

Using fprintf in MATLAB

The `fprintf` statement provides control the way that numeric and string data are printed to the command window or a file.

Syntax

The syntax of `fprintf` is based on the `fprintf` function from the C language.

```
fprintf(format)
fprintf(format,variables)
fprintf(fid,format,variables)
```

where `format` is a text string that controls the appearance of the output, `variables` is a comma-separated list of variables to be displayed according to the specification in `format`, and `fid` is a *file identifier* for the file to which the output is sent. The value of `fid` is only set when output is sent to a file.

Table 1: Summary of format codes for use with `fprintf` and `fscanf`.

Code	Conversion instruction
<code>%s</code>	format as a string
<code>%d</code>	format with no fractional part (integer format)
<code>%f</code>	format as a floating-point value
<code>%e</code>	format as a floating-point value in scientific notation
<code>%g</code>	format in the most compact form of either <code>%f</code> or <code>%e</code>
<code>\n</code>	insert newline in output string
<code>\t</code>	insert tab in output string

Controlling Field Width and Precision

The `%d` and `%s` format specifiers can be modified to control the field width used to display a value. This is best explained via example. Suppose that the `k` and `s` variables are assigned values

```
>> k = 2^7;    s = 'big red barn';
```

Without specifying a field width these fields are displayed with the minimum field width, i.e., the minimum number of characters necessary to display the contents of the variables.

```
>> fprintf('%d\n',k)
128

>> fprintf('%s\n',s)
big red barn
```

The field width specifier is an integer that precedes the `d` in `%d` or the `s` in `%s`. For example, to print the value of `k` width of 6 characters use the `%6d` format specifier.

```
>> fprintf('%6d\n',k)
      128
    123456789
```

The second line of output makes it easy to see that the value of 128 is on the right-hand end of a field that is six characters wide. Similarly

```
>> fprintf('%16s\n',s)
      big red barn
    12345678901234567890
```

Examples

Define some data and print it in the simple-minded way

```
n = 5; x = linspace(-pi,pi,n); y = x/1e6; z = x*1e6;

fprintf('\nSimple fprintf statement\n');
for j=1:length(x)
    fprintf('%d %f %f\n',j,x(j),y(j),z(j));
end
```

Produces the following output

```
1 -3.141593 -0.000003 -3141592.653590
2 -1.570796 -0.000002 -1570796.326795
3 0.000000 0.000000 0.000000
4 1.570796 0.000002 1570796.326795
5 3.141593 0.000003 3141592.653590
```

The following sequence of modifications to the format string provide increasing levels of control over the column appearance.

Specify the Precision of the Floating Point Fields

```
fprintf('\nEqual precision control for all float types\n');
for j=1:length(x)
    fprintf('%d %.2f %.2f %.2f\n',j,x(j),y(j),z(j));
end

Equal precision control for all float types
1 -3.14 -0.00 -3141592.65
2 -1.57 -0.00 -1570796.33
3 0.00 0.00 0.00
4 1.57 0.00 1570796.33
5 3.14 0.00 3141592.65
```

Use Column Width and Precision for All Floating Point Fields

```
fprintf('\nField width and precision control for all float types\n');
for j=1:length(x)
    fprintf('%d %8.2f %11.8f %11.0f\n',j,x(j),y(j),z(j));
end

Field width and precision control for all float types
1 -3.1416 -0.00000314 -3141593
2 -1.5708 -0.00000157 -1570796
3 0.0000 0.00000000 0
4 1.5708 0.00000157 1570796
5 3.1416 0.00000314 3141593
```

Use Column Width, Precision Specification, and Scientific Notation

```
fprintf('\nField width, precision control and scientific notation\n');
for j=1:length(x)
    fprintf('%d %8.2f %13.3e %13.3e\n',j,x(j),y(j),z(j));
end

Field width, precision control and scientific notation
1 -3.1416 -3.142e-06 -3.142e+06
2 -1.5708 -1.571e-06 -1.571e+06
3 0.0000 0.000e+00 0.000e+00
4 1.5708 1.571e-06 1.571e+06
5 3.1416 3.142e-06 3.142e+06
```

Add Column Headers and Field Width for Index Column

```
fprintf('\nFinal touches:\n');
fprintf('\n j x(j) y(j) z(j)\n');
for j=1:length(x)
    fprintf('%3d %8.2f %13.3e %13.3e\n',j,x(j),y(j),z(j));
end
```

Final touches:

j	x(j)	y(j)	z(j)
1	-3.1416	-3.142e-06	-3.142e+06
2	-1.5708	-1.571e-06	-1.571e+06
3	0.0000	0.000e+00	0.000e+00
4	1.5708	1.571e-06	1.571e+06
5	3.1416	3.142e-06	3.142e+06

File Output with Optional Use of Tabs to Separate Columns

```
fprintf('\nWrite data to a file, no screen output:\n');
fout = fopen('someData.txt','wt');
fprintf(fout,'j\tx(j)\ty(j)\tz(j)\n');
for j=1:length(x)
    fprintf(fout,'%d\t%e\t%e\t%e\n',j,x(j),y(j),z(j));
end
fclose(fout); % always close the open file handle
```

Using sprintf to Generate file names.

The **sprintf** function is similar to the **fprintf**, but instead of writing text to the command window or a file

Example: Simple string message

```
>> n = 3; y = sin(n*pi/2);
>> s = sprintf('The sine of %d pi/2 is %-6.4f',n,y)
s =
The sine of 3 pi/2 is -1.0000
```

sprintf is useful for creating file names that contain parameters.

Writing Data to a File with the save Function.

See pp. 53-56 in the textbook and the **demoFilePrintf** example code

```
function demoFileprintf
% demoFileprintf Examples of file creation and output with fprintf

% --- Define data
n = [500 1000 1000];
ave = [10 20 30];
sdev = [1.5 2.5 2.5];

% --- Use fprintf to write data to a file one line at a time
fprintf('\nSave with fprintf one line at a time\n')
for i=1:length(n)
    x = ave(i) + sdev(i)*randn(n(i),1);
    y = sin(x);
    fname = sprintf('rand_%d_%d_%d.txt',n(i),ave(i),100*sdev(i));
    fout = fopen(fname,'wt');
    for j=1:n(i)
        fprintf(fout,'%e\t%e\n',x(j),y(j));
    end
    fclose(fout);
    fprintf('%6d %8.2f %8.2f %s\n',n(i),ave(i),sdev(i),fname);
end

% --- Use save to write data to a file all at once
%      save(fname,'-ascii','x','y') stores x and y in sequence
%      not on the same line. The solution is to create a matrix
%      with x and y as the columns
fprintf('\nUse "save" with x and y in the xy matrix\n')
for i=1:length(n)
    x = ave(i) + sdev(i)*randn(n(i),1);
    y = sin(x);
    xy = [x y];
    fname = sprintf('randsave_%d_%d_%d.txt',n(i),ave(i),100*sdev(i));
    save(fname,'-ascii','xy')
    fprintf('%6d %8.2f %8.2f %s\n',n(i),ave(i),sdev(i),fname);
end

% --- Use save to write data as a MAT file. Variable names are
%      embedded in the file
fprintf('\nSave in MAT format with x and y as distinct variables\n')
for i=1:length(n)
    x = ave(i) + sdev(i)*randn(n(i),1);
    y = sin(x);
    fname = sprintf('randmat_%d_%d_%d.mat',n(i),ave(i),100*sdev(i));
    save(fname,'-mat','x','y')
    fprintf('%6d %8.2f %8.2f %s\n',n(i),ave(i),sdev(i),fname);
end
```
