Rayyan J. Jupyter notebook showcase

The whole point of this notebook is to track the learning progress of the various software development techneques that we are introduced to in SEng-265. First thing on the agenda after getting this notebook started was to learn a little bit about markdown and how it works. Being in the Mathematics and Computer Science program, learning markdown/LaTeX is essentially inevitable. Now to get rid of this plain steryotypical monotone background I did have to install a custom theme using do not click me! (sarcasm)

The whole idea of Jupyter notebook is honestly genius; a great way to show code but in a hyper organised method, where code can be run line-by-line and have asthetically pleasing markdown text documenting the code or the procedure of how it was being developed.

An important yet lacking skill for most engineers is to be able to communicate what their ideas are and this is a great way to exponentially increase the readability and understanding of whatever needs to be said to a novice user/listener.

To get a basic hold of markdown I used this intuitive guide found online here that gives a basic understanding of it alongside HTML equivalent which most people are fisrt introduced with.

Links and Equations

Now let us play around with some markdown/LaTeX formatting

Since we already used link insertion above (pretty similar to image insertion) let us see how we can inster an image into this notebook below; we use "" with the link of the respective image in the parenthesis, here's an example;

Now let us play around with some fancy math equations and see how that works; To start a formatting math equations we can sue $$ and end with the same pattern, and inside we use regular ***LaTeX*** commands, an example here is the Cauchy Inequality

$$\left(\sum_{k=1}^{n} a_k b_k\right)^2 \leq \left(\sum_{k=1}^{n} a_k^2\right)\left(\sum_{k=1}^{n} b_k^2\right)$$

Here is some cool LaTeX formatting that I used in my previous Mathematics courses;

Firstly some Matrix row reductions used in M211

$$\left[\begin{array}{cccc|c} 1 & -2 & 4 & -8 & -1 \\ 1 & -1 & 1 & -1 & 7 \\ 1 & 2 & 4 & 8 & -5 \\ 1 & 3 & 9 & -27 & -1 \end{array}\right] \xrightarrow[R_4 - R_1]{R_3 - R_1 R_4 - R_1} \left[\begin{array}{cccc|c} 1 & -2 & 4 & -8 & -1 \\ 0 & 1 & -3 & 7 & 8 \\ 0 & 4 & 0 & 16 & -4 \\ 0 & 5 & 5 & 35 & 0 \end{array}\right]$$

Some proof writing with vectors and matrices again; Adding the above two matrices to get $\begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}$, now using the definition of the set Y we can write the equation out for the new matrix:

$$(x_1 + y_1)v_1 + (x_2 + y_2)v_2$$

$$x_1v_1 + y_1v_1 + x_2v_2 + y_2v_2$$

After rearranging the terms we get :

$$(x_1v_1 + x_2v_2) + (y_1v_1 + y_2v_2)$$

Here's a way we can write symbols for Algeabric fields like Real's, Complex and Rationals or an arbitarary vector space; $\mathbb{R}$ $\mathbb{C}$ $\mathbb{Q}$ $\mathbb{V}$

In [5]:
```
jt -t gruvboxd
```

The above command illustrates the grubvbox theme for python code block

Unix and Bash

Unix systems or even Linux systems have been around since the dawn of modern computing. Most people go about their days without even realizing the fact that our internet driven world runs on one core thing and that is unix systems like GNU/Linux or FreeBSD; every TV electronic system you can find around your house probably runs some form of Unix based system, hell even MacOS was initially developed on top of FreeBSD, android systems are essentially linux systems. This just goes to show that the most used operation system in the grand scheme of things is indeed GNU/Linux.

Bash scripting which is OS-independent is plenty useful to automate certain system specific tasks. Since I've been using Linux since basically highschool (My setup being Arch linux paired with dwm which is a C based window manager); using dwm which is quite barebones I use bash scripts to run a bunch of commands on startup that setup my status bar

```
[1] 2  3  4  5  6  7  8  9  [|=  Team Portfolio - Jupyter Notebook — Mozilla Firefox     [D] Sat Feb 17 09:46:40 PM PST 2024 [Mem] 7.5Gi [Net]  [V] 61 [Bat] 35%
```

using my script

```
while true; do
    d=$(date)
    m=$(free -h | awk '(NR==2){ print $3 }')
    b=$(cat /sys/class/power_supply/BAT0/capacity)
    i=$(cat /sys/class/net/wlp1s0/operstate)
    v=$(pamixer --get-volume)
    xsetroot -name "[D] $d [Mem] $m [Net] $i [V] $v [Bat] $b%"
    sleep 5
done
```

I have a couple more bash scripts for things like dual display setup for my laptop which I can run when I connect my monitor and it sets-up all my orientation and all preferences as desired;

```
xrandr --dpi 276 --fb 6640x3960 \
    --output eDP --mode 2880x1800 \
    --output DisplayPort-0 --scale 2x2 --pos 2880x0 --panning 3840x2160+3200+0
```

These are things that most desktop-environments in Unix based systems automatically implement anyway so without even knowing it you could be using bash in your daily life!!

C Programming

My background in C before starting this course was minimal to say the least. I had the basic idea about the syntax and whatnot but no real programming. My method to learn new things is always to **do** something with it which creates a way for me to learn random things along the learning process and work on making the method more efficient as I go on my journey. C relative to modern programming languages like *Python* or *Java* is considered ***lower*** level. Most things like machine learning or web development have much more **accesible** languages which are used today like Python or Javascript along with other things due to extensive library sets. But the core *i.e* OS development is still rich in the C language. One really good application that comes to my mind which is written in C is **dwm** which is written by the wonderful folks at , their whole aim is to develop jargon-free, minimilistic daily software written in pure C avoiding unnecessary libraries and maintaining a low line count for max efficiency. All their software can be ran on BSD absed systems and GNU/Linux and is compilation based which is great. Doing the first assignment was not too challenging but running into errors like segmentation faults and memory errors were quite annoying after getting used to the interface of simpler languages like python and java in first year.

Here is the dwm folder

```
rayyan@archlabs ~/dwm % ll
total 416K
-rw-rw-r-- 1 rayyan users 8.4K Mar 13  2022 config.def.h
-rw-rw-r-- 1 rayyan users 7.2K Jan 18  2021 config.def.h.orig
-rw-rw-r-- 1 rayyan users 1.4K Jan 18  2021 config.def.h.rej
-rw-rw-r-- 1 rayyan users 7.0K Sep 16  2022 config.h
-rw-rw-r-- 1 rayyan users  932 Jan 18  2021 config.mk
-rw-rw-r-- 1 rayyan users  984 Jan 18  2021 config.mk.orig
-rw-rw-r-- 1 rayyan users  11K Jun 19  2020 drw.c
-rw-rw-r-- 1 rayyan users 1.7K Jun 19  2020 drw.h
-rw-r--r-- 1 root   root   11K Sep 16  2022 drw.o
-rwxr-xr-x 1 root   root   70K Sep 16  2022 dwm*
-rw-rw-r-- 1 rayyan users 5.2K Jul 21  2020 dwm.1
-rw-rw-r-- 1 rayyan users 5.2K Jul 21  2020 dwm.1.orig
-rw-rw-r-- 1 rayyan users  62K Mar 13  2022 dwm.c
-rw-rw-r-- 1 rayyan users 4.3K Nov 10  2018 dwm-centeredmaster-6.1.diff
-rw-rw-r-- 1 rayyan users  58K Mar 13  2022 dwm.c.orig
-rw-rw-r-- 1 rayyan users 1.2K Jul 21  2020 dwm.c.rej
-rw-r--r-- 1 rayyan users 6.7K Sep 13  2021 dwm-moveresize-20210823-a786211.diff
-rw-r--r-- 1 root   root   59K Sep 16  2022 dwm.o
-rw-rw-r-- 1 rayyan users  373 Jun 19  2020 dwm.png
-rw-rw-r-- 1 rayyan users 7.9K Nov 10  2018 dwm-swallow-20170909-ceac8c9.diff
-rw-rw-r-- 1 rayyan users 9.8K Dec 10  2020 dwm-swallow-20201211-61bb8b2.diff
drwxr-xr-x 7 rayyan users 4.0K Mar 11  2022 .git/
-rw-rw-r-- 1 rayyan users 1.9K Jun 19  2020 LICENSE
-rw-rw-r-- 1 rayyan users 1.2K Jun 19  2020 Makefile
-rw-rw-r-- 1 rayyan users    0 Jul  7  2020 .nvimlog
-rw-rw-r-- 1 rayyan users 1.1K Jun 19  2020 README
-rw-rw-r-- 1 rayyan users  847 Jun 19  2020 transient.c
-rw-rw-r-- 1 rayyan users  517 Jun 19  2020 util.c
-rw-rw-r-- 1 rayyan users  305 Jun 19  2020 util.h
-rw-r--r-- 1 root   root  2.2K Sep 16  2022 util.o
rayyan@archlabs ~/dwm %
```

We can then create diff files (also called patches) to apply to the dwm.c or config.h files which essentially look like as below;

```
 5
 6 ---
 7  config.def.h |   7 ++-
 8  config.mk    |   2 +-
 9  dwm.c        | 190 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++--
10  3 files changed, 190 insertions(+), 9 deletions(-)
11
12 diff --git a/config.def.h b/config.def.h
13 index a9ac303..2d26a23 100644
14 --- a/config.def.h
15 +++ b/config.def.h
16 @@ -26,9 +26,10 @@ static const Rule rules[] = {
17       *  WM_CLASS(STRING) = instance, class
18       *  WM_NAME(STRING) = title
19       */
20 -   /* class      instance    title       tags mask     isfloating   monitor */
21 -   { "Gimp",     NULL,       NULL,       0,            1,           -1 },
22 -   { "Firefox",  NULL,       NULL,       1 << 8,       0,           -1 },
23 +   /* class      instance    title       tags mask     isfloating   isterminal noswallow monitor */
24 +   { "Gimp",     NULL,       NULL,       0,            1,           0,         0,        -1 },
25 +   { "Firefox",  NULL,       NULL,       1 << 8,       0,           0,         0,        -1 },
26 +   { "st",       NULL,       NULL,       0,            0,           1,         1,        -1 },
27  };
28
29  /* layout(s) */
30 diff --git a/config.mk b/config.mk
31 index 80dc936..5ed14e3 100644
32 --- a/config.mk
33 +++ b/config.mk
34 @@ -22,7 +22,7 @@ FREETYPEINC = /usr/include/freetype2
35
```

These diff files have two types of lines one with -- which essentially remove lines of code and ** which add lines of code, obviously these can be done manually but to keep track of updates which are added these are super helpful!

Anyhting you want different? Well you can open up the .c file and change as you please. True software freedom is something all developers should strive for especially in a time where this very basic control has been taken away in most places (MS Software/Apple based software).