Rotinas de Manipulação de Arquivos

No IDL existem muitas rotinas de manipulação de arquivos. Uma lista destas rotinas, com uma explicação da sua finalidade, está na tabela abaixo.

Rotinas	Finalidade
FILEPATH	Constrói uma *string* contendo o caminho absoluto do arquivo.
FILE_BASENAME	Esta função retorna o nome base do arquivo. O nome base é o segmento mais à direta no final do caminho até o arquivo.
FILE_DIRNAME	Esta função retorna o nome do diretório do caminho até o arquivo. O nome do diretório é todo o caminho até o arquivo exceto o segmento mais à direita.
FILE_EXPAND_PATH	Qualificação total do caminho do diretório e até o arquivo.
FILE_CHMOD	Permite que você altere as permissões de acesso ao arquivo.
FILE_COPY	Nos permite copiar arquivos, e ou diretórios dos arquivos, para uma nova localização.
FILE_DELETE	Apaga arquivos ou esvazia diretórios.
FILE_INFO	Esta função retorna informações sobre o arquivo, incluindo seu nome, tamanho, permissões, acessos e tempos de modificações.
FILE_LINES	Informa o número de linhas de texto contidas no arquivo.
FILE_LINK	Este procedimento cria links para arquivos UNIX, regulares e simbólicos.
FILE_MKDIR	Cria um novo diretório nos arquivos de sistema.
FILE_MOVE	Move arquivos, ou diretórios de arquivos, para uma nova localização.
FILE_READLINK	Retorna o caminho apontado para o Link simbólico do UNIX.
FILE_SAME	Esta função é usada para determinar se dois arquivos com nomes diferentes referem-se ao mesmo arquivo fundamentado.
FILE_SEARCH	Encontra os arquivos cujos os nomes combinam com a *string* especificada.
FILE_TEST	De o caminho até o arquivo, que esta rotina retorna 1 se o arquivo existir, senão ele retorna 0.
FILE_WHICH	Procura por um arquivo específico no caminho de busca do diretório.
FSTAT	Relata informações sobre um arquivo aberto.
DIALOG_PICKFILE	Esta rotina nos permite selecionar interativamente um arquivo, ou diversos arquivos, ou um diretório.
EOF	Esta função uma unidade específica de um arquivo para a condição de fim do arquivo (end-of-file).
FLUSH	Este procedimento força todas as saída protegidas nas unidades especificadas do arquivo a ser escrito.
PATH_SEP	Retorna um delimitador apropriado do caminho de um arquivo para o atual sistema de operação.
COPY_LUN	Copia os dados entre dois arquivos abertos.
POINT_LUN	Marca ou obtém a posição atual do ponteiro de um arquivo em um arquivo especificado.
SKIP_LUN	Lê dados de um arquivo aberto e move o arquivo de

	ponteiro.
TRUNCATE_LUN	Este procedimento divide os conteúdos de um arquivo (que deve estar aberto em modo de escrita) na posição atual do ponteiro de arquivo.
SOCKET	Abre um socket TCP/IP para o cliente como uma unidade de arquivo do IDL.

Abaixo estão alguns exemplos de uso das rotinas da tabela acima.

Usando o *FILE_WhICH* para localizar o arquivo contendo o procedimento *LOADCT*.

```
IDL> file = file_which('loadct.pro')
IDL> print, file
        C:\RSI\IDL61\lib\loadct.pro
```

Extraindo o nome base e o nome do diretório deste arquivo com o *FILE_BASENAME* e o *FILE_DIRNAME*.

```
IDL> print, file_basename(file)
        loadct.pro
IDL> print, file_dirname(file)
C:\RSI\IDL61\lib
```

Use o *FILE_LINES* para contar o número de linhas que temos no arquivo **loadct.pro**.

```
IDL> print, file_lines(file)
   185
```

Exiba o diretório atual com *FILE_EXPAND_PATH*.

```
IDL> print, file_expand_path('.')
        C:\RSI\IDL61
```

Exiba o nome dos arquivos começados com a letra a no subdirectório **lib** do IDL.

```
IDL> str = !dir + path_sep() + 'lib' + path_sep() + 'a*'
IDL> print, file_basename(file_search(str, /fold_case))
        a_correlate.pro adapt_hist_equal.pro amoeba.pro annotate.pro array_indices.pro arrow.pro
        ascii_template.pro
```

Veja o IDL Online Help para maiores informações das rotinas listadas na tabela acima.

Rotinas de Alto Nível para Arquivos

Rotinas de alto nível para arquivos são fáceis de serem usadas, mas lhe dão menos controle sobre a descrição de dados em um arquivo no formato ASCII ou binário.

Texto Simples e Arquivos Binários

A tabela abaixo nos apresenta quatro rotinas designadas para simplificar o processo de leitura de textos simples e de arquivos binários para o IDL.

Rotinas	Finalidade
ASCII_TEMPLATE	Um programa de UI (interface para o usuário) que pode ser utilizado para descrever o formato de um arquivo de texto. Retornando uma variável de estrutura que possa ser usada pelo *READ_ASCII* para ler o arquivo.
READ_ASCII	Lê os dados de um arquivo de texto para uma variável de estrutura. O formato deste arquivo pode ser especificado com palavras-chave ou pelo *ASCII_TEMPLATE*.
BINARY_TEMPLATE	Como o *ASCII_TEMPLATE*, um programa de UI que pode ser usado para descrever o conteúdo de um arquivo binário. Retornando uma variável de estrutura que possa ser utilizada pelo *READ_BINARY* para ler o arquivo.
READ_BINARY	Lê os dados de um arquivo binário para uma estrutura ou uma matriz. A organização deste arquivo pode ser especificada por palavras-chave ou pelo *BINARY_TEMPLATE*.

Abra o arquivo **ascii.txt** no subdiretório (do diretório de instalação do IDL) **examples**/**data** com um editor de texto a sua escolha. Note que existem quatro linhas de cabeçalho, seguidas por uma linha em branco, depois 7 colunas e 15 linhas com uma vírgula delimitando cada dado. Abaixo temos um exemplo de como usar o *READ_ASCII* para ler as informações deste arquivo para o IDL.

```
IDL> x = filepath('ascii.txt', subdir = ['examples', 'data'])
IDL> dados = read_ascii (x, data_start=5, header=cabeçalho)
IDL> help, dados, /structures
** Structure <10a0a30>, 1 tags, length=420, data length=420, refs=1:
   FIELD1        FLOAT     Array[7, 15]
```

Os dados do arquivo foram lidos para uma variável de estrutura contendo um campo: ponteiro para uma matriz flutuante de 7X15. Extraia uma coluna desta matriz.

```
IDL> elevacao = dados.(0) [2,*]
IDL> print, elevacao [0:5]
       399.000      692.000      1003.00      1333.00      811.000      90.0000
```

Tente ler este arquivo usando um modelo (template) de definição com o *ASCII_TEMPLATE*.
ASCII_TEMPLATE fornece uma interface gráfica com um processo de três etapas descrevendo o conteúdo de um arquivo. *ASCII_TEMPLATE* retorna uma variável de estrutura definindo um modelo reutilizável.

```
IDL> x_template = ascii_template(x)
```

No IDL Online Help na página do *ASCII_TAMPLATE* tem alguns exemplos mais detalhados do seu uso, e também captações da tela (screenshoots). Use o *READ_ASCII* com o modelo que você criou para ler o conteúdo contido no arquivo **ascii.txt**.

```
IDL> dados = read_ascii (x, template=x_template)
IDL> wind_speed = dados.(5)
IDL> print, wind_speed [0:5]
10        8        10        0        8        10
```

O arquivo **convec.dat** no subdiretório **exemplos/data** é um exemplo de arquivo binário. Leia o seu conteúdo para a variável *binario* do IDL com o *READ_BINARY*.

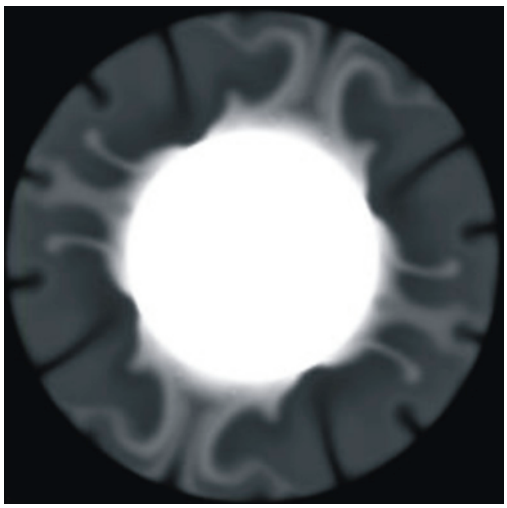
```
IDL> y = filepath('convec.dat', subdir = ['examples', 'data'])
IDL> binario = read_binary(y)
IDL> help, binario
BINARIO       BYTE     = Array[61504]
```

O arquivo foi lido, mas precisamos saber o que pode ser feito com estes dados. No arquivo **index.txt** no subdiretório **exemplos/data**, podemos ver que este arquivo contem a descrição de diversos arquivos incluindo o arquivo **convec.dat** que é uma matriz de *bytes* de 248 X 248 elementos representando um modelo de um manto de condução de calor. Esta informação suplementar é necessária para compreender o conteúdo do arquivo. Tendo esta informação, use a palavra-chave DATA_DIMS para o *READ_BINARY* ler o conteúdo do arquivo.

```
IDL> binario = read_binary(y, data_dims=[248,248])
IDL> help, binario
BINARIO       BYTE     = Array[248, 248]
```

Veja os dados como imagem com o comando *TV*.

```
IDL> tv, binario
```



Maiores informações sobre as rotinas que foram listadas na tabela anterior podem ser encontradas no IDL Online Help.

Rotinas de Baixo Nível para Arquivos

Esta seção descreve o uso do grupo de rotinas *OPEN*, *CLOSE*, *READ*, e *WRITE*. O uso destas rotinas de baixo nível para arquivos requer um pouco mais de conhecimento do IDL, tipos de arquivos e operações de sistemas, mas como um usuário, você terá um grande controle de como esta sendo executada a entrada/saída atual do arquivo.

Ao utilizar rotinas de baixo nível para arquivos do IDL, ajuda ter em mente algumas simples etapas para executar arquivos de entrada ou saída.

1. Encontre o arquivo no sistema de arquivos. Isto envolve tipicamente especificar o caminho de um arquivo e armazená-lo em uma variável, por exemplo, *FILEPATH*, *FILE_SEARCH* ou *DIALOG_PICKFILE*.

2. Defina o formato dos dados para o conteúdo do arquivo. Decidir como o IDL deve representar os dados em um arquivo é geralmente a parte mais difícil de usar rotinas de baixo nível.

3. Abra o arquivo para leitura, escrita ou atualização.

4. Executa arquivos de operações (queryng, reading, writing, positioning) dento do arquivo.

5. Feche o arquivo.

Abrindo e Fechando Arquivos

A tabela abaixo lista as rotinas, para abrir e fechar arquivos do IDL.

Rotinas	Finalidade
OPENR	Abre um arquivo para leitura.
OPENW	Abre um arquivo para escrita.
OPENU	Abre um arquivo para atualização (leitura e/ou escrita).
CLOSE	Fecha um arquivo ou uma lista de arquivos.
FREE_LUN	Fecha arquivos e limpa as unidades de arquivo.

Abaixo temos um exemplo da sintaxe usada para abrir e fechar um arquivo.

```
IDL> a = filepath('ascii.txt', subdir = ['examples','data'])
IDL> openr, 1, a
```

O procedimento *OPENR* é usado para abrir um arquivo em modo de leitura. O valor 1 é número da unidade lógica para o arquivo. Use a palavra-chave FILES para o comando *HELP* e veja quais arquivos estão abertos na sua sessão do IDL.

```
IDL> help, /files
Unit   Attributes                Name
   1   Read                      C:\RSI\IDL61\examples\data\ascii.txt
```

Feche o arquivo com o procedimento *CLOSE*.

```
IDL> close, 1
```

Números de Unidades Lógicas

Um número de unidade lógica (LUN) é um número simples associado a um arquivo do IDL. Todos arquivos abertos são determinados por um LUN quando são abertos, e todas rotinas de entrada/saída deste arquivo serão referidas a este número. Vários arquivos podem ser abertos simultaneamente no IDL, cada um com um número de unidade lógica diferente.

Três números de unidades lógicas são reservados para uso do sistema operacional.

0, stdin : Este LUN representa o caminho padrão de entrada, que é geralmente o teclado.
-1, stdout : Este LUN representa a caminho padrão da saída, que é geralmente a tela do terminal.
-2, stderr : Este LUN representa a caminho padrão de erro, que é geralmente a tela do terminal.

Há 128 números de unidade lógicas disponíveis ao usuário.

1 – 99	Especifico diretamente a rotina *OPEN*.
100 – 128	Especifico para o *GET_LUN*, ou a palavra-chave GET_LUN para a rotina *OPEN*.

Um exemplo da especificação direta de um LUN é dado acima. Quando um LUN na escala de 1-99 é atribuído a um arquivo, não pode ser atribuído novamente até que o arquivo esteja fechado, a sessão atual do IDL seja reiniciada (reset), ou quando você sai do IDL.

O procedimento *GET_LUN*, ou a palavra-chave GET_LUN para o procedimento *OPEN*, pode ser usado deixando que o IDL especifique um LUN na escala de 100 – 128. Isto é particularmente usado para prevenir conflitos com outros LUNs já em uso.

Por exemplo, abra o arquivo **convec.dat** para leitura, deixe que o IDL escolha o LUN.

```
IDL> b = filepath('convec.dat', subdir=['examples','data'])
IDL> openr, lun, b, /get_lun
IDL> help, lun
LUN             LONG      =        100
```

Veja que quando utilizamos a palavra-chave GET_LUN no exemplo acima a variável *lun* recebe o valor 100.
Verifique que o arquivo está aberto com o procedimento *HELP*.

```
IDL> help, /files
Unit   Attributes                    Name
100          Read, Reserved
C:\RSI\IDL61\examples\data\convec.dat
```

Uma unidade de arquivo alocado com *GET_LUN* é liberado com o procedimento *FREE_LUN*. O procedimento *FREE_LUN* fecha o arquivo e limpa o LUN.

```
IDL> free_lun, lun
```