Essential Astronomy Computing Skills
2007 STFC Summer School

- Unix/Linux basics
- Generic tools
- Scripting
- Computer Languages
  - Perl, Python, IDL
- Essential tools/links

Unix/Linux

"Two of the most famous products of Berkeley are LSD and Unix.
I don't think that this is a coincidence" - Anonymous

Unix/Linux 101

Key commands:

- man – display online help
- cd – change directory; cd .. , cd -
- chmod – changes file permissions
- cp – copies one or more files; -p option preserves file information
- df – reports amount of free disk space
- diff – compares two files
- find – finds files with given conditions
- grep – search files for particular strings
- lpr – sends files to the printer; lprm to remove file from queue
- ls – lists contents of directories; ls –lrt
- more – display files one screenful at a time
- mv – move or rename files and directories
- ps – list the processes that are running
- rm – deletes one or more files; there is not an undelete!
- sort – sorts lines of files
- tail – print the last lines of a file

TAB key for file completion and <Control>D to list options

tcsh shell (/bin/tsch/)

Defaults set in $home/.cshrc and $home/.login
e.g. PATH, LD_LIBRARY_PATH, aliases set here

In my .cshrc file I have

```
set noclobber
alias jj 'ps –ef | grep jrl'
```

Shell commands/features in /bin/tcsh
(other shells like /bin/sh similar)

cmd > file	Sends output of cmd to file
cmd >! file	As above but overwrites file
cmd >> file	Appends output to file
cmd >& file	Sends both standard output and standard error to file
cmd < file	Takes input for cmd from file
cmd1 | cmd2	Uses output from cmd1 as input to cmd2
	ls –l | sort +4n
cmd &	Runs process in the background
cmd1;cmd2	Puts two commands on the same line

Email clients

pine - text-based, basic email client
configure, file carbon copy (fcc:)
- very simple and fast; good for low bandwidth connections

Thunderbird - free, cross-platform email client from Mozilla Foundation, rapidly improving

Outlook - only available on M$ windows; should be called look-out

Evolution - outlook-like but works under linux

Web-based mailers - googlemail, fastmail, etc.

Text Editors

For plain text, not a word processor!

GNU emacs - screen editor of choice, $home/.emacs sets defaults
- concise summary of keyboard short-cuts from www.refcards.com
Turn on syntax highlighting!

ViM - **Vi** **IM**proved, multiplatform, has a steep learning curve

pico (nano) - default editor of pine
- very simple and fast; good for low bandwidth connections
- beware of tabs being removed in makefile!

Nedit - popular for the X Windows Systems, similar to text editors on M$ Windows and MACs.

Essential Generic Tools

gzip/gunzip – compress/uncompress files

ftp – transfers files to and from computers elsewhere
(mostly superseded by scp)

ssh – secure remote login programme
(can set up password-less login easily)

scp – secure remote copy

make - to maintain groups of programs

cron - for running commands at a set time

(g)awk - pattern scanning utility

tar - (tape) archive program to un/pack sets of files

rsync - provides fast incremental file transfer, great for backups

putty - free SSH client for M$ windows

Scripting

Scripting is a powerful and time-saving tool for
performing repetitive jobs,
creating data-processing pipelines,
encapsulating useful recipes, etc.

Shell scripting

A shell script is a text file containing a set of commands and constructs that perform a routine task.

May include
commands like **mv**, **rm**, **awk**,
have parameters, subroutines,/functions,
conditional parts and loops,
other scripts/programmes.

You can create and modify scripts with a text editor.

Shell Script to rename a set of files

```
#!/bin/csh
#
Kappa  # to define command fitsval
#
foreach file (d*.sdf)
   set fname = `fitsval $file:r NUMRUN`
   echo $fname
   \mv $file $fname{.sdf}
   gzip -v $fname{.sdf}
end
```

pstogif.csh (converts a postscript file to a gif file)

```
#! /bin/csh
set ncol = 256
set dpi  = 72
if ($1 == "") then
  echo usage pstogif  file.ps [angle] [scale]
  echo will create file.gif
  echo converts a .ps to a .gif with optionally specified rotation and sc
  echo employs gs and ppmtogif also if needed pnmrotate and pnmscale
  echo
  echo defaults:  resolution $dpi  dpi    max colours $ncol
  exit
endif
#
gs -sOutputFile=$1:r.ppm -sDEVICE=ppmraw -dNOPAUSE -q  -r$dpi  $1 -c quit
#
# Rotate and scale if requested
if ($2 != "") then
  if ($2 != "0") then
   echo rotating through $2 degrees
   mv $1:r.ppm $1:r.rppm
   pnmrotate $2 $1:r.rppm   >! $1:r.ppm
```

Shell script for masking and background-fitting

```
# Loop through the remaining arguments, assuming that these are NDFs.
     foreach file ($ndfs[1-])

     # Obtain the NDF's name.
        set file1=$file:r

     # Obtain the centre of the galaxy, assuming it is the pixel with the
     # largest value.
        stats $file1 > /dev/null

     # Store the maximum (centre) co-ordinates values in shell variables.
        set centre = `parget maxcoord stats`

     # Create a two-line ARD file.  The first is for the bright galaxy, ar
     # the other is the second brightest galaxy.  We assume a constant off
     # Use CALC to evaluate the expressions, as the centres are strings ar
     # might be floating point.
        echo 'ELLIPSE( '$centre[1]', '$centre[2]', 82, 44, 152 )' > $file1

        set aa = `calc exp=\"$centre[1] + 68\"`
        set bb = `calc exp=\"$centre[2] - 59\"`
```

Scripting languages (cont:)

Can write sophisticated scripts in the shells (bash, csh), e.g. pipelines for data reduction, see Starlink SC4 C-shell cookbook.

Shell scripts (e.g. csh) are useful for programming repetitive tasks and simple manipulation of files but are not suitable for general programming (e.g. no floating point arithmetic, etc).

Scripting languages, like Python or Perl, are far far better at this.

Other scripting languages include Ruby, Tcl, Jython, JRuby, …

Programming Languages in Astronomy

Rank	Language
1	Fortran, F90, F95
2	C, C++
3	Perl
4	IDL
5	Python
6	Java

Source: Paul Francis, MSSSO

Perl (Practical Extraction and Report Language)

A very popular scripting language, C-like syntax, large library of code (CPAN library) (See http://www.perl.com/)

Example Perl Script: factorial.pl

```
use strict 'vars';
die "Usage: $0 number\n" unless @ARGV == 1;

my $n = factorial($ARGV[0]);
print "$ARGV[0] factorial is $n.\n";

sub factorial {
  my ($n) = @_;
  if ($n == 0) { return 1 }
  else { return $n * factorial($n-1) }
}
```

Part of the perl package Astro::FITS::Header::CFITSIO

```
sub configure {
  my $self = shift;

  my %args = ( ReadOnly => 0, @_ );

  # itialise the inherited status to OK.
  my $status = 0;
  my $ifits;

  return $self->SUPER::configure(%args)
    if exists $args{Cards} or exists $args{Items};

  # read the args hash
  if (exists $args{fitsID}) {
     $ifits = $args{fitsID};
  } elsif (exists $args{File}) {
     $ifits = Astro::FITS::CFITSIO::open_file( $args{File},
               $args{ReadOnly} ? Astro::FITS::CFITSIO::READON
                               Astro::FITS::CFITSIO::READWRI
                                       $status );
  } else {
```

Perl (continued)

Key disadvantage: (My view)

Messy inconsistent syntax and can be cryptic hard to read someone else's code (or your own code a few months later), hard to write and maintain large programmes.

Note: All languages can be used badly!

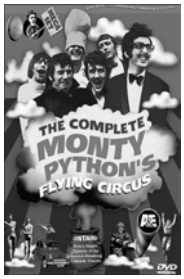
Python

Python is a fun and extremely easy-to-use programming language that combines remarkable power with very clear syntax.

Python is optimized for software quality, portability, integration, and productivity. It fosters maintainable systems that run just about everywhere (Linux, Mac OS X, Windows) and can be freely mixed with other software components.

Great on-line material at www.python.org

Python supports raising and catching exceptions.



Example Python Script

```
def addS(s):
    return s+'s'

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print ' %s [label="%s"' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print ' %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

"99 Bottles of Beer" Python Script

```
for quant in range(99, 0, -1):
   if quant > 1:
      print quant, "bottles of beer on the wall,", quant, "bottles of beer."
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print "1 bottle of beer on the wall, 1 bottle of beer."
      suffix = "no more beer on the wall!"
   print "Take one down, pass it around,", suffix
   print "--"
```

"99 Bottles of Beer" Perl Script

```
$nBottles = 100

foreach (reverse(1 .. $nBottles)) {
    $s = ($_ == 1) ? "" : "s";
    $oneLessS = ($_ == 2) ? "" : "s";
    print "\n$_ bottle$s of beer on the wall,\n";
    print "$_ bottle$s of beer,\n";
    print "Take one down, pass it around,\n";
    print $_ - 1, " bottle$oneLessS of beer on the wall\n";
}
```

OR

```
sub b{$n=99-@_-$_||No;"$n bottle"."s"x!!--$n." of beer"};$w=" on the wall";
die map{b."$w,\n".b.",\nTake one down, pass it around,\n".b(0)."$w.\n\n"}0..98
```

Aghhhhhhhhhhhhhhh!

Python: continued

Great for people who can't really programme, i.e. most of us!

Great for interaction with the OS.

Great for accessing web services (SIMBAD, DSS, NED, 2MASS, SDSS, etc)

Great for scientific programming via SciPy (pronounced "Sigh Pie"), see http://www.scipy.org/

Great for astronomy data analysis, e.g. pyfits, numpy, pyraf, etc.

"Using Python for Interactive Data Analysis"
by Perry Greenfield and Robert Jedrzejewski available from http://www.scipy.org/wikis/topical_software/Tutorial

Check-out Appendix B in the above.
"Why would I switch from IDL to Python (or not)?

Example Python script: Extracting info from sdss spectra files

```
import sys, pyfits, glob
filelist = glob.glob('*.fit')
master=open('master.lis','w')
for x in filelist:
  f=pyfits.open(x)
  ie=filename.index('.fit')
  gname=x[7:ie]
  prihdr=f[0].header
  z=prihdr['Z']
  vel_dis=prihdr['VEL_DIS']
  em_tab=f[2].data
  em_col=f[2].columns
  l6563=em_tab.field('ew')[36]
  f.close()
  master.write(gname+","+str(z)+","+str(vel_dis)+","+str(l6563)+'\n')
master.close()
```

Simple Python Script: To change an image mask

```
import pyfits, numarray, os

hdulist=pyfits.open('jmask.fits')
scidata=hdulist[0].data
head = hdulist[0].header
hdulist.close()

new =  numarray.logical_not(scidata) + 0
hdu=pyfits.PrimaryHDU(new,head)
hdu2=pyfits.HDUList([hdu])

if os.path.exists('ext_mask.fits'): os.unlink('ext_mask.fits')
hdu2.writeto('ext_mask.fits')
```

Perl vs Python Wars

The Empire Strikes Back reinterpreted to serve a valuable moral lesson for aspiring programmers *(and researchers)*

EXTERIOR: DAGOBAH -- DAY
With Yoda strapped to his back, Luke climbs up one of the many thick vines that grow in the swamp until he reaches the Dagobah statistics lab. Panting heavily, he continues his exercises -- grepping, installing new packages, logging in as root, and writing replacements for two-year-old shell scripts in Python.

YODA: Code! Yes. A programmer's strength flows from code maintainability. But beware of Perl. Terse syntax... more than one way to do it... default variables. The dark side of code maintainability are they. Easily they flow, quick to join you when code you write. If once you start down the dark path, forever will it dominate your destiny, consume you it will.

LUKE: Is Perl better than Python?

YODA: No... no... no. Quicker, easier, more seductive.

LUKE: But how will I know why Python is better than Perl?

YODA: You will know. When your code you try to read six months from now.

IDL Interactive Data Language

a programming language from ITT (ex-RSI, ex-Kodak, ex-???) that "analysis, and visualization of any kind of data with a suite of development tools".

Commercial product, very expensive (~£600 per licence) and ITT have taken explicit steps to prevent data compatibility with other environments. IDL originated from early VAX//Fortran, and its syntax still shows its heritage.

Very simple to use interactively.
Popular because iraf is so "user-hostile".

IDL Astronomy User's Library see http://idlastro.gsfc.nasa.gov/
Extensive set of routines

A GUIDE TO IDL FOR ASTRONOMERS by Bob O'Connell
http://www.astro.virginia.edu/class/oconnell/astr511/IDLguide.html

Book: "Practical IDL programming" by Liam E Gumley

IDL function: reads a FITS header

```
function HEADFITS, filename, EXTEN = exten, Compress = compress, $
                   ERRMSG = errmsg, SILENT = silent
;+
; CALLING SEQUENCE:
;       Result = HEADFITS(Filename/Fileunit , [ ERRMSG =, EXTEN= , COMPRESS=,
;                                              /SILENT ])
; ......
;-
  unitsupplied = size(filename,/TNAME) NE 'STRING'
  if unitsupplied then unit = filename else begin
     unit = FXPOSIT( filename, exten, errmsg = errmsg, $
                    /READONLY,compress = compress, SILENT=silent)
     if unit EQ -1 then begin
          if printerr then  $
                message,'ERROR - ' + errmsg,/CON
       return,-1
     endif
     if eof(unit) then begin
        free_lun,unit
        message = 'Extension past EOF'
        if not printerr then errmsg = message else $
```

Fortran vs C vs Perl vs Python vs IDL

Easy to Write

High level: Python, IDL, Perl(?)
Low level: C, Fortran

~5X more lines of code needed for low level languages
Bugs scale as the number of lines of code.

Fast to Run

Fast: C and Fortran
Slow: Perl, Python, IDL

Recommendations

Fortran and C for high performance applications.
If write-time is much greater than the run time: use Python.
Python for general scripting.
IDL (if you afford it!) for general data analysis and visualization.

Essential tools/links

Starlink	www.starlink.ac.uk data reduction/analysis/programming tools Many noteworthy tools include GAIA, KAPPA, CCDPACK, ORAC-DR, TOPCAT, slalib Project ended in 2005 but some support from the Joint Astronomy Centre (JAC) released "Puana (Procyon)" on 22-Aug-2007.
Iraf	iraf.noao.edu a general purpose software system for the reduction and analysis of astronomical data but "user-hostile"
LaTeX	a high-quality typesetting system www.latex-project.org
Numerical Recipes	www.nr.com

Plotting:

SM	www.astro.princeton.edu/~rhl/sm standalone system
Pgplot	www.astro.caltech.edu/~tjp/pgplot called from Fortran or C, wrap-rounds for Perl and Python (ppgplot)
IDL	www.ittvis.com/idl Data Visualization and Analysis
TOPCAT	interactive graphical viewer and editor for tabular data (http://www.star.bris.ac.uk/~mbt/topcat/)
MATLAB	numerical computing environment and language by MathWorks
matplotlib	2D plotting library for python (syntax similar to MATLAB)

Conclusions: Generic Advice

Start by using the default packages/languages used in your research group. This means you have access to local experts.

Other tools may exist that are more efficient and explore these,
e.g.
Python,
IDL,
TOPCAT,
R,
….

Research Skills to Develop

General

- Research styles
- Testing theories
- Random and systematic errors

What you must do:

- Have fun
- Never forget the big picture
- Look at astro-ph (daily) and the journals regularly and be critical of what you read
- Ask your supervisor, etc what papers are worth studying
- Look at the different styles of papers
- Become an expert in your field
- Produce results, e.g. graphs
- Write up regularly what you do; reports, applications, papers, etc
- Ask your supervisor and others difficult questions
- Go to as many talks and seminars as you can (including those at **STScI** via the web); be critical
- Ask questions at talks and seminars (bonus points)
- Argue with your supervisor and others about the real issues
- Monitor your own progress

Perl vs Python Wars II Non-PC version

"**Perl** skills are not very useful when it comes to bedding hot chicks. Unfortunately, for better or worse, perl has become associated with Unix, and for most normal people Unix is synonymous with smelly bearded unwashed hippies. So if you are trying to impress the ladies, best not mention that killer pearl regexp you came up with the other day. "

"**Python** is far more sexy than perl. Python is associated with successful dotcommers, who have survived the dot-bomb and are still making crazy dollars in the cut-throat (but sexy) world of b2b and b2c web commerce. In my experience, admitting a knowledge of python has caused many a young lady to succumb to the temptations of Satan and start "coming on" to me."