

## Tipos de Programas

IDL suporta três tipos de programas: principal, procedimentos e funções. Procedimentos e funções são a chave para modelar a programação estruturada. Encorajamos você a escrever procedimentos e funções ao invés de escrever tudo no programa principal. Todos programas precisam ser compilados antes de serem executados. Compilar é o ato de interpretar as instruções do código fonte para um arquivo de código byte armazenado na memória. O código byte na memória que é executado quando você executa o programa.

### Programa Principal

Um programa principal consiste em uma seqüência de instruções do IDL terminados com a instrução END. Pode existir apenas um programa principal por sessão do IDL. Um exemplo de um programa principal é apresentado logo abaixo. Este programa principal existe em um arquivo **hello.pro**.

```
hello.pro  
Print, 'Hello World'  
End
```

Compile e execute este programa principal com o comando de execução .run:

```
IDL> .run hello  
Hello World
```

### Procedimentos e Funções

Procedimentos e funções contem módulos que dividem grandes tarefas em pequenas, mais manejáveis. Programas modulares simplificam a correção de erros e a manutenção, pois eles podem ser usados novamente, eles diminuem a quantidade de código requerida por novas aplicações.

Novos procedimentos e funções podem ser escritos no IDL e serem chamados da mesma maneira que os procedimentos ou funções definidas pelo sistema. Quando um procedimento ou uma função acaba, é executada uma instrução *RETURN*. Funções sempre retornam um resultado explícito. Um procedimento é chamado por uma instrução de chamada de procedimento, enquanto a função é chamada por uma referência da função.

#### Procedimento

O procedimento já está contido no programa do IDL. Um procedimento começa com a instrução de declaração do procedimento, que consiste em uma palavra-chave reservada PRO, e depois o nome do procedimento, e qualquer parâmetro para o mesmo. Um procedimento está terminado com uma instrução de END.

Um exemplo de um procedimento é apresentado logo abaixo. Este procedimento existe no arquivo **pwd.pro**.

### **pwd.pro**

```
Pro pwd
  Cd, current = c
  Print, c
End
```

Compile o procedimento usando comando de execução `.compile`:

```
IDL> .compile pwd
```

Execute o procedimento chamando pelo nome:

```
IDL> pwd
C:\RS\IDL61
```

Se o arquivo que contem este procedimento não estiver no diretório do IDL, poderia ser compilado especificando o caminho até o arquivo após o comando de execução `.compile`. Por exemplo, se **pwd.pro** pode estar armazenado no diretório **C:\temp**, com isso fora do diretório do IDL, o comando de execução `.compile` pode ser:

```
IDL> .compile "C:\temp\pwd.pro"
```

O procedimento pode ser executado chamando pelo nome, da mesma forma que era antes.

## Função

Uma função outro programa que está contido no IDL; entretanto, uma função retorna informações. Uma função começa com a instrução de declaração da função, que consiste na palavra-chave reservada `FUNCTION`, o nome da função, e todos os seus parâmetros. O corpo da função, incluindo por ultimo a instrução `RETURN`. Uma função é terminada com a instrução `END`.

Um exemplo de uma função é apresentado logo abaixo. Esta função existe no arquivo **is\_positive.pro**.

### **is\_positive.pro**

```
Function is_positive, x
  Return, x gt 0
End
```

Para compilar a função use o comando de execução `.compile`

```
IDL> .compile is_positive
```

A sintaxe para chamar esta função é:

```
IDL> x = is_positive (5)
IDL> print, x
1
```

Da mesma forma que o procedimento se o arquivo contendo a função não estiver em um diretório do IDL, ele poderá ser compilado especificando todo o caminho até o arquivo após o comando de execução `.compile`.

## Parâmetros de posição e de palavra-chave

Parâmetros são usados para passar informações entre os programas. No IDL temos dois tipos de parâmetros: de posição e por palavra-chave. Embora os dois tipos podem ser usados para enviar ou receber informações, parâmetros de posições são tipicamente usado para requerer informações, enquanto palavras-chave são usadas para informações opcionais. Como os parâmetros são empregados, embora, seja uma escolha do programador.

A ordem dos parâmetros de posição em uma chamada à um programa importante. Por exemplo, se  $x$  e  $y$  são vetores, então:

```
IDL> x = findgen (20)
IDL> y = x ^ 2
```

Então está instrução `PLOT`

```
IDL> plot, x, y
```

É diferente deste outro

```
IDL> plot, y, x
```

Parâmetros de palavra-chave, por outro lado, podem ser listados em qualquer ordem. Por exemplo, as seguintes instruções de um procedimento tem o mesmo resultado:

```
IDL> plot, x, y, xtitle = 'Tempo', ytitle = 'Velocidade'
IDL> plot, x, y, ytitle = 'Velocidade', xtitle = 'Tempo'
```

Parâmetros de palavras-chave, podem ser abreviado. Isto é útil ao usar IDL interativamente pela **Linha de Comando**, mas não é recomendado que se faça isto no código de um programa, pode confundir o próprio programador.

Palavras-chave booleanas podem ser determinadas com uma `"/` antes da palavra-chave. Por exemplo, as seguintes instruções de um procedimento tem o mesmo resultado:

```
IDL> plot, x, y, /nodata
IDL> plot, x, y, nodata = 1
```

## Passando por Parâmetros

Quase todos os programas em IDL usam parâmetros. Esta sessão descreve os dois mecanismos distintos para passagem de parâmetro.

Examine a diferença entre os argumentos usados nestas duas chamadas para o procedimento `PLOT`:

```
IDL> x = findgen (10)
IDL> plot, x           ; Primeira chamada
IDL> plot, findgen (10) ; Segunda chamada
```

Ambas chamadas tem os mesmos dados e o mesmo resultado, uma linha retilínea indo de 0 até 9. A diferença encontra-se na maneira que os dados são passados para o *PLOT*: uma variável é passada na primeira chamada, uma expressão (o que retornar da chamada da função) na segunda chamada.

No IDL existe dois mecanismos de passagem de parâmetros: passagem por valor e passagem por referência. Passagem por valor significa que cada parâmetro pega uma copia do valor dos argumentos, de modo que as mudanças ao parâmetro não afetem os argumentos, mesmo se as variáveis tiverem o mesmo nome, e as mudanças serão perdidas quando retornar para o programa.

Com a passagem por referência, um argumento não será copiado para os parâmetros. Qualquer modificação do parâmetro no programa muda o argumento pois os dois referenciam-se a mesma memória.

Muitas rotinas do IDL utilizam parâmetros para passar informações de volta para quem as chamou. Pegue, como exemplo, esta chamada a função *WHERE*:

```
IDL> x = 0
IDL> index = where(x lt 5, count)
IDL> print, index, count
      0
      1
```

O segundo argumento no *WHERE* é usado para retornar o número de vezes que a expressão condicional no primeiro argumento é compatível. Este argumento precisa ser passado por referência, ou as informações não serão retornadas.

As regras para determinar como um parâmetro deve ser passado no IDL são:

- Expressões, incluindo sobrescrever os elementos de uma matriz ou vetor, campos de uma estrutura, variáveis de sistema e constantes são passados por valor.
- Variáveis nomeadas são passadas por referência.

O mecanismo de passagem por referência torna possível a informação de uma variável ser modificada quando ela é passada para o programa.

## Chamada de Programas

O mecanismo de chamada de programas do IDL é uma seqüência de etapas que localizam, solvem e executam um procedimento ou função, quando o procedimento ou função é chamado por nome, da **linha de comando** ou dentro de um programa do IDL. Por exemplo, quando chamamos o procedimento *LOADCT* pela **linha de comando**:

```
IDL> loadct, 5
% Compiled module: LOADCT.
% Compiled module: FILEPATH.
% Compiled module: PATH_SEP.
% LOADCT: Loading table STD GAMMA-II
```

O mecanismo chamado é usado para localizar, solver e executar *LOADCT*, carregando a tabela de cor número 5 na sessão do IDL.

O mecanismo de chamada consiste em quatro etapas.

1. Procura por uma rotina nomeada na tabela de rotinas do sistema do IDL. Se a rotina estiver listada na tabela do sistema, executa ela; senão, processa a próxima etapa.
2. Verifica se a rotina se encontra em um estado compilado na sessão atual do IDL. Isto pode ser diagnosticado usando o *HELP* com a palavra chave *ROUTINE*. Se o programa já estiver compilado, então roda ele. Se ele não estiver vai para a próxima etapa.
3. Procura por um arquivo ou um arquivo de sistema que tenha o mesmo nome base que a rotina chamada, com a extensão **.pro** ou **.sav**. A procura começa no diretório atual do IDL, então ele prossegue através dos diretórios listados na variável de sistema *!path*. Se o arquivo for encontrado, o IDL compila qualquer unidade de programa encontrada no arquivo, começando pelo topo, até que a rotina chamada seja encontrada. A rotina chamada é então compilada e executada. Se um arquivo com o mesmo nome da rotina chamada não for encontrado, ou se um arquivo com o mesmo nome for encontrado, porém ele não contiver a rotina chamada, então iremos para última etapa.
4. Emite uma mensagem de erro, informando que a função ou procedimento solicitado não pode ser achado.

No exemplo anterior foi chamado o procedimento *LOADCT*, na terceira etapa do mecanismo de chamada. *LOADCT* não é uma rotina do sistema, por isso não foi compilado mais cedo. Entretanto, o arquivo **loadct.pro** existe no subdiretório **lib**, do qual é parte do diretório de busca padrão do IDL. O IDL abre o arquivo, compila o procedimento *LOADCT* interiormente e executa ele.

Note que as rotinas *FILEPATH* e *PATH\_SEP* foram executados também na chamada do *LOADCT*.