NEW IRAF ABC (ver. 2. 5. 2)
(IRAF를 이용한 측광 자료 전처리)

2012. 3. 25 (최근 업데이트 2013. 8. 13)

박 근 홍, 전 이 슬, 홍 주 은, 황 호 성

0. 들어가는 글

2009년 처음으로 천문관측 및 실험 1의 조교를 맡았습니다. 그간 천문학과에서 널리 사랑을 받아온 선배님들의 안내서들[1](ver. 1. 0)을 바탕으로 하여 내용을 추가, 구조를 재편하여 두 개의 안내서[2](ver. 2. 0)를 만들었습니다.

그로부터 3년이 지났습니다. 그 사이 천문학과는 전산 실습실을 갖췄고, 덕분에 학생들은 관측 자료 처리를 그간 사용되어왔던 astro10 서버가 아닌 전산 실습실에서 할 수 있게 되었습니다. 그래서 astro10에 깔려 있던 cl(command language)기반의 구버전 IRAF가 아닌, ecl(enhanced command language)기반의 최근 IRAF를 기준으로 IRAF 안내서를 다시 손보았습니다. 3년 전의 안내서에서 크게 달라진 점은, 첫 번째로 리눅스 명령어 부분을 보강하고 독립시켜, 총 3개의 안내서로 재구성 하였습니다. 두 번째로 전에 누락시켰던 IRAF 설치를 비롯한 일부 내용을 추가하였습니다. 세 번째로 앞서 말한바와 같이 ecl 기반으로 변경했습니다. 결과적으로 순서나 구성이나 세부 내용에는 변경된 것이 있더라도, 황호성 선배님의 IRAF ABC와 같은 부분을 다루게 되어, 제목도 NEW IRAF ABC로 명명하였습니다.

불량화소 보정 부분의 작성을 맡아준 홍주은 학생, 이미지 합침 및 프린지 제거에 많은 도움을 준 전이슬 학생, 그리고 좋은 조언을 해주신 김성진 선배님께 지면을 빌어 감사를 드립니다. 해당 안내서의 오류 및 건의 사항이 있으면 khpark@astro.snu.ac.kr로 언제든지 알려주시면 수정 혹은 적극 반영토록 하겠습니다. 부족하지만 측광 작업 입문자에게 도움이 되길 바랍니다.

2012년 3월, 서울대학교에서
박 근 홍

(※ SAGA 홈페이지[3]에서 가장 최근 버전의 안내서를 받을 수 있습니다.)

---

1) IRAF ABC (황호성, 이명균,2004), DAOPHOT을 이용한 PSF측광 (황호성, 이명균, 2004), IRAF/PHOT을 이용한 변광성 측광 (황호성, 이명균, 2006), PHOTCAL Tutorial (이명균, 이준협, 김상철, 2002)
2) Linux 명령어 및 IRAF를 이용한 전처리 (황호성, 이명균, 박근홍, 이광호, 이상각, 박홍수, 류진혁, 2009), IRAF를 이용한 구경측광, PSF측광 그리고 표준화 (황호성, 이명균, 이준협, 박근홍, 이광호, 이상각, 박홍수, 류진혁, 2009)
3) http://astro.snu.ac.kr/~grad

Ⅰ. IRAF 실행

1) 터미널 및 ds9 실행하기

리눅스에는 xterm, gterm, hanterm 등 다양한 터미널이 있다. IRAF는 xgterm에서 최적화 되어 있으므로 IRAF를 쓰기 위해서는 xgterm을 실행시키도록 한다.

```
$ xgterm –sb&
```

Notice 1. $ 표시는 리눅스 터미널에서 실행함을 의미한다.

이 때 - 뒤에 붙는 것은 옵션 값이다. sb는 스크롤바를 의미하며, &는 새 창으로 실행하겠다는 뜻이다. 터미널의 색을 바꾸고 싶을 때는 -fg 색깔(글자색 변경) -bg 색깔(배경색 변경)을 입력한다. 또한 폰트 설정을 변경하고 싶을 때는 -font 옵션을 입력해주면 된다. 가령 배경을 검게, 글자를 상아색, 폰트를 좀 더 크고 굵게 하고 싶다면, 다음 명령어를 입력해보자.

```
$ xgterm -fg ivory -bg black -font 9*15bold &
```

또한 이미지 보기 및 IRAF와 호환하여 이미지 상의 값을 읽어주고, 자료처리를 하는 프로그램인 ds9을 실행하자.

```
$ ds9&
```

tip1. 창을 여러 개 사용하자. xgterm에서 IRAF를 실행 하면서, 동시에 xterm으로 파일 확인 및 간단한 리눅스 명령어 입력에 활용하면 편리하다. 특히 IRAF에 익숙하지 않을 때는, xgterm을 2개에 각각 IRAF를 실행하여 하나는 작업에, 다른 하나는 help 명령어로 도움말을 보는데 활용하자.

tip2. 기본적으로 path가 성공적으로 잡힌 실행파일들은 위치에 상관없이 프로그램을 실행할 수 있다. 하지만 path가 잡혀 있지 않은 실행파일을 실행하고자 할 때는 파일 경로를 다 입력을 해주어야한다. (ex. /usr/local/bin/xxxxx)

Notice 2. 터미널을 종료할 때는 우측에 x박스를 누르기보다는 exit라는 명령으로 종료를 해주도록 한다.

2) IRAF 실행 및 종료하기

IRAF를 실행에 앞서, IRAF 실행 파일인 login.cl을 만들어야 한다. login.cl 파일을 생성하고자 하는 디렉토리로 이동 후 mkiraf를 입력한다. 이 후 터미널 타입을 묻는 항목에서 xgterm이라고 적자.

```
$ mkdir M50
$ cd M50
$ mkiraf
-- creating a new uparm directory
Terminal types: xgterm,xterm,gterm,vt640,vt100,etc.
Enter terminal type: xgterm
A new LOGIN.CL file has been created in the current directory.
You may wish to review and edit this file to change the defaults.
$ ls
login.cl   uparm
```

그 다음 login.cl의 내용을 몇 군데 수정한다.

```
$ vi login.cl
```

stdimage는 디스플레이 되는 이미지의 속성을 설정하는 것이다. 자신의 이미지의 크기에 맞춰서 설정하거나(1K CCD는 1024, 2K CCD는 2048), 아니면 넉넉하게 4096이나 8192 정도로 큰 값을 입력하자. 예제 파일은 1K CCD (픽셀이 1024 × 1024)를 사용하므로 #set stdimage = imt800을 imt1024로 수정해준다. (앞에 #표를 지우는 것에 주의 할 것!!!) 그리고 imtype은 IRAF에서 사용할 이미지 타입을 결정한다. 우리가 사용할 데이터는 확장자가 fits이므로 #set imtype = "imh"를 set imtype = "fits" 로 수정해준다. (앞에 #표를 지우는 것에 주의 할 것!!!) 이제부터 IRAF 상에서 fits파일은 확장자를 써 주지 않아도 되며, 모든 이미지 결과물은 fits로 저장이 된다.

```
# Uncomment and edit to change the defaults.
#set     editor            = vi
#set     printer           = lp
#set     pspage              = "letter"
set      stdimage           = imt1024                <- 여기 수정, 앞에 # 제거
#set     stdimcur          = stdimage
#set     stdplot           = lw
#set     clobber            = no
#set     filewait          = yes
#set     cmbuflen           = 512000
#set     min_lenuserarea = 64000
set      imtype             = "fits"                 <- 여기 수정, 앞에 # 제거
#set     imextn             = "oif:imh fxf:fits,fit plf:pl qpf:qp stf:hhh,??h"
```

자, 이제 모든 준비가 끝났다. 이제 IRAF를 실행해보자. login.cl 파일이 있는 디렉토리에서 cl이라고 입력하면 IRAF가 실행된다.

```
$ cl
ecl>
```

IRAF를 종료할 때는 logout이라고 종료하면 된다. (log까지만 입력해도 된다)

```
ecl> log
$
```

Notice 3. 이후 작업에서 제대로 설정을 했다고 생각했음에도, IRAF가 오류 메시지를 출력하는 경우, 높은 확률로 login.cl이 없는 곳에서 cl 명령어를 입력한 경우다. 반드시 login.cl 이 있는 곳에서 IRAF를 실행하자.

tip3. 당연하지만, 한 번 login.cl을 만들면 다음 IRAF 실행할 때는 mkiraf 명령어를 입력할 필요없다.

tip4. 다수의 관측소에서 관측한 자료를 처리할 경우, 각 디렉토리마다 login.cl 파일을 만들어두면 RDnoise나 Gain과 같은 CCD 관련 상수값을 자료처리 할 때마다 수정해줄 필요가 없어서 편리하다. 예를 들어 Data란 디렉토리 아래 BOAO(보현산 천문대), SOAO(소백산 천문대), SNUO(신천문대 자료)의 하위 디렉토리가 있고, 각 디렉토리 안에 해당 천문대의 자료가 있다면 세 디렉토리에 각각 login.cl을 생성해서 각 천문대 CCD에 해당하는 RDnoise나 Gain을 한 번 입력 해주면 이후 필요한 천문대 자료를 처리할 때 해당 login.cl을 구동하면 덜 번거로워진다.

tip5. IRAF상에서 rm 명령어가 듣지 않는다. 이 때 앞에 느낌표를 붙여주자.

```
ecl> cp ../haha.txt .
ecl> ls
haha.txt   login.cl   uparm
ecl> rm haha.txt
ERROR: ambiguous task `rm'
ecl> !rm haha.txt
ecl> ls
login.cl   uparm
```

마찬가지로, IRAF 상에서 먹히지 않는 리눅스 명령어들은 앞에 느낌표를 붙여주면 IRAF에서 실행된다. 또한 반복 명령을 수행할 때, 화살표 ↑키를 통해 이전에 사용한 명령어를 쉽게 불러올 수 있다.

tip 6. 어떠한 이유에서 돌아가는 프로그램을 멈추고 싶을 때는 ctrl+c키를 누르면 작업이

멈춘다. 다만 가끔씩 그 이후에 프로그램이 제대로 돌아가지 않을 때가 있는데 이 경우는 해당창을 닫아주고 새로 IRAF를 실행하면 다시 제대로 수행된다.

notice 4. 연습 예제파일은 M50을 UBV 필터에 대해 측광한 자료이다. 다운로드는 http://astro.snu.ac.kr/~khpark/practice/M50.tar 에서 받을 수 있다.

tip 7. 학교에서 관측하여 확장자가 fit으로 되어 있는 경우, 확장자를 fits로 바꿔야 한다. 하나하나 수정할 필요 없이 IRAF를 실행 한 후에 rename명령어를 통해서 간단히 확장자를 바꿀 수 있다. 순서는 rename [대상 파일명] [변경되고자 하는 부분] [범위] 순이다. 예를 들어 fit 으로 끝나는 파일을 대상으로, fits라고 확장자를 변경하고 싶을 경우 rename *fit fits field=extn 이 된다.

```
ecl> ls
Autosave-001bias.fit        Autosave-007dark_20.fit   fr01.fit
Autosave-001dark_10.fit   Autosave-007dark_30.fit   fr02.fit
Autosave-001dark_20.fit   Autosave-007dark_60.fit   fr03.fit
...
ecl> rename *fit fits field=extn
ecl> ls
Autosave-001bias.fits        Autosave-007dark_20.fits   fr01.fits
Autosave-001dark_10.fits   Autosave-007dark_30.fits   fr02.fits
Autosave-001dark_20.fits   Autosave-007dark_60.fits   fr03.fits
...
```

3) 이미지 디스플레이 및 헤더 정보 다루기

ds9이 실행되어 있을 때 어떠한 파일을 ds9에 디스플레이하고 싶으면 display (파일명) (프레임번호) fi+ 라고 입력하면 된다. fi+는 꽉채워서 display하라는 명령어이다.

```
ecl> disp M50_v10_1 1 fi+
z1=547. z2=2887.143
```

이미지가 디스플레이 된 상태에서 imexam 명령을 실행시키면 ds9창의 커서가 화살표에서 동그라미로 바뀌게 된다. 이 상태에서 살펴보고자 하는 천체 위에 커서를 놓고 다음과 같은 키를 눌러주면 천체에 대한 정보를 살펴 볼 수 있다.

```
ecl> imexam
```

r : 반경에 따른 별의 밝기 분포 그래프
s : 별의 밝기 분포를 surface plot으로 나타낸 그래프
e : 별의 밝기 분포를 contour plot으로 나타낸 그래프
a : 별의 중심좌표, 등급 등등

h : 별의 밝기 분포의 히스토그램
m : 통계
z : 커서 주변의 픽셀값 표시
q : imexam을 종료하고 터미널창으로 돌아가기

헤더는 해당 이미지의 노출시간, 노출시작시간, 필터와 같은 관측시의 여러 가지 정보를 담고 있다. 이러한 정보를 보는 명령어는 imheader이다.
imheader (파일명)을 입력하면 해당 파일에 가장 기본적인 정보만 디스플레이 된다.

```
ecl> imheader M50*
M50_b10_1.fits[512,512][ushort]:
M50_b10_2.fits[512,512][ushort]:
M50_b10_3.fits[512,512][ushort]:
...
```

헤더의 모든 내용을 보길 원한다면 뒤에 lo+를 입력하면 된다.

```
ecl> imheader M50_b10_1 lo+ | page
```

헤더에 특정 정보만 보고자 한다면 hselect명령어를 사용하자. 원래 hselect 명령어는 특정 조건을 만족하는 파일의 헤더 정보를 보는 명령어이다. hselect [대상파일명] [보고 싶은 헤더 종류] [조건문] 이다. $I는 파일명을 의미한다. 조건문에서 등호(equal)는 ==, 같지 않다(not equal)은 !=, 미만(less than)은 <, 이하(less than or equal)은 <=, 초과(greater)는 >, 이상(greater or equal)은 >= 라고 입력하고, 조건이 문자라면 따옴표를 붙여준다. 2개 이상의 조건을 설정하고 싶으면 조건들 사이에 'AND'는 &&, 'OR"는 || 를 넣어주고 (예> 'FILTER =="V" && EXPTIME == 5.0') 이 때 조건을 주지 않으려면 조건의 위치에 yes라고 입력을 하면 된다.

```
ecl> hselect 'f*' '$I,FILTER,EXPTIME' 'FILTER == "V"'
fv01.fits       V       2.0000000000000000
fv02.fits       V       2.0000000000000000
fv03.fits       V       2.0000000000000000
...
ecl> hselect 'f*' '$I,FILTER,EXPTIME' yes
fb01.fits       B       2.0000000000000000
fb02.fits       B       2.0000000000000000
fb03.fits       B       2.0000000000000000
...
ccdred> hsel f* '$I,EXPTIME' 'FILTER=="V" && EXPTIME >=3'
fv07.fits       3.0000000000000000
```

헤더에 내용을 수정·추가하고 싶을 때는 hedit을 이용한다. hedit [파일명] [헤더명] [헤더내용]을 입력하자.

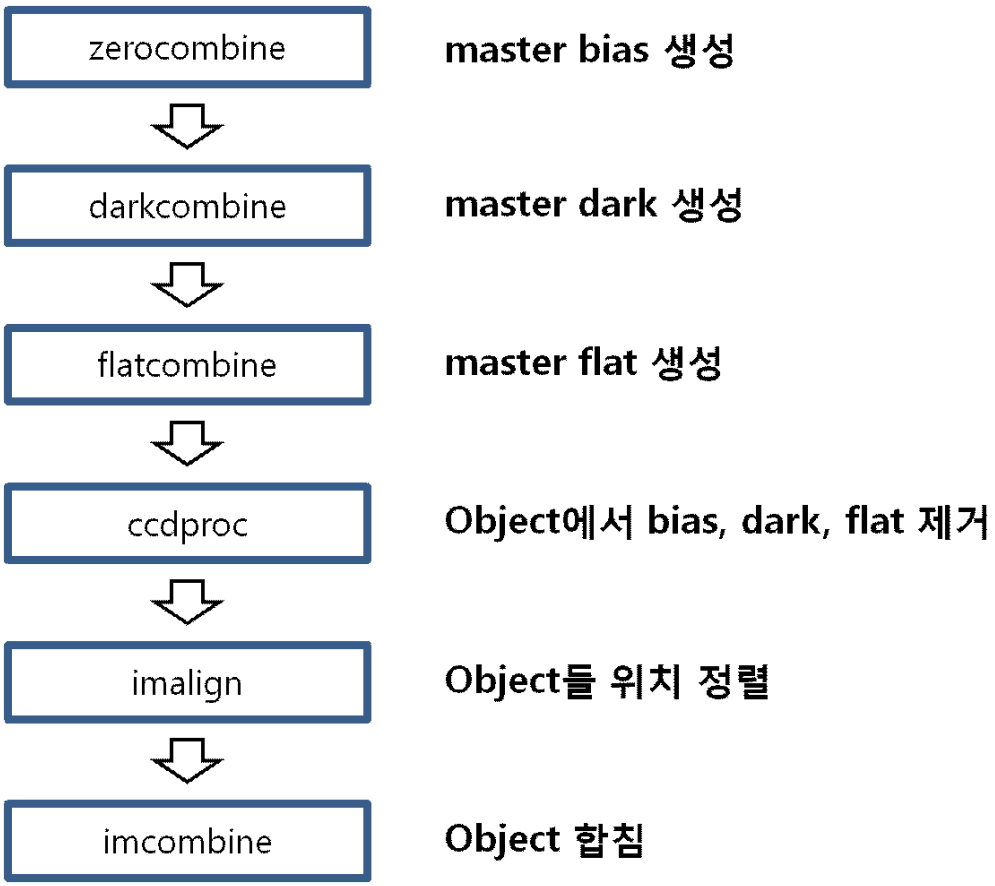
```
ecl> hsel 'M50*' '$I,IMAGETYP' yes
M50_b10_1.fits   LIGHT
M50_b10_2.fits   LIGHT
M50_b10_3.fits   LIGHT
...
ecl> hedit M50* IMAGETYP "Light Frame"
M50_b10_1.fits,IMAGETYP (LIGHT -> "Light Frame"):
M50_b10_1.fits,IMAGETYP: LIGHT -> "Light Frame"
update M50_b10_1.fits ? (yes): y
...
ecl> hsel 'M50*' '$I,IMAGETYP' yes
M50_b10_1.fits   "Light Frame"
M50_b10_2.fits   "Light Frame"
M50_b10_3.fits   "Light Frame"
...
```

tip 8. 위의 예시에서 눈치챈 학생들도 있겠지만, IRAF의 대부분의 명령어는 꼭 모든 철자를 다 칠 필요는 없고 적당히 축약해도 돌아간다. 다만 잘못 축약하다가는 다른 명령이 실행될 수도 있으니 유의하도록 하자.

notice 5. 자료처리에 앞서서 원본 데이터들은 반드시 다른 곳에 백업해두도록 하자.

notice 6. 대소문자 및 띄어쓰기는 매우 중요하다.

Ⅱ. 전처리(preprocessing)



[그림 1] 전처리 작업의 순서

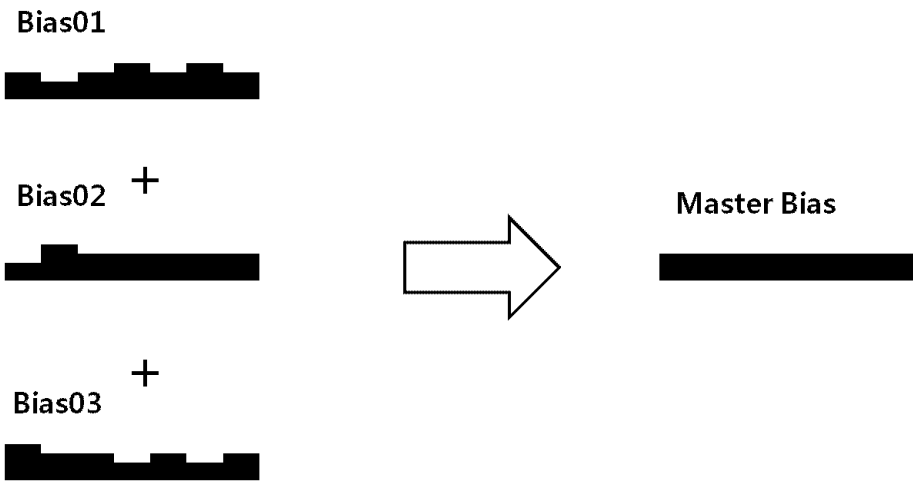
전처리란 [그림 1]과 같이 CCD의 특성인 Bias(CCD에 기본적으로 깔려 있는 전자의 개수), Dark(열복사로 인해 발생하는 전자의 개수), Flat(같은 자극에 대해, 각 픽셀이 반응하는 정도의 차이)을 보정하는 작업 등을 말한다. 이 작업을 위해 noao 패키지의 image reduction . ccd reduction패키지를 사용한다.

```
ecl> noao.imred
        argus.      crutil.     echelle.    iids.       kpnocoude.  specred.
        bias.       ctioslit.   generic.    irred.      kpnoslit.   vtel.
        ccdred.     dtoi.       hydra.      irs.        quadred.

imred> ccdred
        badpiximage     ccdmask         flatcombine     mkskyflat
        ccdgroups       ccdproc         mkfringecor     setinstrument
        ccdhedit        ccdtest         mkillumcor      zerocombine
        ccdinstrument   combine         mkillumflat
        ccdlist         darkcombine     mkskycor

ccdred>
```

1) Bias처리



[그림 2] Master Bias 생성

imstat을 사용하여 Bias 자료의 통계를 보고, 평균값이나 표준편차가 유달리 튀는 이미지는 제거하도록 한다.

```
ccdred> imstat *bias*
#                IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
 Autosave-001bias.fits     262144      127.     9.047       91.       965.
 Autosave-002bias.fits     262144     125.6     8.932       91.       579.
 Autosave-003bias.fits     262144     125.2     8.898       92.       212.
 Autosave-004bias.fits     262144     125.4     8.913       88.       210.
 Autosave-005bias.fits     262144     124.2     8.873       87.       211.
 Autosave-006bias.fits     262144     124.3     8.883       91.       213.
 Autosave-007bias.fits     262144     125.4     8.883       89.       211.
 Autosave-008bias.fits     262144     124.9     8.857       88.       195.
 Autosave-009bias.fits     262144     123.9      8.89       88.       212.
 Autosave-010bias.fits     262144     123.5     8.869       89.       210.
```

이제 사용할 bias이미지를 목록 파일로 만들어준다. Bias 이미지가 10장이므로, 5장씩 2개의 묶음으로 묶은 후, 각 묶음을 각각 중간 값으로 합쳐준다. 그리고 그 두 파일을 평균을 낸다. (Dark나 Flat, Object에 해당하는 천체도 이와 같은 방법으로 combine 및 보정작업 이전에 파일을 묶어주면 된다. 이러한 파일을 묶는 명령어는 이후부터는 알아서 잘 묶었으리라 생각하고 생략한다.) 주의할 사항은, ls 명령어를 통해 생성된 파일 역시 ls로 묶일 수 있는 것이다. 가령 ls *bias* > bias1.list 라고 명령을 내린다면 bias1.list라는 리스트 파일 역시 *bias*라는 조건을 만족하기 때문에, bias1.list안에 bias1.list가 적히게 되고, 이는 이후 오류의 원인이 된다.

```
ccdred> ls *bias* > Bias1.list
ccdred> ls *bias* > Bias2.list
```

vi 명령어로 Bais1.list는 0~5번, Bias2.list는 6~10번을 남기자.

```
ccdred> cat Bias1.list
Autosave-001bias.fits
Autosave-002bias.fits
Autosave-003bias.fits
Autosave-004bias.fits
Autosave-005bias.fits
ccdred> cat Bias2.list
Autosave-006bias.fits
Autosave-007bias.fits
Autosave-008bias.fits
Autosave-009bias.fits
Autosave-010bias.fits
```

또는 앞서 설명했던 hselect 명령어를 통해서 헤더에 IMAGETYP이 Bias인 파일만 묶어줄 수 있다.

```
ccdred> hselect '*fits' '$I' 'IMAGETYP == "BIAS"'
Autosave-001bias.fits
Autosave-002bias.fits
Autosave-003bias.fits
...
ccdred>  hselect '*fits' '$I' 'IMAGETYP == "BIAS"' > 'Bias.list'
cccred> cat Bias.list
Autosave-001bias.fits
Autosave-002bias.fits
Autosave-003bias.fits
...
```

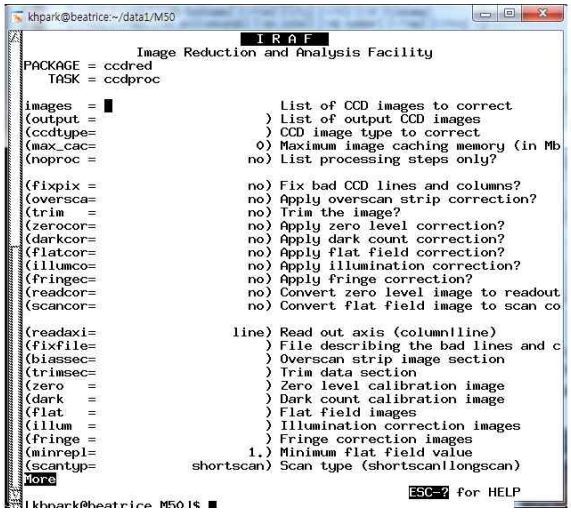
마스터 Bias를 만드는 명령은 zerocombine이다. 하지만 그에 앞서서 ccdproc에서 설정을 변경해야한다. Bias작업은 당연히 Bias, Dark, Flat처리를 하지 않을 것이기에 해당 처리를 꺼준다. 앞으로 epar라는 명령어가 자주 나올 것인데 epar는 edit parameter의 줄임말이다. (parameter를 확인 할 때는 look parameter의 줄임말 lpar라는 명령어를 쓴다.) 어떤 명령을 실행할 때는 epar를 통해 하나하나 파라미터를 수정한 후 실행하는 방법과, 창에 명령어와 파라미터를 직접 써주는 두 가지 방법이 있는데, IRAF를 처음 시작할 때는 epar를 사용하는 것을 추천한다. 명령어를 직접 써주는 방법은 반복 작업이나 스크립트 등에 사용을 하는데 용이하다.

```
ccdred> epar ccdproc
```

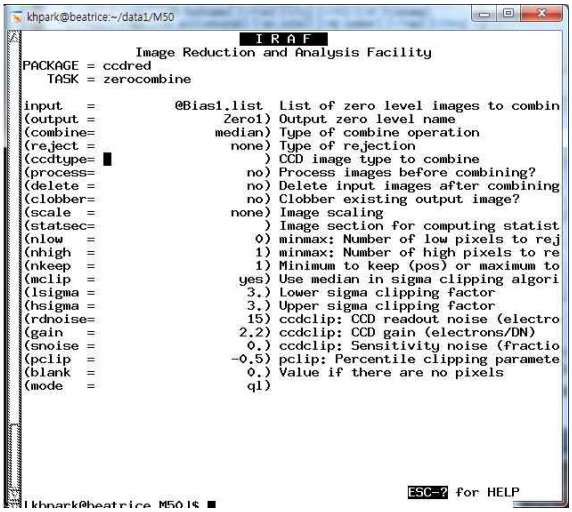
그 후 [그림 3]과 같이 zerocor, darkcor, flatcor을 모두 꺼준다. (그 외 ccdtype도 끄고, trim 등도 특별히 할 게 없다면 꺼둔다.) 변경하였으면 저장 (ctrl+D 또는 :wq)하고 나오도록 하자. (저장을 안하고 나오는 명령은 ctrl+C 또는 :q)

이제 zerocombine을 통해 파일을 합치도록 하자.

```
cc> epar zeroc
```



[그림 3] Bias 처리전 ccdproc 설정



[그림 4] Zerocombine 설정

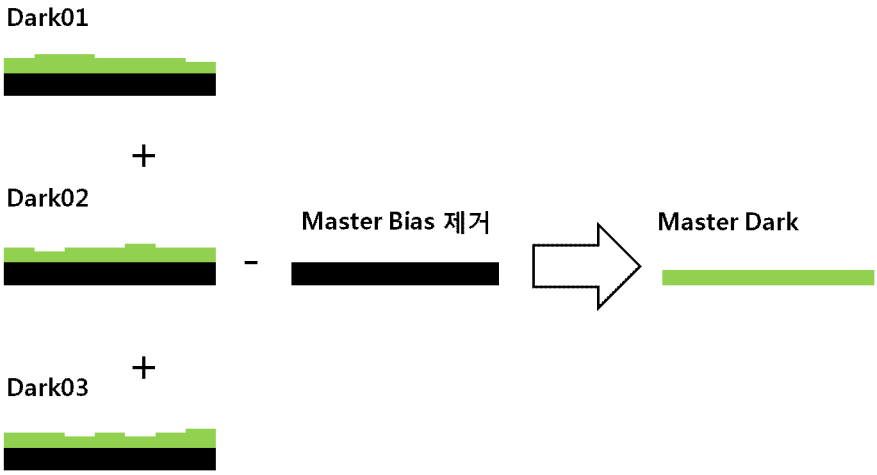
[그림 4]과 같이 input 파일, output파일, RDnoise, Gain값과 combine방식을 정해주도록 한다. combine방식에는 평균(average)과 중간값(median) 두 방법이 많이 사용된다. 평균은 S/N을 높이는데 가장 좋은 효과를 가지고 오지만, 우주선(cosmic ray)이나, 불량화소 (bad pixel, hot pixel) 등 튀는 값에 영향을 크게 받고, 중간값은 튀는 값의 영향을 최소화 하는 장점이 있지만, 평균에 비해 S/N비는 감소하고, 짝수 개의 프레임을 합칠 때는 권장되지 않는다. 이 매뉴얼에서는, 앞서 말한 대로, 10장의 bias를 5장씩 두 그룹으로 나누어, 각 그룹은 중간값을 취하고, 이후 결과물로 나온 두 이미지를 평균낼 것이다. 입력을 마치면 :go라고 입력하여 zerocombine을 실행한다. 마찬가지로, Bias2.list를 Zero2로, 그리고 Zero1, Zero2를 Zero로 zerocombine을 실행하자.

또는 명령창에 다음과 같이 입력해도 된다. \키는 줄이 바뀌더라도 명령문은 끝나지 않았다는 뜻이고 “”는 빈칸을 의미한다. 또한 +,-는 각각 yes와 no를 뜻한다.

```
ccdred> zerocomb @Bias1.list out=Zero1 combine=median reject=none \
>>> ccdtype="" scale=none rdnoise=15 gain=2.2
ccdred> zerocomb @Bias2.list out=Zero2 combine=median reject=none \
>>> ccdtype="" scale=none rdnoise=15 gain=2.2
ccdred> zerocomb Zero1,Zero2 out=Zero combine=average reject=none \
>>> ccdtype="" scale=none rdnoise=15 gain=2.2
ccdred> ls Zero*
Zero1.fits  Zero2.fits  Zero.fits
```

notice 7. 오류 발생 시, ccdproc이나 zerocombine에서 ccdtype을 꺼두었는지, 다시 한 번 체크해보자.

2) Dark처리



[그림 5] Master Dark 생성

```
ccdred> hsel M50* '$I,EXPTIME' yes
M50_b10_1.fits   10.000000000000000
M50_b10_2.fits   10.000000000000000
M50_b10_3.fits   10.000000000000000
M50_u20_1.fits   20.000000000000000
M50_u20_2.fits   20.000000000000000
M50_u20_3.fits   20.000000000000000
M50_v10_1.fits   10.000000000000000
M50_v10_2.fits   10.000000000000000
M50_v10_3.fits   10.000000000000000
```

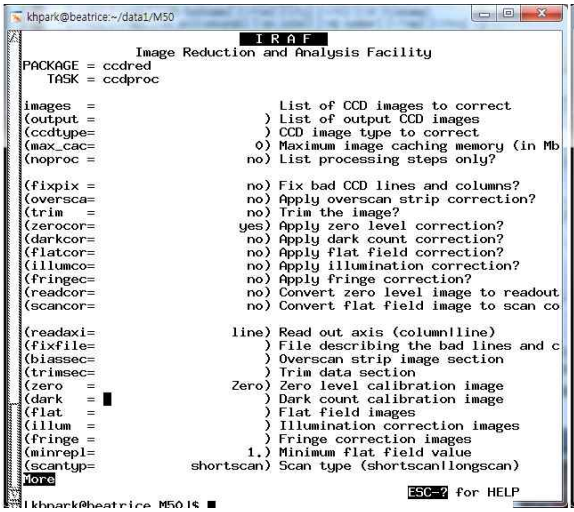
예제 파일에는 10초, 20초, 30초, 60초를 찍은 Dark가 있지만, 실제 M50을 관측하는 데는 10초, 20초 노출만 사용되었으므로, 10초, 20초의 Dark들만 사용한다.

Dark파일 역시 imstat명령을 통해서 튀는 데이터를 제거해주도록 한다. Dark는 열전자에 의한 노이즈로, Dark 이미지의 값은 이론적으로는 시간에 따라 선형증가하기에, 어떤 노출시간의 Dark 이미지가 있으면 다른 노출시간의 Dark값도 계산이 가능하지만, 실제 자료처리를 할 때는 노출 시간 별로 같은 노출 시간의 Dark를 사용한다.

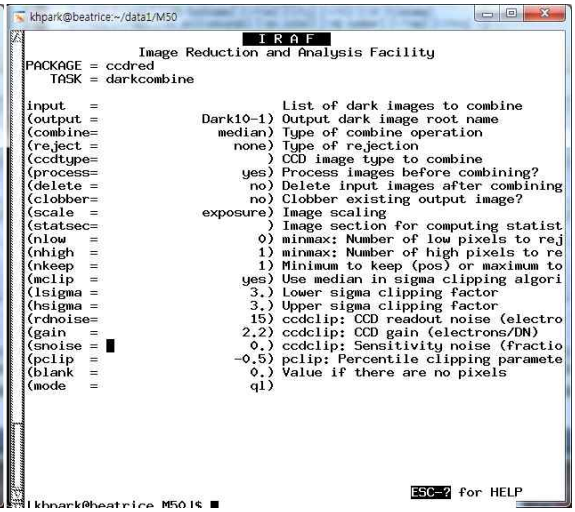
Dark는 Bias 값이 포함되어 있기 때문에 Master Dark를 만드는 과정에서 Bias를 제거하여야 한다. 따라서 [그림 6]과 같이 ccdproc에서 Zerocor을 켜주고, zero에 앞서 만든 Zero.fits를 추가하고 ctrl+D 혹은 :wq로 저장하고 나온다. 이후 아래와 같이 list파일을 만든다.

```
ccdred> ls *001dark_10* *002dark_10* *003dark_10* *004dark_10* *005dark_10* > Dark10-1.list
ccdred> ls *006dark_10* *007dark_10* *008dark_10* *009dark_10* *010dark_10* > Dark10-2.list
ccdred> ls *001dark_20* *002dark_20* *003dark_20* *004dark_20* *005dark_20* > Dark20-1.list
ccdred> ls *006dark_20* *007dark_20* *008dark_20* *009dark_20* *010dark_20* > Dark20-2.list
```

[그림 7]와 같이 epar를 통해서 하거나, 혹은 아래와 같이 직접 명령어를 넣어, 각 그룹 내에서 median 합성 후, 10초/20초 각각 두 결과물을 평균으로 합친다.



[그림 6] Dark 처리전 ccdproc 설정



[그림 7] Darkcombine 설정

ccdred> darkc @Dark10-1.list out=Dark10-1 combine=median reject=none ccdtype="" process=yes scale=exposure rdnoise=15 gain=2.2

ccdred> darkc @Dark10-2.list out=Dark10-2 combine=median reject=none ccdtype="" process=yes scale=exposure rdnoise=15 gain=2.2

ccdred> darkc @Dark20-1.list out=Dark20-1 combine=median reject=none ccdtype="" process=yes scale=exposure rdnoise=15 gain=2.2

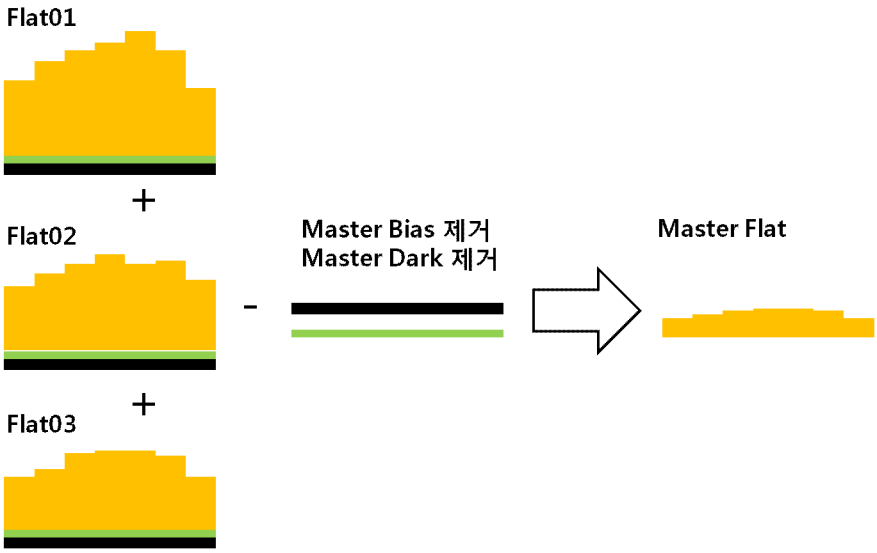
ccdred> darkc @Dark20-2.list out=Dark20-2 combine=median reject=none ccdtype="" process=yes scale=exposure rdnoise=15 gain=2.2

ccdred> darkc Dark10-1,Dark10-2 out=Dark10 combine=average reject=none ccdtype="" process=no scale=exposure rdnoise=15 gain=2.2

ccdred> darkc Dark20-1,Dark20-2 out=Dark20 combine=average reject=none ccdtype="" process=no scale=exposure rdnoise=15 gain=2.2

notice 8. 뒤에 process가 no로 바뀌는 이유는, 앞에서 이미 bias를 제거했기 때문이다. 이렇게 해주지 않으면 bias를 두 번 빼주게 되므로 주의하자.

3) Flat처리



[그림 8] Master Flat 생성

Bias와 Dark는 필터의 영향을 받지 않지만, Flat부터는 각 필터별로 진행한다. imstat으로 튀는 데이터는 사전에 제거한다.

먼저, Flat의 헤더에 subset 정보를 넣어주자. 헤더에 subset 정보를 넣어주는 이유는 flatcombine을 할 때 옵션의 subset 항목을 사용해주면, 자동적으로 같은 subset이 있는 항목끼리 모아서 일을 처리하여 편리하기 때문이다. 예제 파일에서는, UBVRI 모든 필터에 대해 플랫을 얻었기 때문에, 상관없는 R, I 플랫들은 삭제해주자.

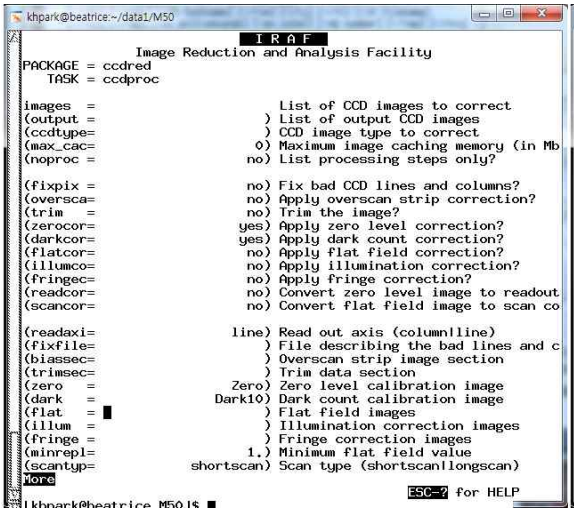
```
ccdred> !rm fr* fi*
ccdred> ccdhedit fu* subset U
ccdred> ccdhedit fb* subset B
ccdred> ccdhedit fv* subset V
ccdred> ls f* > Flat.list
```

Flat이미지는 Bias, Dark에 의한 노이즈를 포함하고 있기 때문에 [그림 9]와 같이 ccdproc에서 Bias와 Dark 보정을 키도록 하자. Flat 역시 각 노출 시간에 따라 해당하는 노출의 Dark를 보정해야 하는 게 원칙이지만, 보통 Dark의 ADU값은, Flat의 그것에 비해 무시 가능 할 정도로 작기 때문에, 아무 마스터 Dark 파일을 사용해도 그렇게 큰 문제가 생기지는 않는다.

마찬가지로 [그림 10]과 같이 epar를 설정하고 실행해주거나 명령창에 아래와 같이 입력한다.

```
ccdred> flatcomb @Flat.list out=Flat combine=median \
reject=avsigclip ccdtype="" process+ subsets+ scale=mean \
statsec=[51:460,51:460] rdnoise=15 gain=2.2
```

제대로 수행되었다면 다음과 같이 한 번의 작업으로 필터별로 Flat을 얻은 것을 확인할 수 있다.

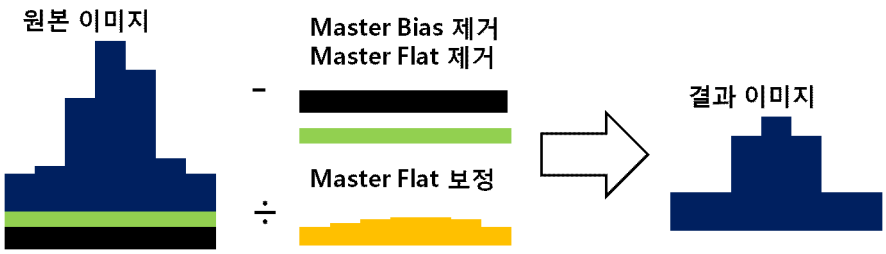


[그림 9] Flat 처리전 ccdproc 설정

[그림 10] Flatcombine 설정

```
ccdred> ls Flat*fits
FlatB.fits  FlatU.fits  FlatV.fits
```

4) Object

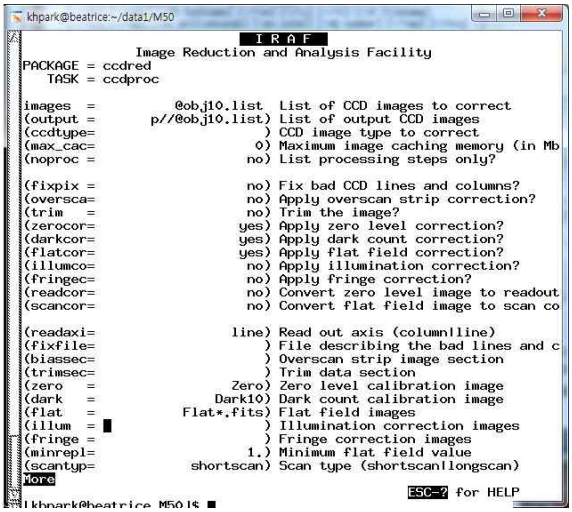


[그림 11] Object 이미지에서, Bias, Dark, Flat 처리하기

이제 object이미지에 ccdproc에 Bias, Dark, Flat 보정을 해주면 된다. object역시 필터 별로 분리해서 자료를 처리해야하므로 헤더에 subset정보를 넣어준다.

```
ccdred> ccdhedit M50_u* subset U
ccdred> ccdhedit M50_b* subset B
ccdred> ccdhedit M50_v* subset V
ccdred> ls M50*10* > obj10.list
ccdred> ls M50*20* > obj20.list
```

[그림 12]과 같이 epar로 설정 및 실행해주거나, 아래와 같은 명령어를 입력해준다. (p//@의 의미는 해당되는 리스트 파일 앞에 p를 붙여주라는 의미이다. 이전까지는 여러 리스트의 파일을 하나의 결과물로 받아줬지만, 이와 같이 입력물과 결과물이 1:1 대응이 되는 경우 //@가 자주 사용된다.)



[그림 12] Object 처리 설정

ccdred> ccdproc @obj10.list out=p//@obj10.list ccdtype="" fixpix- oversc- trim- zerocor+ darkcor+ flatcor+ zero=Zero dark=Dark10 flat=Flat*.fits

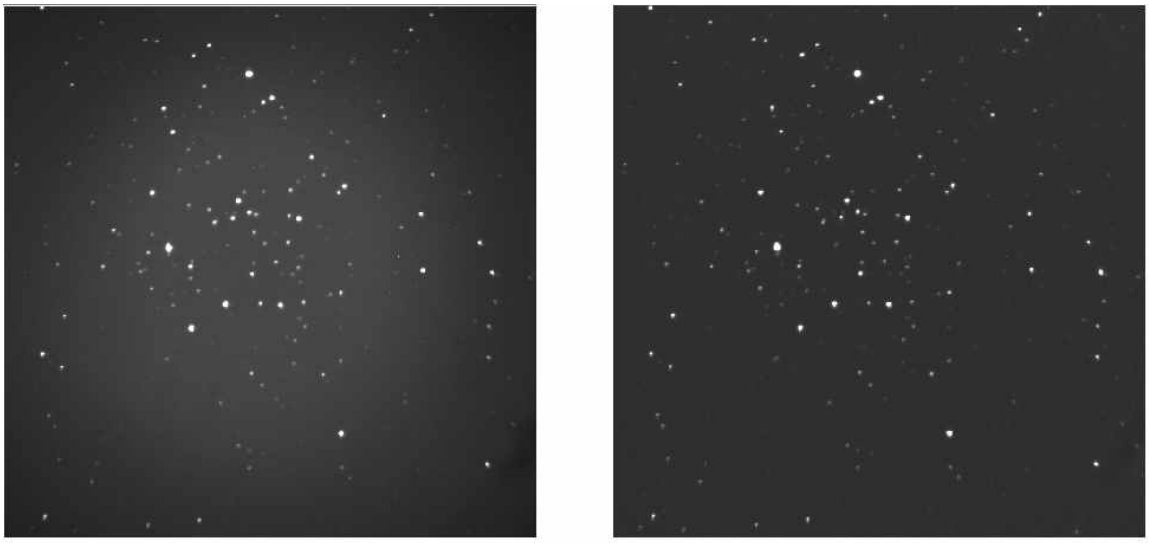
ccdred> ccdproc @obj20.list out=p//@obj20.list ccdtype="" fixpix- oversc- trim- zerocor+ darkcor+ flatcor+ zero=Zero dark=Dark20 flat=Flat*.fits

이제 전처리 작업이 완료된 p로 시작되는 파일들이 만들어졌을 것이다. [그림 13]과 같이, 전처리 과정을 통해 Bias, Dark, Flat이 제거

ccdred> ccdlist p*
pM50_b10_1.fits[512,512][real][unknown][B][ZDF]:M50
pM50_b10_2.fits[512,512][real][unknown][B][ZDF]:M50
pM50_b10_3.fits[512,512][real][unknown][B][ZDF]:M50
...

ZDF는 각각 Zero, Dark, Flat 보정이 완료되었다는 뜻이다.

tip 9. 지금까지 작업을 하면서 어렴풋이 느꼈을 수도 있지만, 헤더에 많은 정보를 기록하지 않는 교내 천문대에서 관측을 할 때는 약간 번거롭더라도 저장할 때부터 파일이름을 하나의 로그북 기록하듯이 저장을 하면 이후 자료처리가 훨씬 용이해진다. 가령 예시에 든 파일들은 '타겟_필터+찍은 순서_노출시간.fits' 와 같은 형식으로 파일명이 기록되어 있다. 즉, M50_V10_2.fits 파일은 'M50을 V필터로 2번째 찍은 사진이고 노출시간은 10초다' 라는 식이다. 보현산 천문 등 외부 천문대의 관측데이터는 대부분 헤더 정보가 상세하므로 hselect 명령어를 사용하여 헤더로 묶으면 되기에 굳이 이런 일을 할 필요는 없다.



[그림 13] 원본 이미지(좌)와 전처리가 완료된 이미지(우)

5) 불량화소 보정

[그림 14]의 좌측 이미지와 같이 불량화소가 있을 경우, fixpix를 이용해서 메울 수 있다. (불량 화소가 없다면, 다음 단계로 넘어가자.) 불량화소는 CCD 자체의 특성일 수 있고, 혹은 인공위성이 지나가거나, 지나치게 밝은 별에 의한 포화로 발생한다.

우선, 불량화소가 있는 파일들을 묶어준다.

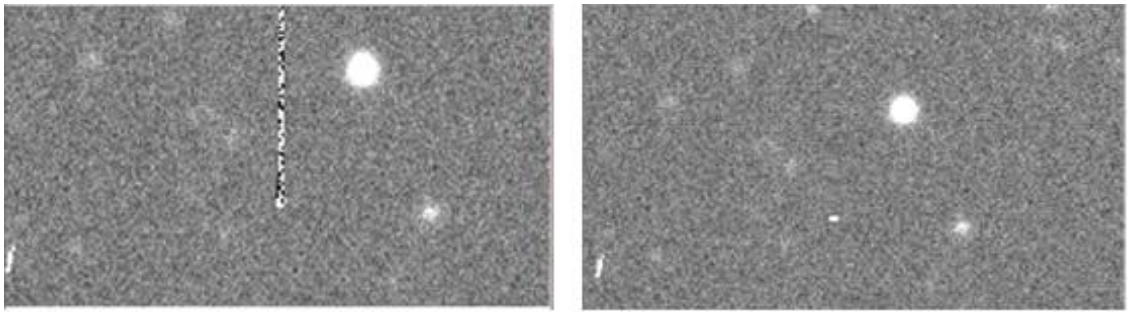
```
ccdred> vi fixpix
pccd058.fits
pccd059.fits
pccd060.fits
pccd061.fits
pccd062.fits
```

fixpix에 저장한 파일의 순서대로 badpixel의 x,y위치의 시작과 끝(x_i x_f y_i y_f)을 적어준다. ds9 을 열어서 image로 나타나는 x,y좌표이다.

```
ccdred> vi badpix
1012 1014 644 2048
1012 1014 644 2048
1012 1014 644 2048
1012 1014 644 2048
1012 1014 644 2048
```

Fixpix와 badpix 파일이 준비되었으면 fixpix 를 사용하여 다음과 같이 적어준다.

```
ccdred> fixpix @fixpix badpix
```



[그림 14] 불량화소 제거 전 이미지(좌)와 제거 후 이미지(우)

이미지를 열어서 확인해본다. [그림 14]의 우측 이미지처럼 불량화소가 제대로 제거 된 것을 알 수 있다.

```
ccdred> !ds9 pccd* &
```

6) 프린지 제거

일부 CCD에서는 I 밴드와 같은 장파장에서 물결치는 듯한 CCD 특유의 패턴이 나타난다. (없다면 다음 단계로 넘어가자.) 이를 프린지(Fringe)라 한다. 이를 제거해주기 위해서 마스터 프린지가 필요하다. 같은 노출 시간의 대상 이미지들을 중간 값으로 합쳐준다.

```
ccdred> imcombine @I.list output=Fringe combine=median reject=none rdnoise=15 gain=2.2
```

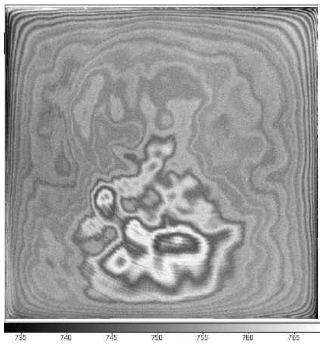
tip 10. 물론, 디더링(dithering)이라 하여 천체를 CCD 픽셀에서 이곳저곳 옮겨가는 작업을 관측 시 수행해야한다. 비단 프린지 제거 뿐만 아니라, CCD의 핫픽셀의 영향을 보정하기 위해서 반드시 관측시 디더링을 해주자. (물론 교내 천문대는 의도하지 않아도 자동으로 디더링을 해주는 훌륭한(?) 추적 능력을 갖추고 있다)

이러면 [그림 15]와 같은 마스터 프린지가 완성되었다. 다크 처리와 마찬가지로, 이 마스터 프린지를 모든 I 밴드 대상 이미지에서 제거해 준다. 천체의 이미지와 마스터 프린지 이미지의 값의 크기에 차이가 있기 때문에 바로 빼줄 수 없기에 배경 값으로부터 스케일을 맞춰줄 필요가 있다. 배경 값을 비교하는데는 여러 방법이 있지만, 예시에서는 imstat으로 측정하겠다.

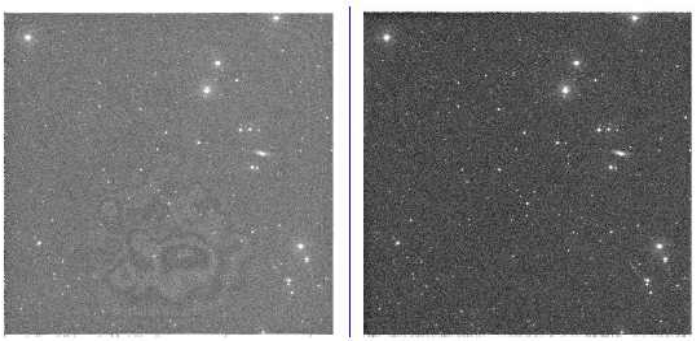
만약 처리할 이미지에서 별의 개수가 적다면 배경 값을 구하기 위해 imstat를 이미지 전체에 사용해도 큰 상관은 없다. 하지만 그렇지 않다면 별이 없는 부분 영역을 찾아서 그 부분에 대해서만 통계값을 구하도록 하자. 예를 들어 X좌표 100~200, Y좌표 50~150 영역에 별이 적을 경우, 명령어는 **imstat 파일명[100:200, 50:150]**이 된다.

우선, 마스터 프린지의 통계 값을 살펴보자.

```
ccdred> imstat Fringe.fits
```



[그림 15] 마스터 프린지



[그림 16] 프린지 제거 전(좌)과 후(우)

```
#                IMAGE        NPIX        MEAN      STDDEV         MIN         MAX
          Fringe.fits      262144       124.9       6.038       109.5       210.5
```

즉 현재 마스터 프린지의 평균값은 124.9가 되는 것을 알 수 있다. 만약 우리가 빼고자 하는 이미지들의 평균 값이 140.7이 나왔다 가정할 때 아래와 같이 imarith(IRAF 상에서 사칙연산을 계산할 때 사용) 명령어로 [그림 16]과 같이 프린지를 제거할 수 있다.

```
ccdred> imarith Fringe * 1.1265 Fringe_nor.fits
ccdred> imarith Obj_I – Fringe_nor.fits fObj_I
```

7) 이미지 정렬

같은 대상을 찍은 이미지도 별들의 위치가 조금씩 다르기 때문에, 기준 이미지에 대해 다른 이미지를 이동하는 작업이 필요하다. ds9에 정렬을 수행할 이미지를 띄우고 목록파일을 만들어준다.

```
ccdred> !ds9 pM* &
ccdred> ls pM50* > M50.list
ccdred> cat M50.list
pM50_b10_1.fits
pM50_b10_2.fits
pM50_b10_3.fits
pM50_u20_1.fits
pM50_u20_2.fits
pM50_u20_3.fits
...
```

그 다음 이미지 정렬 작업 수행 시 기준이 되어줄 별들을 정하기 위해 imexam의 설정을 [그림 17]과 같이 바꿔준다. imexam을 실행한 후

```
ccdred> imexam
Log file M50.cod open
```

기준 이미지(여기서는 제일 앞에 있는 pM50_b10_1.fits를 기준으로 잡는다. ds9에서 frame>

single > first를 순서대로 눌러주면 pM50_b10_1.fits가 나올 것이다.)에서 고루 분포한 5-10개의 별에 대해서 a키를 눌러준 후 q를 눌러 imexam을 종료한다.

그 다음으로 [그림 18]와 같이 keeplog와 logfile을 꺼주고 다시 imexam을 실행한다.

```
ccdred> imexam
```

기준 image에서 아무 별이나 골라서 a를 눌러 좌표를 얻고, 다른 프레임들에 대해서도(ds9의 frame에서 next를 누르자) 같은 별에 a키를 눌러서 좌표를 얻어준다.

[그림 19]와 같이, 각 프레임에서 해당 별의 좌표가 나오는 것을 알 수 있다. 기준 이미지에서 (142.94, 278.76)에 있던 별이, 그 다음 이미지에서는 (140.64, 279.18), (129.07, 279.72) 순으로 위치하는 것을 볼 수 있다. 이로부터 x(기준)-x(이미지), y(기준)-y(이미지)를 계산하여 M50.shf란 파일을 만든다.

```
cc> vi M50.shf
0.0 0.0
2.3 -0.4
13.9 -1.0
10.7 -1.4
5.9 -1.7
... ...
```

각 줄은 목록파일의 순서에 따라 x축과 y축이 이동한 정도를 의미한다. 목록 파일 중 제일 처음에 있는 천체가 기준이미지였다면, 당연히 첫줄은 0이 된다. 아래 두 줄은 cod파일의 2번째 줄에 있는 별의 좌표와 기준 별과의 좌표의 차이, 3번째 줄에 있는 별의 좌표와 기준 별과의 좌표의 차이를 뜻한다. 이제 [그림 20]과 같이 imalign 명령어를 통해 이미지를 정렬할 수 있다.

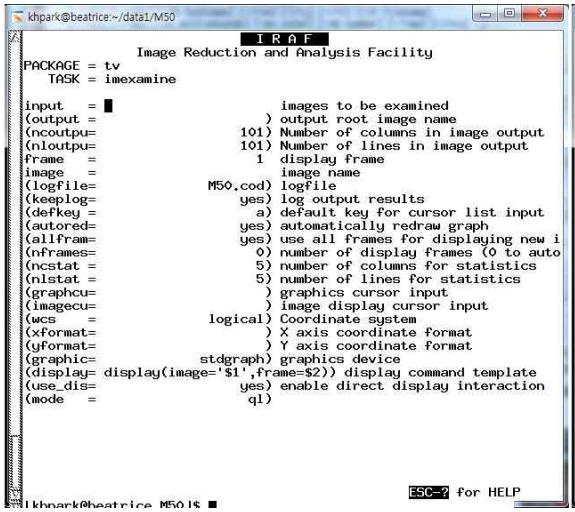
```
ccdred> epar imalign
```

이미지 정렬 작업이 끝나면 alp가 붙은 전처리·정렬 작업이 끝난 이미지가 생성되는 것을 볼 수 있다.

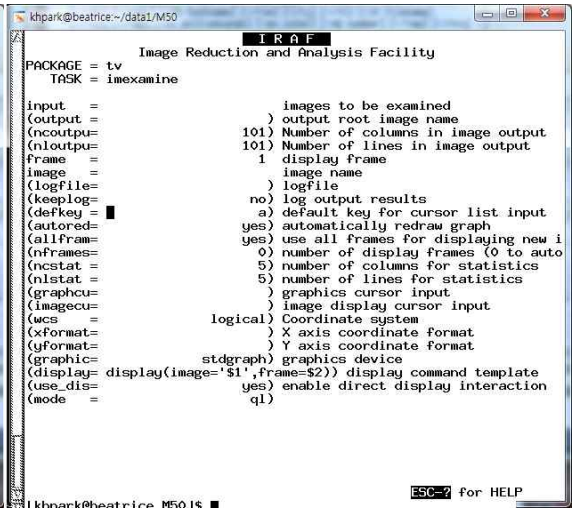
```
ccdred> ls alp*
alpM50_b10_1.fits   alpM50_u20_1.fits   alpM50_v10_1.fits
alpM50_b10_2.fits   alpM50_u20_2.fits   alpM50_v10_2.fits
alpM50_b10_3.fits   alpM50_u20_3.fits   alpM50_v10_3.fits
```

8) 이미지 합침

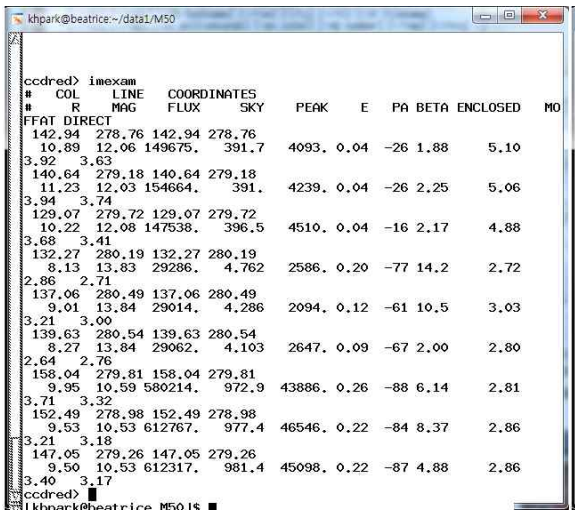
```
ccdred> ls alp*u* > M50u.list
ccdred> ls alp*b* > M50b.list
ccdred> ls alp*v* > M50v.list
```



[그림 17] imexam 설정 (기준 별 선정 전)

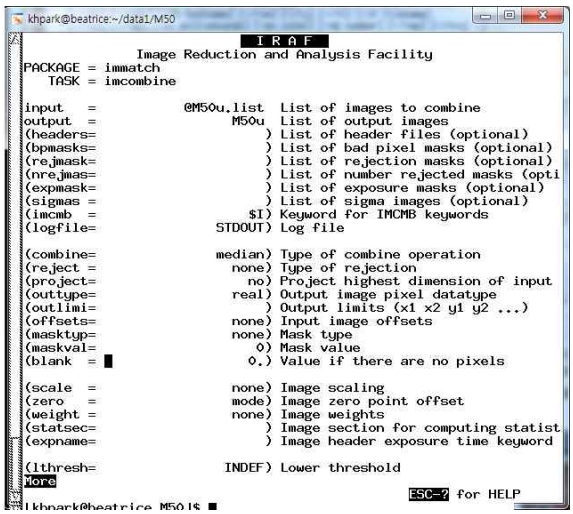


[그림 18] imexam 설정(기준 별 선정 후)

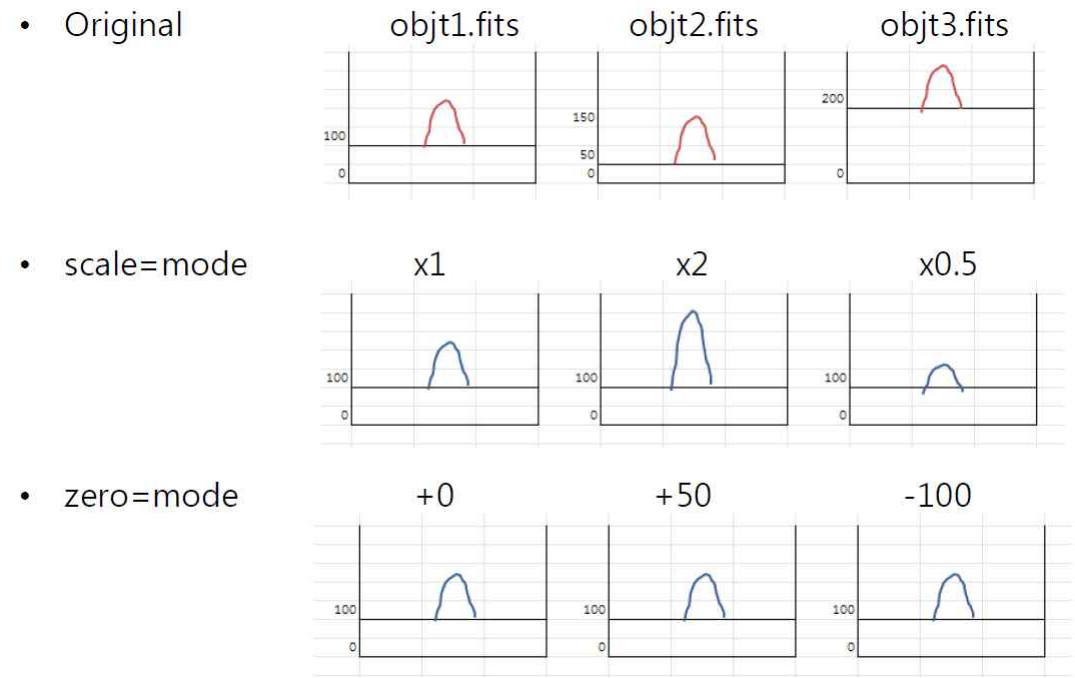


[그림 19] imexam으로 한 별의 좌표 측정

[그림 20] 이미지 정렬작업



[그림 21] 이미지 합침 작업



[그림 22] imcombine에서 scale과 zero를 mode로 둘 때의 효과

이제 [그림 21]과 같이 imcombine명령을 통해서 이미지들을 하나로 합칠 수 있다. (그림에는 보이지 않지만, 뒤에 RDnoise와 Gain값 역시 설정한다.) combine은 보통 평균과 중간값 둘 중에 고르면 된다. 중간값을 택하면 우주선이나 불량화소 등 튀는 값을 제거할 수 있다. 시상이 불안정하여 합치려는 영상의 천체의 PSF들이 각 장마다 다르다면 평균을 써야 한다. 중간값을 쓸 경우 플럭스 손실(flux loss)이 일어날 수 있다.

scale=none으로 하자. scale을 해주면 배경 값에 맞추어 대상들의 값이 재조정되어 ([그림 22] 참조), 그 천체의 ADU와 물리적 플럭스(physical flux) 간의 비율이 문제가 생기게 된다. 이는 나중 표준화 시에 문제가 된다.

또한, zero=mode로 해야 한다. 배경 값이 조금씩 달라질 수가 있는데, 평균을 택할 경우에는 문제가 없지만, 중간값을 택할 경우에는 역시 플럭스 손실이 생길 수 있다. zero=mode를 선택해 주면 배경을 첫 번째 이미지에 대해서 맞춰주어 플럭스 손실을 막을 수 있다.

명령어는 아래와 같이 입력하면 된다.

```
ccdred> imcombine @M50u.list output=M50u combine=median reject=none rdnoise=15 gain=2.2 scale=none zero=mode
ccdred> imcombine @M50b.list output=M50b combine=median reject=none rdnoise=15 gain=2.2 scale=none zero=mode
ccdred> imcombine @M50v.list output=M50v combine=median reject=none rdnoise=15 gain=2.2 scale=none zero=mode
```

6) 컬러 이미지 제작

ds9을 이용하면 간편하게 컬러이미지를 제작할 수 있다. 터미널에 다음과 같이 입력하자.

```
$ ds9 -rgb -red M50v.fits -green M50b.fits –blue M50u.fit &
```

이 때, 당연히 사용하고자 하는 이미지는 위치 정렬이 완료되어 있어야하며, blue-green-red 순으로 단파장에서 장파장으로 가는 규칙은 지켜주어야 한다.

각 이미지의 스케일이 다르기 때문에 특정 색깔이 유달리 강한 것을 볼 수 있다.
위의 메뉴에서 Frame > RGB, 그리고 Scale > Scale Paramter를 띄우자.

RGB 탭에서 각 색상을 클릭하고, Scale Parameter 탭에서 각각 최소, 최대값을 바꿔가며 가장 마음에 들도록 '꾸미자.'

스케일 조정이 끝난 뒤에는 File>Save Image As를 통해 원하는 파일 형식으로 저장이 가능하다.

ds9 이외에도 IDL이나 포토샵 등을 이용해서도 컬러 이미지를 만들 수 있다.

tip11. RDnoise나 Gain값은 CCD 회사 홈페이지나 헤더를 통해서 얻을 수 있다. 하지만 이 값이 맞는지 의심이 될 때는 다음과 같은 계산 방법으로 확인해 볼 수 있다. 우선 3쌍 이상의 bias 이미지와 3쌍 이상의 flat이미지가 필요하다.

한쌍, 즉 2장의 flat 이미지의 '평평한 부분' 에서 각각의 값을 더해준 다음 해당 범위 내의 모든 픽셀에 대해 그 값을 평균낸 것을 Fs라 정의한다. 그 다음 이번에는 차이를 구해 해당 범위 내의 모든 픽셀에 대해 구한 표준편차를 SEf라고 정의한다. 한쌍의 Bias 이미지에도 동일한 작업을 수행하여 이를 각각 Bs, SEb라고 정의한다.
이 때 Gain은 $G=(Fs-Bs)/(SEf^2-SEb^2)$가 되고 RDnoise는 $RDnoise=G\times SEb/\sqrt{2}$가 된다.

이 값들을 3쌍 이상에 대해서 구한 뒤 그 값들을 평균내어 Gain과 RDnoise를 측정할 수 있다.

Ⅲ. 참고 문헌

-Bruce L. Gray. 2007, 'Exoplanet observing for amateurs',2007
-심현진, '관측 1 보충수업 기본 명령어 정리 >_<', 2005
-황호성, 이명균, 'IRAF ABC', 2004

Appendix A. IRAF 설치하기

본 문서는 Fedora 12 (64bit) + IRAF 2.15를 기반으로 작성되었다. 버전 혹은 개인 컴퓨터 설정에 따라 링크 위치 및 설치 방법이 다소간 차이가 있으므로, 문제가 발생 시 인터넷에 검색하거나, 혹은 각자 기지를 가지고 이것저것 시도를 해보거나, 주위 선배에게 문의하는 등 유연하게 해결하자.

터미널을 띄우고, 관리자 계정으로 전환한다.

```
# su
```

IRAF 및 X11IRAF 설치에 필요한 것들을 받고 설치하자

```
# yum install tcsh
# yum install libXmu.so.6 libncurses.so.5
```

1) X11IRAF 설치

IRAF가 설치될 폴더를 만들어주자.

```
# mkdir /iraf
# mkdir /iraf/iraf
# mkdir /iraf/iraf/local
# mkdir /iraf/iraf/x11iraf
```

http://iraf.net/ftp/iraf/x11iraf/ 에 가서 redhat 버전을 다운 받고 /iraf/iraf/x11iraf로 복사한 후, 압축을 풀어주고 설치하자.

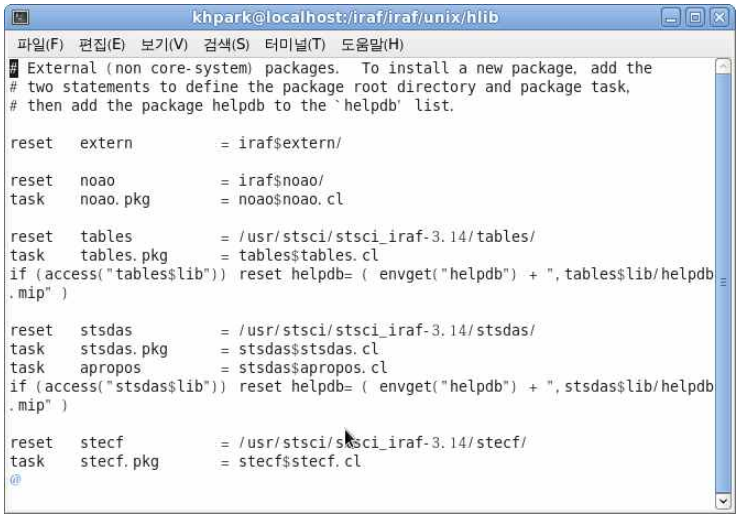
```
# cp /home/khpark/다운로드/x11iraf-v1.5DEV-bin.redhat.tar.gz .
# tar -xzvf x11iraf-v1.5DEV-bin.redhat.tar.gz
# ./install
```

터미널에서 xgterm -sb&를 입력하여, xgterm을 실행하여 설치 여부를 확인하자.

2) IRAF 설치

시스템 > 관리 > 사용자 및 그룹 항목 (System > Administration > Users and Groups) 에 가서 유저를 추가하자. [그림 A.1]과 같이 유저 ID는 iraf, 로그인 쉘은 /bin/tcsh, 홈 디렉토리는 /iraf/iraf/local로 잡아준다.

[그림 A.1] iraf 계정 추가 화면



[그림 A.2] extern.pkg의 내용

iraf 폴더의 권한을 iraf 계정으로 넘겨준다.

```
# chown –R iraf:iraf /iraf
```

(혹시 에러가 난다면 그룹 지정에 관련된 문제이다. chown -R iraf /iraf를 입력해보자.)

http://iraf.noao.edu/ 에 가서 Linux 64-bit iraf를 다운 받고, /iraf/iraf폴더로 옮겨 압축을 해제한다.

```
# cd /iraf/iraf
# cp /home/khpark/다운로드/iraf.lnux.x86_64.tar.gz .
# tar -xzvf iraf.lnux.x86_64.tar.gz
```

설치 폴더로 이동한 뒤, 터미널의 계정을 iraf로 옮겨 다음과 같은 명령을 실행한다.

```
# cd /iraf/iraf/unix/hlib/
# su iraf
> setenv iraf /iraf/iraf
> source irafuser.csh
```

다시 관리자 계정으로 접속 한 다음, 설치를 해주자.

```
#./install
```

이 때 다른 건 건드리지 않아도 되지만, 네트워크 설치에 관한 명령은 no로 해준다.

```
Would you like to return to networking setup?  (yes): n
```

설치가 다 되었으면, xgterm 상에서 mkiraf로 login.cl 파일을 만들고, cl로 IRAF를 실행하여 설치가 제대로 되었는지 확인해보자.

3) STSCI의 확장툴 설치

http://www.stsci.edu/resources/software_hardware/stsdas/download 에서 STSDAS, TABLE, STECF 통합본의 linux 버전을 다운받는다.

다운로드가 완료되면 압축을 풀고, 설치 스크립트를 실행해준다.

```
# tar zxf stsci_iraf-3.14.linux.tar.gz
# cd stsci_iraf-3.14
# ./install_helper
```

혹, 자동 설치가 되지 않는다면, /iraf/iraf/unix/hlib/extern.pkg에서 수동으로 경로를 잡아주면 된다. 위 압축 파일을 해제해보면, extern.pkg 파일이 들어 있고, 이 파일을 보면 STSDAS, TABLE, STECF의 경로 설정이 /usr/stsci로 잡혀있으므로, 파일을 그곳으로 옮겨준 후, [그림 A.2]처럼 압축 해제 된 파일 내의 extern.pkg 내용을 /iraf/iraf/unix/hlib/ extern.pkg 에 복사해서 추가한다.

```
# mkdir /usr/stsci/
# cd /usr/stsci
# cp /iraf/iraf/stsci_iraf-3.14.redhat.tar.gz .
# tar -xzvf stsci_iraf-3.14.redhat.tar.gz
```

IRAF 로그인 후, stsdas를 입력해보자. stsdas> 로 넘어가면 설치가 성공적으로 끝났다.

4) DS9 설치

http://hea-www.harvard.edu/RD/ds9/ 에서 linux64 의 DS9을 받고, 압축을 푼다.

```
# tar -xzvf ds9.linux64.6.2.tar.gz
```

압축 해제된 ds9 파일을 /usr/local/bin으로 옮기면 설치가 끝난다.

```
# mv ds9 /usr/local/bin
```

터미널에 ds9&을 입력하여 ds9이 실행되는 것을 확인해보자.