Lab-2a: Introduction to shell-scripting Feb.-19, 2010

Step-0: In this exercise you will create, download, and manipulate a file (meta-data input) that contains a list of MODIS file names. This task requires manual interaction with NASA at

http://oceancolor.gsfc.nasa.gov/cgi/browse.pl

1. Select a satellite (e.g., MODIS-Terra and MODIS-Aqua);
2. Select "Swath containing ALL of the area of interest";
3. Select "Mission" in the time table;
4. Select a geographic bounding box (e.g., N:42 , S:37, E:130, W:115);
5. Click on List L0 and save this list as a file to your disk, this is your meta-data for now);
6. Move this file.list to a suitably created directory on your account on the muenchow.cms.udel.edu server.

Step-1: You will use segments of file.list to select and process some data files but not others in an automated fashion. For now, however, you want to know how many "image frames" (5-minutes of Modis raw data) you may have available:

1. Write, make executable, and execute a c-shell script called step-1.csh that contains
   Line-1: #! /bin/csh
   Line-2: echo "Display content of file file.list"
   Line-3: more file.list
2. comment out line-3 as it is not terribly helpful and rewrite as
   Line-3: #more file.list
   Line-4: /usr/bin/nawk 'END{print NR}' file.list
3. I just introduced you to a powerful unix scripting tool called "nawk" which processes text strings one line at a time; its executes the file /sw/bin/nawk with a text string 'END{print NR}' that is the entire program. The program is executed with the file file.list as input. The program reads one line after another storing the line number into a built-in variable called NR. The END{…} section of the program is executed only after all lines of the file file.list are read in, at which time NR is send to an output device. Adapt to these ideas by adding the following lines (one at a time, you want to study what happens why):
   Line-5: /usr/bin/nawk '$1 ~ "2000" {print}' file.list >file.list2000
   Line-6: /usr/bin/nawk 'END{print "Modis data files in 2000: ",NR} file.list2000
   Line-7: /usr/bin/nawk '$1 ~ "2010" {print}' file.list >file.list2010
   Line-8: /usr/bin/nawk 'END{print "Modis data files in 2010: ",NR}' file.list2010
   Line-9: echo "#! /bin/csh" >file1.csh
   Line-10: /usr/bin/nawk 'NR == 1 {print "command.here",substr($1,12,1)}' file.list >>file1.csh

4. The new operators ">" and ">>" are the Unix shell-scripting way to create new files. So, rather than directing output from commands to the (default) screen terminal, these redirects open new files and place the output there. The ">" will over-write any existing files in the current directory that have this name while the ">>" will append the output should that file already exist in the current directory. The above also introduced $x which refers to column-x in the input file, file.list in this case.
5. Please note that in 3. you made a file that potentially constitutes an automatically generated new executable. Please modify Line-10 to generate yourself a shell script file1.csh that would automatically download a data file from NASA's web site?
6. Please add 2 lines that make file1.csh executable (Line-11) and execute it (Line-12).