MATLAB® Digest

2003 Issues
May
March
January
2002 Issues
November
September
July
May
March
January
2001 Issues
November
September
June
March
2000 Issues
December
September
June
March
Archived Articles
1999-1998
Subscribe Now

What's the big `deal`?
Getting the most out of the `deal` function

by Clay Thompson

What is `deal`?
The `deal` function is an adapter that allows cell arrays and structure arrays to be cross assigned to each other. It gets its name from the metaphor of dealing a round of cards. `deal` takes each input argument and deals it out to the corresponding output argument, in much the same way that you would deal cards out to the players of a card game.

For example, suppose `C` is a cell array and `S` is a structure array with a `.name` field, then the statements

```
[C{:}] = deal(S.name)
```

and

```
[S.name] = deal(C{:})
```

allow the elements of the structure field to be assigned to the cells of C, and vice versa.

The `deal` function is commonly used to

- Assign cell array contents to a structure field
- Assign structure field contents to a cell array
- Initialize structure fields
- Extract varargin input arguments into simple variables
- Transfer field contents from one structure to another

Why `deal`?
The `deal` function was created to work around a limitation in the MATLAB® language that was introduced when the comma-separated list syntax was conceived. It should be possible in the MATLAB language to type

```
S.name = C{:}
```

You would expect the above expression to assign the values in the cell array to the structure. However, since this expression is equivalent to

```
S(1).name,S(2).name,...,S(end).name = C{1},C{2},...,C{end}
```

by the rules of the comma separated list, MATLAB issues the following error:

```
??? Illegal right hand side in assignment. Too many elements.
```

The workaround for this is to encapsulate the comma-separated lists inside [ ] or ( ). Using `deal`, the above statement can be written

```
[S.name] = deal(C{:});
```

This retains the simplicity and readability of the original statement but avoids the error.

How does `deal` work?
The simplest way to code the `deal` function is by typing

```
function [varargout] = deal(varargin)

varargout = varargin;
```

The `deal` function that ships with MATLAB includes the ability to scalar expand a single rhs, and therefore requires more code. The `deal.m` function file can be found in the $MATLAB\toolbox\matlab\datatypes, where $MATLAB is the MATLAB root directory. You can also find further information on this function in the MATLAB Help Desk by choosing 'Documentation Roadmap', 'Online Manuals', 'MATLAB Function Reference Volume I : Language'.

Interesting uses of the `deal` function

Input argument parsing
`deal` is unexpectedly useful for extracting varargin inputs into separate arrays. Take the following simple function, for instance:

```
function h = mountain(varargin)

%MOUNTAIN Display data as a mountain.

%   MOUNTAIN(Z) displays the heights Z as a mountain.

%   MOUNTAIN(X,Y,Z) displays the points (x,y,z) as a mountain.

if nargin==1

  z = varargin{1};

  x = 1:size(z,2);

  y = 1:size(z,1);

elseif nargin==3

  [x,y,z] = deal(varargin{1:3});

end

h = surf(x,y,z)
```

`deal` is used to extract the three input arguments into the simple `x`, `y`, and `z` variables in a single line in order to improve readability.

Initializing structure fields
Suppose we have a structure with the fields (`.name`, `.type`, `.value`). We can initialize the fields

to constant values via the `deal` function by typing in

```
[A(1:4).name] = deal(''); % Initialize all names to empty

[A.type] = deal('simple'); % Initialize all types to 'simple'

[A.value] = deal(0); % Initialize all values to 0
```

Assigning multiple structure fields

The `deal` function is very useful when trying to assign multiple structure fields without a FOR loop. The `addtype` function below uses `deal` in this way to prepend a type string to the values in the `.name` field of the structure `S`. So, for example, if

```
s = struct('name',{'Elephant','Cardinal'});
```

then

```
s = addtype(s,{'mammal','bird'})
```

produces a structure array where `s(1).name` is '`mammal: Elephant`' and `s(2).name` is '`bird: Cardinal`'.

```
function r = addtype(s,t)

%ADDTYPE Add type code to structure name field

%   R = ADDTYPE(S,T) prepends the strings in the cell array T

%   to the S.name fields. S must contain a .name field.

if length(t)==1

   t = t(ones(size(s))); % Scalar expand t to the size of s

end


if ~isequal(size(s),size(t))

  error('S and T must the same size.');

end


name = strcat(t(:)',{': '},{s.name});


%R is the same as S except that the name field has type info

        r = s;

[r.name] = deal(name{:});
```

Note that `deal` is used only in the last line.

related topics:

News & Notes | Using MathWorks Products For... | MATLAB Based Books | Third-Party Products