SpecTcl

Online/Offline Analysis Code Tutorial

Kevin Carnes

3/26/2018

Online DAQ Linux computers:

Fizeau – Itzik's group, ECR area

Geissler – Artem's group, PULSAR

Newer version of Linux Ubuntu

Avogadro – Daniel's group, PULSAR

Ampere – Raju, Sq. Room

Henry – New machine under development

Offline analysis Linux computers:

All rack mount machines without consoles, housed in CW31

Auger

Balmer

Fourier

Oppy

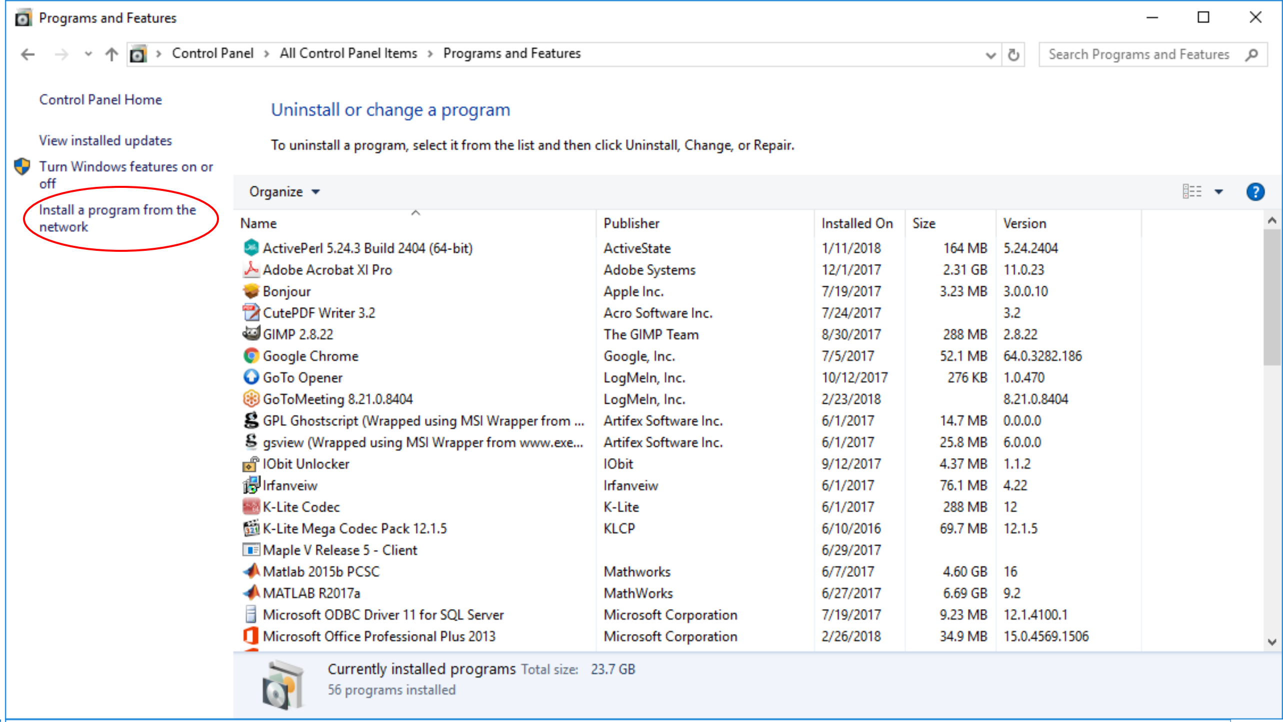
Stark

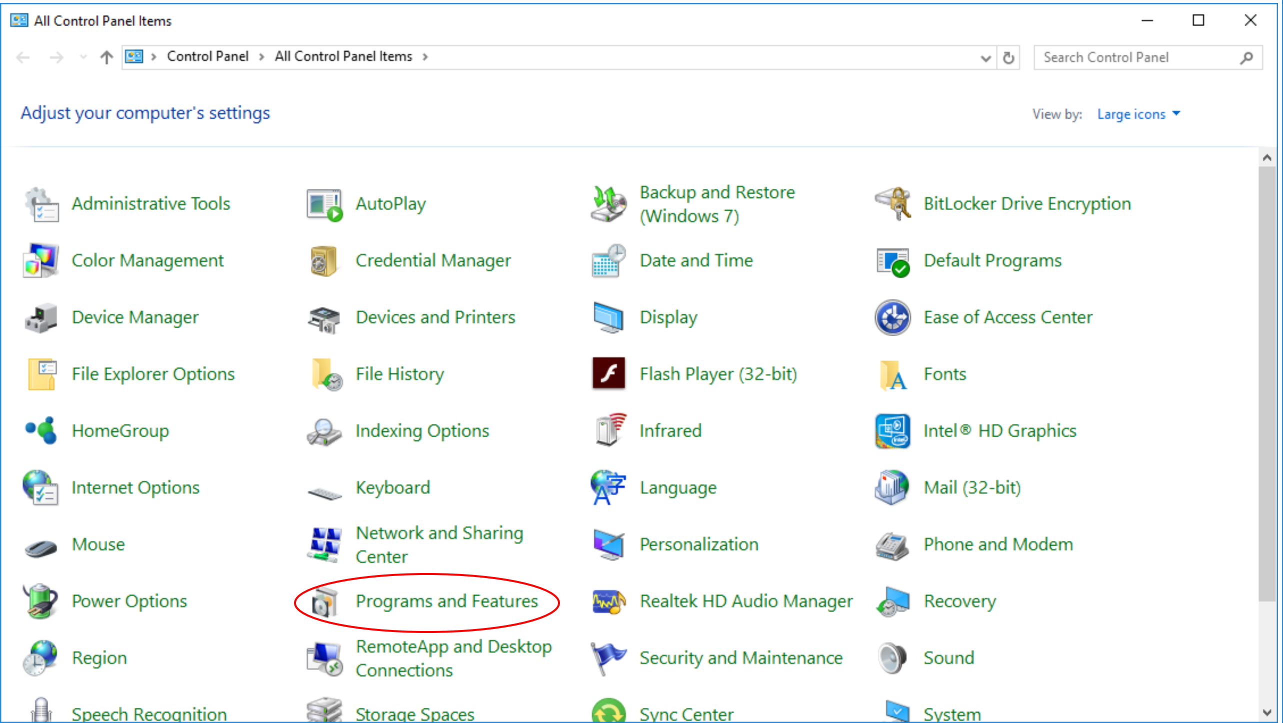
Stern

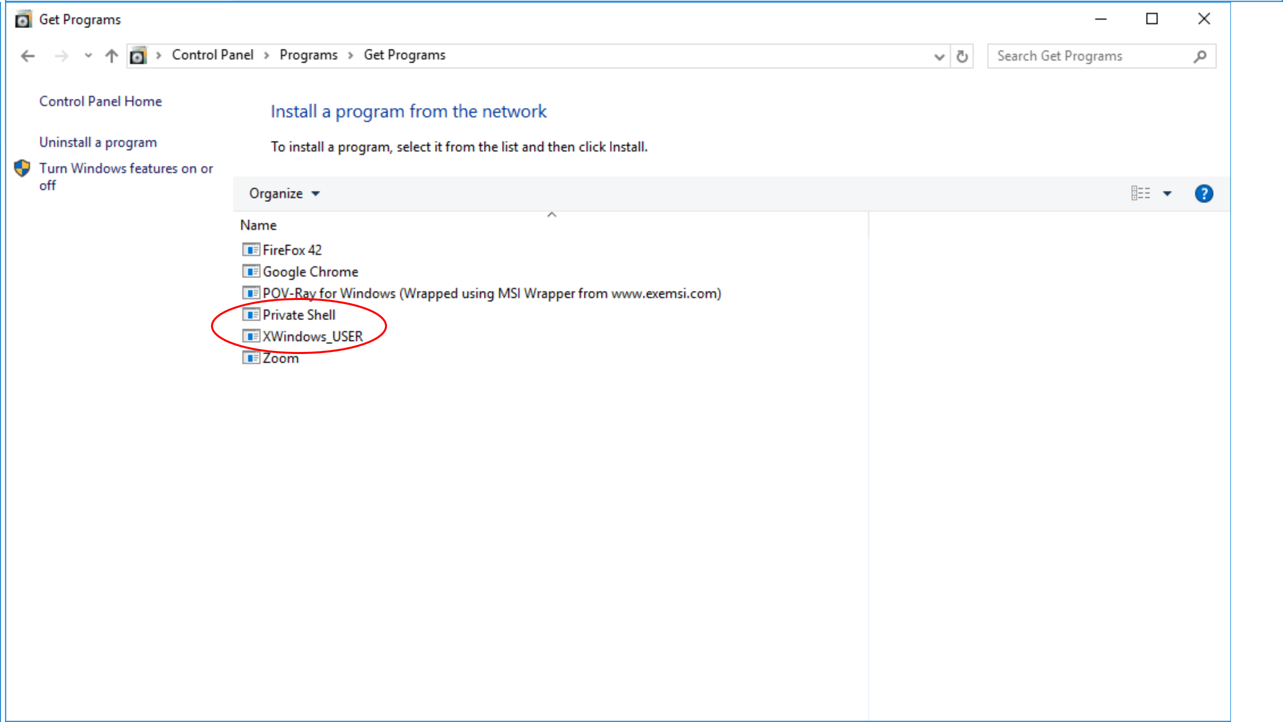
Use Private Shell and Xming to connect to Linux PC from Windows

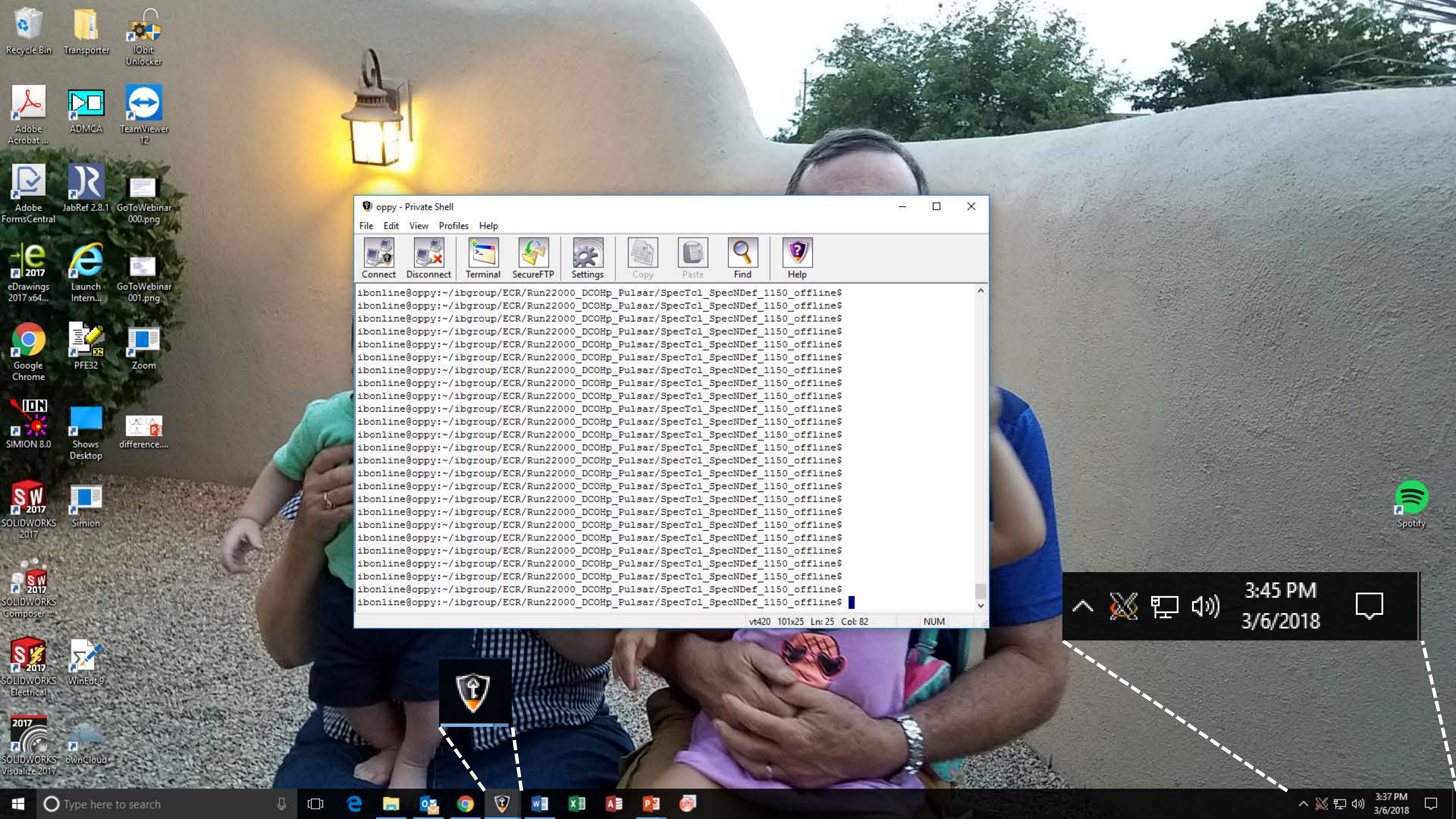
These can be installed on Windows 7 using “Run Advertised Programs”

For Windows 10, search for Control Panel







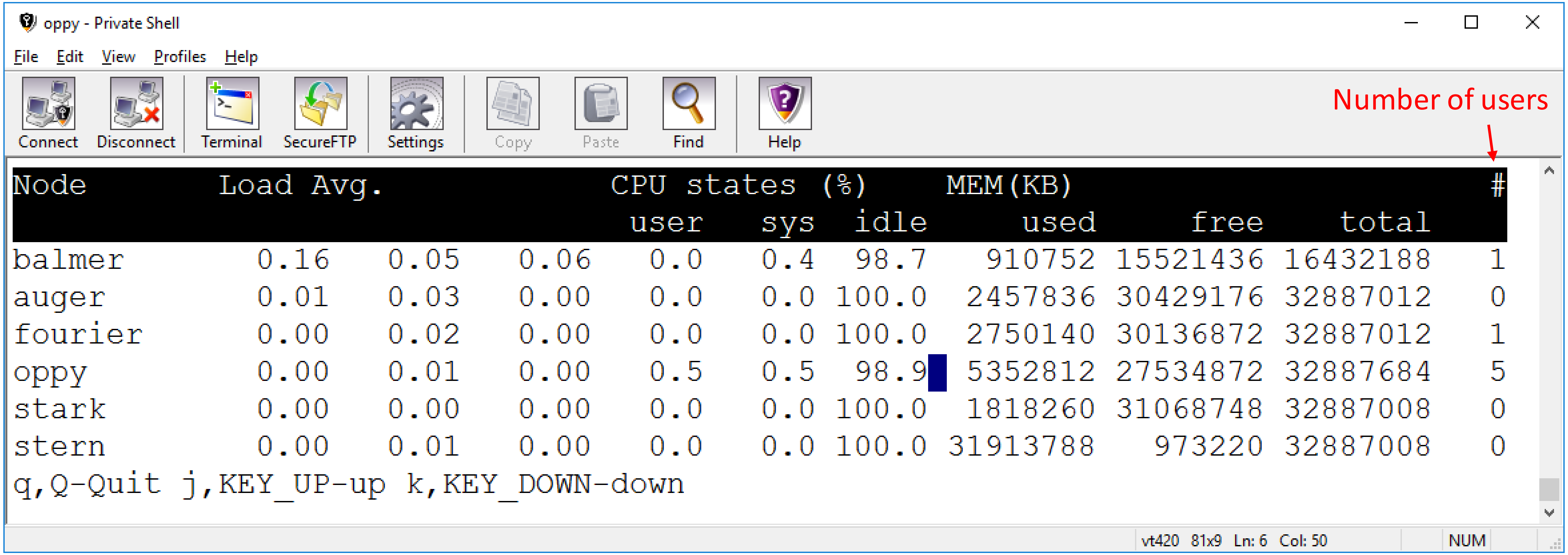


Recycle Bin
Transporter
IObit Unlocker
Adobe Acrobat ...
ADMCA
TeamViewer 12
Adobe FormsCentral
JabRef 2.8.1
GoToWebinar 000.png
eDrawings 2017 x64...
Launch Intern...
GoToWebinar 001.png
Google Chrome
PFE32
Zoom
SIMION 8.0
Shows Desktop
difference....
SOLIDWORKS 2017
Simion
SOLIDWORKS Composer ...
SOLIDWORKS Electrical
WinEdt 9
SOLIDWORKS Visualize 2017
ownCloud
Spotify
oppy - Private Shell
File
Edit
View
Profiles
Help
Connect
Disconnect
Terminal
SecureFTP
Settings
Copy
Paste
Find
Help
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
ibonline@oppy:~/ibgroup/ECR/Run22000_DCOHp_Pulsar/SpecTcl_SpecNDef_1150_offline$
vt420 101x25 Ln: 25 Col: 82
NUM
3:45 PM
3/6/2018
Type here to search
3:37 PM
3/6/2018

nctop – Linux command to check load and number of users on networked machines

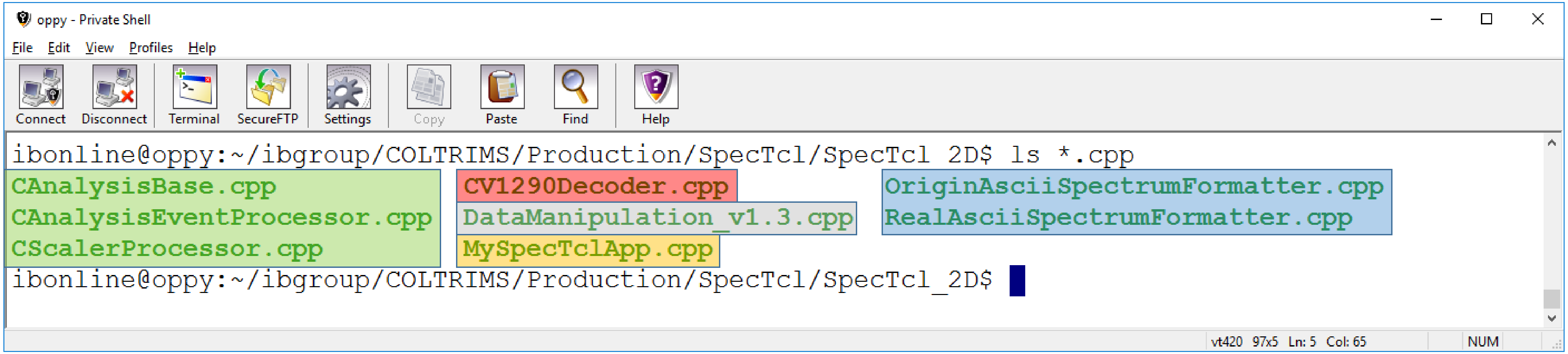
Use this to decide which machine to do analysis on. Only works with offline analysis machines.

Number of users



Node	Load Avg.			CPU states (%)			MEM(KB)			#
				user	sys	idle	used	free	total	
balmer	0.16	0.05	0.06	0.0	0.4	98.7	910752	15521436	16432188	1
auger	0.01	0.03	0.00	0.0	0.0	100.0	2457836	30429176	32887012	0
fourier	0.00	0.02	0.00	0.0	0.0	100.0	2750140	30136872	32887012	1
oppy	0.00	0.01	0.00	0.5	0.5	98.9	5352812	27534872	32887684	5
stark	0.00	0.00	0.00	0.0	0.0	100.0	1818260	31068748	32887008	0
stern	0.00	0.01	0.00	0.0	0.0	100.0	31913788	973220	32887008	0

```
q,Q-Quit j,KEY_UP-up k,KEY_DOWN-down
```



CV1290Decoder.cpp – Actual sorting code, the only file here that is regularly modified

MySpecTclApp.cpp – Registers the sorting code, only changed if additional decoders added

CAnalysisBase.cpp
CAnalysisEventProcessor.cpp – Files related to scalers, not to be edited
CScalerProcessor.cpp

OriginAsciiSpectrumFormatter.cpp – Files related to exporting spectra, not to be edited
RealAsciiSpectrumFormatter.cpp

DataManipulation_v1.3.cpp – File used to further process data, such as delay line fitting or random generation

Makefile, contains all *.cpp files to compile and link

After modification, or moving to a new directory or machine:
make clean
make

Makefile – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

```
#
#   Append your objects to the definitions below:
#

OBJECTS=MySpecTclApp.o  OriginAsciiSpectrumFormatter.o RealAsciiSpectrumFormatter.o CV1290Decoder.o CAnalysisBase.o \
        CAnalysisEventProcessor.o CScalerProcessor.o


#
#  Finally the makefile targets.
#


SpecTcl: $(OBJECTS) DataManipulation_v1.3.o
        $(CXXLD)  -o SpecTcl $(OBJECTS) $(USERLDFLAGS) \
        $(LDFLAGS)
        $(CXXLD) -o DataManipulation_v1.3 DataManipulation_v1.3.o
```

Line: 44 Col: 1 LINE INS Makefile Makefile

MySpecTclApp.cpp – COLTRIMS

MySpecTclApp.cpp – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

V785 Slot

V1290 Slot

Scaler processing

```
#include "CV1290Decoder.h"

// Local Class definitions:

CV1290Decoder decoder1 = CV1290Decoder(2, 4);

//  Function:
//    void CreateAnalysisPipeline(CAnalyzer& rAnalyzer)
//  Operation Type:
//     Override
/*
Purpose:

  Sets up an analysis pipeline.  This function is required and must
be filled in by the SpecTcl user.  Pipeline elements are objects
which are members of classes derived from the CEventProcessor
class. They should be added to the Analyzer's event processing pipeline
by calling RegisterEventProcessor (non virtual base class member).
*/

void
CMySpecTclApp::CreateAnalysisPipeline(CAnalyzer& rAnalyzer)
{

    RegisterEventProcessor(decoder1);
    SpecTcl* api = SpecTcl::getInstance();
    CTCLInterpreter* pInterp = api->getInterpreter();
//Register Scaler Event Processor
    RegisterEventProcessor(*(new CAnalysisEventProcessor(new ScalerProcessor(*pInterp))), "Scalers");
}
```

Line: 39 Col: 1 LINE INS C++ MySpecTclApp.cpp

MySpecTclApp.cpp – ECR

MySpecTclApp.cpp [modified] – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

Three decoders for ECR code

```cpp
#include "C1290EventDecoder.h"
#include "C1290Experiment.h"
#include "C1290Physics.h"

// Local Class definitions:

C1290EventDecoder decoder1 =  C1290EventDecoder(2, 4);   // Decode -> raw parameters.
C1290Experiment decoder2;        // Intermediate calculations
C1290Physics decoder3;           // Physics calculations
/*
Purpose:

  Sets up an analysis pipeline.  This function is required and must
be filled in by the SpecTcl user.  Pipeline elements are objects
which are members of classes derived from the CEventProcessor
class. They should be added to the Analyzer's event processing pipeline
by calling RegisterEventProcessor (non virtual base class member).
*/

void
CMySpecTclApp::CreateAnalysisPipeline(CAnalyzer& rAnalyzer)
{
    RegisterEventProcessor(decoder1);  // First step is -> raw params.
    RegisterEventProcessor(decoder2);  // Intermediate calculations
    RegisterEventProcessor(decoder3);  // Physics calculations
//    RegisterEventProcessor(scaldecoder);  // Scaler processor
    SpecTcl* api = SpecTcl::getInstance();
    CTCLInterpreter* pInterp = api->getInterpreter();
//Register Event Processor included in NSCLScaler files

    RegisterEventProcessor(*(new CAnalysisEventProcessor(new ScalerProcessor(*pInterp))), "Scalers");

}
```

Line: 39 Col: 1 LINE INS C++ MySpecTclApp.cpp

userscaler.tcl

Tcl/Tk command script updating scaler strip charts

userscaler.tcl – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

```
proc UserBeginRun {} {
  global strchan
  set strchan 0
}
proc UserUpdate {} {
  global strchan
  global Scaler_Increments
  channel -set photodiode $strchan $Scaler_Increments(1)
  channel -set mcpfront $strchan $Scaler_Increments(2)
  channel -set scalerx1 $strchan $Scaler_Increments(3)
  channel -set scalerx2 $strchan $Scaler_Increments(4)
  channel -set scalery1 $strchan $Scaler_Increments(5)
  channel -set scalery2 $strchan $Scaler_Increments(6)
  channel -set mcpback $strchan $Scaler_Increments(7)
  channel -set power $strchan $Scaler_Increments(8)
  incr strchan
}
```

Line: 3 Col: 16 LINE INS Tcl/Tk userscaler.tcl

Scaler channel name

Scaler channel number starting at 0

setup.tcl – defines parameters and spectra

setup.tcl – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

```
# (C) Copyright Michigan State University 2014, All rights reserved
#
#   Define the parameters:
#
#====================== SCALER DATA (strip charts) ============================
parameter photodiode  70  14
spectrum photodiode 1 photodiode 14
parameter mcpfront  71  14
spectrum mcpfront 1 mcpfront 14
parameter scalerx1  72  14
spectrum scalerx1 1 scalerx1 14
parameter scalerx2  73  14
spectrum scalerx2 1 scalerx2 14
parameter scalery1  74  14
spectrum scalery1 1 scalery1 14
parameter scalery2  75  14
spectrum scalery2 1 scalery2 14
parameter mcpback  76  14
spectrum mcpback 1 mcpback 14
parameter power  77  14
spectrum power 1 power 14
#===========
```

parameter number-must be unique

parameter/spectrum size in 2^n channels

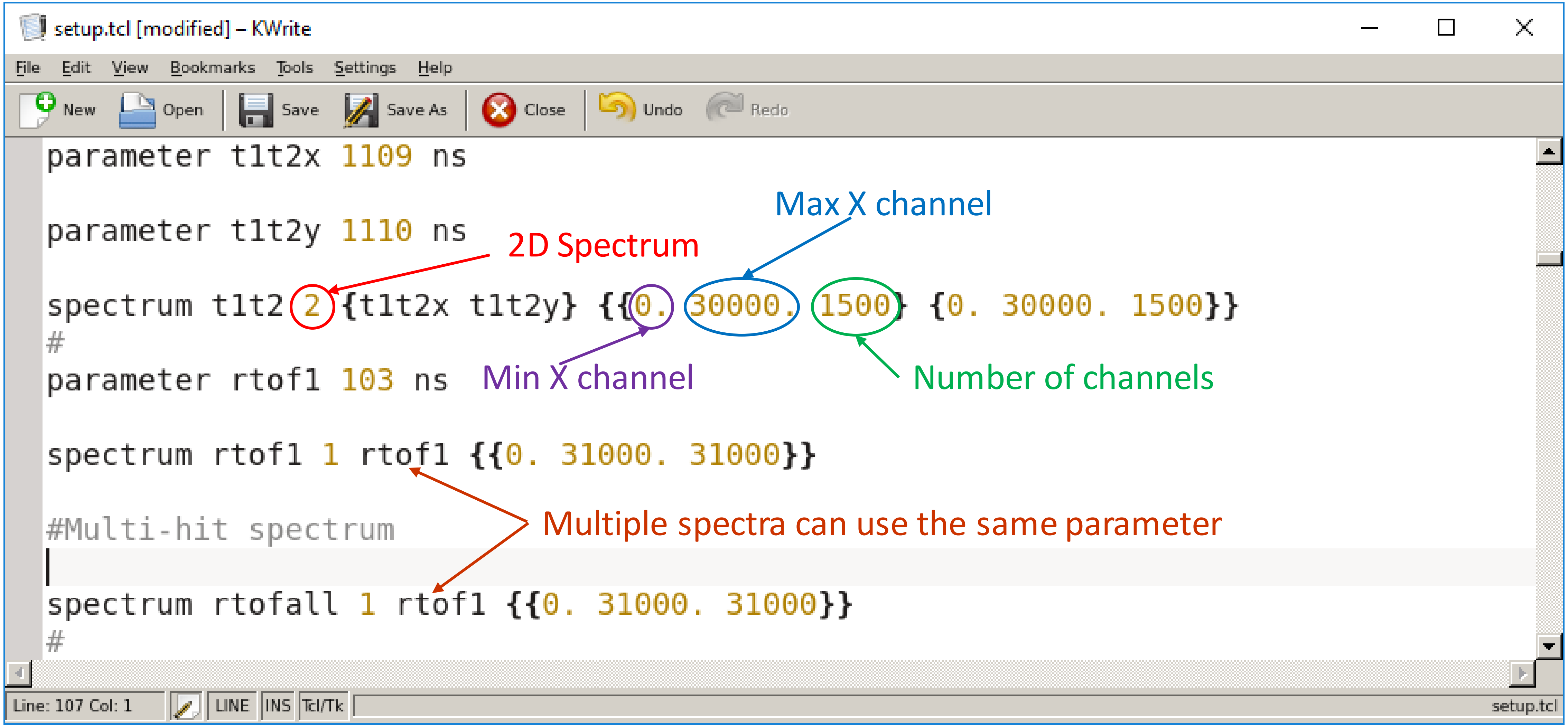
parameter name

spectrum name

1D spectrum

Line: 15 Col: 30 LINE INS Tcl/Tk setup.tcl

setup.tcl, continued



setup.tcl, continued

setup.tcl [modified] – KWrite

```
set baseparam 1060

set basename "tofion"

set paramtlist [list]

for {set i 0} {$i < 8} {incr i} {

 set name [format "$basename.%02d" $i]

 parameter $name $baseparam ns

 lappend paramtlist $name

 incr baseparam

}

spectrum tofion g1 $paramtlist {{0. 31000. 31000}}
```

Define parameter for each hit, up to 8

Single 1D spectrum based on all 8 parameters in paramtlist

Line: 359 Col: 1 LINE INS Tcl/Tk setup.tcl

setup.tcl, continued

setup.tcl [modified] – KWrite

File Edit View Bookmarks Tools Settings Help

New Open Save Save As Close Undo Redo

```
set baseparam 1040

set basenamex "rxy_gatedontimex"

set basenamey "rxy_gatedontimey"

set paramlist [list]

for {set i 0} {$i < 8} {incr i} {

 set namex [format "$basenamex.%02d" $i]

 set namey [format "$basenamey.%02d" $i]

 parameter $namex $baseparam mm

 incr baseparam

 parameter $namey $baseparam mm

 lappend paramlist $namex $namey

 incr baseparam

}

spectrum rxy_gatedontime m2 $paramlist {{-100. 100. 401} {-100. 100. 401}}
```

Define x and y parameters for each hit, up to 8

Multiply incremented 2D spectrum based on all 8 parameter pairs in paramlist

Line: 322 Col: 1 LINE INS Tcl/Tk setup.tcl

setup.tcl, continued

```
sbind -all ;#plotting all the spectra

# Add an attach online button
proc Attach hostname {
        global tcl_platform
        set ringHelper [file join /usr/opt/daq/current bin ringselector]
        set url [join [list tcp: "" $hostname $tcl_platform(user)] /]
        puts $ringHelper
        puts $url
        attach -format ring -pipe \
            $ringHelper --source=$url --sample=PHYSICS_EVENT
        start
}
button .attach -command "Attach localhost" -text "Attach Online"
pack .attach -fill x
# Add an attach file button
proc Attachf {} {
        set typelist {
            {"Event Files" {".evt"}}
        }
        set fname [tk_getOpenFile -filetypes $typelist -initialdir ../stagearea/complete]
        attach -format ring -file $fname
}
button .attachf -command Attachf -text "Attach File"
pack .attachf -fill x

pack [source ~/ibgroup/Utility/SpecRealWrite.tcl] -fill x
pack [source ~/ibgroup/Utility/SpecIntWrite.tcl] -fill x

source userscaler.tcl
```

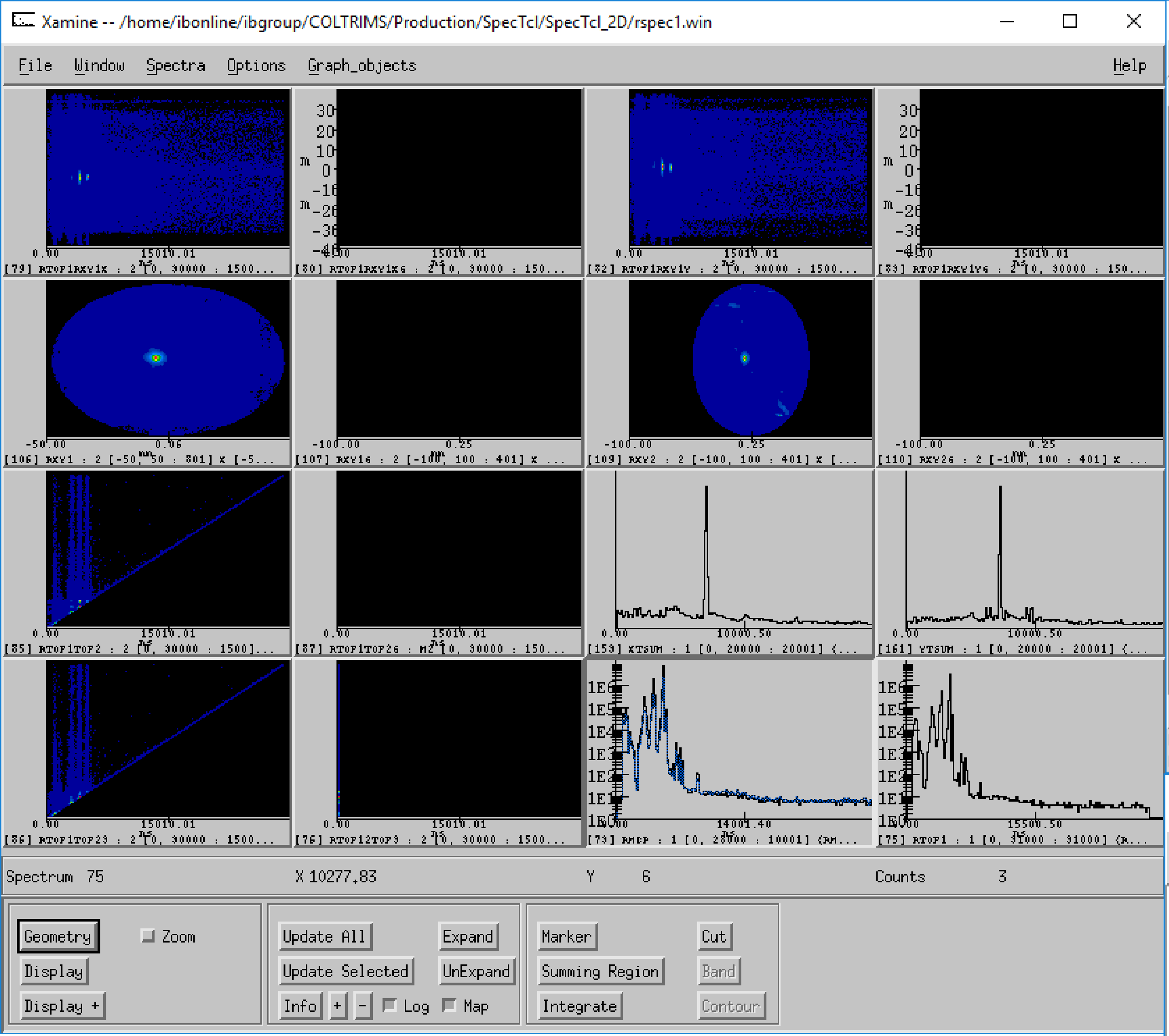
Let display know about all defined spectra

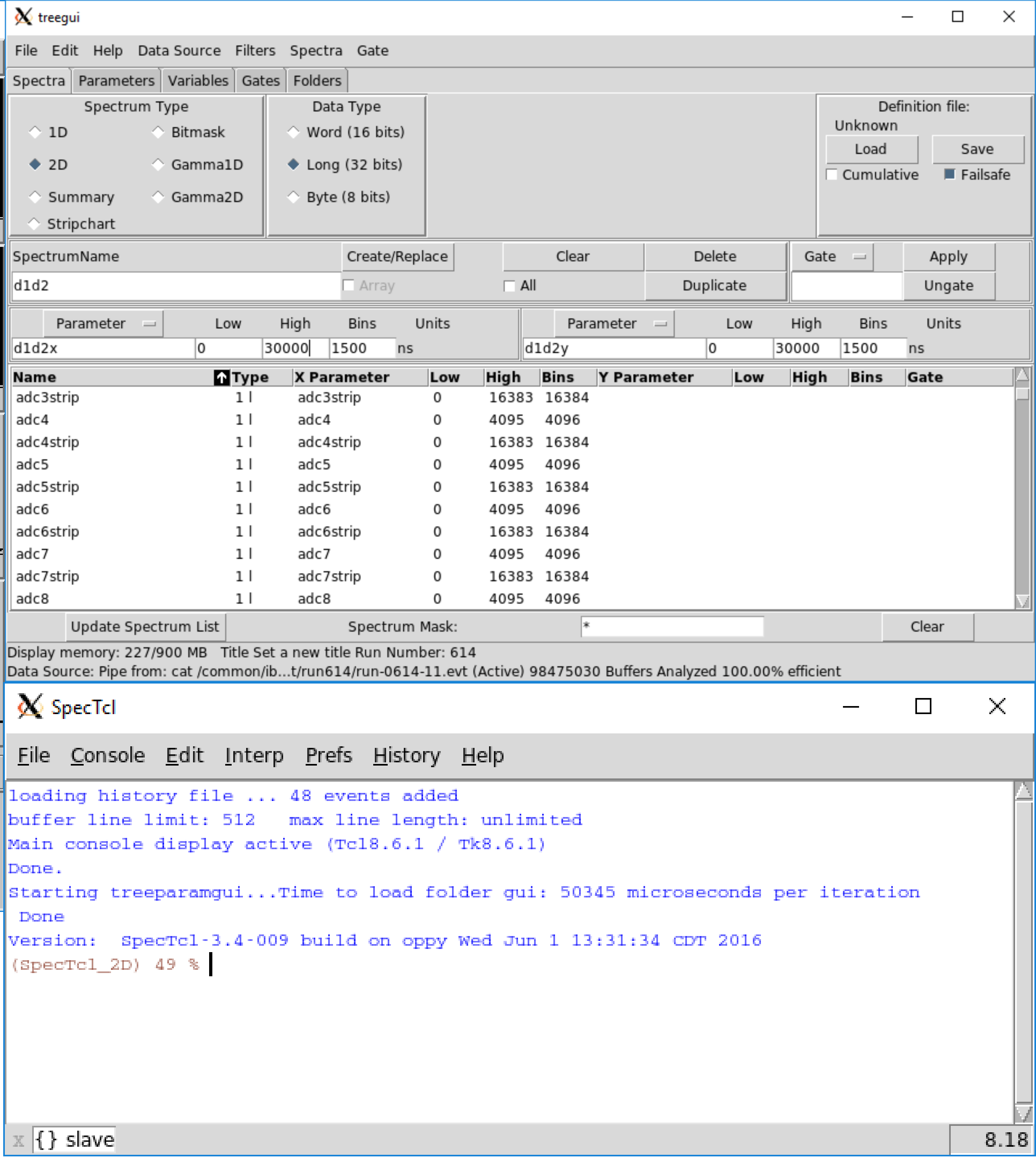
Define procedure

Define button

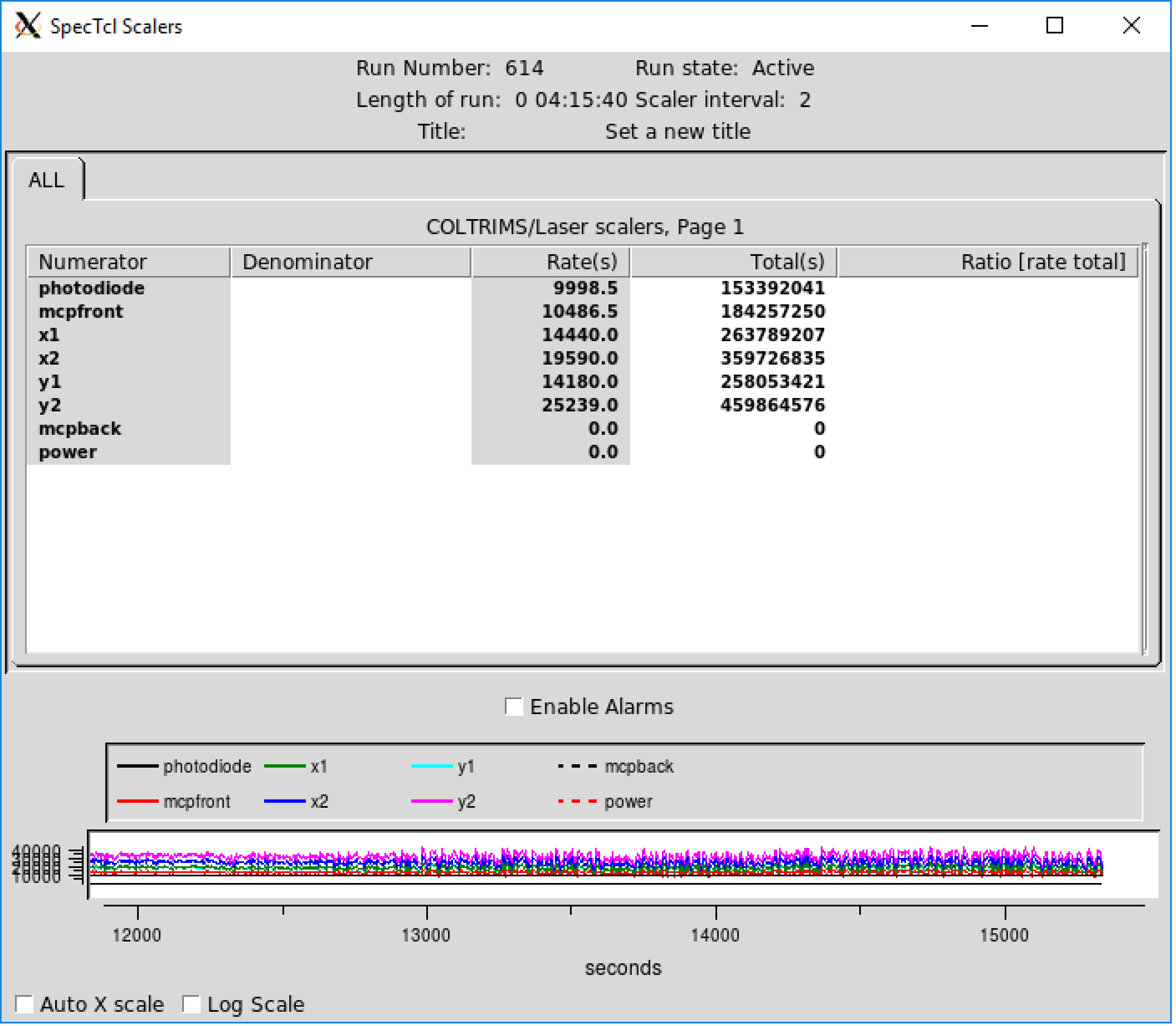
Add button to menu

Scaler update script; source is Tcl command to run

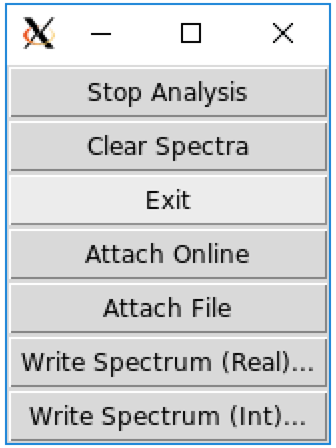




To run, type: ./SpecTcl, then in window
type: source setup.tcl
Or, type: ./SpecTcl < setup.tcl

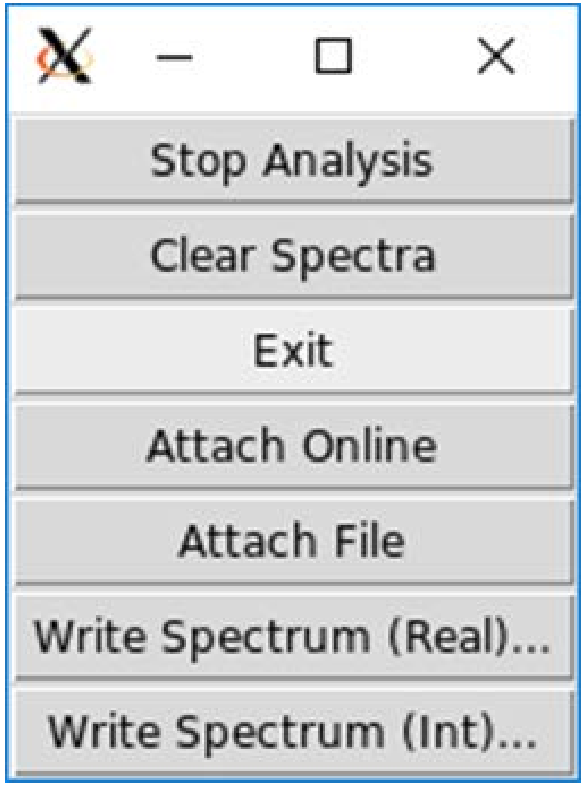


Live scaler display



Sorting control menu

Sorting Control Menu



Start/Stop Run

Clear all spectra

Exit program

Attach Online for DAQ

Select *.evt file

Write out spectrum for Origin plotting