## CONTENTS

1 CSX-1 (23-ID-1) Beamline Documentation
   1.1 Contents ......................................................... 3

2 Guides and Tutorials
   2.1 HELP!! The %$^$#@% just crashed .......................... 7
   2.2 Controls Account Setup Guide .............................. 8
   2.3 Installing a Personal Conda Environment .................. 10
   2.4 Installing Linxbrew ........................................... 12
   2.5 UNIX Primer and Account Setup ............................ 13
   2.6 Markdown Cheat Sheet ......................................... 15
   2.7 h1 Heading ....................................................... 16
   2.8 Vim Commands Cheat Sheet ................................. 24
   2.9 tmux shortcuts & cheatsheet ............................... 30

3 CSXTOOLS Documentaation
   3.1 Introduction ................................................... 35
   3.2 Contents ....................................................... 35

4 Example Notebooks
   4.1 iPython Notebook To Export CSX Data as a suitcase ....... 37

5 Downloads ........................................................... 39
These pages are the documentation of the CSX beamlines (23-ID-1 and 23-ID-2) at the NSLS-II.

Fig. 1: CSX Beamlines control area at the NSLS-II
1.1 Contents

1.1.1 Fast CCD Detector

Introduction

The FastCCD installed in the endstation at CSX-1 is of the LBNL Fast CCD design. The sensor contains 1920 x 960 pixels of 30 µm x 30 µm and is arranged into two halves of 960 rows by 960 columns with the columns parallel to the long CCD axis. There is one output for each 10 columns (a “super column”) which results in 192 individual outputs and analogue to digital converters (ADC). The CCD camera can either be used in a traditional CCD with an x-ray shutter exposing the full chip, or in a framestore (frame transfer) mode by covering two quarters of the CCD with a light (x-ray) block effectively exposing half the chip along the column direction.

The analogue CCD signal is digitized by a custom designed fCRIC. Each fCRIC has 16 analogue inputs and digitizes with 13 bit precision and had 16 bit dynamic range. This is accomplished by having 3 gain ranges of 8x, 4x and 1x with an auto gain feature. In order to allow negative charge injection. The ADC is biased at a value of approximately 4096 (0x1000 in hex) with the exact value dependent on the ADC channel. The gain settings are stored in the two most significant bits of each ADC reading. The schematic of a single fCRIC channel is shown in the LBNL fCRIC Circuit Diagram.

![LBNL fCRIC Circuit Diagram](image)

Fig. 1.1: LBNL fCRIC Circuit Diagram

The specifications of the CCD are summarized below:

- Pixel Size: 30 µm x 30 µm
• Active Area: 1920 pixels (column) x 960 pixels (row)
• 192 super columns = 192 outputs (480 rows x 10 columns)
• Back illuminated
• 250 µm - 350 µm thickness
• Full well : ~900k e⁻ per pixel
• Sensitivity : 6 e⁻ / ADU for 8x gain (max gain)
• Pixel readout time: 500 µs
• Digitization time: 2 µs at 120 Hz
• 100 Hz maximum data collection

**Data Format**

In treating the raw CCD data from the FastCCD there are a few important considerations related to the multi-gain behaviour of the fCRIC amplifier and digitizer. The raw 16 bit values that are recorded in the data file follow the *16 Bit fCRIC Data Format* shown below with the two gain bits following the *fCRIC Gain Setting*.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>09</th>
<th>08</th>
<th>07</th>
<th>06</th>
<th>05</th>
<th>04</th>
<th>03</th>
<th>02</th>
<th>01</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>G0</td>
<td>ERR</td>
<td>D12</td>
<td>D11</td>
<td>D10</td>
<td>D09</td>
<td>D08</td>
<td>D07</td>
<td>D06</td>
<td>D05</td>
<td>D04</td>
<td>D03</td>
<td>D02</td>
<td>D01</td>
<td>D00</td>
</tr>
</tbody>
</table>

**Table 1.2: fCRIC Gain Setting**

<table>
<thead>
<tr>
<th>G1</th>
<th>G0</th>
<th>Gain</th>
<th>Pre-factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>x1</td>
<td>x8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>x2</td>
<td>x4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x1</td>
<td>x8</td>
</tr>
</tbody>
</table>

Here the two most significant bits record the gain setting for the encoded value. The least significant 13 bits hold the measured analogue value. The actual value is therefore related to the measured value by:

\[
A_{corr} = G(A_{meas} - O)
\]

where \(A_{corr}\) is the corrected intensity, \(A_{meas}\) is the measured value by the ADC, \(G\) is the gain of the ADC and \(O\) is the bias offset.

**Dark Image Subtraction**

Due to the multi gain nature of the fCRIC it is therefore necessary to take 3 dark images at different gain settings to obtain the different ADC offsets under these modes. As the lower gain settings are not subject to considerable contribution due to dark current it is usually justifiable to measure only the highest gain dark image repeatedly. Given 3 dark images for the different gain settings the images the following python pseudo code can be used to correct for dark current and gain:

```python
import numpy as np

def subtract_background(image, dark_image, gain = [1, 4, 8]):
    gain_mask_8 = (image & 0xC000) == 0xC000
    gain_mask_4 = (image & 0xC000) == 0x8000
    gain_mask_1 = (image & 0xC000) == 0x0000

    cor_image = image.astype(np.float16)
```

---

**4 Chapter 1. CSX-1 (23-ID-1) Beamline Documentation**
cor_image -= gain_mask_8 * dark_image[2]  
cor_image -= gain_mask_4 * dark_image[1]  
cor_image -= gain_mask_1 * dark_image[0]  

gain_image = (gain_mask_8 * gain[2]) + (gain_mask_4 * gain[1]) + (gain_mask_1 * gain[0])  

return (cor_image * gain_image), gain_image

Useful Links

- LBNL Fast CCD Site
- csxtools python analysis routines
- libcin low level c driver
- areaDetector Driver
GUIDES AND TUTORIALS

Contents:

2.1 HELP!! The %$^$#@% just crashed

2.1.1 Managing IOCs

Soft IOCs are managed through the `manage-iocs` script. To obtain a list of softiocs running on a NSLS-II computer use the command `manage-iocs report` an example is shown below for xf23id1-ioc3:

```
[swilkins@xf23id1-ioc3 ~]$ manage-iocs report
nBASE | IOC | USER | PORT | EXEC
/epics/iocs | apcupsd | root | 5000 | /epics/iocs/apcupsd/st.cmd
/epics/iocs | cam-diag1 | softioc | 4202 | /epics/iocs/cam-diag1/st.cmd
/epics/iocs | cam-diag6 | softioc | 4300 | /epics/iocs/cam-diag6/st.cmd
/epics/iocs | cam-dif1 | softioc | 4204 | /epics/iocs/cam-dif1/st.cmd
/epics/iocs | cam-dif2 | softioc | 4205 | /epics/iocs/cam-dif2/st.cmd
/epics/iocs | cam-dif3 | softioc | 4206 | /epics/iocs/cam-dif3/st.cmd
/epics/iocs | cam-dif-beam | softioc | 4201 | /epics/iocs/cam-dif-beam/st.cmd
/epics/iocs | ct-eps | softioc | 4002 | /epics/iocs/ct-eps/st.cmd
/epics/iocs | es-dg645 | softioc | 5013 | /epics/iocs/es-dg645/st.cmd
/epics/iocs | es-K2611 | softioc | 4302 | /epics/iocs/es-K2611/st.cmd
/epics/iocs | es-tcrtl1 | softioc | 5010 | /epics/iocs/es-tcrtl1/st.cmd
/epics/iocs | es-vortex | softioc | 4301 | /epics/iocs/es-vortex/st.cmd
/epics/iocs | mc11 | softioc | 5001 | /epics/iocs/mc11/st.cmd
/epics/iocs | mc12 | softioc | 5002 | /epics/iocs/mc12/st.cmd
/epics/iocs | mc13 | softioc | 5003 | /epics/iocs/mc13/st.cmd
/epics/iocs | omegaM4061 | softioc | 5012 | /epics/iocs/omegaM4061/st.cmd
/epics/iocs | simdetector | softioc | 4203 | /epics/iocs/simdetector/st.cmd
/epics/iocs | simmotor | softioc | 8001 | /epics/iocs/simmotor/st.cmd
/epics/iocs | timestamp | softioc | 6001 | /epics/iocs/timestamp/st.cmd
/epics/iocs | va-bakeout-01 | softioc | 4001 | /epics/iocs/va-bakeout-01/st.cmd
/epics/iocs | zebra | softioc | 5011 | /epics/iocs/zebra/st.cmd
```

To connect to the IOC console, telnet to localhost at the port that is shown in the table. For example to connect to the mc12 console issue the command:

```
[swilkins@xf23id1-ioc3 ~]$ telnet localhost 5002
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
@@@ Welcome to procServ (procServ Process Server 2.6.0)
```
In order to reboot the IOC, type [CTRL] + X. To leave the console type [CTRL] + ] and type close at the telnet> prompt

To start all IOCs configured on the system issue the command sudo manage-iocs startall and if needed to stop all IOCs issue the command sudo manage-iocs stopall

2.1.2 OLog Glassfish Server

To reboot the glassfish server on xf23id-ca.cs.nsls2.local execute:

```
swilkins@xf23id-ca:$ sudo su - glassfish
glassfish@xf23id-ca:$ cd glassfish3/bin/
glassfish@xf23id-ca:$ /glassfish3/bin$ ./asadmin stop-domain domain1
glassfish@xf23id-ca:$ /glassfish3/bin$ ./asadmin stop-domain domain2
glassfish@xf23id-ca:$ /glassfish3/bin$ ./asadmin start-domain domain1
glassfish@xf23id-ca:$ /glassfish3/bin$ ./asadmin start-domain domain2
```

2.2 Controls Account Setup Guide

2.2.1 Introduction

NSLS-II controls accounts allow access to all computers at CSX. Once a new account has been created, the following is suggested to get the account working correctly.

2.2.2 Fixing Permissions

Firstly, all accounts are created with global read permissions. If this is OK by you then skip the next step. However, if you want to restrict access to files in your home directory then do the following:

```
[swilkins@xf23id1-srv2 ~ ]$ chmod o-rwx,g-w+rx,u+rwx /home/<uid>
```

2.2.3 Setting up an SSH Key

Create the RSA Key Pair

The first step is to create an SSH key pair in your home directory:

```
ssh-keygen -t rsa
```

the ssh-keygen command will ask you a few questions:
Enter file in which to save the key (/home/demo/.ssh/id_rsa):

You can press enter here, saving the file to the user home (in this case, the user is called demo).

Enter passphrase (empty for no passphrase):

You should enter a passphrase. The security of a key, no matter how encrypted, still depends on the fact that it is not visible to anyone else. Should a passphrase-protected private key fall into an unauthorized user's possession, they will be unable to log in to its associated accounts until they figure out the passphrase.

The entire key generation process looks like this:

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/demo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/demo/.ssh/id_rsa.
Your public key has been saved in /home/demo/.ssh/id_rsa.pub.
The key fingerprint is:
The key's randomart image is:
++[ RSA 2048]----+
<p>| .oo. |
| . o.E |
| + . o |
| . = = . |
| = $ = . |
| o + = + |
| . o + o . |</p>
<table>
<thead>
<tr>
<th>. o</th>
</tr>
</thead>
</table>
```

The public key is now located in /home/demo/.ssh/id_rsa.pub The private key (identification) is now located in /home/demo/.ssh/id_rsa

Copy the Public Key

Once the key pair is generated, you can now install it on the controls system to allow access. This can be done with the `ssh-copy-id` command:

```
ssh-copy-id user@box64-2
```

where user is your username. If successful, you should see something like:

```
[swilkins@xf23id1-srv2 ~/Scratch ]$ ssh-copy-id swilkins@box64-2
********************
* NOTICE TO USERS *
* This is a Federal computer system (and/or it is directly connected *
* to a BNL local network system) and is the property of the United *
* States Government. It is for authorized use only. Users (authorized or *
* unauthorized) have no explicit or implicit expectation of privacy. *
* Any or all uses of this system and all files on this system may be *
* intercepted, monitored, recorded, copied, audited, inspected, and *
* disclosed to authorized site, Department of Energy, and law *
```
Now try logging into the machine, with "ssh 'swilkins@box64-2'", and check in:

~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

Now you are done ... you should be able to login using SSH kets now.

2.3 Installing a Personal Conda Environment

2.3.1 Introduction

This guide covers installing the conda packaging system miniconda into your own space. This has a number of advantages, including full control on package versions. Due to the large nature of the files, it is suggested not to install into your home directory if you are using NFS home directories. At CSX conda should be installed on the GPFS file system in the directory /GPFS/xf23id/users/<uid> where <uid> is your user ID.

To create a directory on the GPFS system, do the following:

```bash
[swilkins@23id1-srv2 ~]$ sudo mkdir /GPFS/xf23id/users/<uid>
[swilkins@23id1-srv2 ~]$ sudo chown <uid> /GPFS/xf23id/users/<uid>
[swilkins@23id1-srv2 ~]$ sudo chmod 750 /GPFS/xf23id/users/<uid>
```

Where <uid> is your userid.

2.3.2 Installing Miniconda

Install the latest miniconda. This can be done by downloading the latest miniconda binary installer from conda.pydata.org.

Once the file is downloaded make the file executable and run the installer. DO NOT let the installer change your .bashrc file:

```bash
[swilkins@xf23id1-srv2 ~/Downloads]$ chmod u+x Miniconda3-latest-Linux-x86_64.sh
[swilkins@xf23id1-srv2 ~/Downloads]$ ./Miniconda3-latest-Linux-x86_64.sh
```

When prompted, enter the GPFS path for the install location, for example:

```bash
Do you approve the license terms? [yes|no]
[no] >>> yes

Miniconda3 will now be installed into this location:
```
To add the miniconda to your path, edit your .bashrc file with your favorite editor and add the following lines.

```bash
if [-e "/GPFS/xf23id/users/swilkins/miniconda3" ]; then
    export PATH="/GPFS/xf23id/users/swilkins/miniconda3/bin:$PATH"
fi
```

Substituting `swilkins` for your user ID. To enable the path, now source your .bashrc file:

```
[swilkins@xf23id1-srv2 ~/Downloads]$ source ~/.bashrc
```

### 2.3.3 Installing a Custom Conda Environment

Once miniconda is installed and in the path, configure conda to use the NSLS-II anaconda cloud server:

```
[swilkins@xf23id1-srv2 ~/Downloads]$ conda install anaconda-client conda-build --yes
[swilkins@xf23id1-srv2 ~/Downloads]$ conda config --add channels anaconda
[swilkins@xf23id1-srv2 ~/Downloads]$ conda config --add channels latest
[swilkins@xf23id1-srv2 ~/Downloads]$ conda config --add create_default_packages pip
[swilkins@xf23id1-srv2 ~/Downloads]$ conda config --add create_default_packages anaconda-client
[swilkins@xf23id1-srv2 ~/Downloads]$ anaconda config --set url https://conda.nsls2.bnl.gov/api
[swilkins@xf23id1-srv2 ~/Downloads]$ conda config --remove channels defaults --force
[swilkins@xf23id1-srv2 ~/Downloads]$ conda update --all --yes
```

Congratulations! You now have a personal installation of miniconda connected to the NSLS-II anaconda cloud server. Now you can create a new environment for doing your analysis. To create and activate the environment type:

```
[swilkins@xf23id1-srv2 ~/Downloads]$ conda create -n analysis python=3.5
[swilkins@xf23id1-srv2 ~/Downloads]$ source activate analysis
(analysis)[swilkins@xf23id1-srv2 ~/Downloads]$ conda install dataportal
(analysis)[swilkins@xf23id1-srv2 ~/Downloads]$ conda install ipython-notebook
```

### 2.3.4 Installing a Custom Notebook Kernel

To get your custom environment to work with the notebook server, you have to create a kernel file in your .python/kernel directory (where `my-analysis-kernel` can be any name you wish to know this kernel by):

```
(analysis)[swilkins@xf23id1-srv2 ~]$ cd ~/.ipython/kernels/
(analysis)[swilkins@xf23id1-srv2 ~/.ipython/kernels]$ mkdir my-analysis-kernel
```

Now create a kernel json file to let the notebook server know how to run this kernel. Create a file `kernel.json` in the directory `my-analysis-kernel` with your favorite text editor such as:

```json
{
    "argv":["/home/swilkins/miniconda3/envs/analysis/bin/python3.5",
    "-m", "IPython.kernel", "-f", "[connection_file]"],
    "display_name": "xf23id1-srv2 - Analysis Conda Env",
    "host": "xf23id1-srv2"
}
```

### 2.3. Installing a Personal Conda Environment
Where the path /home/swilkins/miniconda3/envs/analysis/bin/python3.5 should point to the path of python in your home directory conda environment. display_name should be a nice name for this kernel, and the host is the computer on which the kernel should run.

If all works OK, the new kernel should show up in the kernel list on notebook.nsls2.bnl.gov

Running a new notebook from that option will now run a kernel in the new conda environment:

Congratulations!!

2.4 Installing Linxbrew

2.4.1 Introduction

Linxbrew is a package manager for Linux which can be used to have a local installation of such utilities such as git, gist and tmux. It is especially useful to get around systems which have relatively out of date core utilities (not looking at you debian). As it doesn’t need superuser privileges it can be installed in a home directory for example.

2.4.2 Installing

To install at CSX, it is prefertable to install in the GPFS system in your userspace. To do this:

```
[swilkins@xf23id1-srv2 ~]$ git clone https://github.com/NSLS-II-CSX/linuxbrew.git /GPFS/xf23id/users/<uid>/linuxbrew
```

where <id> should be substituted for your username. Now edit your .bashrc file and add the following:
if [ -e "/GPFS/xf23id/users/<id>/linuxbrew" ]; then
    export PATH="/GPFS/xf23id/users/<id>/linuxbrew/bin:$PATH"
    export MANPATH="/GPFS/xf23id/users/<id>/linuxbrew/share/man:$MANPATH"
    export INFOPATH="/GPFS/xf23id/users/<id>/linuxbrew/share/info:$INFOPATH"
fi

where as before <id> should be replaced with your username. You are now ready to install some useful programs. As a suggestion, the following are useful:

[swilkins@xf23id1-srv2 ~]$ brew install openssl
[swilkins@xf23id1-srv2 ~]$ brew install git
[swilkins@xf23id1-srv2 ~]$ brew install gist
[swilkins@xf23id1-srv2 ~]$ brew install tmux

2.5 UNIX Primer and Account Setup

2.5.1 File Permissions

In a unix system every file (and directory) has an owner, an associated group, and a set of permission flags that specify separate read, write, and execute permissions for the user (owner), group, and other. Other is also sometimes known as world permissions, and applies to all users who can login to the system.

The permission flags can be seen by the ls -l command:

[swilkins@xf23id1-srv1 ~/Presentations]$ ls -l
total 24M
-rw-rw---- 1 swilkins csx 22M Mar 5 2014 csx.pdf
The first field is a set of 10 permission flags. The first flag is the file type. For regular files it is not set (-). Directories are indicated by d and links by l. The remaining 9 flags are 3 groups of 3 for read (r) / write (w) / execute (x) for the user, group and other. In the example above the files are read/write permitted for both the user (swilkins) and any members of the csx group. Any other users cannot read or write the files. Note that the directory other has the execute bit set for only the user and group to restrict listing to only the user and members of the csx group.

To set file permissions, UNIX uses the octal representation of file permissions, with 4 digits representing the special, user, group and other. These permissions are outlined in the table "octal file permissions".

<table>
<thead>
<tr>
<th>Octal digit</th>
<th>chmod Permissions</th>
<th>umask Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No permissions</td>
<td>Any permission may be set</td>
</tr>
<tr>
<td>1</td>
<td>Execute permission only</td>
<td>Setting of execute permission is prohibited</td>
</tr>
<tr>
<td>2</td>
<td>Write permission only</td>
<td>Setting of write permission is prohibited</td>
</tr>
<tr>
<td>3</td>
<td>Write and execute permissions</td>
<td>Setting of write and execute permission is prohibited</td>
</tr>
<tr>
<td>4</td>
<td>Read permission only</td>
<td>Setting of read permission is prohibited</td>
</tr>
<tr>
<td>5</td>
<td>Read and execute permissions</td>
<td>Setting of read and execute permission is prohibited</td>
</tr>
<tr>
<td>6</td>
<td>Read and write permissions</td>
<td>Setting of read and write permission is prohibited</td>
</tr>
<tr>
<td>7</td>
<td>Read, write and execute permissions</td>
<td>All permissions are prohibited from being set</td>
</tr>
</tbody>
</table>

The file permissions can be changed using the chmod command with the permissions specified in an octal format and can be applied recursively using the -R option. For example:

```
[swilkins@xf23id1-srv1 ~/Presentations]$ ls -l
total 24M
-rw-rw---- 1 swilkins csx 22M Mar 5 2014 csx.pdf
-rw-rw---- 1 swilkins csx 1.2M Mar 3 2014 csx.ppt
drwxrwx--- 2 swilkins csx 6 Nov 24 06:15 other
[swilkins@xf23id1-srv1 ~/Presentations]$ chmod 700 other
[swilkins@xf23id1-srv1 ~/Presentations]$ ls -l
total 24M
-rw-rw---- 1 swilkins csx 22M Mar 5 2014 csx.pdf
-rw-rw---- 1 swilkins csx 1.2M Mar 3 2014 csx.ppt
drwx------ 2 swilkins csx 6 Nov 24 06:15 other
```

UNIX uses the umask to define how (new or default) files permissions are set. This can be set on a per-user basis using the umask command to set the default (allowed) permissions. The umask command uses the octal representation the same as chmod to define permissions and should have the restricted permissions set - i.e. the inverse of what you want the files to be.

```
$ umask 0227  # allow user rwx group r-x and other r-x  (-rwxr-xr-x)
$ umask 0227  # allow user rwx group r-x and restrict other (-rwxr-x---)
$ umask 0007  # allow group and user rwx and restrict other (-rwrxrwx---)
$ umask 0007  # allow only user rwx  (-rwx------)
```

The following example illustrates the use of the umask command:

```
[swilkins@xf23id1-srv1 ~/Example]$ umask 0007
[swilkins@xf23id1-srv1 ~/Example]$ touch hello_world
[swilkins@xf23id1-srv1 ~/Example]$ ls -l
total 0
-rw-rw---- 1 swilkins csx 0 Nov 24 06:55 hello_world
```
To set the umask for every session the command should be added to your .bashrc file. CSX Team members should add the following to their .bashrc:

```bash
umask 0027
```

### 2.5.2 Bash Shell Setup

The BASH shell is initialized by two main files .bashrc and .bash_profile. These are executed by the shell differently depending on how the shell is executed. This is often a source of confusion. These files are summarized below:

- **.bashrc**: This is run by interactive shells. These are shells that are connected to a terminal (or pseudo-terminal) such as a xterm running under a windowing system.
- **.bash_profile**: This is run by login shells. These are shells that are started when you login from another host or you login from the text terminal on a local machine.

BASH is also different from other shells in that .bashrc and .bash_profile are mutually exclusive (i.e. only one is run on the shell startup). To get round this problem, most people place teh following in the .bash_profile so that the shell is initialized the same way for both interactive and login shells:

```bash
if [ -f ~/.bashrc ]; then
    source ~/.bashrc
fi
```

### 2.6 Markdown Cheat Sheet

#### 2.6.1 Headings

Headings from h1 through h6 are constructed with a # for each level:

- `# h1 Heading`
- `## h2 Heading`
- `### h3 Heading`
- `#### h4 Heading`
- `##### h5 Heading`
- `###### h6 Heading`

Renders to:
2.7 h1 Heading

2.7.1 h2 Heading

h3 Heading

h4 Heading

h5 Heading  h6 Heading

HTML:

```html
<h1>h1 Heading</h1>
<h2>h2 Heading</h2>
<h3>h3 Heading</h3>
<h4>h4 Heading</h4>
<h5>h5 Heading</h5>
<h6>h6 Heading</h6>
```

2.7.2 Horizontal Rules

The HTML `<hr>` element is for creating a “thematic break” between paragraph-level elements. In markdown, you can create a `<hr>` with any of the following:

- `___`: three consecutive underscores
- `---`: three consecutive dashes
- `***`: three consecutive asterisks

renders to:

```

```

2.7.3 Body Copy

Body copy written as normal, plain text will be wrapped with `<p>` tags in the rendered HTML.

So this body copy:

```
markdown Lorem ipsum dolor sit amet, graecis denique ei vel, at duo primis mandamus. Et legere occurreret pri, animal tacies compectitur ad cum. Cu eum inermis inimicus efficiendi. Labore officiis his ex, soluta officiis concludaturque ei qui, vide sensibus vim ad. renders to this HTML:
```

```
<p>Lorem ipsum dolor sit amet, graecis denique ei vel, at duo primis mandamus. Et legere occurreret pri, animal tacies compectitur ad cum. Cu eum inermis inimicus efficiendi. Labore officiis his ex, soluta officiis concludaturque ei qui, vide sensibus vim ad.</p>
```

```
2.7.4 Emphasis

Bold

For emphasizing a snippet of text with a heavier font-weight. The following snippet of text is rendered as bold text.

markdown **rendered as bold text** renders to:

rendered as bold text

and this HTML

<strong>rendered as bold text</strong>

Italics

For emphasizing a snippet of text with italics. The following snippet of text is rendered as italicized text.

_*rendered as italicized text_*

renders to:

rendered as italicized text

and this HTML:

<em>rendered as italicized text</em>

strikethrough

In GFM you can do strikethroughs.

markdown ~~Strike through this text.~~ Which renders to:

[STRIKEOUT:Strike through this text.]

2.7.5 Blockquotes

For quoting blocks of content from another source within your document. Add > before any text you want to quote.

Add `>` before any text you want to quote.

Renders to:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

and this HTML:

<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
</blockquote>

Blockquotes can also be nested:
> Donec massa lacus, ultricies a ullamcorper in, fermentum sed augue. Nunc augue augue, aliquam non hendrerit ac, commodo vel nisi. 
>> Sed adipiscing elit vitae augue consectetur a gravida nunc vehicula. Donec auctor odio non est accumsan facilisis. Aliquam id turpis in dolor tincidunt mollis ac eu diam. 
>>> Donec massa lacus, ultricies a ullamcorper in, fermentum sed augue. Nunc augue augue, aliquam non hendrerit ac, commodo vel nisi.

Renders to:


### 2.7.6 Lists

**Unordered**

A list of items in which the order of the items does not explicitly matter.

You may use any of the following symbols to denote bullets for each list item:

- valid bullet
- valid bullet
- valid bullet

For example

markdown + Lorem ipsum dolor sit amet + Consectetur adipiscing elit + Integer molestie lorem at massa + Facilisis in pretium nisl aliquet + Nulla volutpat aliquam velit - Phasellus iaculis neque - Purus sodales ultricies - Vestibulum laoreet porttitor sem - Ac tristique libero volutpat at + Faucibus porta lacus fringilla vel + Aenean sit amet erat nunc + Eget porttitor lorem

Renders to:

- Lorem ipsum dolor sit amet
- Consectetur adipiscing elit
- Integer molestie lorem at massa
- Facilisis in pretium nisl aliquet
- Nulla volutpat aliquam velit
- Phasellus iaculis neque
- Purus sodales ultricies
- Vestibulum laoreet porttitor sem
- Ac tristique libero volutpat at
- Faucibus porta lacus fringilla vel
- Aenean sit amet erat nunc
- Eget porttitor lorem

And this HTML
Ordered

A list of items in which the order of items does explicitly matter.

markdown
1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. Facilisis in pretium nisl aliquet
5. Nulla volutpat aliquam velit
6. Faucibus porta lacus fringilla vel
7. Aenean sit amet erat nunc
8. Eget porttitor lorem Renders to:

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. Facilisis in pretium nisl aliquet
5. Nulla volutpat aliquam velit
6. Faucibus porta lacus fringilla vel
7. Aenean sit amet erat nunc
8. Eget porttitor lorem

And this HTML:

```html
<ol>
  <li>Lorem ipsum dolor sit amet</li>
  <li>Consectetur adipiscing elit</li>
  <li>Integer molestie lorem at massa</li>
  <li>Facilisis in pretium nisl aliquet</li>
  <li>Nulla volutpat aliquam velit</li>
  <li>Faucibus porta lacus fringilla vel</li>
  <li>Aenean sit amet erat nunc</li>
  <li>Eget porttitor lorem</li>
</ol>
```

**TIP:** If you just use 1. for each number, GitHub will automatically number each item. For example:

```
1. Lorem ipsum dolor sit amet
1. Consectetur adipiscing elit
1. Integer molestie lorem at massa
```
1. Facilisis in pretium nisl aliquet
1. Nulla volutpat aliquam velit
1. Faucibus porta lacus fringilla vel
1. Aenean sit amet erat nunc
1. Eget porttitor lorem

Renders to:

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. Facilisis in pretium nisl aliquet
5. Nulla volutpat aliquam velit
6. Faucibus porta lacus fringilla vel
7. Aenean sit amet erat nunc
8. Eget porttitor lorem

2.7.7 Code

**Inline code**

Wrap inline snippets of code with `. For example, `<section></section>` should be wrapped as “inline”.

For example, `LOCITY>` should be wrapped as "inline".

**Indented code**

Or indent several lines of code by at least four spaces, as in:

```verbatim
// Some comments
line 1 of code
line 2 of code
line 3 of code
```

**Block code “fences”**

Use “fences” ` ``` to block in multiple lines of code.

Sample text here...

**HTML:**

```html
<pre>
  <p>Sample text here...</p>
</pre>
```
Syntax highlighting

GFM, or “GitHub Flavored Markdown” also supports syntax highlighting. To activate it, simply add the file extension of the language you want to use directly after the first code “fence”, ``, and syntax highlighting will automatically be applied in the rendered HTML. For example, to apply syntax highlighting to JavaScript code:

Renders to:

```javascript
grunt.initConfig({
  assemble: {
    options: {
      assets: 'docs/assets',
      data: 'src/data/*.{json,yml}',
      helpers: 'src/custom-helpers.js',
      partials: ['src/partials/**/*.{hbs,md}']
    },
    pages: {
      options: {
        layout: 'default.hbs'
      },
      files: {
        './': ['src/templates/pages/index.hbs']
      }
    }
  }
});
```

And this complicated HTML:

```html
<div class="highlight"><pre>
grunt.initConfig({
  assemble: {
    options: {
      assets: 'docs/assets',
      data: 'src/data/*.{json,yml}',
      helpers: 'src/custom-helpers.js',
      partials: ['src/partials/**/*.{hbs,md}']
    },
    pages: {
      options: {
        layout: 'default.hbs'
      },
      files: {
        './': ['src/templates/pages/index.hbs']
      }
    }
  }
});
</pre></div>
```

2.7.8 Tables

Tables are created by adding pipes as dividers between each cell, and by adding a line of dashes (also separated by bars) beneath the header. Note that the pipes do not need to be vertically aligned.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>path to data files to supply the data that will be passed into templates.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------:</td>
<td>-----------:</td>
</tr>
<tr>
<td>data</td>
<td>path to data files to supply the data that will be passed into templates.</td>
</tr>
<tr>
<td>engine</td>
<td>engine to be used for processing templates. Handlebars is the default.</td>
</tr>
<tr>
<td>ext</td>
<td>extension to be used for dest files.</td>
</tr>
</tbody>
</table>

2.7.9 Links

Basic link

[Assemble](http://assemble.io)

 Renders to (hover over the link, there is no tooltip): Assemble

 HTML:
Add a title

[Upstage](https://github.com/upstage/ "Visit Upstage!")

Renders to (hover over the link, there should be a tooltip):
Upstage

HTML:

<a href="https://github.com/upstage/" title="Visit Upstage!">Upstage</a>

Named Anchors

Named anchors enable you to jump to the specified anchor point on the same page. For example, each of these chapters:

markdown # Table of Contents
* [Chapter 1](#chapter-1) * [Chapter 2](#chapter-2) * [Chapter 3](#chapter-3) will jump to these sections:

```markdown
## Chapter 1 Content for chapter one.

### 2.7.10 Chapter 2

Content for chapter one.

### 2.7.11 Chapter 3

Content for chapter one. ***NOTE*** that specific placement of the anchor tag seems to be arbitrary. They are placed inline here since it seems to be unobtrusive, and it works.

### 2.7.12 Images

Images have a similar syntax to links but include a preceding exclamation point.

markdown ![Minion](http://octodex.github.com/images/minion.png)

or

markdown ![Alt text](http://octodex.github.com/images/stormtroopocat.jpg "The Stormtroopocat") ![Alt text]

Like links, Images also have a footnote style syntax:

markdown ![Alt text][id]

With a reference later in the document defining the URL location:

[id]: http://octodex.github.com/images/dojocat.jpg "The Dojocat"

2.7. h1 Heading

2.8 Vim Commands Cheat Sheet

For the orignal please go here.

2.8.1 How to Exit

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:q[uit]</td>
<td>Quit Vim. This fails when changes have been made.</td>
</tr>
<tr>
<td>:q[uit]!</td>
<td>Quit without writing.</td>
</tr>
<tr>
<td>:wq</td>
<td>Write the current file and exit.</td>
</tr>
<tr>
<td>:wq!</td>
<td>Write the current file and exit always.</td>
</tr>
<tr>
<td>:wq {file}</td>
<td>Write to {file}. Exit if not editing the last</td>
</tr>
<tr>
<td>:wq! {file}</td>
<td>Write to {file} and exit always.</td>
</tr>
<tr>
<td>:[range]wq[!]</td>
<td>Write to {file} Same as above, but only write the lines in [range].</td>
</tr>
<tr>
<td>ZZ</td>
<td>Write current file, if modified, and exit.</td>
</tr>
<tr>
<td>ZQ</td>
<td>Quit current file and exit (same as &quot;:q!&quot;).</td>
</tr>
</tbody>
</table>

2.8.2 Editing a File

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e[dit]</td>
<td>Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim.</td>
</tr>
<tr>
<td>:e[dit]!</td>
<td>Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.</td>
</tr>
<tr>
<td>:e[dit] {file}</td>
<td>Edit {file}.</td>
</tr>
<tr>
<td>:e[dit]! {file}</td>
<td>Edit {file} always. Discard any changes to the current buffer.</td>
</tr>
<tr>
<td>gf</td>
<td>Edit the file whose name is under or after the cursor. Mnemonic: “goto file”.</td>
</tr>
</tbody>
</table>

2.8.3 Inserting Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Append text after the cursor [count] times.</td>
</tr>
<tr>
<td>A</td>
<td>Append text at the end of the line [count] times.</td>
</tr>
<tr>
<td>i</td>
<td>Insert text before the cursor [count] times.</td>
</tr>
<tr>
<td>I</td>
<td>Insert text before the first non-blank in the line [count] times.</td>
</tr>
<tr>
<td>gl</td>
<td>Insert text in column 1 [count] times.</td>
</tr>
<tr>
<td>o</td>
<td>Begin a new line below the cursor and insert text, repeat [count] times.</td>
</tr>
<tr>
<td>O</td>
<td>Begin a new line above the cursor and insert text, repeat [count] times.</td>
</tr>
</tbody>
</table>

2.8.4 Inserting a file

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:r[ead] {name}</td>
<td>Insert the file {name} below the cursor.</td>
</tr>
<tr>
<td>:r[ead] ![cmd]</td>
<td>Execute {cmd} and insert its standard output below the cursor.</td>
</tr>
</tbody>
</table>
2.8.5 Deleting Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Del&gt;</code> or <code>x</code></td>
<td>Delete [count] characters under and after the cursor</td>
</tr>
<tr>
<td><code>X</code></td>
<td>Delete [count] characters before the cursor</td>
</tr>
<tr>
<td><code>d[motion]</code></td>
<td>Delete text that <code>motion</code> moves over</td>
</tr>
<tr>
<td><code>dd</code></td>
<td>Delete [count] lines</td>
</tr>
<tr>
<td><code>D</code></td>
<td>Delete the characters under the cursor until the end of the line</td>
</tr>
<tr>
<td><code>{Visual}x</code> or <code>{Visual}d</code></td>
<td>Delete the highlighted text (for <code>{Visual}</code> see Selecting Text).</td>
</tr>
<tr>
<td><code>{Visual}CTRL-H</code> or <code>{Visual}</code></td>
<td>When in Select mode: Delete the highlighted text</td>
</tr>
<tr>
<td><code>{Visual}X</code> or <code>{Visual}D</code></td>
<td>Delete the highlighted lines</td>
</tr>
<tr>
<td><code>:range]d[delete</code></td>
<td>Delete <code>range</code> lines (default: current line)</td>
</tr>
<tr>
<td><code>:range]d[delete] [count]</code></td>
<td>Delete <code>count</code> lines, starting with <code>range</code></td>
</tr>
</tbody>
</table>

2.8.6 Changing (or Replacing) Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r[char]</code></td>
<td>replace the character under the cursor with <code>char</code>.</td>
</tr>
<tr>
<td><code>R</code></td>
<td>Enter Insert mode, replacing characters rather than inserting</td>
</tr>
<tr>
<td><code>~</code></td>
<td>Switch case of the character under the cursor and move the cursor to the right. If a [count] is given, do that many characters.</td>
</tr>
<tr>
<td><code>~{motion}</code></td>
<td>switch case of <code>{motion}</code> text.</td>
</tr>
<tr>
<td><code>{Visual}~</code></td>
<td>Switch case of highlighted text</td>
</tr>
</tbody>
</table>

2.8.7 Substituting

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>:range]s[substitute]/[pattern]/[string]/[c][e][g][r][i][I] [count]</code></td>
<td>in [range] replace a match of [pattern] with [string].</td>
</tr>
<tr>
<td><code>:range]s[substitute] [c][e][g][r][i][I] [count]</code></td>
<td>Repeat last :substitute with same search pattern and substitute string, but without the same flags. You may add extra flags</td>
</tr>
</tbody>
</table>

The arguments that you can use for the substitute commands:
[c] Confirm each substitution. Vim positions the cursor on the matching string. You can type:
'y' to substitute this match
'n' to skip this match
to skip this match
'a' to substitute this and all remaining matches (not in Vi)
'q' to quit substituting (not in Vi)
CTRL-E to scroll the screen up (not in Vi)
CTRL-Y to scroll the screen down (not in Vi).
[e] When the search pattern fails, do not issue an error message and, in particular, continue in maps as if no error occurred.
[g] Replace all occurrences in the line. Without this argument, replacement occurs only for the first occurrence in each line.
2.8.8 Copying and Moving Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“(a-zA-Z0-9.%#:-”)</td>
<td>Use register {a-zA-Z0-9.%.#:-“} for next delete, yank or put (use uppercase character to append with delete and yank) ((.%#:) only work with put).</td>
</tr>
<tr>
<td>:registers</td>
<td>Display the contents of all numbered and named registers.</td>
</tr>
<tr>
<td>:registers {arg}</td>
<td>Display the contents of the numbered and named registers that are mentioned in {arg}.</td>
</tr>
<tr>
<td>:display {arg}</td>
<td>Same as :registers.</td>
</tr>
<tr>
<td>[&quot;x&quot;]y{motion}</td>
<td>Yank {motion} text [into register x].</td>
</tr>
<tr>
<td>[&quot;x&quot;]yy</td>
<td>Yank [count] lines [into register x].</td>
</tr>
<tr>
<td>[&quot;x&quot;]Y</td>
<td>Yank [count] lines [into register x] (synonym for yy).</td>
</tr>
<tr>
<td>[Visual]&quot;x&quot;y</td>
<td>Yank the highlighted text [into register x] (for {Visual} see Selecting Text).</td>
</tr>
<tr>
<td>[Visual]&quot;x&quot;Y</td>
<td>Yank the highlighted lines [into register x].</td>
</tr>
<tr>
<td>[:range]y[ank] {x}</td>
<td>Yank [range] lines [into register x].</td>
</tr>
<tr>
<td>[:range]y[ank] {x} {count}</td>
<td>Yank {count} lines, starting with last line number in [range] (default: current line), [into register x].</td>
</tr>
<tr>
<td>[&quot;x&quot;]p</td>
<td>Put the text [from register x] after the cursor [count] times.</td>
</tr>
<tr>
<td>[&quot;x&quot;]P</td>
<td>Put the text [from register x] before the cursor [count] times.</td>
</tr>
<tr>
<td>[&quot;x&quot;]gp</td>
<td>Just like “p”, but leave the cursor just after the new text.</td>
</tr>
<tr>
<td>[&quot;x&quot;]gP</td>
<td>Just like “P”, but leave the cursor just after the new text.</td>
</tr>
<tr>
<td>[:line]put[t] {x}</td>
<td>Put the text [from register x] after [line] (default current line).</td>
</tr>
<tr>
<td>[:line]put[t]! {x}</td>
<td>Put the text [from register x] before [line] (default current line).</td>
</tr>
</tbody>
</table>

2.8.9 Undo/Redo/Repeat

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>Undo [count] changes.</td>
</tr>
<tr>
<td>:undo</td>
<td>Undo one change.</td>
</tr>
<tr>
<td>CTRL-R</td>
<td>Redo [count] changes which were undone.</td>
</tr>
<tr>
<td>:redo</td>
<td>Redo one change which was undone.</td>
</tr>
<tr>
<td>U</td>
<td>Undo all latest changes on one line. {Vi: while not moved off of it}</td>
</tr>
<tr>
<td>.</td>
<td>Repeat last change, with count replaced with [count].</td>
</tr>
</tbody>
</table>

2.8.10 Moving Around

Basic motion commands:

```
  k
  h   l
  j
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h or</td>
<td>[count] characters to the left (exclusive).</td>
</tr>
<tr>
<td>l or</td>
<td>[count] characters to the right (exclusive).</td>
</tr>
<tr>
<td>k or CTRL-P</td>
<td>[count] lines upward</td>
</tr>
<tr>
<td>j or CTRL-J or CTRL-N</td>
<td>[count] lines downward (linewise).</td>
</tr>
<tr>
<td>0</td>
<td>To the first character of the line (exclusive).</td>
</tr>
<tr>
<td>&lt;Home&gt;</td>
<td>To the first character of the line (exclusive).</td>
</tr>
<tr>
<td>^</td>
<td>To the first non-blank character of the line</td>
</tr>
<tr>
<td>$ or &lt;End&gt;</td>
<td>To the end of the line and [count - 1] lines downward</td>
</tr>
<tr>
<td>g0 or g&lt;Home&gt;</td>
<td>When lines wrap (‘wrap on’): To the first character of the screen line (exclusive). Differs from “0” when a line is wider than the screen. When lines don’t wrap (‘wrap off’): To the leftmost character of the current line that is on the screen. Differs from “0” when the first character of the line is not on the screen.</td>
</tr>
<tr>
<td>g^</td>
<td>When lines wrap (‘wrap’ on): To the first non-blank character of the screen line (exclusive). Differs from “^” when a line is wider than the screen. When lines don’t wrap (‘wrap’ off): To the leftmost non-blank character of the current line that is on the screen. Differs from “^” when the first non-blank character of the line is not on the screen.</td>
</tr>
<tr>
<td>g$ or g&lt;End&amp;gr;</td>
<td>When lines wrap (‘wrap’ on): To the last character of the screen line and [count - 1] screen lines downward (inclusive). Differs from “$” when a line is wider than the screen. When lines don’t wrap (‘wrap’ off): To the rightmost character of the current line that is visible on the screen. Differs from “$” when the last character of the line is not on the screen or when a count is used.</td>
</tr>
<tr>
<td>f{char}</td>
<td>To [count]’th occurrence of {char} to the right. The cursor is placed on {char} (inclusive).</td>
</tr>
<tr>
<td>F{char}</td>
<td>To the [count]’th occurrence of {char} to the left. The cursor is placed on {char} (inclusive).</td>
</tr>
<tr>
<td>t{char}</td>
<td>Till before [count]’th occurrence of {char} to the right. The cursor is placed on the character left of {char} (inclusive).</td>
</tr>
<tr>
<td>T{char}</td>
<td>Till after [count]’th occurrence of {char} to the left. The cursor is placed on the character right of {char} (inclusive).</td>
</tr>
<tr>
<td>;</td>
<td>Repeat latest f, t, F or T [count] times.</td>
</tr>
<tr>
<td>,</td>
<td>Repeat latest f, t, F or T in opposite direction [count] times.</td>
</tr>
<tr>
<td>- &lt;minus&gt;</td>
<td>[count] lines upward, on the first non-blank character (linewise).</td>
</tr>
<tr>
<td>+ or CTRL-M or &lt;CR&gt;</td>
<td>[count] lines downward, on the first non-blank character (linewise).</td>
</tr>
<tr>
<td>_ &lt;underscore&gt;</td>
<td>[count] - 1 lines downward, on the first non-blank character (linewise).</td>
</tr>
<tr>
<td>&lt;C-End&gt; or G</td>
<td>Goto line [count], default last line, on the first non-blank character.</td>
</tr>
<tr>
<td>&lt;C-Home&gt; or gg</td>
<td>Goto line [count], default first line, on the first non-blank character.</td>
</tr>
</tbody>
</table>

2.8. Vim Commands Cheat Sheet
These commands move over words or WORDS.

A word consists of a sequence of letters, digits and underscores, or a sequence of other non-blank characters, separated with white space (spaces, tabs, ). This can be changed with the ‘iskeyword’ option.

A WORD consists of a sequence of non-blank characters, separated with white space. An empty line is also considered to be a word and a WORD.

| ( | [count] sentences backward |
| ) | [count] sentences forward |
| { | [count] paragraphs backward |
| } | [count] paragraphs forward |
| | [count] sections forward or to the next ‘{ ’ in the first column. When used after an operator, then the ‘ } ’ in the first column. |
| } | [count] sections forward or to the next ‘}’ in the first column |
| [ | [count] sections backward or to the previous ‘{ ’ in the first column |
| ] | [count] sections backward or to the previous ‘}’ in the first column |

Screen movement commands

| 26. | Center the screen on the cursor |
| zt | Scroll the screen so the cursor is at the top |
| zb | Scroll the screen so the cursor is at the bottom |

### 2.8.11 Marks

| m{a-zA-Z} | Set mark {a-zA-Z} at cursor position (does not move the cursor, this is not a motion command). |
| m’ or m’ | Set the previous context mark. This can be jumped to with the “’” or “’” command (does not move the cursor, this is not a motion command). |
| :[range]ma[rk] {a-zA-Z} | Set mark {a-zA-Z} at last line number in [range], column 0. Default is cursor line. |
| :[range]k{a-zA-Z} | Same as :mark, but the space before the mark name can be omitted. |
| ‘{a-z} | To the first non-blank character on the line with mark {a-z} (linewise). |
| ‘{A-Z0-9} | To the first non-blank character on the line with mark {A-Z0-9} in the correct file |
| ‘{a-z} | To the mark {a-z} |
| ‘{A-Z0-9} | To the mark {A-Z0-9} in the correct file |
| :marks | List all the current marks (not a motion command). |
| :marks {arg} | List the marks that are mentioned in {arg} (not a motion command). For example: |
### 2.8.12 Searching

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/{pattern}/</td>
<td>Search forward for the count'th occurrence of {pattern}</td>
</tr>
<tr>
<td>/{pattern}/{offset}</td>
<td>Search forward for the count'th occurrence of {pattern} and go {offset} lines up or down.</td>
</tr>
<tr>
<td>/&lt;CR&gt;</td>
<td>Search forward for the count' th latest used pattern</td>
</tr>
<tr>
<td>//{offset}&lt;CR&gt;</td>
<td>Search forward for the count' th latest used pattern with new. If {offset} is empty no offset is used.</td>
</tr>
<tr>
<td>?{pattern}?{offset}&lt;CR&gt;</td>
<td>Search backward for the count' th previous occurrence of {pattern} and go {offset} lines up or down.</td>
</tr>
<tr>
<td>?&lt;CR&gt;</td>
<td>Search backward for the count' th latest used pattern</td>
</tr>
<tr>
<td>??{offset}&lt;CR&gt;</td>
<td>Search backward for the count' th latest used pattern with new {offset}. If {offset} is empty no offset is used.</td>
</tr>
<tr>
<td>n</td>
<td>Repeat the latest “/” or “?” count times.</td>
</tr>
<tr>
<td>N</td>
<td>Repeat the latest “/” or “?” count times in opposite direction.</td>
</tr>
</tbody>
</table>

### 2.8.13 Selecting Text (Visual Mode)

To select text, enter visual mode with one of the commands below, and use motion commands to highlight the text you are interested in. Then, use some command on the text.

The operators that can be used are:
- switch case
- d delete
- c change
- y yank
- > shift right
- < shift left
- ! filter through external command
- = filter through 'equalprg' option command
- gq format lines to 'textwidth' length

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>start Visual mode per character.</td>
</tr>
<tr>
<td>V</td>
<td>start Visual mode linewise.</td>
</tr>
<tr>
<td>&lt;Esc&gt;</td>
<td>exit Visual mode without making any changes</td>
</tr>
</tbody>
</table>

### 2.8.14 How to Suspend

CTRL-Z  Suspend Vim, like “:stop”. Works in Normal and in Visual mode. In Insert and Command-line mode, the CTRL-Z is inserted as a normal character.

:suspend[]! or :stop[]!  Suspend Vim. If the ‘!’ is not given and ‘autowrite’ is set, every buffer with changes and a file name is written out. If the ‘!’ is given or ‘autowrite’ is not set, changed buffers are not written, don’t forget to bring Vim back to the foreground later!
2.9 tmux shortcuts & cheatsheet

start new:
```bash
tmux
```

start new with session name:
```bash
tmux new -s myname
```

attach:
```bash
tmux a # (or at, or attach)
```

attach to named:
```bash
tmux a -t myname
```

list sessions:
```bash
tmux ls
```

kill session:
```bash
tmux kill-session -t myname
```

Kill all the tmux sessions:
```bash
tmux ls | grep : | cut -d. -f1 | awk '{print substr($1, 0, length($1)-1)}' | xargs kill
```

In tmux, hit the prefix `ctrl+b` (my modified prefix is `ctrl+a`) and then:

2.9.1 Sessions

```bash
:new<CR> new session
s list sessions
$ name session
```

2.9.2 Windows (tabs)

```bash
c create window
w list windows
n next window
p previous window
f find window
, name window
& kill window
```

2.9.3 Panes (splits)

```bash
% vertical split
" horizontal split
o swap panes
q show pane numbers
```
2.9.4 Sync Panes

You can do this by switching to the appropriate window, typing your Tmux prefix (commonly Ctrl-B or Ctrl-A) and then a colon to bring up a Tmux command line, and typing:

:setw synchronize-panes

You can optionally add on or off to specify which state you want; otherwise the option is simply toggled. This option is specific to one window, so it won’t change the way your other sessions or windows operate. When you’re done, toggle it off again by repeating the command. tip source

2.9.5 Resizing Panes

You can also resize panes if you don’t like the layout defaults. I personally rarely need to do this, though it’s handy to know how. Here is the basic syntax to resize panes:

```
PREFIX : resize-pane -D (Resizes the current pane down)
PREFIX : resize-pane -U (Resizes the current pane upward)
PREFIX : resize-pane -L (Resizes the current pane left)
PREFIX : resize-pane -R (Resizes the current pane right)
PREFIX : resize-pane -D 20 (Resizes the current pane down by 20 cells)
PREFIX : resize-pane -U 20 (Resizes the current pane upward by 20 cells)
PREFIX : resize-pane -L 20 (Resizes the current pane left by 20 cells)
PREFIX : resize-pane -R 20 (Resizes the current pane right by 20 cells)
PREFIX : resize-pane -t 2 20 (Resizes the pane with the id of 2 down by 20 cells)
PREFIX : resize-pane -t -L 20 (Resizes the pane with the id of 2 left by 20 cells)
```

2.9.6 Copy mode:

Pressing PREFIX [ places us in Copy mode. We can then use our movement keys to move our cursor around the screen. By default, the arrow keys work. we set our configuration file to use Vim keys for moving between windows and resizing panes so we wouldn’t have to take our hands off the home row. tmux has a vi mode for working with the buffer as well. To enable it, add this line to .tmux.conf:

```
setw -g mode-keys vi
```

With this option set, we can use h, j, k, and l to move around our buffer.

To get out of Copy mode, we just press the ENTER key. Moving around one character at a time isn’t very efficient. Since we enabled vi mode, we can also use some other visible shortcuts to move around the buffer.

For example, we can use “w” to jump to the next word and “b” to jump back one word. And we can use “f”, followed by any character, to jump to that character on the same line, and “F” to jump backwards on the line.
### 2.9.7 Misc

d  detach
t  big clock
?  list shortcuts
:  prompt

### 2.9.8 Configurations Options:

# Mouse support - set to on if you want to use the mouse
* setw -g mode-mouse off
* set -g mouse-select-pane off
* set -g mouse-resize-pane off
* set -g mouse-select-window off

# Set the default terminal mode to 256color mode
set -g default-terminal "screen-256color"

# enable activity alerts
setw -g monitor-activity on
set -g visual-activity on

# Center the window list
set -g status-justify centre

# Maximize and restore a pane
unbind Up bind Up new-window -d -n tmp \; swap-pane -s tmp.1 \; select-window -t tmp
unbind Down
bind Down last-window \; swap-pane -s tmp.1 \; kill-window -t tmp

2.9.9 Resources:

- tmux: Productive Mouse-Free Development
- How to reorder windows

2.9.10 Thanks

Thanks to the original creator of this cheatsheet.
3.1 Introduction

Python library for tools to be used at the Coherent Soft X-ray scattering beamline at NSLS-II, (CSX, 23-ID) To chat to the CSX team:

3.2 Contents

3.2.1 Image Processing Routines
API Reference

3.2.2 FastCCD Routines
API Reference

3.2.3 Utilities
API Reference

3.2.4 IPython Notebook Routines
API Reference
EXAMPLE NOTEBOOKS

Contents:

4.1 iPython Notebook To Export CSX Data as a suitcase

```python
from dataportal import DataBroker as db
import os
import sys
from suitcase import export
from time import sleep
from __future__ import print_function

FILENAME = 'scan-'  # no file extension; we take care of that below
SCAN_ID = 52326
START = SCAN_ID
STOP = 52329
OVERWRITE = True  # If true, I will delete the file FILENAME if it exists.

# sanity check that we can see the GPFS. If this fails, give up and email Rob Petkus. There is no hope.
!ls '/GPFS/

xf23id

h = db[SCAN_ID]

for SCAN_ID in range(START,STOP+1):
    h = db[SCAN_ID]
    SCANNUMBER=str(SCAN_ID)
    fn = '{filename}.h5'.format(filename=FILENAME+SCANNUMBER)
    if os.path.isfile(fn):
        if OVERWRITE:
            os.remove(fn)
        else:
            raise Exception("That file already exists. Set OVERWRITE = True if you want to overwrite.
    export([h], fn)
```
Download the CSX Documentation as a PDF