This page will document the process of installing Scientific Linux and installing the required components for the LZ Electronics Chain Test. The software standards adopted are on ElectronicsChainTestSoftware.

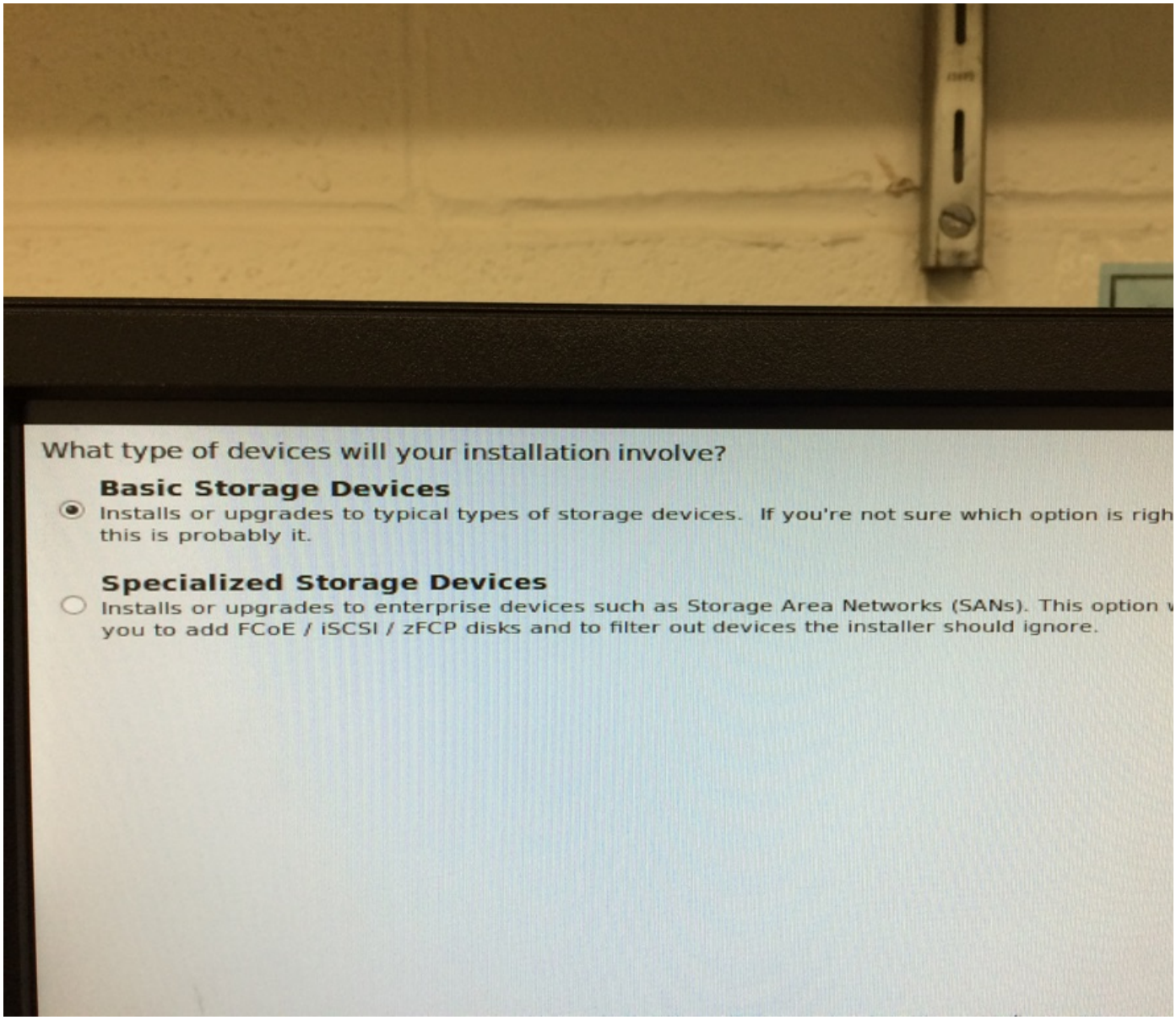
Installing Scientific Linux

1. "Install with basic video drivers"
Stuff happens; opens up the installer

2. There will be an option to "Check disk". I did this to ensure that the disk is ready for installation/no corruption. It will eject the disk after it is done. Replace the disk and it will find the installation media and it should proceed to the welcome screen automatically.

3. It will now ask for some simple language settings; Language -> English. Keyboard -> US English.

4. It will now ask for the storage system to be used. I chose the "Basic Storage System" as opposed to something Network or RAID based. This is shown in the image below

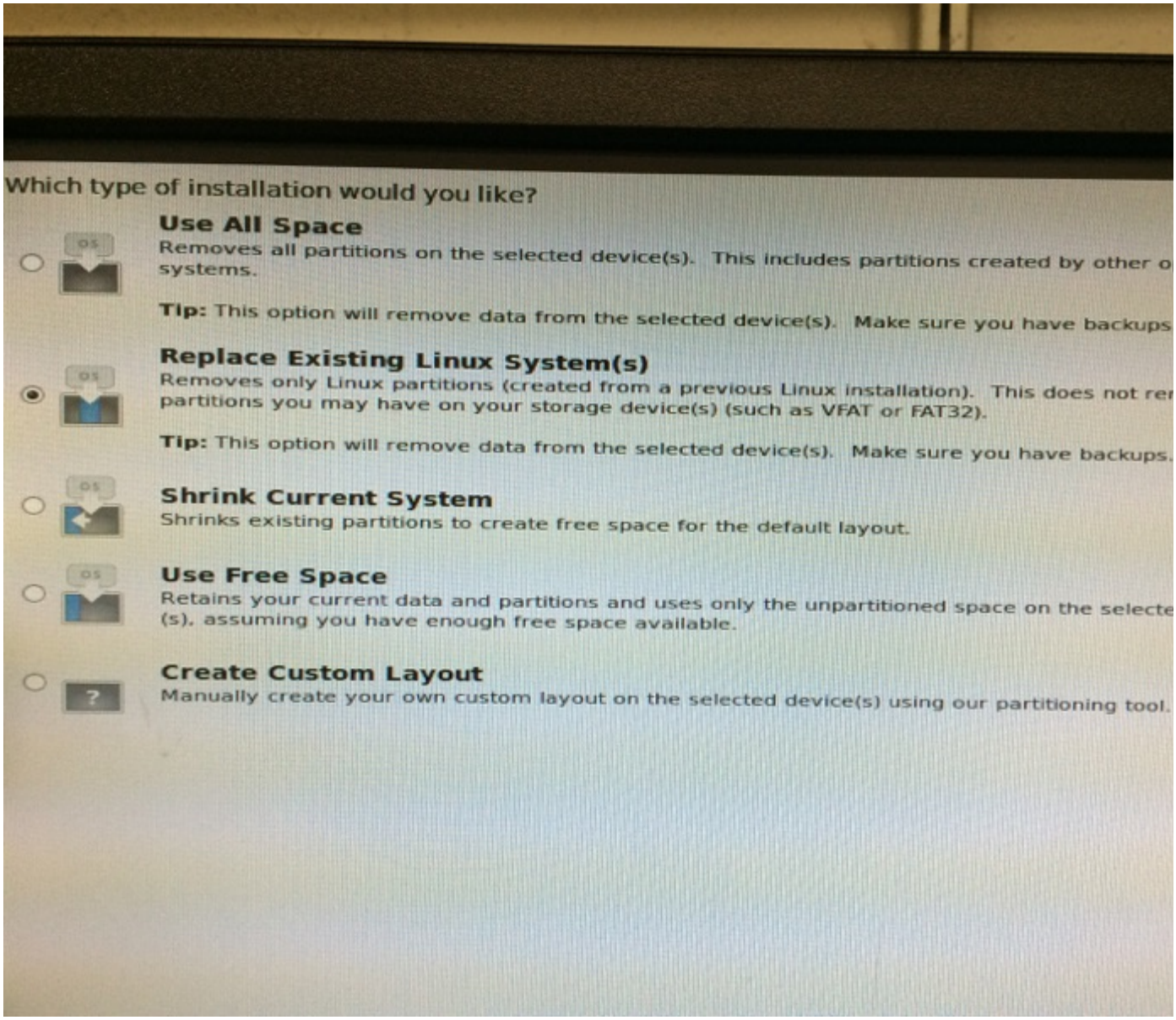


5. It will request a hostname, I used "ANSEL1".

6. Choose timezone. America/New York.

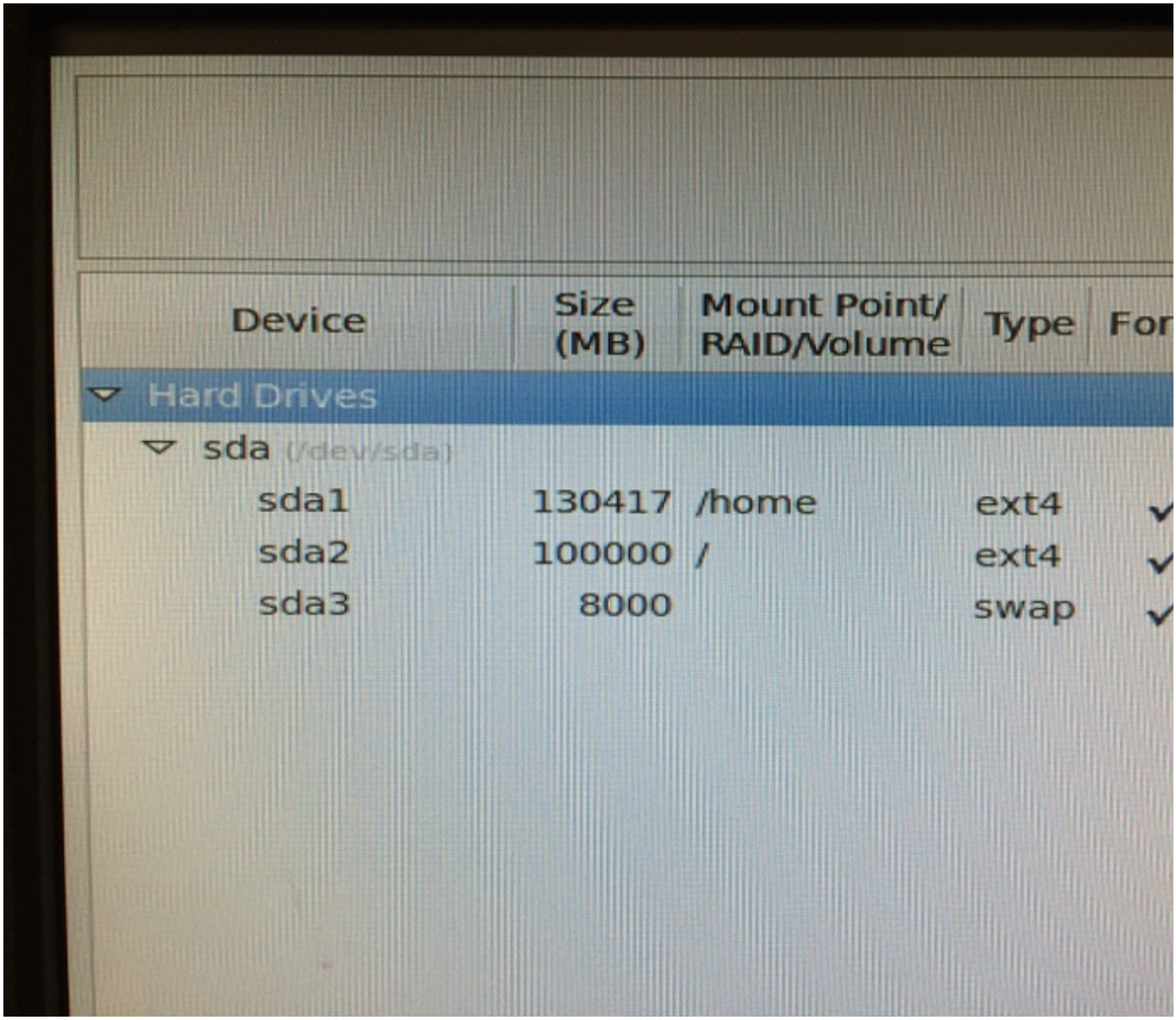
7. It will now ask for a root password. Hint: there Exists no Non Trivial Unitary Finite Dimensional Representation of a Non Compact Group.

8. Disk Formatting. Refer to image below. I chose "Create Custom Layout".

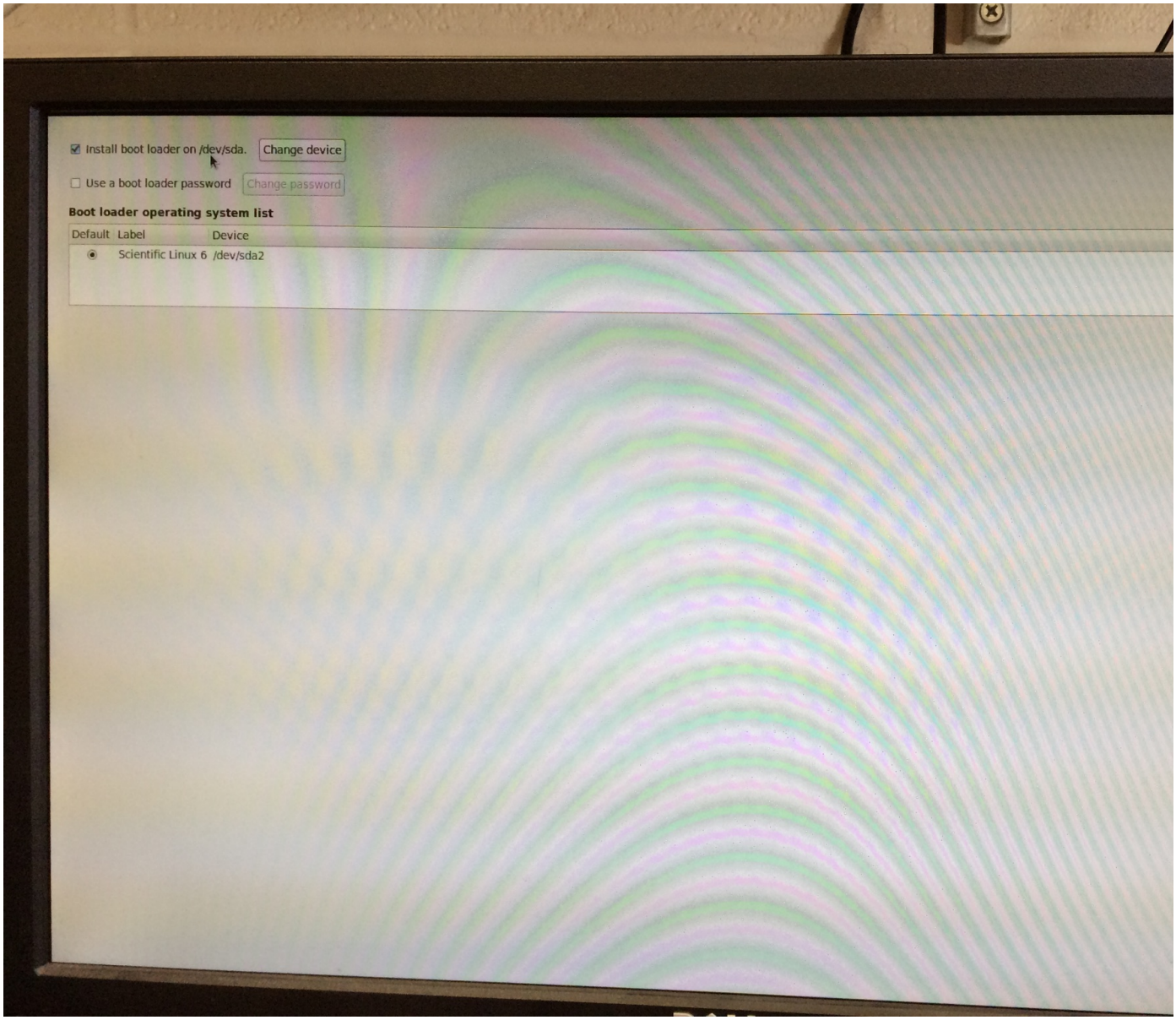


9. A screen similar to the one below was displayed. I deleted the old partitions and created three new ones:

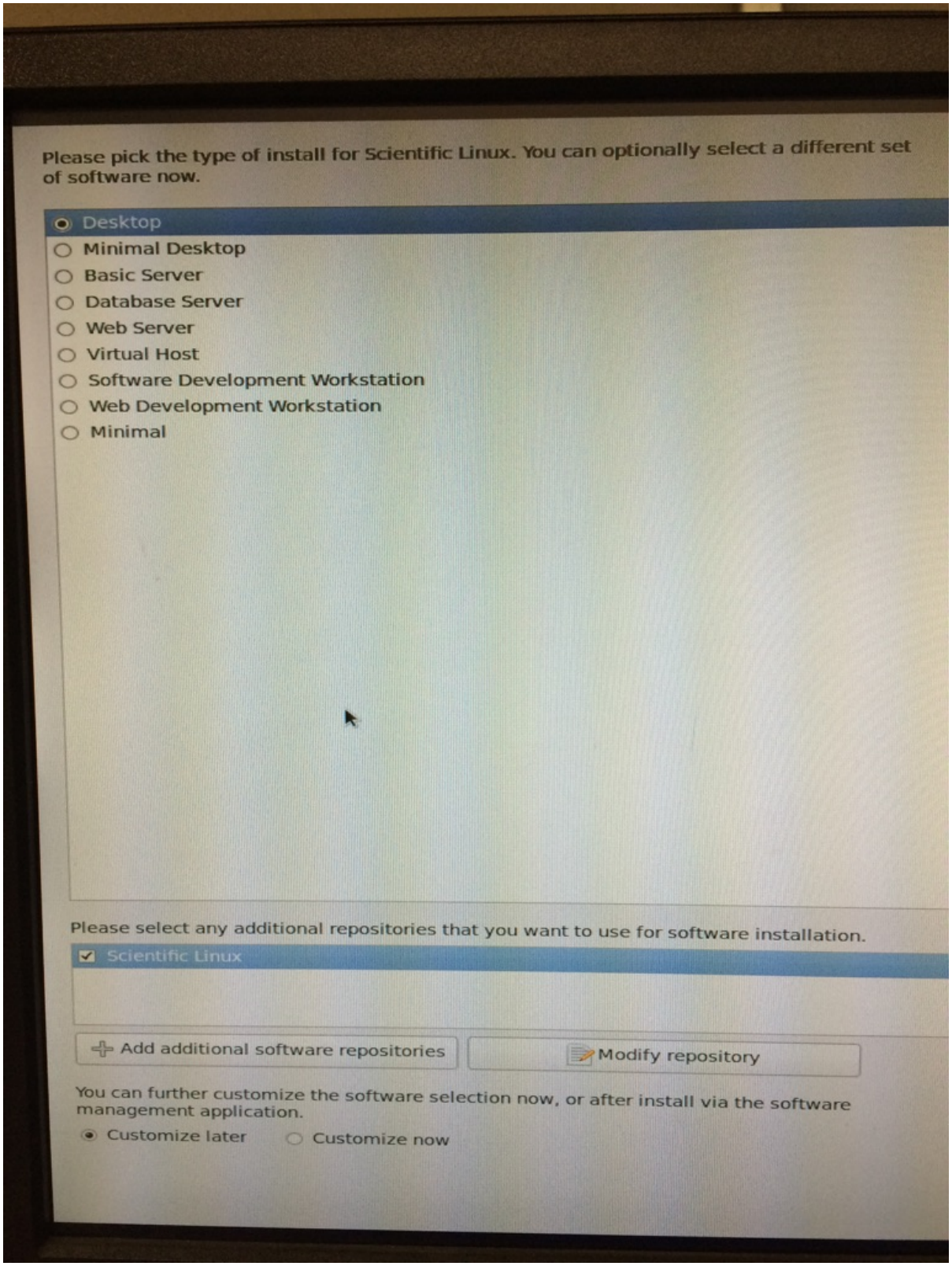
* 100 GB for '/'
* 8 GB for 'swap'
* Rest for '/home' as shown in the image below



9b. It will now ask for which drive to install the boot disk on. If you had a configuration similar to the previous image you should choose /dev/sda2 (take a look below). And then commit the changes to the master boot loader.

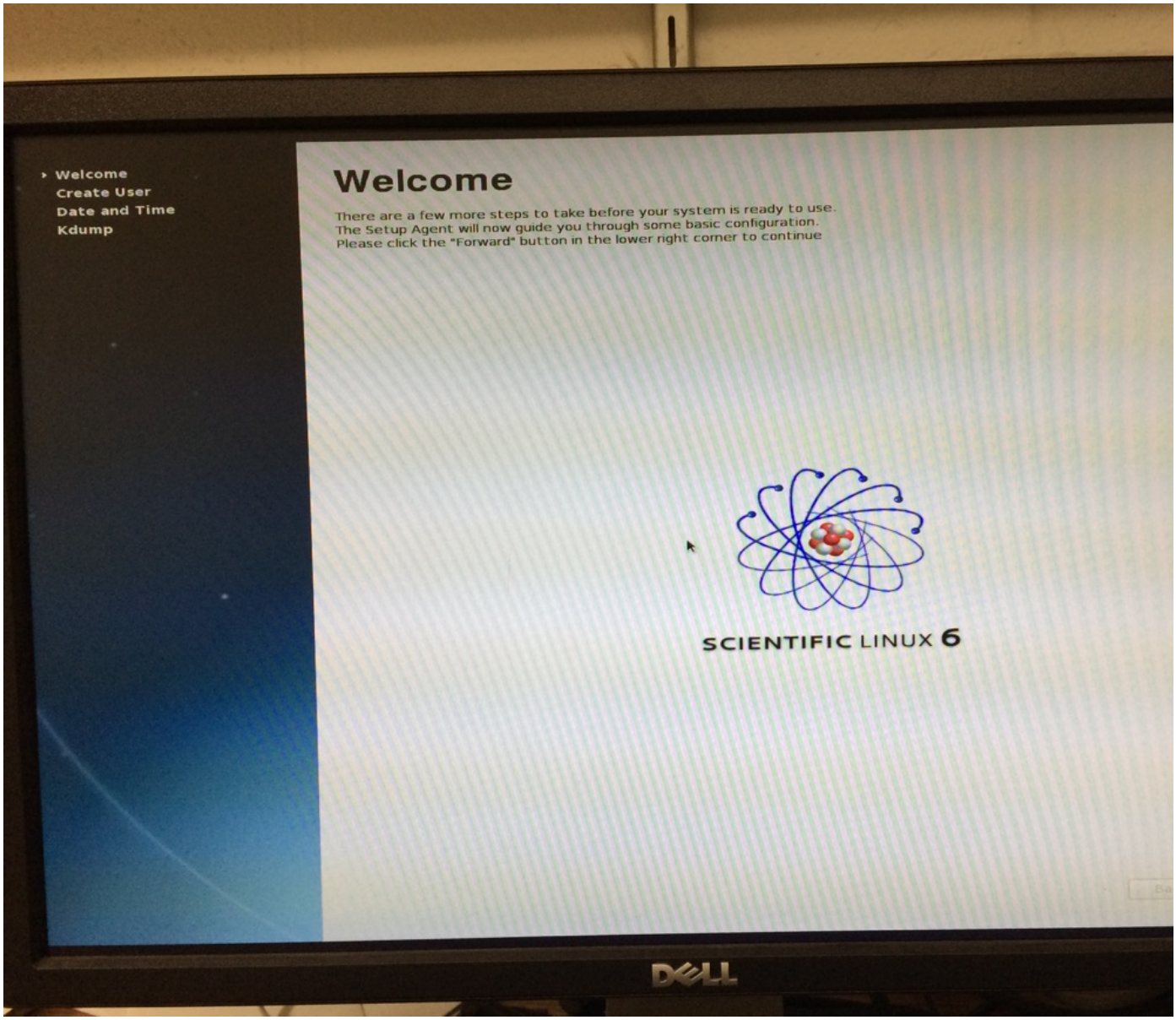


10. It will now ask to select the repositories to install. Once again I chose the defaults as shown in the image below.

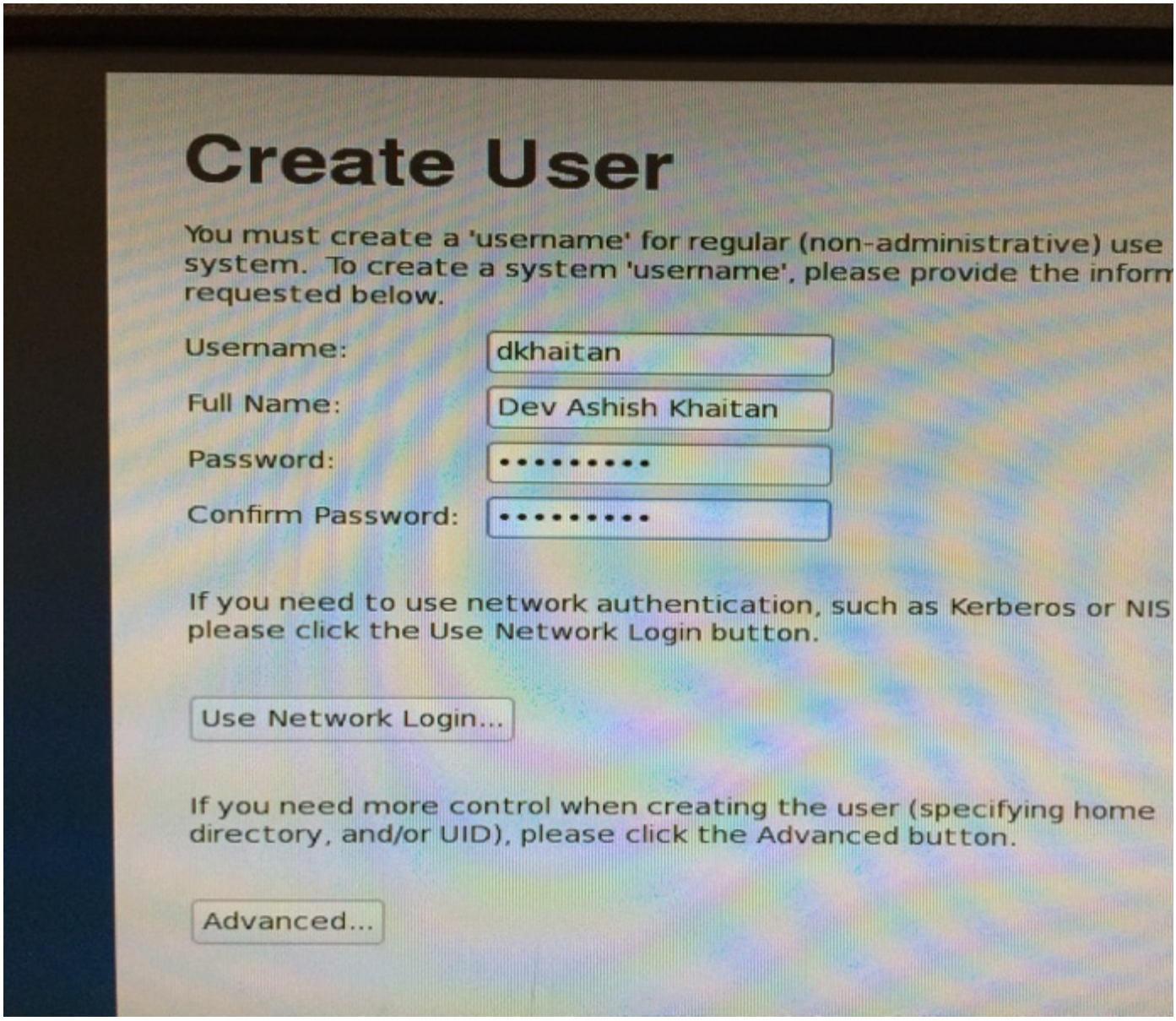


11. Now it will ask for confirmation to install Scientific Linux in '/'. Agree. This step takes a while.

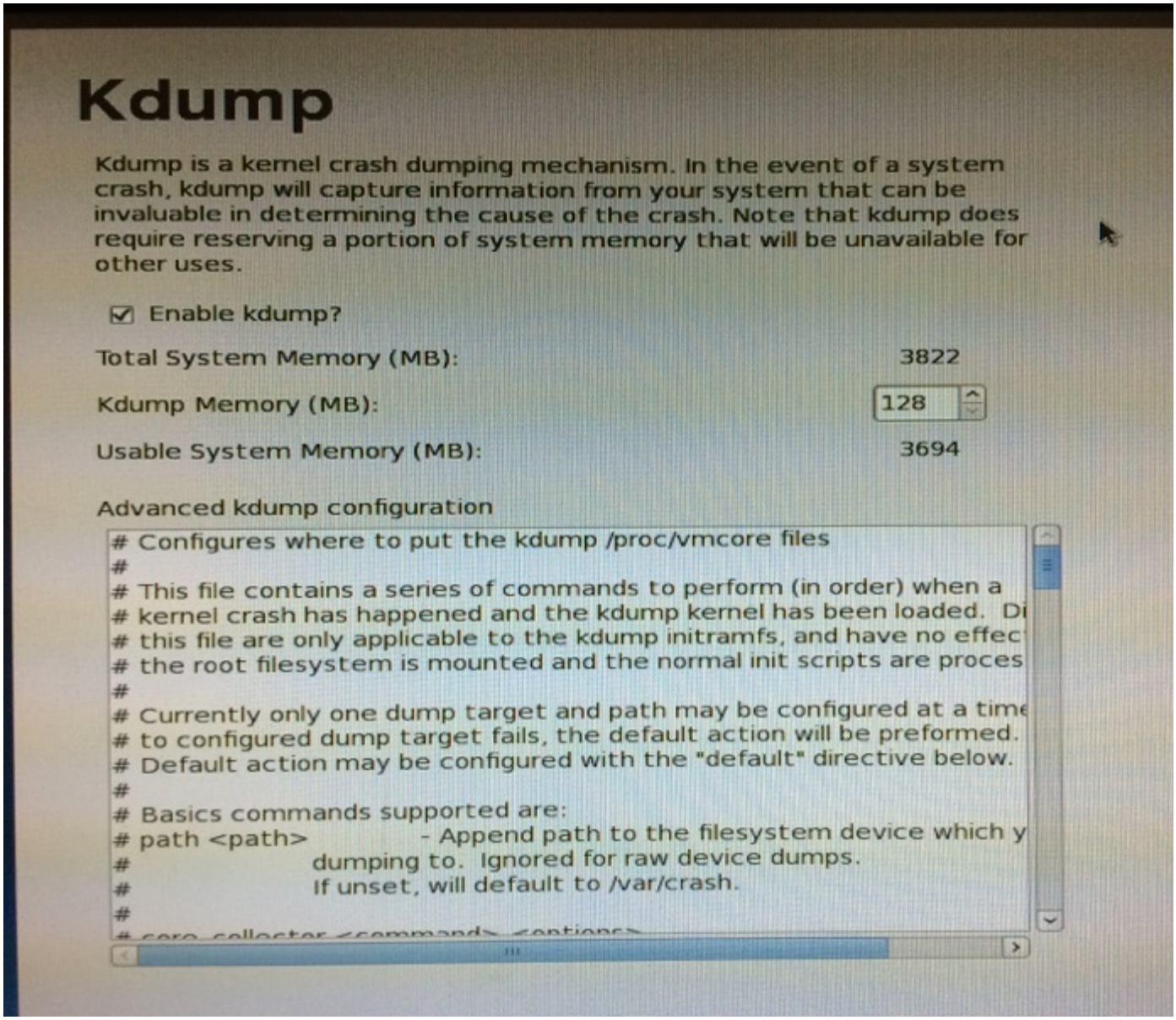
12. When it is done it will have a welcome screen as shown below.



13. It will now ask for a Username and Password. Image shown below. It will want the date and time after this screen; choose sync over the network or enter manually.



14. Lastly it will wan to Create the Kdump. I don't know what this is so I accepted the default settings shown below.



15. It will now reboot and you can log into the user you created in step 13.

Transferring root/admin privileges

http://www.server-world.info/en/note?os=Scientific_Linux_6&p=initial_conf&f=8

1. Log in as the root user. (take a look above in the steps to install for a hint)

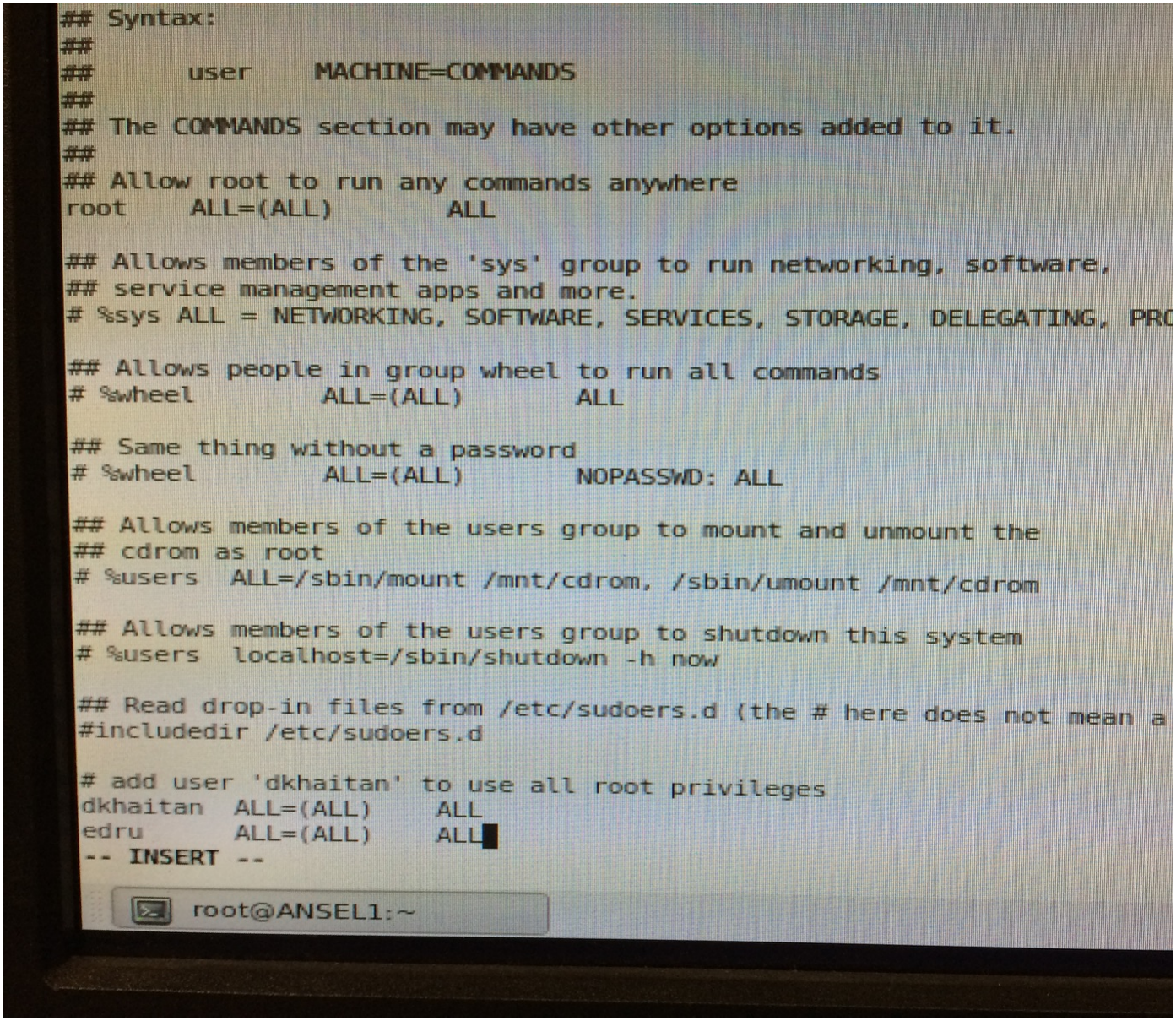
2. In a terminal type "visudo". This will be open up the sudo settings file in vim. To use vim refer to the following page .

3. Scroll to the bottom on the file opened in vim and hit the following keys:

- ":e"
- Character return
- Any key

Now at the bottom of the page you should see in bold "-- INSERT --". We are now ready to edit the file.

4. Now add the last three lines shown in the image below. This grants users "edru" and "dkhaitan" to the super user list and grants them all privileges.



5. Hit the "Esc" key to exit the editing more.

6. Type ":wq" to exit the file saving changes or ":q!" to exit without saving changes. This should exit the file and return you to the command line.

7. You can run the command "visudo" again in the terminal and scroll to the bottom of the file to confirm the changes have been made.

Getting on the Network, Connecting to the Internet

This is a solution that was used to get DSP-1 online. We spoofed the DSP-1 MAC address with the ANSEL-1 MAC address. ANSEL-1 was already registered with the DHCP server so we are basically using that registration. This is a temporary solution as long as ANSEL-1 MAC is not connected. For a permanent solution you should contact

Dave Munson to get on the network.

```
sudo ifconfig em1 hw 00:21:9B:55:58:10
sudo ifdown em1
sudo ifup em1
```

Installing google-chrome

http://www.tecmint.com/install-google-chrome-on-redhat-centos-fedora-linux/

Get NAUTILUS to behave

Open a folder in the regular "non tree" way. Go to Edit -> Behaviour and check the box for "Always Open in Browser Windows".

Installing Python 3.3

```
1. Navigate to http://continuum.io/downloads and fetch the latest full installer for linux x86_64.

2. Save the .sh file to /anaconda (you will need root privileges to make this folder).

3. From within the directory run the command "bash Anaconda-2.x.x-Linux-x86[_64].sh"

4. Agree to the terms and install this in /Anaconda

5. Add the path to our environment.
$sudo gedit ~/.bashrc
//Add the line to the end of the file
PATH=$PATH:/Anaconda/bin

6. Change permission to folder so all users can access, update and edit (somewhat dangerous)
$sudo chmod -R 777 /Anaconda

7. Now we update and add python 3 and ipython notebook.
$conda update conda
$conda create -n py3k python=3 anaconda
$source activate py3k (to launch a python 3 session)
$conda update ipython ipython-notebook ipython-qtconsole



8. Add python 3 path to the environment:
$sudo gedit ~/.bashrc
//Add the line to the end of the file
PATH=$PATH:/Anaconda/envs/py3k/bin/:

9. To run python we now use python3 for all our purposes.
```

Installing gcc/gcc-c++

There isn't a newer version of gcc available for RHEL (Red Hat Enterprise Linux) at the time of writing this page. So we will install GCC 4.4.7 (What is listed on the ElectronicsChainTestSoftware page). Additionally all the testing for various other programs has been done with this version of GCC so it is recommended even if a newer version is available. Since we are installing it directly from the repository manager (no hacking; this is easy to do).

You might need sudo privileges to run these commands but that's about it.

```
yum install gcc
yum install gcc-c++
```

Installing Blackfin gcc

Installing the Blackfin compiler is slightly more intense than the gcc compiler but I think I have simplified the process for future use. NOTE: Because the version supported of gcc support on RHEL is so old we need to use an older toolchain. I started with the latest (2014-R1) and kept working backwards till I found one that is compatible with our version of gcc. The 2011R1-RC4 toolchain was found to work.

1. From the website grab the three .rpm files. The website is: http://sourceforge.net/projects/adi-toolchain/files/2011R1/2011R1-RC4/GCC4.3/x86_64/ and we will need the following rpms: blackfin-toolchain-2011R1-RC4.x86_64.rpm ; blackfin-toolchain-elf-gcc-4.3-2011R1-RC4.x86_64.rpm and; blackfin-toolchain-uclibc-default-2011R1-RC4.x84_56.rpm

2. Now install these RPMS:

sudo yum install ~/Downloads/blackfin-toolchain-2011R1-RC4.x86_64.rpm
sudo yum install ~/Downloads/blackfin-toolchain-elf-gcc-4.3-2011R1-RC4.x86_64.rpm
sudo yum install ~/Downloads/blackfin-toolchain-uclibc-default-2011R1-RC4.x86_64.rpm

3. These rpms should now end up in /opt/uClinux/. Check that they are there as:

$ ls -l /opt/uClinux/
total 16
drwxr-xr-x 9 root root 4096 Mar 28 16:34 bfin-elf
drwxr-xr-x 9 root root 4096 Mar 28 16:33 bfin-linux-uclibc
drwxr-xr-x 9 root root 4096 Mar 28 16:33 bfin-uclinux

4. From this page download futils.tar.gz. (tarred with tar -zcvf)

5. Now create a directory called "futils" in /opt/uClinux/

mkdir /opt/uClinux/futils

6. Now we can untar the contents for futils.tar.gz

sudo tar -zxvf ~/Downloads/futils.tar.gz -C /opt/uClinux/futils/

7. Now we need to update our path variables. So edit ~/.bashrc to include the line below:

PATH=$PATH:/opt/uClinux/bfin-elf/bin:/opt/uClinux/bfin-linux-uclibc/bin:/opt/uClinux/bfin-uclinux/bin:/opt/uClinux/futils

8. I have found that for whatever reason this path doesn't always pick up the futils so I always hardcode those include statements in like:

#include "/opt/uClinux/futils/bitmasks.h" /*useful bitmasks*/
#include "/opt/uClinux/futils/fpga_4futils.h" /*FPGA addresses*/
#include "/opt/uClinux/futils/fcommon.c" /*FPGA utility functions*/

etc....

9. When compiling the code always use bfin-linux-uclibc-gcc. This will produce an "a.out" which you can rename to whatever you want; example:

bfin-linux-uclibc-gcc fbinary_5waves.c //this produces an executable called a.out

mv a.out fbinary_5wave //renaming this executable so I Know what it is.

ALTERNATIVELY, you can also use

bfin-linux-uclibc-gcc fbinary_5waves.c -o fbinary_5wave

Installing ICE 3.5.1

This is very easy to install. The rpm commands on the website work well:

```
wget https://zeroc.com/download/Ice/3.5/Ice-3.5.1-el6-x86_64-rpm.tar.gz
tar xvf Ice-...
cd Ice-...
rm db53-devel-5.3.21-1ice.el6.x86_64.rpm # only needed for building ice from source, dependency issue
sudo yum install ./*.rpm
```

Installing ICE-E 1.2.0

This is more complicated. A bunch of my commands here I came up with by trial and error.

1. Get the Ice-E 1.2 Translator (slice2cppe command for Ice-E)

```
mkdir icee
cd icee
wget https://zeroc.com/download/IceE/1.2/IceE-trans-1.2.0.tar.gz
tar xvf ./IceE-trans-1.2.0.tar.gz
cd IceE-trans-1.2.0
```

2. When I attempted to install this I ran into issue where a whole lot of files couldn't locate the "cstring" package for c. I added them manually and then it works fine.

```
Add the command
#include <cstring>
somewhere with all the other #include commands in the following files.

In the folder IceE-trans-1.2.0/src/IceUtil the files that need editing are:
ArgVector.cpp
MD5.cpp
OutputUtil.cpp
Random.cpp
ThreadException.cpp



In the folder IceE-trans-1.2.0/src/Slice the files that need editing are:
Grammar.cpp
Parser.cpp
CPlusPlusUtil.cpp
Preprocessor.cpp
PythonUtil.cpp (this one #include <limits.h>)
DotNetNames.cpp

In the folder IceE-trans-1.2.0/src/slice2javae the file that needs editing are:
Gen.cpp
```

3. One more file that needs where an include needs to be added

```
Add the command
#include <limits.h>
somewhere with all the other #include commands in the following files.

In the folder IceE-trans-1.2.0/src/Slice the files that need editing are:
PythonUtil.cpp
```

4. We are now ready to compile and install.

```
In the folder IceE-trans-1.2.0 running the following commands:
$ make
$ make install
```

5. This should now compile the whole package without any errors. Let's now add the path to the environment. Edit ~/.bashrc to include this line:

```
PATH=$PATH:/opt/IceE-1.2.0/bin
```

6. Download the IceE package to the same folder where we downloaded and installed IceE

```
wget https://zeroc.com/download/IceE/1.2/IceE-1.2.0.tar.gz
tar xvf IceE-1.2.0.tar.gz
cd IceE-1.2.0
```

7. Not sure about this step but it seems to make things work.

```
Somewhere near line 100 in  IceE-1.2.0/include/IceE/Comfig.h

add || defined(BFIN) to

#if
    ...
#   define ICE_LITTLE_ENDIAN
```

8. Once again we need to add more packages to some files in which they are missing.

```
Add the command
#include <memory>
somewhere with all the other #include commands in the following files.

In the folder IceE-1.2.0/include/IceE/ the files that need editing are:
Outgoing.h
Connection.h
```

9. We need to change a couple of things in the make.rules and make.rules.linux:

```
in Ice-1.2.0/config/Make.rules
change
prefix = /opt/IceE-$(VERSION)
to
prefix = /opt/IceE-bfin-$(VERSION)

in ./config/Make.rules.Linux
comment the lines (around line 60)
CXXARCHFLAGS    = -m64
and
CXXARCHFLAGS    = -m32
```

10. Now we are ready to compile all this fun stuff. We are going to cross compile IceE for tha blackfin.

```
in IceE-1.2.0/ run the following commands

CXX=bfin-linux-uclibc-c++ AR=bfin-linux-uclibc-ar make
sudo make install
```

11. Everything should be good to run. The libraries will now be in /opt/. Happy Coding.

Telnet And Connecting to DDC10s

```
yum install telnet telnet-server -y
sudo yum install samba

sudo gedit /etc/selinux/config
<<Change SELINUX=enforcing to SELINUX=disabled>>



sudo su
service smb stop
mkdir /media/share/

smb.conf:
cp /etc/samba/smb.conf /home/dkhaitan/
gedit /etc/samba/smb.conf
<<I may have added some things here, username level, allow guests, probably not needed just experimnetation>><
[[global]]
#    workgroup = MYGROUP
#    server string = Samba Server Version %v
#2015/03/17 Commenting line above and adding line below
        workgroup = WORKGROUP
        server string = samba
        username level = 2
        usershare allow guests = yes
;    netbios name = MYSERVER

<< I changed the password backend to smbpasswd, because I am familiar with it, tbdsam would probably work fine
        passdb backend = smbpasswd

<<SHARE DEFINITION>>
<< Can probably turn off guest, it is important that the share is NOT in the home folder.
Maybe make a /data in root or something? >>
        [SHARE]
        path = /media/share/
        guest ok = yes
        read only = no
        valid users = lz_devel
    create mask = 0777
    directory mask = 0777

<< Create a user lz_devel, create an smbpasswd for that user >>
<< chown -R lz_devel:lz_devel /media/share
   or whatever folder you use for the share >>
ls -l /media/

service smb start
```

```
<< On the DDC:
mkdir /mnt/share
mkdir /mnt/DDCTest
smbmount //192.168.1.100/SHARE /mnt/share/ -o username=lz_devel
smbmount //192.168.1.100/SHARE /mnt/share/ -o username=lz_devel,password=W1MPfindr\$
smbmount //192.168.1.100/DDCTest /mnt/DDCTest/ -o username=lz_devel,password=W1MPfindr\$
cp /mnt/share/fbinary_2f /usr/bin/
>>
<< Note that you have to escape the $ in the password... >>
```

Installing Video Driver

```
lspci -v
<<Look for VGA Compatible controller in this case we have:>>
00:02.0 VGA compatible controller: Intel Corporation 82Q35 Express Integrated Graphics Controller (rev 02) (pr
   Subsystem: Dell OptiPlex 755
   Flags: bus master, fast devsel, latency 0, IRQ 25
   Memory at fea00000 (32-bit, non-prefetchable) [size=512K]
   I/O ports at ec90 [size=8]
   Memory at d0000000 (32-bit, prefetchable) [size=256M]
   Memory at feb00000 (32-bit, non-prefetchable) [size=1M]
   Expansion ROM at <unassigned> [disabled]
   Capabilities: <access denied>
   Kernel modules: i915

https://01.org/linuxgraphics/downloads/2015/intelr-graphics-installer-linux-1.0.8

$ wget --no-check-certificate https://download.01.org/gfx/RPM-GPG-KEY-ilg-2 ; \
$ sudo rpm --import RPM-GPG-KEY-ilg-2

$ wget --no-check-certificate https://download.01.org/gfx/RPM-GPG-KEY-ilg-2 ; \
$ sudo rpm --import RPM-GPG-KEY-ilg-2
```

Installing NTFS/Getting Packages from EPEL

http://www.unixmen.com/install-epel-repository-rhel-centos-scientific-linux-6/

yum install --disablerepo="*" --enablerepo=epel

Installing X Go/Remote Desktop/Perl

sudo yum install perl-DBI
sudo yum install perl-DBD-Pg
sudo yum install perl-DBD-SQLite
sudo yum install perl-Config-Simple
sudo yum install perl-File-Which
sudo yum install --disablerepo="*" --enablerepo=epel x2goserver-xsession

Installing Octave 4

```
mkdir Octave
wget ftp://ftp.gnu.org/gnu/octave/octave-4.0.0.tar.gz
tar -zxvf octave-4.0.0.tar.gz
cd octave-4.0.0

Then when you run "./configure" you should get a whole bunch of missing packages. Time to download them.

sudo yum install gcc gcc-c++ kernel-devel make mercurial libtool libtool-ltdl-devel libtool-ltdl autoconf cmak
lapack pcre-devel readline-devel readline fftw-devel glpk-devel suitesparse suitesparse-devel gnuplot libcurl-
flex texlive gperf qt-devel bison ghostscript-devel

sudo yum install perl-libwww-perl.noarch libffi-devel mesa-libOSMesa mesa-libOSMesa-devel


sudo yum install --disablerepo="*" --enablerepo=epel icoutils librsvg2 llvm-devel qscintilla-devel arpack-deve

./configure
make

./run-octave    - to run in place to test before installing
make install    - to install (PREFIX=/usr/local)
```

Installing LUXSim

These are the installations instructions I got from Sean Fallon, a PhD student at SUNY Albany working with Matthew Syzdagis. They work phenomenally well on SciLinux, less well on debian based linuxes. I'll add my own installation notes soon. Also worth noting that modifying the .bashrc makes the X2goclient unable to connect. You can work around this by creating a copy of the .bashrc profile call is .LUXSim_bashrc; in this you place the three additional statements required for geant4. Everytime you want to run LUXSimExectuable you run "source .LUXSim_bashrc" and that should be a good enough work around.

- LUXSimAdvancedTraining.pdf: LUXSimAdvancedTraining.pdf
- geant494p04_instruction.pdf: geant494p04_instruction.pdf

Installing LZSim

VERY similar to above but newer versions of packages.

```
1. CLHEP-2.1.3.1
http://wiki.opengatecollaboration.org/index.php/New_Compilation_ProcedureV7.0#CLHEP
$ cd /usr/share/
$ sudo mkdir clhep2131
==> Download the package clhep 2.1.3.1 from http://proj-clhep.web.cern.ch/proj-clhep/DISTRIBUTION/
$ sudo mv ~/Downloads/clhep-2.1.3.1.tgz /usr/share/clhep2131/
$ cd /usr/share/clhep2131/
$ sudo tar -zxvf clhep-2.1.3.1.tgz
$ sudo mkdir build
$ sudo mkdir install
$ cd build
$ sudo cmake -DCMAKE_INSTALL_PREFIX=/usr/share/clhep2131/install/ -DCLHEP_BUILD_DOCS=ON /usr/share/clhep2131/2
$ sudo make
$ sudo make test
$ sudo make install


2. Geant-4/4.9.5.
http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/ch02.html#sect.UnixBuild

some stuff
in /usr/share/geant4/build_g495/
$ sudo cmake -DCMAKE_INSTALL_PREFIX=/usr/share/geant4/install_g495/  -DGEANT4_INSTALL_DATA=ON  -DBUILD_STATIC_

sudo cmake -DCMAKE_INSTALL_PREFIX=/usr/share/geant4/install_g495/  -DGEANT4_INSTALL_DATA=ON  -DBUILD_STATIC_LI

sudo cmake -DCMAKE_INSTALL_PREFIX=/usr/share/geant4/install_g495/  -DGEANT4_INSTALL_DATA=ON  -DBUILD_SHARED_LI


http://geant4.cern.ch/UserDocumentation/UsersGuides/InstallationGuide/BackupVersions/V9.5_Ver01_20111202/fo/Bo

sudo cmake -DCMAKE_INSTALL_PREFIX=/usr/share/geant4/install_g495/  -DGEANT4_INSTALL_DATA=ON  -DBUILD_SHARED_LI
```

NI-VISA and Python

First you need the NI-VISA drivers. You can download them from: http://www.ni.com/download/ni-visa-15.0/5410/en/ Version 15 supports 64-bit linux platforms. Install NI-VISA as described in the README file:

```
You should be logged in as 'root' to perform this installation.

1) Download the ISO image "NI-VISA-#.#.#.iso" from the ftp site into a
   temporary directory.

2) Open a Terminal (command prompt) window.

3) Mount the ISO image. This may be done by typing
   "mount -o loop <iso-location>/NI-VISA-#.#.#.iso <mount-location>".

4) Make the mount location your current working directory. This may be done
   by typing "cd <mount-location>".

5) Type "./INSTALL" to execute the install script, and follow the instructions
   as prompted.
```

Next you have to install PyVisa to be able to talk to NI-VISA drivers from Python.

Since we are working with the Anaconda package you:

1. Open a terminal (as a regular user, no root or sudo)
2. Activate a given anaconda environment (root or py3k):
    - `source activate root`
    - `source activate py3k`
3. Install PyVisa package through pip (with no sudo or root privillages):
    - `pip install pyvisa`

If the install gave no errors you can try to open python in the chosen anaconda environment and try executing the following:

```
>>> import visa
>>> rm = visa.ResourceManager()
>>> print(rm.list_resources())
```

and you should see something like:

```
('ASRL1::INSTR', 'ASRL2::INSTR', 'ASRL3::INSTR', 'ASRL4::INSTR', 'ASRL5::INSTR', 'ASRL6::INSTR')
```

I	Attachment	Action	Size	Date	Who
	IMG_0735.jpg	manage	793.8 K	2015-01-29 - 14:06	DevKhaitan
	IMG_0786.jpg	manage	515.4 K	2015-03-29 - 16:42	DevKhaitan
	IMG_1570.jpg	manage	173.5 K	2015-01-20 - 23:40	DevKhaitan
	IMG_4284.jpg	manage	171.8 K	2015-01-20 - 23:40	DevKhaitan
	IMG_4880.jpg	manage	209.5 K	2015-01-20 - 23:40	DevKhaitan
	IMG_5209.jpg	manage	146.3 K	2015-01-20 - 23:40	DevKhaitan
	IMG_7066.jpg	manage	156.8 K	2015-01-20 - 23:40	DevKhaitan
	IMG_7729.jpg	manage	207.4 K	2015-01-20 - 23:40	DevKhaitan
	IMG_8122.jpg	manage	151.5 K	2015-01-20 - 23:40	DevKhaitan
	LUXSimAdvancedTraining.pdf	manage	736.6 K	2015-10-06 - 15:01	DevKhaitan
	futils.tar.gz	manage	93.7 K	2015-03-30 - 09:40	DevKhaitan
	geant494p04_instruction.pdf	manage	38.7 K	2015-10-06 - 15:01	DevKhaitan

Topic revision: r27 - 2016-01-19 - ErykDruszkiewicz