateti gives (Six-Clubs)

   player hanen asked for Fours
   player dr. mateti has no Fours

   (Queen-Clubs), (Jack-Clubs), (Nine-Spades), (Eight-Diamonds),
   (Nine-Hearts), (Nine-Clubs), (Queen-Spades), (Queen-Diamonds)
   Ranks: Eight, Nine, Jack, Queen
   choose from 1 to 4
   ```

2.0.2.2 Guess the Number

1. Original Source Code - The game-play on this Guess the Number game is very simple. A random number generator is used to ”think of a number” and the player puts in numbers until they guess the number that was chosen at random. The user cannot specify the maximum of the range and they are not given any feedback on how far off they are from the selected number.

   The following is the current state of the code:

```
#!/usr/bin/env ocaml

let () =
  Random.self_init();
  let n =
    if Random.bool () then
      let n = 2 + Random.int 8 in
      print_endline "Please guess a number between 1 and 10 excluded";
      (n)
    else
      let n = 1 + Random.int 10 in
      print_endline "Please guess a number between 1 and 10 included";
      (n)
  in
  while read_int () <> n do
    print_endline "The guess was wrong! Please try again!"
  done;
  print_endline "Well guessed!"
```

   The following is the current game-play in the terminal:

```
hanen@hanen:~/Desktop/gofish$ ocamlc -g -o guessnum guessnum.ml
hanen@hanen:~/Desktop/gofish$ ./guessnum
Please guess a number between 1 and 10 excluded
1
The guess was wrong! Please try again!
2
The guess was wrong! Please try again!
3
The guess was wrong! Please try again!
4
The guess was wrong! Please try again!
5
The guess was wrong! Please try again!
```

```
6
Well guessed!
```

2. Let User Specify Max of Range - I wanted to give the user the ability to specify what the max of the range should be for the number that is selected randomly for guessing.

   The following is the new state of the code after the adjustment:

```
let () =
  Random.self_init();
  let n =
    if Random.bool () then
      let n = 2 + Random.int ((int_of_string Sys.argv.(1)) - 2) in
      Printf.printf "Please guess a number between 1 and %s excluded \n" Sys.
      (n)
    else
      let n = 1 + Random.int (int_of_string Sys.argv.(1)) in
      Printf.printf "Please guess a number between 1 and %s included \n" Sys.
      (n)
  in
  while read_int () <> n do
    print_endline "The guess was wrong! Please try again!"
  done;
  print_endline "Well guessed!"
```

   The following is the new game-play in the terminal:

```
hanen@hanen:~/Desktop/gofish$ ocamlc -g -o guessnum guessnum.ml
hanen@hanen:~/Desktop/gofish$ ./guessnum 5
Please guess a number between 1 and 5 excluded
2
The guess was wrong! Please try again!
3
Well guessed!
```

3. Warm/Cold Indicator For Guess - I wanted to be able to implement in some logic that would let the user know if they are close or far off with their guess. Now that I'm allowing the user to choose any number for the max of the range, this is a nice-to-have feature. If they choose their max at 100, it would be helpful to know if they are close or not with their guess.

CHAPTER 3

OCamlFUSE on Linux

3.0.1 Task 3: Explore OCamlFUSE Source Code

(https://github.com/astrada/google-drive-ocamlfuse)

3.0.1.1 OCaml Development Environment

3.0.1.2 Study OCamlFUSE Source Code

1. OCaml development environment
   (https://github.com/janestreet/install-ocaml)

   a) Install opam
   `sudo add-apt-repository ppa:avsm/ppa`
   `sudo apt update`
   `sudo apt install -y opam m4` This was a success!

   b) Initialize opam
   `sudo opam init -y --compiler=4.07.1` `sudo opam update -uy` `sudo echo $(opam` This was a success!

   c) Install libraries and tools `sudo opam install -y async core js_of_ocaml js_of_oc` `merlin utop ocp-indent` This installed so many things and took a while, but was a success!

   d) Test Installation

```
hanen@hanen:~$ git clone https://github.com/janestreet/install-ocaml
Cloning into 'install-ocaml'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 58 (delta 10), reused 14 (delta 6), pack-reused 38
Unpacking objects: 100% (58/58), done.
hanen@hanen:~$ cd install-ocaml/01-hello-world
hanen@hanen:~/install-ocaml/01-hello-world$ dune build hello_world.exe
hanen@hanen:~/install-ocaml/01-hello-world$ dune exec ./hello_world.exe
Hello, World
```

   e) Run Tests

```
hanen@hanen:~/install-ocaml/01-hello-world$
cd ../02-expect-tests
hanen@hanen:~/install-ocaml/02-expect-tests$
dune runtest
Done: 17/19 (jobs: 1)File "expect_test_example.ml",
line 1, characters 0-0:
diff (internal) (exit 1)
(cd _build/default && /usr/bin/diff -u
expect_test_example.ml expect_test_example.ml.corrected)
--- expect_test_example.ml      2019-06-08 15:37:18.946700012
-0400
+++ expect_test_example.ml.corrected    2019-06-08 15:37:21.494597583
-0400
@@ -2,5 +2,5 @@

let%expect_test _ =
let () = printf "foo" in
-  [%expect {| bar |}]
  +  [%expect {| foo |}]
    ;;
```

The test failed because there is a different in the actual results versus what was expected. The following commands will copy the results into what was expected and show that the tests will pass after that because there will no longer be a difference.

```
hanen@hanen:~/install-ocaml/02-expect-tests$ dune promote
Promoting _build/default/expect_test_example.ml.corrected to
expect_test_example.ml.
hanen@hanen:~/install-ocaml/02-expect-tests$ dune runtest
hanen@hanen:~/install-ocaml/02-expect-tests$ git diff
diff --git a/02-expect-tests/expect_test_example.ml b/
02-expect-tests/expect_test_example.ml
index 75a19d9..9bb1c70 100644
--- a/02-expect-tests/expect_test_example.ml
+++ b/02-expect-tests/expect_test_example.ml
@@ -2,5 +2,5 @@ open! Core

let%expect_test _ =
let () = printf "foo" in
-  [%expect {| bar |}]
  +  [%expect {| foo |}]
    ;;
```

f) Editor Setup: Installed Visual Studio Code. Install a plugin for OCaml through Visual Studio Code by opening Visual Studio Code, pressing

Ctrl+P, and entering `ext install hackwaly.ocaml`[1] into the text field. Once enter is pressed, Visual Studio code will automatically install the OCaml plugin.

1. Study OCamlFUSE source code source code I downloaded the source code from GitHub (https://github.com/astrada/google-drive-ocamlfuse) and opened it in Visual Studio Code.

   a) ML versus MLI file The first thing I noticed was that there were duplicate files in the source folder that simply had different extensions. After some research, I found that the extension ML stands for Meta Language which is the umbrella programming language that contains OCaml. I suspect that the extension MLI stands for Meta Language Interface, but I cannot be certain of that since I was unable to find its expansion in my research. The reason I suspect it is Interface is because of what I observed when I looked at the files themselves. I opened bufferPool.ml and bufferPool.mli and compared them to each other. In the ML file, I found full function implementations while in the MLI file, I found a listing of function signatures. So, the MLI files must be interfaces that are used by other classes so as to hide the implementation in the ML from outsiders. pmateti: Compiled from ML to MLI.

   b) Code Exploration

      i. There is a Make module in the concurrentGlobal file.

      ii. The file drive.ml appears to have the bulk of logic behind this implementation.

      iii. I picked a function from the buffer.mli interface file called `write_to_block` and did a project wide search for it. I found it referenced in the drive.ml file as `Buffering.MemoryBuffers.write_to_block` I do not understand why the case is different in the reference. Buffering when calling the function versus buffering in the definition of the file. I would not have made that connection before, but now am aware of it.

      iv. Gapi shows up all over the code. It stands for Google API.

pmateti: Study OCamlFUSE source code: Study every file. Also, include sloccount.

[1] Insert citation.

CHAPTER 4

Mount Google Drive on Android

4.0.1 Task 4: Mount Google Drive

4.0.1.1 Install OCamlFUSE

4.0.1.2 Allow Permissions through Browser

4.0.1.3 Make Directory and Mount Drive

1. Run commands to install OCamlFUSE

```
sudo add-apt-repository ppa:alessandro-strada/ppa
sudo apt-get-update
sudo apt-get install google-drive-ocamlfuse
sudo google-drive-ocamlfuse
```

2. Allow Permissions through Browser Allow gdfuse to see, edit, create, and delete all of your Google Drive files. Select Google account and allow access permissions.

3. Make Directory and Mount Drive

```
hanen@hanen:mkdir ~/GoogleDrive
hanen@hanen:google-drive-ocamlfuse ~/GoogleDrive
hanen@hanen:cd GoogleDrive
hanen@hanen:ls
'2019-Hanen-CS 6970'    Misc
```

That final output matches the contents of that Google Drive in the browser as seen below. pmateti: Placement of figures is not quite in our control. So we must refer to afig by its number. See Figure 4.2

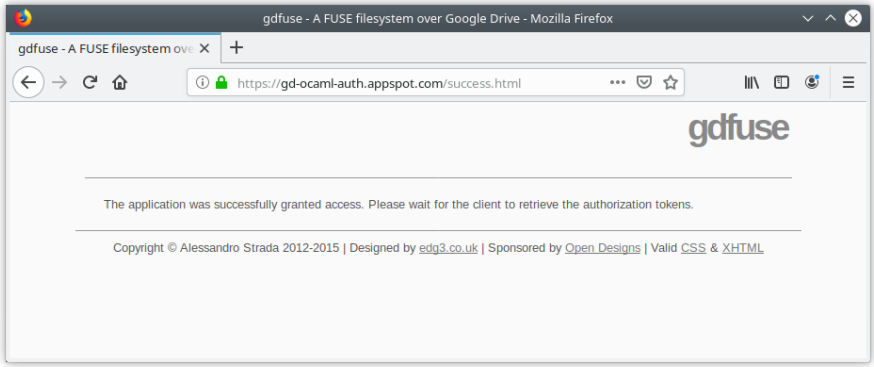

Figure 4.1: Screenshot ocaml7.png.

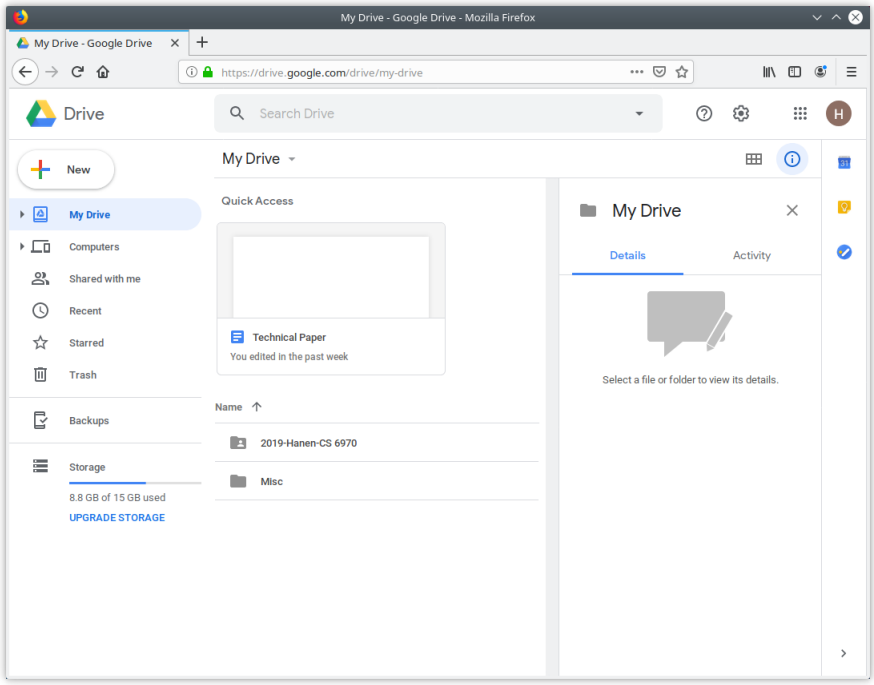

Figure 4.2: Content of GD as seen through the browser.

CHAPTER 5

Concluding Remarks

To be decided...

APPENDIX A

Project Scope

A.1 Introduction

This document outlines the scope of the project that is being explored, researched, and implemented for CS 6970. It includes the description, scope, and deliverables of the project as well as details on what criteria is expected to be met in order for this project to be deemed accepted. Another important purpose for this document is so that all stakeholders involved in this project have a common understanding of the scope, expectations, and goals of this project. August 1, 2019 is the expected completion date and submission of deliverables for this project.

A.2 Project Purpose and Justification

The Cloud Storage Mounting on Android OS project has been approved for researching, planning, designing, building, and implementation. The project involves mounting a cloud storage directory at the Operating System level so that the directory can be accessed from only application running on the mobile device. The purpose is to make that cloud storage appear to be native to the device rather than hosted elsewhere. The successful implementation of this project will meet the requirements for the course CS 6970 as well as produce software that will greatly reduce the friction of cloud storage access for mobile device users.

A.3 Scope Description

The scope of the Cloud Storage Mounting on Android OS project will involve research, planning, designing, building, and implementing a solution for mounting a cloud storage at the Operating System level of an Android mobile device. Open source resources will be used as tools for implementation as well as starting points.

A.4 High Level Requirements

The only requirement specified for this particular project is to properly perform mounting of a cloud storage directory so that it can be accessed from anywhere on the mobile device.

A.5 Boundaries

The scope of the Cloud Storage Mounting on Android OS project includes all work involving research, planning, designing, building, and implementing a solution for mounting a cloud storage at the Operating System level of an Android mobile device. Tasks that will also be involved include gathering requirements, writing up requested documents and documentation as well as the technical report, deploying the solution, and testing the solution on an Android mobile device. The scope of this project will not include an implementation for Windows or Apple mobile devices or implementations for other cloud storage providers outside of Google Cloud Platform.

A.6 Strategy

The Cloud Storage Mounting on Android OS project will be implemented by one developer using a machine running Linux and has Android Studio installed. The developer will research open source options to use as a template to begin the implementation. The implementation will be tested locally on the Linux machine to debug and make sure functionality is operating as expected. Then, either by using the mobile device emulator or the physical Android device (Samsung Galaxy Tab A6), the implementation will be tested on a mobile device for completeness.

A.7 Deliverables

The Cloud Storage Mounting on Android OS project will yield an Android APK file that can be downloaded and installed on an Android mobile device. A technical report detailing the project will be delivered as well.

A.8 Acceptance Criteria

Acceptance criteria have been established for the PMD Project to ensure thorough vetting and successful completion of the project. The acceptance criteria are both qualitative and quantitative in nature. All acceptance criteria must be met in order to achieve success for this project:

Meet all deliverables within scheduled time and budget tolerances. Reduce schedule delays by at least 30Reduce budget overruns by at least 30Improve Acme Consulting's resource allocation ability. Accomplish an overall performance improvement in program metrics.

Read more: https://www.projectmanagementdocs.com/template/project-documents/scope-statement/ixzz5pnIgkmLx

A.9 Constraints

The project must be completed within the range of days between May 13, 2019 and August 1, 2019. That is about 12 weeks. Development will be performed on a personal Linux (Ubuntu 18.10) machine. Testing will be performed on the same machine and a Samsung Galaxy Tab A6 mobile device that has been rooted.

A.10 Assumptions

The application will be installed on only Android devices. The user on the device has root access. The directory being mounted is on Google Cloud provider, not any other provider.

APPENDIX B

Development Plan

APPENDIX C

Lab Reports

C.1 Mount USB Through OTG Using ADB

First, the USB device needs to be authorized based on it's IP address.

```
hanen@hanen:~$ adb tcpip 5555
error: device unauthorized.
This adb server's $ADB_VENDOR_KEYS is not set
Try 'adb kill-server' if that seems wrong.
Otherwise check for a confirmation dialog on your device.
hanen@hanen:~$ adb connect 192.168.1.105:5555
connected to 192.168.1.105:5555
```

However, sometimes that isn't enough, especially once the computer running adb is disconnected from being attached to the tablet over USB cable.

```
hanen@hanen:~$ adb devices
List of devices attached
192.168.1.105:5555      unauthorized
0a655f09        device
```

The tablet was restarted. When prompted to allow for enabling USB debugging, a checkbox was selected to always allow that permission. This fixed the issue once the tablet was unplugged.

```
hanen@hanen:~$ adb kill-server
hanen@hanen:~$ adb start-server
* daemon not running; starting now at tcp:5037
* daemon started successfully
hanen@hanen:~$ adb connect 192.168.1.105:5555
connected to 192.168.1.105:5555
hanen@hanen:~$ adb devices
List of devices attached
192.168.1.105:5555      device
0a655f09        device
```

Now, it is time to partition the drive in order to mount.

```
hanen@hanen:~$ adb -s 192.168.1.105:5555 shell
1|shell@flo:/ $ sm list-disks
disk:8,0
shell@flo:/ $ sm partition disk:8,0 private
shell@flo:/ $
```

Under Settings -> Storage USB, it will show that the portable device (the flash drive) was removed/no longer exists. This is the point in which the tablet restarts. Once it turns back on, the USB Drive has moved out of portable devices and into the collection of internal storage.

C.2 OCamlFUSE

C.3 RClone

C.4 ODrive

C.5 InSync

C.6 X-Plore

C.7 Write FileSystem with FUSE

C.8 ES File Explorer

C.9 Mounting using SSHFS

APPENDIX D

Meeting Minutes

D.1 5/14/2019

This initial meeting of the summer involved deciding on what project would be worked on for this independent study course. Dr. Mateti presented two topics we had discussed in meetings prior to this after we had discussed other options and narrowed them down to a project dealing with Cloud Computing or a project dealing with Cloud Storage. Both projects would involve Android. The Cloud Computing topic would be one in which I would research different cloud computing approaches. The Cloud Storage topic would involve implementing a solution for mounting different cloud storage folders onto an Android device at the operating system level. There are many apps available that will do this very thing on the application level, where a user interacts with their cloud storage through the application itself, but cannot access that mounted storage from any other app on the device. Dr. Mateti showed me some apps like that on his phone which included ES File Explorer and X-plore. The goal would be mounting the storage at the OS level so that the user can access it from any app as if it were simply just another folder on their Android file system.

I decided to select the topic of Cloud Storage, because I really wanted to implement something this summer and this sounded both interesting and personally useful. Once I selected the topic I wanted to focus on, Dr. Mateti elaborated further on expectations and next steps. We will meet in his office at 6pm on Tuesdays or Thursdays on an as-needed basis. In the meantime, I will be uploading my meeting minutes and technical document drafts to a shared Google Drive folder that Dr. Mateti has been given access. I will, also, be organizing the tasks needed to accomplish this project using an application called Trello. Dr. Mateti has been given access to this board on Trello. The technical document will be around 50 pages once it is complete and will use Software Engineering principles. I was tasked to explore open source options for mounting Google Cloud and/or Firebase. Other things to read about and look into included Fuse and SSHFS. The plan will be to implement any solution on a Linux PC first. Then, when it is working to our satisfaction, it will be ported over to Android.

Dr. Mateti also provided me with an Android tablet to be used for this course. I will be bringing back the other tablet I borrowed in our next meeting.

D.2 6/4/2019

Dr. Mateti and I met to primarily discuss the development plan for my project for the summer. The goal for the summer is to be able to demonstrate mounting Google Drive at the operating system level on an Android device. The secondary goal is to try the same for another cloud storage provider, but we will determine later if we have time for that.

On KDE, I need to install GDrive (KIO). There is an authentication error that appears that can be resolved by going into System Settings and adding an Online Account for Google Drive. There was also another error that could be resolved by going into Personal Settings and enabling the KDE Wallet. If all else fails, remove the account under System Settings and add it back.

Dr. Mateti provided some helpful hints on using Latex such as for using a typewriter script to distinguish a certain chunk of text from the remaining report. beginverbatim terminal output endverbatim was another helpful tip.

Dr. Mateti also provided me with a usb input cable that I will use to create a DIY OTG cable, so that I can plug a USB into my tablet. I will plug the USB into my tablet and mount it using the code I develop.

Some action items that Dr. Mateti sent me away with were:

1. Put together a Development Plan document.
2. Revisit the init lab from the Android Security and Internals course from last semester.
3. Browse the OcamlFUSE code and determine comfort level with using it. Do lab report.
4. Rclone might be able to do mounting at the OS level and can handle more cloud storage providers. Do lab report. Include SLOC count of source code.
5. Odrive. Do lab report.
6. InSync. Try the free trial. Do lab report. Do lab report on mounting a USB drive.

D.3 6/11/2019

Dr. Mateti provided me with a tablet that has already been rooted. This will help me make progress without the risk of bricking the device like I did before. Dr. Mateti reviewed his feedback to my OCamlFUSE lab report. He shared a Latex format he would like me to use for my final report. He asked me to mount a usb drive using an OTG cable. He, also, asked that I make some changes to an existing OCaml project. The plan is to focus on these two items. The feedback given by Dr. Mateti was uploaded on Google Drive. I will need to upload the Latex files he provided and fill things in based on his suggestions.

APPENDIX E

Acronyms

OS operating system