McIDAS-X Tutorial

Scripting in Python

updated February 2017 (software version 2017.1)

Introduction

The McIDAS-X Python package allows users to create python scripts to run McIDAS-X commands. The advantages of running McIDAS-X in a Python environment include but are not limited to:

- Setting up the "mcenv" environment is simpler and removes the shell scripting concepts of EOF and exit 0.
- 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.

The package is part of McIDAS-XRD so certain McIDAS-X commands are not compatible with Python syntax. Specifically, any McIDAS-X command using single quote marks cannot be used and any command using a double quote must use curly brackets. Additionally, the package is only compatible with Linux and OS X operating systems and at least McIDAS-X 2017.1.

Installing McIDAS-X Python

Assuming a standard installation of McIDAS-X 2017.1, where McIDAS was installed as user *mcidas* and is being run from the user account that is set up for McIDAS-X access, run the following commands:

```
cd $HOME
mcxpyinstall
```

mcxpyinstall is the installation script to set up the Python "subprocess" module.

How McIDAS-X Python Works

The Python "subprocess" module is used to spawn an instance of the "mcenv shell" as a background process. McIDAS commands are started via the "mcenv" session using Python functions. Command line parameters passed as a single string. For example:

```
mcenv.logon('DEMO 1234')
mcenv.dataloc('ADD BLIZZARD GEOARC.SSEC.WISC.EDU')
mcenv.dsinfo('I BLIZZARD')
```

Neither **dataloc()** nor **dsinfo()** 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

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

```
import mcidasx
```

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.mcenv()
```

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.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.mcenv(f='10@480x640')
```

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'
```

The following is a simple example of the use of the command **IMGLIST**:

```
#!/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']
mc = mcidasx.mcenv()
logonOut = mc.logon('DEMO 1234')
datalocOut = mc.dataloc('ADD BLIZZARD GEOARC.SSEC.WISC.EDU')
dsinfoOut = mc.dsinfo('I BLIZZARD')
imglistOut = mc.imglist('BLIZZARD/IMAGES.ALL')
print imglistOut.stdout
print imglistOut.stderr
print imglistOut.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:

```
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']
mc = mcidasx.mcenv()
```

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.

Stdout, Stderr, and Return Codes

When a mcenv command is run, a named tuple containing values for “stdout”, “stderr”, and “retcode” are 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()** and check if the command finished successfully:

```
mc.logon('DEMO 1234')
dataloc_result = mc.dataloc('ADD BLIZZARD GEOARC.SSEC.WISC.EDU')
if dataloc_result.retcode == 0:
    imglist_result = mc.imglist('BLIZZARD/IMAGES FORM=ALL')
    print imglist_result.stdout
    print imglist_result.retcode
```