McIDAS-X Scripting in Python

McIDAS Training Workshop

Madison, WI
June 11, 2015

McIDAS-X can be used interactively using the McIDAS-X text window, or scripts can be written to run McIDAS-X commands. These scripts can take several forms including McBASI scripts, batch files, or shell scripts. This workshop will introduce you to using McIDAS-X commands in a Python script.

This workshop assumes that you have some knowledge of McIDAS-X commands and basic Python syntax. This workshop assumes that both Python 2.6 and McIDAS-X 2015.1 are installed.

Why Run McIDAS-X in a Python Environment?

The advantages of running McIDAS-X in a Python environment include but are not limited to:

- Setting up the "mcenv" environment is simpler and less kludgy.
- Users can take advantage of Python's superior text handling capabilities.
- Users can take advantage of Python's superior date/time functionality.
- Python has many libraries for doing math, image manipulation and other data transformations.
- Python is more like a programming language than other traditional McIDAS scripting languages.

Setting up the Environment

1. Open a terminal window (***Applications menu -> System Tools -> Terminal***).

2. Download and install the mcidasx-python module using the following commands:

   wget ftp://ftp.ssec.wisc.edu/pub/mug/mug_meeting/2015/python/mcidasx-python_0.4dev.tar.gz
   pip install --user mcidasx-python_0.4dev.tar.gz

   Alternative installation methods:

   easy_install mcidasx-python_0.4dev.tar.gz

   # or

tar zxvf mcidasx-python_0.4dev.tar.gz

cd mcidasx
python setup.py install

How mcidasx-python Works

The Python 'subprocess' module is used to spawn an instance of the "mcenv shell" as a background process. "mcenv" shell commands are run in that mcenv session using Python functions, with any command line parameters passed as a single string. For example:

mcenv.logon('DEMO 1234')
mcenv.dataloc('ADD DATASET SERVER.DOMAIN')

Neither **logon()** nor **dataloc()** are explicitly defined functions. When an implicit function **mccmd('arg1 arg2 arg3')** is called, the mcenv instance searches the PATH environment variable for a **mccmd.k** McIDAS command/program (which corresponds to the "MCCMD" McIDAS-X command), and then runs **mccmd.k arg1 arg2 arg3** in the mcenv shell subprocess.

Syntax Rules and Examples

1. To use a Python module in a Python program/script, the module must be "imported":

   import mcidasx

2. To begin using the mcidasx module's mcenv "session", create an instance of the mcenv() object and assign it to a local variable ("mc" in this example):

   mc = mcidasx.mcidas.mcenv()

3. The **-f** (frame size), **-i** (image colors), and **-g** (graphics colors) mcenv options can be passed as arguments to the mcenv() object's instantiation:

   mc = mcidasx.mcidas.mcenv(f=['3@1000x2000', '4@500x500'], i=150, g=16)

The argument passed to **f=** can be either a list of strings (above), or just an individual string:

```
mc = mcidasx.mcidas.mcenv(f='10@480x640'])
```

4. To run the mcenv command **logon.k DEMO 1234** (equivalent to **LOGON DEMO 1234** in McIDAS-X), call the **logon()** method of our mcenv() instance "mc", passing the entire set of parameters ("arguments and keywords") "DEMO 1234" as a single string:

```
mc.logon('DEMO 1234')
```

Oddities

The mcenv executable must be found in the **PATH** environment variable, otherwise the mcenv() instantiation will fail.

Existing **PATH** and **MCPATH** environment variables may be sufficient for some uses, but defining these explicitly within a script may be desirable:

```
import os
os.environ['PATH'] = '/path/to/mcidas/dir/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '/path/to/project/data/dir:/path/to/mcidas/data'
```

McIDAS-X and mcenv generally write files to the first writeable path in **MCPATH**, although certain situations may arise where this does not occur. This behavior is maintained in mcidasx-python.

Quotation marks (" and ') are not handled well when passed to the mcenv shell subprocess. Curly brackets should be used for comments in DSSERVE commands, for example:

```
mc.dsserve("ADD A/A AREA 1 9999 {comment}")
```

Stdout, Stderr, and Return Codes

When a mcenv command is run, a named tuple containing values for “stdout”, “stderr”, and “retcode” is returned. It is not necessary to capture this tuple unless one of these values is needed.

For example, we might want to add a new remote dataset using **dataloc()**, and then print the output of an **imglist()** call if the **dataloc()** command was successful:

```
dataloc_result = mc.dataloc('ADD GROUP server.domain')
if dataloc_result.retcode == 0:
   imglist_result = mc.imglist('GROUP/DESCRIPTOR FORM=ALL')
   print imglist_result.stdout
```

Some commands might not produce meaningful output, and thus there is no need to capture the output:

```
mc.logon('DEMO 1234')
mc.eg('1')
```

IMGLIST Example

The following is a simple example of the use of the command **IMGLIST**. This script can be found in *<local-path>*/**Data/mcidasx/python_examples/imglist_example.py**.

```
#!/usr/bin/env python
import mcidasx
import os

os.environ['PATH'] = '/home/mcidas/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']

mcenv = mcidasx.mcidas.mcenv()
mcenv.logon('DEMO 1234')
mcenv.dataloc('ADD BLIZZARD arcserv2.ssec.wisc.edu')
result = mcenv.imglist('BLIZZARD/IMAGES')
```

```
print result.stdout
print result.stderr
print result.retcode
```

In this example **MCPATH** is still set as it is in other McIDAS-X scripts. Initializing the McIDAS environment is done differently than in other scripts. Rather than starting a mcenv subshell, and then running commands in that subshell, the McIDAS environment is started with the command:

```
mcenv = mcidasx.mcidas.mcenv()
```

Also note that standard out is captured in the variable result and needs to be explicitly written to standard out.

The next example is a slightly more advanced version of the previous **IMGLIST** example that takes advantage of Python text handling and date manipulation capabilities. This script can be found in *<local-path>*/**Data/mcidasx/python_examples/imglist_advanced.py**.

```
#!/usr/bin/env python
import datetime
import mcidasx
import os

user = 'DEMO'
proj = 1234
group = 'BLIZZARD'
descriptor = 'IMAGES'
server = 'arcserv2.ssec.wisc.edu'

os.environ['PATH'] = '/home/mcidas/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']

mcenv = mcidasx.mcidas.mcenv()
mcenv.logon('%s %d' % (user, proj))
mcenv.dataloc('ADD %s %s' % (group, server))
img_date = datetime.date(1993, 3, 13)
result = mcenv.imglist('%s/%s DAY=%s TIME=%s FORM=ALL' % (group, descriptor, img_date.strftime('%y%j'), '12 18'))

print result.stdout
```

Exercise 1: Write a short Python script that displays data in a background McIDAS-X window and saves the image as a GIF image.

- Please use dataset **BLIZZARD/GE-IR-4K** on **ARCSERV2.SSEC.WISC.EDU**.
- Please use logon **DEMO** and project number **1234**.
- An example solution is available on page 8 as well as in the *<local-path>***/Data/mcidasx/python_examples/bash_vs_python.py** script. However, before checking the solution, it is recommended that you try to complete the exercise on your own.
- Hint: here is a bash script that does this:

```
#!/bin/bash
PATH=$PATH:/home/mcidas/bin
MCPATH=$HOME/mcidas/data:/home/mcidas/data
export PATH MCPATH

mcenv << 'EOF'
logon.k DEMO 1234
dataloc.k ADD BLIZZARD ARCSERV2.SSEC.WISC.EDU
imgdisp.k BLIZZARD/IMAGES.-1 1 LAT=38 78
frmsave.k 1 storm_ir_bash.gif
exit 0
EOF

exit
```

Advanced Example

Now for a more advanced example. In this example, we will **IMGCOPY** an archived Meteosat-9 image to a local netcdf dataset, then use netCDF4 and numpy to perform a Normalized Difference Vegetation Index (NDVI) calculation, display the NDVI imagery using matplotlib.pyplot, and finally save the output to a PNG file. This script can be found in *<local-path>***/Data/mcidasx/python_examples/ndvi.py**.

```
#!/usr/bin/env python
import matplotlib.pyplot as pyplot
import mcidasx
import netCDF4
import numpy
import os
import sys

mcidas_dir = os.path.expanduser('~mcidas')

path = [os.environ['PATH'],
```

```
       os.path.join(mcidas_dir, 'bin')]

mcpath = [os.path.dirname(__file__),
          os.path.join(os.environ['HOME'], 'mcidas/data'),
          os.path.join(mcidas_dir, 'data')]

os.environ['PATH'] = ':'.join(path)
os.environ['MCPATH'] = ':'.join(mcpath)

mcenv = mcidasx.mcidas.mcenv()
mcenv.logon('DEMO 1234')

result1 = mcenv.dataloc('ADD MUG2015 arcserv2.ssec.wisc.edu')
if result1.retcode != 0:
    sys.exit(result1.stdout)

mcenv.dsserve('ADD N/A NCDF 1 9999 TYPE=IMAGE')

msg_ndvi_bands = [1, 2]

imgcopy_string = 'MUG2015/NDVI N/A.{band} SIZE=SAME BAND={band} MAG=-8 DAY=2011/08/31 TIME=12 UNIT=REFL'
for band in msg_ndvi_bands:
    imgcopy_result = mcenv.imgcopy(imgcopy_string.format(band=band))
    print imgcopy_result.stdout

try:
    # open the NetCDF files
    redBand = netCDF4.Dataset('A0001.nc', 'r')
    nirBand = netCDF4.Dataset('A0002.nc', 'r')

    # read data into numpy arrays
    redData = numpy.array(redBand.variables['data'][0])
    nirData = numpy.array(nirBand.variables['data'][0])

    check = numpy.logical_and(redData != 0, nirData != 0)
    ndvi = numpy.where(check, (nirData - redData) / (nirData + redData), 0)

    pyplot.imshow(ndvi, cmap=pyplot.get_cmap('PRGn'), vmin=-1, vmax=1)
    pyplot.savefig('ndvi.png')
    pyplot.show()

except:
    sys.exit('An error occurred.')
```

Other Python Modules

These Python modules may offer interesting possibilities in combination with McIDAS-X:

- numpy - package for scientific computing
- netCDF4 - python/numpy interface to netCDF
- basemap - library for plotting 2D data on maps
- cartopy - cartographic tools
- gdal - Geospatial Data Abstraction Library bindings

Integrating McIDAS-X into an existing script or workflow involving any of these modules is now very straight-forward.

Disclaimers and Afterthoughts

This package is **NOT** supported by MUG, McIDAS-X, or any group within SSEC. The software is currently used internally by the SSEC Data Center for experimental use, with operational usage planned for the near future. Hopefully this workshop has inspired you to use McIDAS-X and Python scripting in creative new ways!

Exercise 1: A Python Solution

```
#!/usr/bin/env python
import mcidasx
import os

os.environ['PATH']   = "%s:/home/mcidas/bin" % os.environ['PATH']
os.environ['MCPATH'] = "%s/mcidas/data:/home/mcidas/data" % os.environ['HOME']

mc = mcidasx.mcidas.mcenv()
mc.logon('DEMO 1234')
mc.dataloc('ADD BLIZZARD ARCSERV2.SSEC.WISC.EDU')
mc.imgdisp('BLIZZARD/IMAGES.-1 1 LAT= LAT=38 78’)
frmsave_result = mc.frmsave('1 storm_python.gif')

print frmsave_result.stdout
```