Record of Programs

plotDirectory.py: This program defines a function that takes a directory of files in miniseed format and a list containing the names of previously checked files and plots them, without plotting the same file twice.

from obspy.core import read

import os

checkedfiles = []

def plotDirectory (directory, previousfiles):

isfirst = 0

isfirst += 1

for f in os.listdir(directory):

alreadychecked = False

for fil in previousfiles:

if f == fil:

alreadychecked = True

if not alreadychecked and f.endswith('.mseed') and isfirst == 1:

totalstream = read(directory+'/'+f)

previousfiles += f

elif not alreadychecked and f.endswith('.mseed'):

stream = read(directory+'/'+f)

totalstream += stream

previousfiles += f

totalstream.plot()

plotDirectory("mseed",[])

Seis1.py: This reads data from a serial port and prints it.

# a first script to print what is coming from the digitser

# we need to import an external library to read the data

import serial

# the original SEP serial port digitser appers as devive /dev/ttyUSB0

# the new SEP USB digitiser appears as device /dev/ttyACMO

#port\_name = '/dev/ttyACM0'

port\_name = '/dev/ttyUSB0'

port = serial.Serial(port\_name, 9600, timeout=1)

# this looprints all the data appearing on this port

while(port.isOpen()):

sample = port.readline().strip()

print sample

Seis2.py: This reads data from a serial port, records it in miniseed format and plots it to a graph-DOESN’T WORK

import serial

import numpy

from obspy.core import read,Trace,Stream,UTCDateTime

port\_name= '/dev/ttyACM0'

port = serial.Serial(port\_name, 9600, timeout=1)

datapoints = 100

data=numpy.zeros([datapoints],dtype=numpy.int32)

x=1

starttime=UTCDateTime()

print(starttime)

while(port.isOpen()) and x<datapoints:

sample = port.readline().strip()

#if sample != '':

data[x]=sample

x=x+1

timenow=UTCDateTime()

print sample,timenow

stats= {'netwrok': 'UK',

'station': 'Test',

'location': '00',

'channel': 'BHZ',

'npts': datapoints,

'sampling\_rate': '20',

'mseed' : {'dataquality' : 'D'},

'starttime': starttime}

st =Stream([Trace(data=data, header=stats)])

st.write('test.mseed',format='MSEED',encoding='INT32',reclen=512)

st.plot()

Seis3.py: This program reads data from a serial port, over a period of time and records it as sample miniseed data into a queue.

import serial

import numpy

from obspy.core import read,Trace,Stream,UTCDateTime

#import hashlib

import Queue

from threading import Thread

import time

#serial input spec

port\_name='/dev/ttyACM0'

port = serial.Serial(port\_name, 9600, timeout=1)

#array of zeros to write data into

#block\_length=0

#iterator for writing files

block\_id=1

q = Queue.Queue()

#this is the thread

def save\_data():

#it wait as there won't be anything to save in the first 5 seconds

time.sleep(5)

global block\_id

while True:

#'if' not essential but wil allow waiting to save processing

if not q.empty():

to\_save = q.get()

#write block with id from iterator

to\_save.write('mseed/PHYS' + str(block\_id) + '.mseed',format='MSEED')

block\_id=block\_id+1

q.task\_done()

else:

print 'nothing to save...'

#to save processing bit

time.sleep(5)

def read\_data(block\_length):

starttime=UTCDateTime()

x=1

data=numpy.zeros([block\_length],dtype=numpy.int16)

while (port.isOpen()) and x<block\_length:

#loop continues for block size

sample = port.readline().strip()

data[x]=sample

x=x+1

#'timenow' not essential and isn't stored

timenow=UTCDateTime()

print sample,timenow

stats = {'network': 'UK', 'station': 'PHYS', 'location': '00',

'channel': 'BHZ', 'npts': block\_length, 'sampling\_rate': 20,

'mseed': {'dataquality': 'D'},'starttime': starttime}

#create strem of data and queue it

st =Stream([Trace(data=data, header=stats)])

q.put(st)

for x in range(1):

worker = Thread(target=save\_data)

#worker.Daemon = True

worker.start()

for x in range(50):

read\_data(128)

Seis4.py: This does the same thing as seis2 but uses a jitter to record the time differences between samples to see if the digitiser is sending information at the correct rate.

import serial

import numpy

from obspy.core import read,Trace,Stream,UTCDateTime

#import hashlib

import Queue

from threading import Thread

import time

#serial input spec

port\_name='/dev/ttyACM0'

port = serial.Serial(port\_name, 9600, timeout=1)

#array of zeros to write data into

#block\_length=0

#iterator for writing files

sample\_block\_id=1

jitter\_block\_id=1

samplequeue = Queue.Queue()

jitterqueue = Queue.Queue()

#this is the thread

def save\_data\_sample():

#it wait as there won't be anything to save in the first 5 seconds

time.sleep(5)

global sample\_block\_id

while True:

#'if' not essential but wil allow waiting to save processing

if not samplequeue.empty():

to\_save = samplequeue.get()

#write block with id from iterator

to\_save.write('mseed/PHYS' + str(sample\_block\_id) + '.mseed',format='MSEED')

sample\_block\_id=sample\_block\_id+1

samplequeue.task\_done()

else:

print 'nothing to save...'

#to save processing bit

time.sleep(5)

def save\_data\_jitter():

#it wait as there won't be anything to save in the first 5 seconds

time.sleep(5)

global jitter\_block\_id

while True:

#'if' not essential but wil allow waiting to save processing

if not jitterqueue.empty():

to\_save = jitterqueue.get()

#write block with id from iterator

to\_save.write('mseed/JTR' + str(jitter\_block\_id) + '.mseed',format='MSEED')

jitter\_block\_id=jitter\_block\_id+1

jitterqueue.task\_done()

else:

print 'nothing to save...'

#to save processing bit

time.sleep(5)

def read\_data(block\_length):

starttime=UTCDateTime()

x=1

data=numpy.zeros([block\_length],dtype=numpy.int16)

jitter=numpy.zeros([block\_length],dtype=numpy.int16)

firsttime=True

totaltime=0

lastsample=UTCDateTime()

while (port.isOpen()) and x<block\_length:

#loop continues for block size

sample = port.readline().strip()

data[x]=sample

timenow=UTCDateTime()

sample\_time=timenow-lastsample

jitter[x]=sample\_time

totaltime=totaltime+sample\_time

lastsample=timenow

x=x+1

print sample,timenow

avg\_samplingrate=totaltime/block\_length

print avg\_samplingrate

stats = {'network': 'UK', 'station': 'PHYS', 'location': '00',

'channel': 'BHZ', 'npts': block\_length, 'sampling\_rate': avg\_samplingrate,

'mseed': {'dataquality': 'D'},'starttime': starttime}

#create strem of data and queue it

st =Stream([Trace(data=data, header=stats)])

jt =Stream([Trace(data=jitter)])

samplequeue.put(st)

jitterqueue.put(jt)

for x in range(1):

worker\_sample = Thread(target=save\_data\_sample)

worker\_jitter = Thread(target=save\_data\_jitter)

worker\_sample.start()

worker\_jitter.start()

for x in range(5):

read\_data(32)

Seis5.py: This program sets up a thread to save data from a queueto some files and then puts data from the digitiser into the queue for the thread to read from.

import serial

import numpy

from obspy.core import read,Trace,Stream,UTCDateTime

import Queue

from threading import Thread

import time

#serial input spec

port\_name='/dev/ttyACM0'

port = serial.Serial(port\_name, 9600, timeout=1)

#this is how after how many samples a block is saved

block\_length=10

#iterator for writing files

sample\_block\_id=1

jitter\_block\_id=1

queue = Queue.Queue()

#this is the thread

def save\_data():

global block\_id

while True:

if queue.qsize()>=block\_length:

data=numpy.zeros([block\_length],dtype=numpy.int16)

jitter=numpy.zeros([block\_length],dtype=numpy.int16)

for x in range (block\_length):

packet = queue.get()

data[x] = packet[1]

jitter[x] = packet[2]

totaltime=totaltime+packet[2]

queue.task\_done()

avg\_samplingrate=totaltime/block\_length

#start time is not proper at the mo

stats = {'network': 'UK', 'station': 'PHYS', 'location': '00',

'channel': 'BHZ', 'npts': block\_length, 'sampling\_rate': avg\_samplingrate,

'mseed': {'dataquality': 'D'},'starttime': starttime}

st =Stream([Trace(data=data, header=stats)])

jt =Stream([Trace(data=jitter)])

#write block with id from iterator

to\_save.write('mseed/PHYS' + str(block\_id) + '.mseed',format='MSEED')

to\_save.write('mseed/JTR' + str(block\_id) + '.mseed',format='MSEED')

block\_id=block\_id+1

def read\_data():

starttime=UTCDateTime()

packet=[]

firsttime=True

totaltime=0

lastsample=starttime

while port.isOpen():

#loop continues indefinately

sample = port.readline().strip()

timenow=UTCDateTime()

packet[0]=sample

sample\_time=timenow-lastsample

packet[1]=sample\_time

lastsample=timenow

#print sample,timenow

queue.put(packet)

for x in range(1):

worker\_sample = Thread(target=save\_data)

worker\_sample.start()

for x in range(5):

read\_data(32)

Questions:

What’s a jitter?

The jitter is used to record the time differences between the samples to make sure the sample rate is correct

What’s a Thread?

A Thread is a separate line of code being run that can gather data whilst the main code is doing something else, like submitting data from an array to a file.

What’s a Queue?

A Queue is the computational structure that allows two threads to be running at the same time and submitting data to the same file.