Subject: Re: Reading images from a socket
Posted by Roberto Monaco on Thu, 31 Jul 2003 13:45:19 GMT
View Forum Message <> Reply to Message

Sure!! Thanks,
Roberto

"Andrew Cool" <andrew.cool@dsto.defence.gov.au> wrote in message
news:c6d70400.0307271513.4379d1dd@posting.google.com...
> "Roberto Monaco" <rmonaco@coresw.com> wrote in message
news:<bfqivl$hk0vb$1@ID-201966.news.uni-berlin.de>...
>> I am trying to use a socket to read JPEG images from the Web.
>>
>> I connect to the remote server, and get to the point of sending the GET
to
>> request the image. But at this point I can not read from this unit by
using
>> READ_JPEG (socket is not supported).
>>
>> I tried to read it first into a buffer, and then read it from the buffer
>> with READ_JPEG. But I have no idea how to dimension the buffer correctly
>> beforehand (how large the image is). Even if I make it large enough
>> READ_JPEG gives me error reading from the buffer (conflicting keywords).
>>
>> Also, in this case I know it is a JPEG image, but in general I need to
>> understand the type of image I am reading into the buffer, which I do
not
>> know (in the HTML I have only "img src=name" from where I have no idea).
>>
>> Any hints?
>>
>> Many thanks,
>> Roberto
>
> Hi Roberto,
>
> You can't use Read_JPEG, or any of the other image reading routines
> over
> a Socket.
>
> What you can do is just read the image file as a sequence of bytes.
> Something like this :-
>

```
>   Openw,outlun,'my_new_image.tmp',/GET
>   x=0B
>   While not EOF(socket_lun) Do begin
>     ReadU,socket_lun,x
>     writeu,outlun,x
>   End
>   Free_lun,socket_lun
>   Free_lun,outlun
>
>   Then use IDL's QUERY_IMAGE function to find out what sort of
>   image 'my_new_image.tmp' really is. Rename the .tmp file to suit.
>
>   That easy! ;-)
>
>   Andrew
>
>   (FTP using IDL is also possible. In fact easier, because you can
>    readily return the filesize before you download it.)
>
>
>
>  The QUERY_IMAGE function determines whether a file is recognized as a
>  supported image file. QUERY_IMAGE first checks the filename suffix,
>  and if found, calls the corresponding QUERY_ routine. For example, if
>  the specified file is image.bmp, QUERY_BMP is called to determine if
>  the file is a valid .bmp file. If the file does not contain a filename
>  suffix, or if the query fails on the specified filename suffix,
>  QUERY_IMAGE checks against all supported file types. If the file is a
>  supported image file, an optional structure containing information
>  about the image is returned. If the file is not a supported image
>  file, QUERY_IMAGE returns 0.
>
>  Syntax
>
>  Result = QUERY_IMAGE ( Filename[, Info] [, CHANNELS=variable] [,
>  DIMENSIONS=variable] [, HAS_PALETTE=variable] [, IMAGE_INDEX=index] [,
>  NUM_IMAGES=variable] [, PIXEL_TYPE=variable] [,
>  SUPPORTED_READ=variable] [, SUPPORTED_WRITE=variable] [,
>  TYPE=variable] )
>
>  Return Value
>
>  Result is a long with the value of 1 if the query was successful (the
>  file was recognized as an image file) or 0 on failure. The return
>  status will indicate failure for files that contain formats that are
>  not supported by the corresponding READ_* routine, even though the
>  file may be valid outside the IDL environment.
>
>  Arguments
>  Filename
>
>  A scalar string containing the name of the file to query.
>
```

> Info
>
> An optional anonymous structure containing information about the
> image. This structure is valid only when the return value of the
> function is 1. The Info structure for all image types has the
> following fields:
>
>
>
> Tag Type
> CHANNELS Long
> DIMENSIONS Two-dimensional long array
> FILENAME Scalar string
> HAS_PALETTE Integer
> IMAGE_INDEX Long
> NUM_IMAGES Long
> PIXEL_TYPE Integer
> TYPE Scalar string