drPacs:
A Unix Pipeline Execution

Peter Teubben
ABSTRACT:

Imagine a shell script that runs a simulation or reduces data. If you have made it flexible, it will have parameters to control this process. But now imagine you have to run many many (100s, 1000s) of these scripts, and potentially have to re-run them when something in your pipeline has changed (e.g. one of the programs in your pipeline has a bug). Or make the pipeline longer.
I'll describe a generic Unix (so it works on your Mac as well) solution to this dilemma, to make this process easy.
Theory

Unix pipe:

% grep ^tshirt= mdjo10 | sort | uniq -c

% grep ^lunch= mdjo10 | awk -F= '{sum+=$2} END {print sum}'

PYTHON

>>> f = os.popen("grep ^lunch= mdjo10","r")
>>> lines=f.readlines()
>>> sum=0
>>> for line in lines:
>>>    sum = sum + int(line.split("=")[1])
>>> print sum
Theory (2)

Unix programs “P1”, “P2”, .... form the pipeline

Each program has their own set of keywords

Programs can pass information along in the pipe

\[(o_{11}, o_{12}, ...) = f_1(p_{11}, p_{12}, ...)\]

\[p_{21} = g(o_{11}, o_{12}, ...)\]

\[(o_{21}, o_{22}, ...) = f_2(p_{21}, p_{22}, ...)\]
drPacs

- d=dalton  r=roger  PACS=phase ....
- Code in CVS     ("cvs checkout drpacs")
- Uses “configure” to install
  - configure ; make ; make install
- Minimal dependencies (csh, python, sh)
Example: disk_*
(needs NEMO)

- disk_init
  - nbody=250
  - rcut=1.25
  - Qtoomre=1
- disk_run
- disk_check

Initialize disk for simulation
Run simulation
Check some results
Simple “pipe” commands

- **pipeline**: create a Pipefile (a Makefile)
- **pipepar**: get and set pipeline parameters
- **pipe**: run the pipeline
- **piperun**: run the pipeline in set of directories
- **pipesave**: save pipeline pars
- **pipesetup**: grab previously saved pipeline pars