NSLS-II CSX Beamline Docs Documentation

Release 0.1

Stuart B. Wilkins

November 30, 2015

CONTENTS

CHAPTER

ONE

INTRODUCTION

These pages are the doccumentation of the CSX beamlines (23-ID-1 and 23-ID-2) at the NSLS-II.

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

CHAPTER

TWO

CONTENTS

2.1 CSX-1 (23-ID-1) Beamline Doccumentation

Contents:

2.1.1 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 chanel is shown in the *LBNL fCRIC Circuit Diagram*.

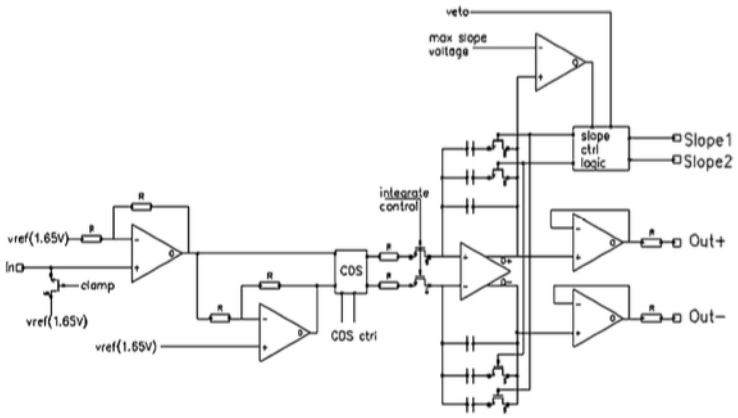


Fig. 2.1: LBNL fCRIC Circuit Diagram

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 2.1: 16 Bit fCRIC Data Format

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
G1	G0	ERR	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00

Table 2.2: fCRIC Gain Setting

G1	G0	Gain	Pre-factor
0	0	x8	x1
1	0	x2	x4
1	1	x1	x8

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:

$$A_{\mathrm{corr}} = G(A_{\mathrm{meas}} - O)$$

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:

```
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)
    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

2.2 Indices and tables

- genindex
- modindex
- search

2.3 Guides and Tutorials

Contents:

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

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-tctrl1       | softioc         |  5010 | /epics/iocs/es-tctrl1/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)
@@@ Use ^X to kill the child, auto restart is ON, use ^T to toggle auto restart
@@@ procServ server PID: 10584
@@@ Server startup directory: /epics/iocs/mc12
@@@ Child startup directory: /epics/iocs/mc12
@@@ Child "mc12" started as: /epics/iocs/mc12/st.cmd
@@@ Child "mc12" PID: 28044
@@@ procServ server started at: Tue Oct 20 17:35:25 2015
@@@ Child "mc12" started at: Fri Nov 13 12:49:49 2015
@@@ 0 user(s) and 0 logger(s) connected (plus you)
```

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`

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
```

2.3.2 Controls Account Setup Guide

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.

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 restict access to files in your home directory then do the following:

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

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 users 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:
4a:dd:0a:c6:35:4e:3f:ed:27:38:8c:74:44:4d:93:67 demo@a
The key's randomart image is:
+--[ RSA 2048]----+
|          .oo.   |
|         .  o.E  |
|        + .  o   |
|     . = = .     |
|      = S = .    |
|     o + = +     |
|      . o + o .  |
|           . o   |
|                 |
+-----------------+
The public key is now located in /home/demo/.ssh/id_rsa.pub The private key (identification) is now
```

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@123.45.56.78
```

Alternatively, you can paste in the keys using SSH:

cat ~/.ssh/id_rsa.pub | ssh user@123.45.56.78 "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys" No matter which command you chose, you should see something like:

```
The authenticity of host '12.34.56.78 (12.34.56.78)' can't be established.
RSA key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '12.34.56.78' (RSA) to the list of known hosts.
user@12.34.56.78's password:
Now try logging into the machine, with "ssh 'user@12.34.56.78'", and check in:

  ~/.ssh/authorized_keys

  to make sure we haven't added extra keys that you weren't expecting.
  Now you can go ahead and log into user@12.34.56.78 and you will not be prompted for a password. How
```

2.3.3 Installing a Personal Conda Environment

Introduction

This guide covers installing the conda packaging system *miniconda* into your own space. This has a number of advantages, including full controll 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:

```
[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.

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.:

```
[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:

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

Miniconda3 will now be installed into this location:
/home/swilkins/miniconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
```

```
  - Or specify a different location below

[/home/swilkins/miniconda3] >>> /GPFS/xf23id/users/swilkins/miniconda3
PREFIX=/GPFS/xf23id/users/swilkins/miniconda3
```

To add the *miniconda* to yor path, edit your `.bashrc` file with your favorite editor and add the following lines.

```
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
```

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
```

Installing a Custom Notebook Kernel

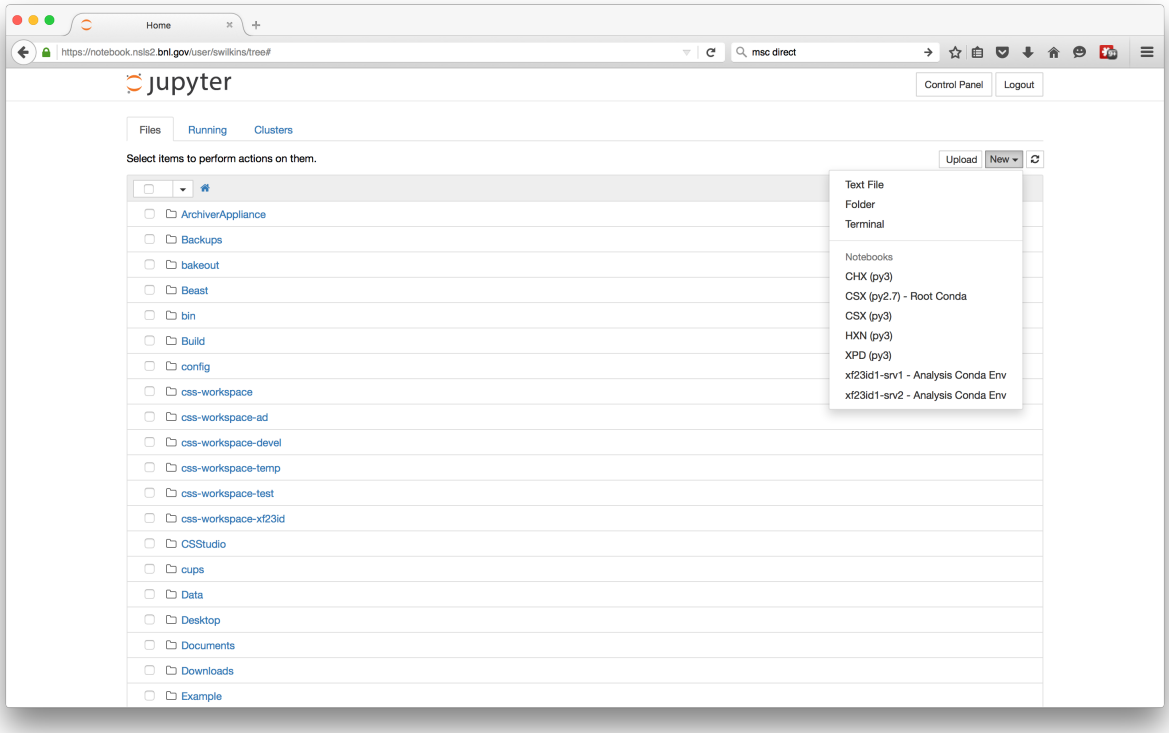
To get your custom environment to work with the notebook server, you have to create a kernal file in your `.python/kernel` directory (where *my-analysis-kernel* can be any name you wish to know this kernel by):

```
(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:

```
{"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"
}
```

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 shouls 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!!

2.3.4 Installing Linxbrew

Introduction

Linuxbrew 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 privilages it can be installed in a home directory for example.

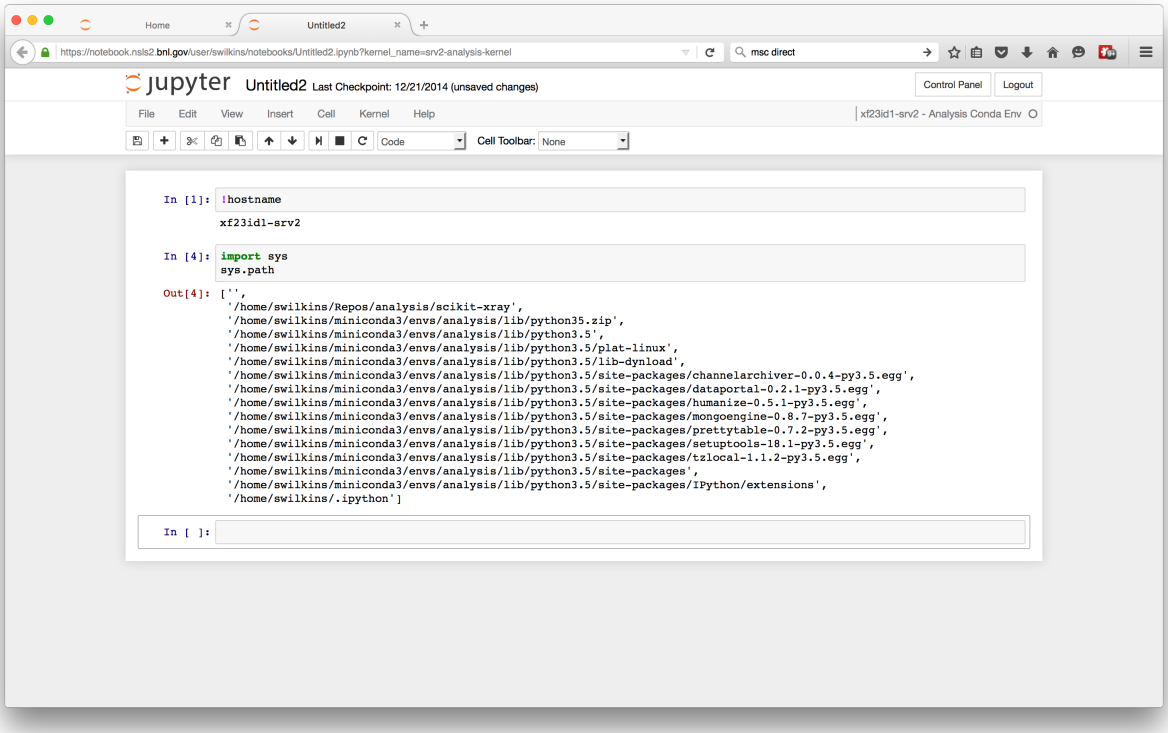
Installing

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
```

2.3.5 UNIX Primer and Account Setup

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
-rw-rw---- 1 swilkins csx 1.2M Mar  3  2014 csx.ppt
drwxrwx--- 2 swilkins csx    6 Nov 24 06:15 other
```

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 2.3: Octal file permissions for chmod and umask

Octal digit	`chmod` Permissions	`umask` Permissions
0	No permissions	Any permission may be set
1	Execute permission only	Setting of execute permission is prohibited
2	Write permission only	Setting of write permission is prohibited
3	Write and execute permissions	Setting of write and execute permission is prohibited
4	Read permission only	Setting of read permission is prohibited
5	Read and execute permissions	Setting of read and execute permission is prohibited
6	Read and write permissions	Setting of read and write permission is prohibited
7	Read, write and execute permissions	All permissions are prohibited from being set

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
[swilkins@xf23id1-srv1 ~/Presentations]$
```

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 0027      # allow user rwx group r-x and restrict other (-rwxr-x---)
$ umask 0007      # allow group and user rwx and restrict other (-rwxrwx---)
$ umask 0077      # 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
[swilkins@xf23id1-srv1 ~/Example]$ umask 0027
[swilkins@xf23id1-srv1 ~/Example]$ touch goodbye_world
[swilkins@xf23id1-srv1 ~/Example]$ ls -l
total 0
-rw-r----- 1 swilkins csx 0 Nov 24 06:55 goodbye_world
-rw-rw---- 1 swilkins csx 0 Nov 24 06:55 hello_world
[swilkins@xf23id1-srv1 ~/Example]$ umask 0023
```

```
[swilkins@xf23id1-srv1 ~/Example]$ touch hello_world_again
[swilkins@xf23id1-srv1 ~/Example]$ ls -l
total 0
-rw-r----- 1 swilkins csx 0 Nov 24 06:55 goodbye_world
-rw-rw---- 1 swilkins csx 0 Nov 24 06:55 hello_world
-rw-r--r-- 1 swilkins csx 0 Nov 24 06:58 hello_world_again
```

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`:

```
umask 0027
```

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:

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

2.3.6 Vim Commands Cheat Sheet

For the origional please go here.

How to Exit

:q[uit]	Quit Vim. This fails when changes have been made.
:q[uit]!	Quit without writing.
:cq[uit]	Quit always, without writing.
:wq	Write the current file and exit.
:wq!	Write the current file and exit always.
:wq {file}	Write to {file}. Exit if not editing the last
:wq! {file}	Write to {file} and exit always.
:[range]wq[!]	[file] Same as above, but only write the lines in [range].
ZZ	Write current file, if modified, and exit.
ZQ	Quit current file and exit (same as ":q!").

Editing a File

:e[dit]	Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim.
:e[dit]!	Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.
:e[dit] {file}	Edit {file}.
:e[dit]! {file}	Edit {file} always. Discard any changes to the current buffer.
gf	Edit the file whose name is under or after the cursor. Mnemonic: "goto file".

Inserting Text

a	Append text after the cursor [count] times.
A	Append text at the end of the line [count] times.
i	Insert text before the cursor [count] times.
I	Insert text before the first non-blank in the line [count] times.
gI	Insert text in column 1 [count] times.
o	Begin a new line below the cursor and insert text, repeat [count] times.
O	Begin a new line above the cursor and insert text, repeat [count] times.

Inserting a file

:r[ead] [name]	Insert the file [name] below the cursor.
:r[ead] !{cmd}	Execute {cmd} and insert its standard output below the cursor.

Deleting Text

<Del> or x	Delete [count] characters under and after the cursor
X	Delete [count] characters before the cursor
d{motion}	Delete text that {motion} moves over
dd	Delete [count] lines
D	Delete the characters under the cursor until the end of the line
{Visual}x or {Visual}d	Delete the highlighted text (for {Visual} see *Selecting Text*).
{Visual}CTRL-H or {Visual}	When in Select mode: Delete the highlighted text
{Visual}X or {Visual}D	Delete the highlighted lines
:[range]d[elete]	Delete [range] lines (default: current line)
:[range]d[elete] {count}	Delete {count} lines, starting with [range]

Changing (or Replacing) Text

r{char}	replace the character under the cursor with {char}.
R	Enter Insert mode, replacing characters rather than inserting
~	Switch case of the character under the cursor and move the cursor to the right. If a [count] is given, do that many characters.
~{motion}	switch case of {motion} text.
{Visual}~	Switch case of highlighted text

Substituting

:[range]s[ubstitute]/{pattern}/{string}/[c][e][g][r][i][I] [count]	For each line in [range] replace a match of {pattern} with {string}.
:[range]s[ubstitute] [c][e][g][r][i][I] [count] :[range]&[c][e][g][r][i][I] [count]	Repeat last :substitute with same search pattern and substitute string, but without the same flags. You may add extra flags

```
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.
[i]  Ignore case for the pattern.
[I]  Don't ignore case for the pattern.
[p]  Print the line containing the last substitute.
```

Copying and Moving Text

"{a-zA-Z0-9.%#:-"}	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).
:reg[isters]	Display the contents of all numbered and named registers.
:reg[isters] {arg}	Display the contents of the numbered and named registers that are mentioned in {arg}.
:di[splay] [arg]	Same as :registers.
["x]y{motion}	Yank {motion} text [into register x].
["x]yy	Yank [count] lines [into register x]
["x]Y	yank [count] lines [into register x] (synonym for yy).
{Visual}["x]y	Yank the highlighted text [into register x] (for {Visual} see *Selecting Text*).
{Visual}["x]Y	Yank the highlighted lines [into register x]
:[range]y[ank] [x]	Yank [range] lines [into register x].
:[range]y[ank] [x] {count}	Yank {count} lines, starting with last line number in [range] (default: current line), [into register x].
["x]p	Put the text [from register x] after the cursor [count] times.
["x]P	Put the text [from register x] before the cursor [count] times.
["x]gp	Just like "p", but leave the cursor just after the new text.
["x]gP	Just like "P", but leave the cursor just after the new text.
:[line]pu[t] [x]	Put the text [from register x] after [line] (default current line).
:[line]pu[t]! [x]	Put the text [from register x] before [line] (default current line).

Undo/Redo/Repeat

u	Undo [count] changes.
:u[ndo]	Undo one change.
CTRL-R	Redo [count] changes which were undone.
:red[o]	Redo one change which was undone.
U	Undo all latest changes on one line. {Vi: while not moved off of it}
.	Repeat last change, with count replaced with [count].

Moving Around

```
Basic motion commands:

        k
      h   l
        j
```

h or	[count] characters to the left (exclusive).
l or or	[count] characters to the right (exclusive).
k or or CTRL-P	[count] lines upward
j or or CTRL-J or or CTRL-N	[count] lines downward (linewise).
0	To the first character of the line (exclusive).
<Home>	To the first character of the line (exclusive).
^	To the first non-blank character of the line
$ or <End>	To the end of the line and [count - 1] lines downward
g0 or g<Home>	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.
g^	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.
g$ or g<End&gr;	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.
f{char}	To [count]'th occurrence of {char} to the right. The cursor is placed on {char} (inclusive).
F{char}	To the [count]'th occurrence of {char} to the left. The cursor is placed on {char} (inclusive).
t{char}	Till before [count]'th occurrence of {char} to the right. The cursor is placed on the character left of {char} (inclusive).
T{char}	Till after [count]'th occurrence of {char} to the left. The cursor is placed on the character right of {char} (inclusive).
;	Repeat latest f, t, F or T [count] times.
,	Repeat latest f, t, F or T in opposite direction [count] times.
- <minus>	[count] lines upward, on the first non-blank character (linewise).
+ or CTRL-M or <CR>	[count] lines downward, on the first non-blank character (linewise).
_ <underscore>	[count] - 1 lines downward, on the first non-blank character (linewise).
<C-End> or G	Goto line [count], default last line, on the first non-blank character.
<C-Home> or gg	Goto line [count], default first line, on the first non-blank character.
	[count] words forward.

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

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:

Searching

/{pattern}[/]	Search forward for the [count]'th occurrence of {pattern}
/{pattern}/{offset}	Search forward for the [count]'th occurrence of {pattern} and go {offset} lines up or down.
/<CR>	Search forward for the [count]'th latest used pattern
//{offset}<CR>	Search forward for the [count]'th latest used pattern with new. If {offset} is empty no offset is used.
?{pattern}[?]<CR>	Search backward for the [count]'th previous occurrence of {pattern}
?{pattern}?{offset}<CR>	Search backward for the [count]'th previous occurrence of {pattern} and go {offset} lines up or down
?<CR>	Search backward for the [count]'th latest used pattern
??{offset}<CR>	Search backward for the [count]'th latest used pattern with new {offset}. If {offset} is empty no offset is used.
n	Repeat the latest "/" or "?" [count] times.
N	Repeat the latest "/" or "?" [count] times in opposite direction.

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
```

v	start Visual mode per character.
V	start Visual mode linewise.
<Esc>	exit Visual mode without making any changes

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.
:sus[pend][!] or :st[op][!]	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!

2.3.7 tmux shortcuts & cheatsheet

start new:

```
tmux
```

start new with session name:

```
tmux new -s myname
```

attach:

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

attach to named:

```
tmux a -t myname
```

list sessions:

```
tmux ls
```

kill session:

```
tmux kill-session -t myname
```

Kill all the tmux sessions:

```
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:

Sessions

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

Windows (tabs)

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

Panes (splits)

```
%  vertical split
"  horizontal split

o        swap panes
q        show pane numbers
x        kill pane
+        break pane into window (e.g. to select text by mouse to copy)
-        restore pane from window
<space>  toggle between layouts
q        (Show pane numbers, when the numbers show up type the key to goto that pane)
```

```
{        (Move the current pane left)
}        (Move the current pane right)
z        toggle pane zoom
```

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

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)
```

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.

```
Function               vi            emacs
Back to indentation    ^             M-m
Clear selection        Escape        C-g
Copy selection         Enter         M-w
Cursor down            j             Down
Cursor left            h             Left
Cursor right           l             Right
```

```
Cursor to bottom line   L
Cursor to middle line   M              M-r
Cursor to top line      H              M-R
Cursor up               k              Up
Delete entire line      d              C-u
Delete to end of line   D              C-k
End of line             $              C-e
Goto line               :              g
Half page down          C-d            M-Down
Half page up            C-u            M-Up
Next page               C-f            Page down
Next word               w              M-f
Paste buffer            p              C-y
Previous page           C-b            Page up
Previous word           b              M-b
Quit mode               q              Escape
Scroll down             C-Down or J    C-Down
Scroll up               C-Up or K      C-Up
Search again            n              n
Search backward         ?              C-r
Search forward          /              C-s
Start of line           0              C-a
Start selection         Space          C-Space
Transpose chars                        C-t
```

Misc

```
d  detach
t  big clock
?  list shortcuts
:  prompt
```

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
```

Resources:

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

Thanks

Thanks to the origional creator of this cheatsheet.

2.4 Indices and tables

- genindex
- modindex
- search

CHAPTER

THREE

DOWNLOADS

Download the CSX Documentation as a PDF

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search