Subject: Re: Fuzzy searching of FITS header
Posted by David Fanning on Tue, 21 Dec 2010 22:23:04 GMT
View Forum Message <> Reply to Message

Gray writes:

> Does anyone have a routine that allows for fuzzy searching of FITS
> header keywords? For example, I have a header where I want the value
> of the following keywords:
>
> D001DEXP
> D002DEXP
> ...
>
> I don't know a priori how many of them there are, and it would be nice
> to be able to have a parsing program that would let me give 'DEXP' as
> a search string and return all of them. Any suggestions?

StrPos?

> I know I
> could construct something that did searches for up to 1000 keywords
> (000-999) and aborted when it didn't find anything, but that would
> defeat the purpose of IDL vectorization, wouldn't it?

It's already vectorized! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Fuzzy searching of FITS header
Posted by wlandsman on Tue, 21 Dec 2010 22:59:33 GMT
View Forum Message <> Reply to Message

On Tuesday, December 21, 2010 5:10:27 PM UTC-5, Gray wrote:
> Hi all,
>
> Does anyone have a routine that allows for fuzzy searching of FITS
> header keywords? For example, I have a header where I want the value
> of the following keywords:

>
> D001DEXP
> D002DEXP

For quicklook purposes hgrep.pro
( http://idlastro.gsfc.nasa.gov/ftp/pro/misc/hgrep.pro )
is a wrapper around STRPOS.   There would be a couple of extra lines of code to make something automated.  --Wayne

---

## Subject: Re: Fuzzy searching of FITS header
## Posted by Gray on Tue, 21 Dec 2010 23:00:36 GMT

View Forum Message <> Reply to Message

On Dec 21, 5:23 pm, David Fanning <n...@dfanning.com> wrote:
> Gray writes:
>> Does anyone have a routine that allows for fuzzy searching of FITS
>> header keywords?  For example, I have a header where I want the value
>> of the following keywords:
>
>> D001DEXP
>> D002DEXP
>> ...
>
>> I don't know a priori how many of them there are, and it would be nice
>> to be able to have a parsing program that would let me give 'DEXP' as
>> a search string and return all of them.  Any suggestions?
>
> StrPos?
>
>> I know I
>> could construct something that did searches for up to 1000 keywords
>> (000-999) and aborted when it didn't find anything, but that would
>> defeat the purpose of IDL vectorization, wouldn't it?
>
> It's already vectorized! :-)
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Well, sure.  But what I WAS using was SXPAR, which pads the search string w/ spaces and searches 8 characters.

I'm envisioning a routine that uses lists and hashes to return the
keyword and value for all keywords that match any number of search
strings... do I have the time/energy to write it?

---

Subject: Re: Fuzzy searching of FITS header
Posted by David Fanning on Tue, 21 Dec 2010 23:38:02 GMT
View Forum Message <> Reply to Message

Gray writes:

> I'm envisioning a routine that uses lists and hashes to return the
> keyword and value for all keywords that match any number of search
> strings... do I have the time/energy to write it?

I think I would still start with STRPOS and be
done in less than a minute. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---

Subject: Re: Fuzzy searching of FITS header
Posted by penteado on Wed, 22 Dec 2010 01:11:20 GMT
View Forum Message <> Reply to Message

On Dec 21, 9:00 pm, Gray <grayliketheco...@gmail.com> wrote:
> I'm envisioning a routine that uses lists and hashes to return the
> keyword and value for all keywords that match any number of search
> strings... do I have the time/energy to write it?

I was just writing something like that as an example for a class.
Similar to what the new graphics do to find elements by search strings
(like axes=plot['*axis*']). I will post it later.

---

Subject: Re: Fuzzy searching of FITS header
Posted by Marc Buie on Wed, 22 Dec 2010 03:49:30 GMT
View Forum Message <> Reply to Message

Wayne -

Why can't this be handled with

dexp=sxpar(hdr,'D*DEXP')

sxpar already handles

naxis=sxpar(hdr,'NAXIS*')

It seems to me that this is a simple extension of what sxpar already does.  All you have to do is support a regular expression and take the vector result.  However, this isn't what I would call a fuzzy search but it would definitely be a handy extension to sxpar.

--Marc

Subject: Re: Fuzzy searching of FITS header
Posted by penteado on Wed, 22 Dec 2010 04:50:39 GMT
View Forum Message <> Reply to Message

On Dec 21, 11:11 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Dec 21, 9:00 pm, Gray <graylikethecо...@gmail.com> wrote:
>
>> I'm envisioning a routine that uses lists and hashes to return the
>> keyword and value for all keywords that match any number of search
>> strings... do I have the time/energy to write it?
>
> I was just writing something like that as an example for a class.
> Similar to what the new graphics do to find elements by search strings
> (like axes=plot['*axis*']). I will post it later.

It is not complete (not even documented), but it already does that:

http://www.ppenteado.net/idl/pp_lib/src/pp_readfits__define. pro

I started this as an example for a class, but due to some recent needs
I noticed when using fits files, I intend to give it a more complete
functionality, with things like processing coordinates through wcs,
calculating wavelengths, and allowing to edit and save files.

An example of how it works now:

IDL> fits=pp_readfits('test.fits')

```
% READFITS: Now reading 256 by 256 array
IDL> help,fits.data
<Expression>    LONG      = Array[256, 256]
IDL> help,fits.header
<Expression>    STRING    = Array[184]
IDL> help,fits.variables
<Expression>    HASH  <ID=746  NELEMENTS=181>
IDL> help,fits.descriptions
<Expression>    HASH  <ID=1117  NELEMENTS=181>
IDL> print,(fits.variables)['NAXIS']
2
IDL> print,(fits.descriptions)['NAXIS']
Number of axes
IDL> print,fits['NAXIS']
NAXIS: 2
IDL> print,fits['NAXIS*']
NAXIS2: 256
NAXIS: 2
NAXIS1: 256
```

---

## Subject: Re: Fuzzy searching of FITS header
## Posted by wlandsman on Wed, 22 Dec 2010 13:06:40 GMT

View Forum Message <> Reply to Message

---

On Tuesday, December 21, 2010 10:49:30 PM UTC-5, Marc Buie wrote:
> Wayne -
>
> Why can't this be handled with
>
> dexp=sxpar(hdr,'D*DEXP')
>
> sxpar already handles
>
> naxis=sxpar(hdr,'NAXIS*')
>
> It seems to me that this is a simple extension of what sxpar already does.

Well, there is a FITS convention for reserved keyword names followed by sequential integers (e.g. NAXIS1, NAXIS2, NAXIS3... ), where (with one exception) you can be sure that the returned values will all be of the same type (in this case integers). But for a general wildcard (e.g. 'D*EXP') the returned values might be a mixture of strings, integers and floats. That is why Paulo's list/hash approach seems preferable this case. --Wayne

P.S. The one exception for reserved keyword names is TSCALi for converting 16 bit integers in a binary table to double/float. In some cases TSCALi returns a float and in other cases it returns a double. That is why MRDFITS currently has a limitation of requiring either all conversions to float or all conversions to double. In some other FITS routines I get around this limitation by using pointers, but it is a pain.   It is a nice application for the new LIST datatype.

---

Subject: Re: Fuzzy searching of FITS header
Posted by Gray on Wed, 22 Dec 2010 13:40:57 GMT
View Forum Message <> Reply to Message

On Dec 22, 8:06 am, wlandsman <wlands...@gmail.com> wrote:
> On Tuesday, December 21, 2010 10:49:30 PM UTC-5, Marc Buie wrote:
>> Wayne -
>
>> Why can't this be handled with
>
>> dexp=sxpar(hdr,'D*DEXP')
>
>> sxpar already handles
>
>> naxis=sxpar(hdr,'NAXIS*')
>
>> It seems to me that this is a simple extension of what sxpar already does.
>
> Well, there is a FITS convention for reserved keyword names followed by sequential integers (e.g. NAXIS1, NAXIS2, NAXIS3... ), where (with one exception) you can be sure that the returned values will all be of the same type (in this case integers).   But for a general wildcard (e.g. 'D*EXP') the returned values might be a mixture of strings, integers and floats.    That is why Paulo's list/hash approach seems preferable this case.   --Wayne
>
> P.S. The one exception for reserved keyword names is TSCALi for converting 16 bit integers in a binary table to double/float.    In some cases TSCALi returns a float and in other cases it returns a double.    That is why MRDFITS currently has a limitation of requiring either all conversions to float or all conversions to double.    In some other FITS routines I get around this limitation by using pointers, but it is a pain.   It is a nice application for the new LIST datatype.

Here's a very basic writeup using lists and hashes and FXPAR() (since
I didn't feel like re-writing the keyword parsing rules).

```
FUNCTION hdregex, header, search
  n = n_elements(search)
  keys = strmid(header,0,8)
  if (n lt 2) then begin
    this = where(stregex(keys,search,/fold,/bool),count)
    if (count eq 0) then return, !null
    out = hash(keys[this])
    for i=0,count-1 do out[keys[this[i]]] = $
      fxpar(header,keys[this[i]],start=this[i],precheck=0)
  endif else begin
    out = list(length=n)
    foreach key,search,i do begin
      this = where(stregex(keys,search,/fold,/bool),count)
      if (count eq 0) then continue
      temp = hash(keys[this])
      for j=0,count-1 do out[keys[this[j]]] = $
        fxpar(header,keys[this[j]],start=this[j],precheck=0)
      out[i] = temp
    endforeach
  endelse
  return, out
end
```

Subject: Re: Fuzzy searching of FITS header
Posted by Gray on Wed, 22 Dec 2010 13:45:44 GMT
View Forum Message <> Reply to Message

On Dec 22, 8:40 am, Gray <graylikethecо...@gmail.com> wrote:
> On Dec 22, 8:06 am, wlandsman <wlands...@gmail.com> wrote:
>
>
>
>
>
>
>
>
>
>> On Tuesday, December 21, 2010 10:49:30 PM UTC-5, Marc Buie wrote:
>>> Wayne -
>
>>> Why can't this be handled with
>
>>> dexp=sxpar(hdr,'D*DEXP')
>
>>> sxpar already handles
>
>>> naxis=sxpar(hdr,'NAXIS*')
>
>>> It seems to me that this is a simple extension of what sxpar already does.
>
>> Well, there is a FITS convention for reserved keyword names followed by sequential integers (e.g. NAXIS1, NAXIS2, NAXIS3... ), where (with one exception) you can be sure that the returned values will all be of the same type (in this case integers). But for a general wildcard (e.g. 'D*EXP') the returned values might be a mixture of strings, integers and floats. That is why Paulo's list/hash approach seems preferable this case. --Wayne
>
>> P.S. The one exception for reserved keyword names is TSCALi for converting 16 bit integers

in a binary table to double/float.    In some cases TSCALi returns a float and in other cases it returns a double.    That is why MRDFITS currently has a limitation of requiring either all conversions to float or all conversions to double.    In some other FITS routines I get around this limitation by using pointers, but it is a pain.   It is a nice application for the new LIST datatype.

```
>
>  Here's a very basic writeup using lists and hashes and FXPAR() (since
>  I didn't feel like re-writing the keyword parsing rules).
>
>  FUNCTION hdregex, header, search
>    n = n_elements(search)
>    keys = strmid(header,0,8)
>    if (n lt 2) then begin
>      this = where(stregex(keys,search,/fold,/bool),count)
>      if (count eq 0) then return, !null
>      out = hash(keys[this])
>      for i=0,count-1 do out[keys[this[i]]] = $
>        fxpar(header,keys[this[i]],start=this[i],precheck=0)
>    endif else begin
>      out = list(length=n)
>      foreach key,search,i do begin
>        this = where(stregex(keys,search,/fold,/bool),count)
>        if (count eq 0) then continue
>        temp = hash(keys[this])
>        for j=0,count-1 do out[keys[this[j]]] = $
>          fxpar(header,keys[this[j]],start=this[j],precheck=0)
>        out[i] = temp
>      endforeach
>    endelse
>    return, out
>  end
```

Oops, typos (that's what I get for copy/paste):

```
FUNCTION hdregex, header, search
  n = n_elements(search)
  keys = strmid(header,0,8)
  if (n lt 2) then begin
    this = where(stregex(keys,search,/fold,/bool),count)
    if (count eq 0) then return, !null
    out = hash(keys[this])
    for i=0,count-1 do out[keys[this[i]]] = $
      fxpar(header,keys[this[i]],start=this[i],precheck=0)
  endif else begin
    out = list(length=n)
    foreach key,search,i do begin
      this = where(stregex(keys,key,/fold,/bool),count)
      if (count eq 0) then continue
    temp = hash(keys[this])
    for j=0,count-1 do temp[keys[this[j]]] = $
      fxpar(header,keys[this[j]],start=this[j],precheck=0)
    out[i] = temp
   endforeach
  endelse
  return, out
end
```

Subject: Re: Fuzzy searching of FITS header
Posted by penteado on Wed, 22 Dec 2010 19:44:55 GMT
View Forum Message <> Reply to Message

On Dec 22, 2:50 am, Paulo Penteado <pp.pente...@gmail.com> wrote:
> It is not complete (not even documented), but it already does that:
>
>  http://www.ppenteado.net/idl/pp_lib/src/pp_readfits__define. pro
>
> I started this as an example for a class, but due to some recent needs
> I noticed when using fits files, I intend to give it a more complete
> functionality, with things like processing coordinates through wcs,
> calculating wavelengths, and allowing to edit and save files.
>
> An example of how it works now:
>

```
> IDL> fits=pp_readfits('test.fits')
> % READFITS: Now reading 256 by 256 array
> IDL> help,fits.data
> <Expression>    LONG      = Array[256, 256]
> IDL> help,fits.header
> <Expression>    STRING    = Array[184]
> IDL> help,fits.variables
> <Expression>    HASH  <ID=746  NELEMENTS=181>
> IDL> help,fits.descriptions
> <Expression>    HASH  <ID=1117  NELEMENTS=181>
> IDL> print,(fits.variables)['NAXIS']
> 2
> IDL> print,(fits.descriptions)['NAXIS']
> Number of axes
> IDL> print,fits['NAXIS']
> NAXIS: 2
> IDL> print,fits['NAXIS*']
> NAXIS2: 256
> NAXIS: 2
> NAXIS1: 256
```

Now that I thought more about this, it seems it would be better to not always return hashes from the brackets overload. If only one element is found, returning directly the value seems more natural ans useful than a 1-element hash. And if none is found, returning !null seems better than an empty hash.