McIDAS-X Shell Scripting

McIDAS Training Workshop

Madison, WI

November 14th, 2016

McIDAS-X commands can be run interactively using the McIDAS-X text window or using Unix shell scripts to create a background McIDAS-X environment. This workshop will introduce you to setting up a McIDAS-X environment and running commands using a UNIX shell script. Further documentation can be found in the McIDAS User's Guide: http://www.ssec.wisc.edu/mcidas/doc/users_guide/current/app_h-1.html

Why Run McIDAS-X using UNIX Shell Scripts?

McIDAS-X is a powerful tool that can process and display data. Users may want to do this processing outside of an interactive session.

- The concepts for setting up a "mcenv" environment are similar to that of an interactive session.
    - PATH
    - MCPATH
    - Other environment variables
        - ADDETIMEOUT
        - MCCOMPRESS
- mcenv is core supported.
- mcenv works well with cron.
- mcenv is ideal for saving graphics to a web page

Basic Concepts

As with an interactive session, the mcenv environment needs to be defined. Arguments are passed to mcenv using standard Unix command line conventions. The following are valid arguments that can be used by mcenv:

mcenv –f *framespec* **-e** *bytes* **-g** *graphics_levels* **-i** *image_levels prog*

-f framespec	specifies a set of frames to include in the McIDAS environment (default=one, 480x640 frame)
-e bytes	specifies a memory pool for the frames
-g number	specifies the number of graphics color levels (default=8)
-i number	specifies the number of image display levels (default=48)
prog	specifies a program, script or command to run; see Using Unix Conventions for the correct format to enter McIDAS commands

Example:

```
mcenv -i 220 -g 32 -f 3@1200x1800
```

Defining Unix Specific Environment

When a user runs a McIDAS-X session interactively, environmental variables are usually defined at login. When running mcenv using a shell script these environmental variables must also be defined.

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

To avoid conflicts between the user’s mcidas/data directory, a good practice is to specify a unique directory to write the McIDAS files. An additional good practice is to direct the output to a log file. Make sure all the directories are created prior to running the scripts!

```
#!/bin/bash

export PROJECT=workshop
export PATH=$DATA_DIR:/home/mcidas/bin:$PATH
export MCPATH=$HOME/$PROJECT/mcidas/data:/home/mcidas/data
export DATA_DIR=$HOME/$PROJECT/mcidas/data


# - This is just for the workshop!
rm -rf $HOME/$PROJECT

mkdir $HOME/$PROJECT
mkdir $HOME/$PROJECT/logs
mkdir $HOME/$PROJECT/mcidas
mkdir $HOME/$PROJECT/mcidas/data

LOG=$HOME/$PROJECT/logs/workshop.log

exec >$LOG
exec 2>>$LOG

cd $DATA_DIR
```

In the above example a DATA_DIR was also defined. This is convenient for accessing files created from the mcenv session later in the script.

Running a mcenv Session

Now that the Unix environment is created, the mcenv session is ready. Here's a simple example that defines the dataset TORNADO to be located on the server PAPPY.SSEC.WISC.EDU

```
#!/bin/bash

export PROJECT=workshop
export PATH=$DATA_DIR:/home/mcidas/bin:$PATH
export MCPATH=$HOME/$PROJECT/mcidas/data:/home/mcidas/data
export DATA_DIR=$HOME/$PROJECT/mcidas/data


# - This is just for the workshop!
rm -rf $HOME/$PROJECT

mkdir $HOME/$PROJECT
mkdir $HOME/$PROJECT/logs
mkdir $HOME/$PROJECT/mcidas
mkdir $HOME/$PROJECT/mcidas/data

LOG=$HOME/$PROJECT/logs/workshop.log

exec >$LOG
exec 2>>$LOG

cd $DATA_DIR



mcenv -i 220 -g 32 -f 3@480x640 << 'EOF'
   dataloc.k ADD TORNADO PAPPY.SSEC.WISC.EDU
   imglist.k TORNADO/GOES13-IR.ALL
EOF
exit 0
```

The `<< 'EOF'` instructs the mcenv to run all commands until it encounters `EOF`. List the contents of `workshop.log` to see the output from the DATALOC and IMGLIST commands.

Exercise 1

Edit the script *<local-path>*/**Data/mcidasx/ python_examples/example-mcenv.bash** that that includes a frame size of 1000 lines and 1000 elements, uses 240 images levels and 8 graphics levels. The script should display the 23:45Z image from Julian day 2011142. Center the image on Springfield, MO (KSGF). Save the image as $USER-joplin.gif. Check the log file for any errors. Send the image to SSEC's ftp site:

```
cd $HOME/workshop/mcidas/data
ftp ftp.ssec.wisc.edu
user anonymous
password e-mail address
cd pub/incoming
bin
mput *joplin.gif
```

Use a browser to check the image. An example script is located at the end of this document, but we encourage you try to write the script before viewing the solution.

Oddities

McIDAS-X can use characters which may cause confusion within the Unix environment. Generally, the problem can be rectified by escaping the character using a backslash (\). Use pairs of double (") and single quotes (') to escape groups of characters. Double quotes escape all characters except other double quotes and dollar signs ($). Single quotes escape all characters except other single quotes. The characters below have special meaning in Unix and require an escape character when entered as part of a McIDAS-X command.

- ampersand (&)
- semicolon (;)
- parentheses ( )
- pipe (|)
- less than and greater than symbols (< >)
- unpaired double quotation marks (")
- unpaired single quotation marks (')
- backslash (\)
- pound sign (#)
- asterisk (*)
- question mark (?)
- dollar sign ($)

Solution to Exercise 1

```
export PROJECT=workshop
export DATA_DIR=/home/username/$PROJECT/mcidas/data
export PATH=$DATA_DIR:/home/mcidas/bin:$PATH
export MCPATH=/home/username/$PROJECT/mcidas/data:/home/mcidas/data

LOG=/home/username/$PROJECT/logs/workshop.log

exec >$LOG
exec 2>>$LOG

cd $DATA_DIR

mcenv -i 240 -g 8 -f 1@1000x1000 << 'EOF'
   dataloc.k ADD TORNADO PAPPY.SSEC.WISC.EDU
   imgdisp.k TORNADO/GOES13-IR TIME=23:45 DAY=2011142 STA=KSGF
   frmsave.k 1 $HOME/$PROJECT/mcidas/data/$USER-joplin.gif
EOF
exit 0
```

McIDAS-X Scripting in Python

McIDAS Training Workshop
Madison, WI
November 14th, 2016

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. McIDAS-X must be installed prior to taking advantage of python scripts. This workshop utilizes Centos 7.0 with Python 2.6 and McIDAS-X 2016.2.

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 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.

Setting up the Environment

1. Open up your ssh application and login to the host dcprod-dev.

2. Download the mcidasx-python module using **wget** command.

```
wget ftp://ftp.ssec.wisc.edu/pub/mug/mug_meeting/2016/python/mcidasx-python_0.6dev.tar.gz
```

3. Install the mcidasx-python module using the following commands:

```
tar zxvf mcidasx-python_0.6dev.tar.gz


cd mcidasx
python setup.py install --user
```

or

```
easy_install --user mcidasx-python_0.6dev.tar.gz
pip install --user mcidasx-python_0.6dev.tar.gz
```

Later in this workshop we will be using both netCDF and matplotlib libraries. Some of these libraries do not come standard with the system installed version of Python. Both libraries are distributed with miniconda or anaconda. Enter the following to create the correct python environment:

```
module load miniconda/2.7-base
```

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('WKSP 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.

Note: logon **WKSP** and project number **1234** will only work for these servers during the workshop. You will need to change these in your own scripts when the workshop is complete.

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

   ```
   mc.logon('WKSP 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.

Double quotation marks (") 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}”)
```

Single quote marks (') do not currently work.

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('WKSP 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/workshop/mcidas/data:/home/mcidas/data' % os.environ['HOME']

        mcenv = mcidasx.mcidas.mcenv()
        mcenv.logon('WKSP 1234')
        mcenv.dataloc('ADD EASTL eastl.ssec.wisc.edu')
        result = mcenv.imglist('EASTL/ALL')

        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 = 'WKSP'   # only necessary for restricted datasets
```

```
proj = 1234     # only necessary for restricted datasets
group = 'EASTL'
descriptor = 'ALL'
server = 'eastl.ssec.wisc.edu'

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

mc = mcidasx.mcidas.mcenv()
mc.logon('%s %d' % (user, proj))
mc.dataloc('ADD %s %s' % (group, server))
image_date = datetime.date(2015, 6, 4)
result = mc.imglist('%s/%s.ALL DAY=%s TIME=%s FORM=ALL' % (group, descriptor, image_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 **EASTL/CONUS** on **EASTL.SSEC.WISC.EDU**.
- Please use logon **WKSP** and project number **1234**.
- An example solution is available on page 14 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/workshop/mcidas/data:/home/mcidas/data
export PATH MCPATH

mcenv << 'EOF'
logon.k WKSP 1234
dataloc.k ADD EASTL  EASTL.SSEC.WISC.EDU
imgdisp.k EASTL/CONUS.-1 1 BAND=1 LAT=43 90
frmsave.k 1 wisconsin_vis_bash.gif
exit 0
EOF

exit
```

Exercise 2: Run the python script <*local-path*>**/Data/mcidasx/python_examples/sfclist.py**. Update the script to print out the dew point depression for 0Z. An example solution is available on page 15 as well as <*local-path*>**/Data/mcidasx/python_examples/dewpt.py**

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'], 'workshop/mcidas/data'),
         os.path.join(mcidas_dir, 'data')]

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

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

result1 = mcenv.dataloc('ADD AMET09 geoarc.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 = 'AMET09/FD 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/workshop/mcidas/data:/home/mcidas/data" % os.environ['HOME']

mc = mcidasx.mcidas.mcenv()
mc.logon('WKSP 1234')
mc.dataloc('ADD EASTL EASTL.SSEC.WISC.EDU')
mc.imgdisp('EASTL/CONUS.-1 1 BAND=1 LAT=43 90')
frmsave_result = mc.frmsave('1 wisconsin_vis_python.gif')

print frmsave_result.stdout
```

Exercise 2: A Python Solution

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

def main(argv):

    if len(argv) < 3:
        print ' Must specify a station and time'
        return
    else:
        station = argv[1]
        time = argv[2]

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

    mcOutput = run_mcidas(station,time)
    dewPTdepression = get_dewPT(mcOutput)
    dewPTdepression = "{:6.2f}".format(dewPTdepression)
    print ' '
    print 'Dew point depression for ' + station + ' = ' + dewPTdepression

def run_mcidas(station,time):
#
# Create McIDAS environment and run commands
#
    mcenv = mcidasx.mcidas.mcenv()
    mcenv.logon('CKMK 6999')

    datalocString='ADD RTPTSRC NOAAPORT.SSEC.WISC.EDU'
    datalocOut = mcenv.dataloc(datalocString)

    sfclistString = station + ' TIME=' + time
    sfclistOut = mcenv.sfclist(sfclistString)

#
# Break output into list
#
    sfclistOutput = mccmdout2list(sfclistOut.stdout)
    return sfclistOutput

def mccmdout2list(mcoutput):
```

```
#
# Inputs output from a McIDAS command and creates a list of lines
#
    line = ''
    count = 0
    lineList = []
    for char in mcoutput:
        if char != '\n':
             line = line + char
        else:
             lineList.append(line)
             count = count + 1
             line = ''
    return lineList

def get_dewPT(outputList):
#
# Remove header and use only hourly observation
#
    outputList.pop(0)
    outputList.pop(0)
    outputList.pop(0)

    recCount = 0
    runningTot = 0
    for line in outputList:
        record = line.split()
        if 'Number' in record[0]:
            break
        elif 'S' in record[0]:
            pass
        else:
            temperature = float(record[4])
            dewPT = float(record[5])
            print temperature, dewPT

    dewPTdepression = temperature - dewPT
    return dewPTdepression

if __name__ == "__main__":
    main(sys.argv)
```