IRAF講習会
ソフト開発編 clスクリプト

大薮進喜
(ISAS/JAXA)
2009年6月29日

その0

前置き

前提

- IRAFとは、NOAOが提供するデータ解析ファシリティーである。
  - ちなみに、Imaging Reduction and Analysis Facilityの略
- IRAFで、一晩は話せる。
  - 悪口でも。
- External Packageのインストールをしたことがある。
- 途方に暮れるぐらいのデータを抱えている。
- プログラム言語を何か知っていると、うれしいな。

前提 その2

- もちろん大薮の知らないこともあるので、おもしろいことは教えて欲しい。

- 大薮は、あんまり大物プログラムは書かない(昔は、少し書いていたが、最近は他の言語を使用することが増えた。)。簡単なコマンドを作って、他のプログラム等でつなぐのが、主流。

- もっと達人がいるなら、次回を託したい。

IRAFでのソフト開発とは？

- 既存の優れたコマンド類を、使い回せる。
- メモリが、上手に使えないので、高速化等には融通が利かないが、とりあえず少ないメモリ環境でも何とかなる。
- デバッカ、プロファイラがないのに開発するのか?
  - 10000行とかあるものは、作らない方がいいだろう。
- 資料が古い、少ない、間違っている。

今回の私のお話

- IRAF Scriptの文法の話
  - 役に立つことから、
  - 役に立たないことまで。
- IRAFの膨大な機能は、話せない。
- SPPの話はありません。
  - SPPまで書くぐらいなら、他の言語に手を出す方がいいでしょう。
- 知っている人に教えることはありません。
- 皆さんが作りたいものは、皆様が考えて下さい。

参考資料

- An Introductory User's Guide to IRAF Scripts
  - http://iraf.noao.edu/iraf/ftp/iraf/docs/script.ps.Z
  - IRAF Version 2.8用に1989年に書かれた文書
- IRAF CL Script Tips & Tricks
  - http://iraf.noao.edu/iraf/ftp/iraf/docs/script_intro.pdf
  - 2003年のプレゼンファイル。いくつかの間違いやすいトピックが、Pick upされて書かれている。
- help language
  - をIRAFコンソールで叩く。
- iraf.netに行く。

大薮からの提供物

- img/*.fits
  - 演習に使うサンプルFITS
- oyabuex*.cl
  - 大薮のサンプルプログラム
- printit.cl
- oyabu
  - サンプルテキストファイル(内容は気にするな。)
- oyabu.cl
  - 念のためパッケージ化。
  - 内容のset oyabu = /home/oyabu/work/iraf/ を適当に編集して。
  - task $oyabu.pkg = <path>/oyabu.clをすると設定できる。

その1

IRAFの文法

たぶん知っていると思うけど、

- 何か動作がおかしいと思ったときは、
  - flprcache
    - Process の使ったcacheを掃除する。
  - unlearn
    - 各種コマンドの使ったパラメータを初期化する。
- それでもおかしいときは、IRAFをLoginし直して下さい。

Command

In command mode (normal interactive commands):

taskname arg1 ... argN par=val ... par=val redir

In compute mode (algebraic mode, for expressions and procedures)

taskname (arg1, ... argN, par=val, ... par=val redir)

Redirect

- > file spool output in a file
- < file read input from a file (rather than the terminal)
- >> file append the output to a file
- >& file spool both error and regular output in a file
- >>& file append both error and regular output to a file
- >[GIP] redirect graphics output to a file, e.g, >G file
- >>[GIP] append graphics output to a file, e.g, >G file

Example

cl> type *.x > spool

cl> type ("*.x", > "spool")

help language

break * Break out of a loop
case * One setting of a switch
commands - A discussion of the syntax of IRAF commands
cursors - Graphics and image display cursors
declarations - Parameter/variable declarations
default * The default clause of a switch
else * Else clause of IF statement
for * C-style for loop construct
if * If statement
goto * Goto statemen
logging - Discussion of CL logging
next * Start next iteration of a loop
parameters - Discussion of parameter attributes
procedure * Start a procedure script
return * Return from script with an optional value
switch * Multiway branch construct
while * While loop

Built-in commands and functions

- access - Test if a file exists
- back - Return to the previous directory (after a chdir)
- beep - Send a beep to the terminal
- bye - Exit a task or package
- cache - Cache parameter files, or print the current cache list
- cd - Change directory
- chdir - Change directory
- cl - Execute commands from the standard input
- clbye - A cl followed by a bye (used to save file descriptors)
- clear - Clear the terminal screen
- defpac - Test if a package is defined
- defpar - Test if a parameter is defined
- deftask - Test if a task is defined
- dparam - Dump a pset as a series of task.param=value assignments
- edit - Edit a text file
- ehistory - Edit history file to re-execute commands
- envget - Get the string value of an environment variable
- eparam - Edit parameters of a task
- error - Print error code and message and abort
- flprcache - Flush the process cache
- fprint * Print a line into a parameter
- fscan * Scan a list
- fscanf * Formatted read from a file, or another parameter
- gflush - Flush any buffered graphics output
- hidetask - Make a task invisible to the user
- history - Display commands previously executed
- jobs - Display status of background jobs
- keep - Make recent set, task, etc. declarations permanent
- kill - Kill a background job
- logout - Log out of the CL
- lparam - List the parameters of a task
- mathfcns - Mathematical routines
- mktemp - Make a temporary (unique) file name
- osfn - Return the host system equivalent of an IRAF filename
- package - Define a new package, or print the current package names
- prcache - Show process cache, or lock a process into the cache
- print - Print a line on the standard output
- printf - Formatted print to the standard output
- putlog - Put a message to the logfile
- radix - Encode a number in the specified radix
- redefine - Redefine a task
- reset - Reset the value of an environment variable
- scan * Scan the standard input
- scanf * Formatted read from the standard input
- service - Service a query from a background job
- set - Set an environment variable
- show - Show an environment variable
- sleep - Hibernate for a specified time
- strings - String manipulation routines
- stty - Set/show terminal characteristics
- task - Define a new task
- time - Print the current time
- unlearn - Restore the default parameters for a task or package
- update - Update a task's parameters (flush to disk)
- wait - Wait for all background jobs to complete
- whereis - locate all occurances of a task in the package list
- which - locate a task in the package list

Sample: printit

```
procedure printit (file_name)
# comment dayo---nn
string file_name
struct *flist
begin
    struct line
    flist = file_name
    while (fscan (flist, line) != EOF)
        print (line)
end
```

(IRAFマニュアルから)

Sample 使い方

タスクをエントリーする。
cl> task printit = path$printit.cl
実行
cl> printit textfile

例えば
cl> printit printit.cl
をすると、コードが出力される。

Parameters(変数)

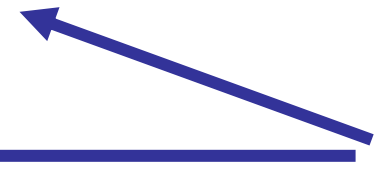
- char, string (同一か?マニュアルで表記が違う。)
  - 64 characters 以下
- Int
  - 32bits
- real
  - double相当
- bool
  - yes or no
- file
  - 基本はcharacterであるが、各種ファイル操作に必要な機能搭載。(でもまだ実装していないと20年前のマニュアルには書いてある。今はどうなっているんだろう。)
- struct
  - 基本はcharacterであるが、fscan, scanと一緒に使われる。
    - See sample(次項)

Parameterの取り扱い(最初のコードで、解説)

```
procedure printit (file_name)
# comment dayo---nn
string file_name
struct *flist
begin
    struct line
    flist = file_name
    while (fscan (flist, line) != EOF)
        print (line)
end
```

隠しパラメータ

内部変数



Argumentとしてあれば、Query パラメータ(必ず聞いてくるやつ。)。なければオプションパラメータ。

file型とstruct型

- file型は、実質stringであるようだ。
  - 詳しい人がいたら教えて欲しい。

- structもほぼstringであるが、stringが空白でターミネートされるのに大して、structは最後まで読み込める。

```
cl > string aline=“aa bb cc”
cl> =aline
aa bb cc
cl> string a,b
cl> =fscan(aline,a,b)
2
cl> =a
aa
cl> =b
bb
cl>struct c
cl>=fscan(aline,a,c)
2
cl> =a
aa
cl>=c
bb cc
```

List Directed Parameters

- IRAF独特の機能
- ファイルの中身を順次読んでくれる。
- EOFがくると勝手にCloseする。
- パラメータ名の前に＊をつけて表現

List Directed Parameters: example

cl > string *ld

cl > ld = “oyabu”

cl> = ld を繰り返してみて下さい。

その他、変数に関すること

- imcur, gcur

cl> !ds9 &

cl> display dev$pix 1

cl> =imcur

- Parameter Attributes
- Arrays

演算子(まとめて解説しているところが見つからない。誰か教えて。)

- 四則演算(+-*/)

cl>=1/2

cl>=1/2. が違うので注意。

- 関係演算子(> < == !=)
- 論理演算子(&& ||)

- 文字列の連結は、//

な感じで基本的なことは揃っている。

標準入出力

- scan系
  - scan -- read parameters from standard input
  - fscan -- read parameters from file, or another parameter
  - scanf -- formatted read from standard input
  - fscanf -- formatted read from file, or another parameter
  - nscan -- get number of parameters scanned
- print系
  - fprint -- print to a parameter
  - print -- print to the standard output
  - printf -- formatted print to the standard output

制御構造

- 大体、標準的なものは使える。
  - 使えないものがあったら上品に悪態をついて、使えるもので代用。
- 選択
  - if,switch,case
- ループ
  - while,for

さらにIRAF独自で便利なもの

- Graphics

- imget

cl> imget dev$pix naxis
cl> lpar imget
cl> =imget.value

- mktemp

cl>string savefile
cl>savefile = mktemp ("tmp$sav")
cl>=savefile

- defpac,deftask

cl> =defpac(“noao”)

その3

Exercise

前置き

- img/にサンプルimageを置いておきました。

Exercise part0

- 市川さんにやれと言われたので、
- Parameterのコード中の取り扱いを注意しないと、ある意味大変なことになる。

cl> oyabuex0dame 1 2

cl> oyabuex0dame

の動作の違いを見比べて下さい。

コードを見る。

そして oyabuex0を実行してみる。

ほらね。皆さん注意しましょう。

Exercise part1

- 複数のFitsファイル(img/obj.*.fits)の統計値を求める。
  - imstatじゃんと突っ込みを入れないように。
  - だから芸を見せるために観測ターゲット(OBJECT)と観測者名(OBSERVER)を統計量を並べてみる。
- ポイントは
  - Wildcardの実現
  - FITS headerの読み出す。
- oyabuex1.cl(実行してもいいから、まだ見ないでね。)

Wildcardの実現

- 一度入力値(ファイル名一個でもファイルリストでもワイルドカード)から、ファイルリストを作る。
- ファイルリストは、mktempを使うとスマート。
- ゴミがたまるので消すのを忘れないように。

例えば、

```
# This is for wild cards.
infile=mktemp("tmp$tako");
sections(file_name,option="root", > infile)
flist=infile
##########################
```

header情報の取り方
(いろいろ方法があると思うが、これがIRAF開発側が想定したものと思う。)

- imget(“img/obs.200s.fits”,”OBJECT”)
  - を実行すると、
  - imgetのvalueというパラメータに返り値が格納される。
- See

  cl> lpar imget

  か

  cl> =imget.value

Exercise part2

- 複数のFitsファイルのシフト量を、ds9上で天体を選択することで、シフトさせて足しあわせる。

- ds9からカーソル位置をとる。
- 多幸せは、もとい足し合わせは、imcombineを使う。

- ds9を立ち上げてから、

> oyabuex2 img/*.fits tako.fits

を実行して見て下さい。

その4

余談

64bit IRAF (メッセージ from 山内さん)

- IRAF64 Project
- IRAFの64bit化を進めている。
- http://www.ir.isas.jaxa.jp/~cyamauch/iraf64/index.html

- 32bit IRAFは、メモリ4GBの壁がある。
  - これからの時代4GBの壁があると大変だ。
- 達人になったら協力して上げて下さい。

最後に、今一度CL ScriptのMeritを。

- IRAFのcl Scriptを用いると、パラメータ渡しや、オプションの制限など簡単にできる(すでに実装されている)ために、細かいところの作り込みが省略でき、その開発コストを下げることできます。
- IRAFがMulti platformであることから、簡単に他のOSへのアプリの提供ができます。
  - これらは外部へ提供するアプリをつくるのには重要なポイント。

- IRAFには多くの便利なコマンドがあります。まずこれをうまく使うことで、コーディングがかなり省略できます。
  - また既存のコードを書き換えて自分のニーズあったものにできます。
  - 人のプログラムを参考にすることはいいことだ。
- ds9とのやりとりも比較的簡単にできます。
  - ここは最近はXPAが発達してきたので、さほどメリットではありませんが。

IRAF Scriptへの懸念

- 最近のコンピュータ事情に追随できていません。
  - 64bit化
  - multi-thread化
  - 大メモリ化

    - なんちゃって解決法はなきにしもあらずだが、本質的には上記はいずれ問題化すると考えている。(本当のコンピュータの能力が活用できない。)

ほんとに最後に

- 外部から、外部のプログラムの使用法を学べる明日のお話を楽しみにしましょう。
- 今回の講義は、私も手探りでしたので、是非感想(大薮とADC宛に)を教えて下さい。
  - もっとDeepにという場合は、さらなる達人が必要でしょう。
  - こういう点が知りたいというリクエストは、ぜひ次回に重要です。
  - ありがとうございました。