

Beyond the Mouse – A Short Course on Programming

5. Matlab IO: Getting data in and out of Matlab

Ronni Grapenthin and Glenn
Thompson

Geophysical Institute, University of Alaska
Fairbanks

October 10, 2011

YOU'LL NEVER FIND A
PROGRAMMING LANGUAGE
THAT FREES YOU FROM
THE BURDEN OF
CLARIFYING
YOUR IDEAS.



"The Uncomfortable Truths Well",
<http://xkcd.com/568> (April 13, 2009)

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc
- 7 Examples

File access 1: Excel-data

`xlsread`

- `[num, txt, raw] = xlsread('myfile.xls', 'sheet23');`
attempts to read sheet 23 (first sheet if parameter omitted)
- `num` – a matrix that contains all numeric data
- `txt` – a cell array that contains all text data
- `raw` – cell array with columns `xlsread` could not interpret

File access 1: Excel-data

`xlsread`

- `[num, txt, raw] = xlsread('myfile.xls', 'sheet23');`
attempts to read sheet 23 (first sheet if parameter omitted)
- `num` – a matrix that contains all numeric data
- `txt` – a cell array that contains all text data
- `raw` – cell array with columns `xlsread` could not interpret

`xlswrite`

- `[status, msg] = xlswrite('myfile.xls', M, 'sheet42');`
attempts to write matrix `M` to sheet 42 of `myfile.xls`
- `status` – 1 on success, 0 on error
- `msg` – error message object with fields `message` and `identifier`

File access 1: Excel-data

`xlsread`

- `[num, txt, raw] = xlsread('myfile.xls', 'sheet23');`
attempts to read sheet 23 (first sheet if parameter omitted)
- `num` – a matrix that contains all numeric data
- `txt` – a cell array that contains all text data
- `raw` – cell array with columns `xlsread` could not interpret

`xlswrite`

- `[status, msg] = xlswrite('myfile.xls', M, 'sheet42');`
attempts to write matrix `M` to sheet 42 of `myfile.xls`
- `status` – 1 on success, 0 on error
- `msg` – error message object with fields `message` and `identifier`

See Also:

`dlmread`, `dlmwrite`, `csvread`, `csvwrite`

File access 1.5: Opening and closing files

fopen

- `fid = fopen('filename', mode);`
Open a file, **do not discard fid!**
- mode is (`>help fopen` for full list):
 - 'r' – read (default)
 - 'w' – write (**overwrite** if file exists - **careful!**)
 - 'a' – append (append if file exists)

Wherever **fid** is used as a parameter with functions below, `fopen`, `fclose` must bracket the function call!

File access 1.5: Opening and closing files

fopen

- `fid = fopen('filename', mode);`
Open a file, **do not discard fid!**
- mode is (`>help fopen` for full list):
 - 'r' – read (default)
 - 'w' – write (**overwrite** if file exists - **careful!**)
 - 'a' – append (append if file exists)

fclose

- `fid = fclose(fid);`
close file with identifier `fid`

Wherever **fid** is used as a parameter with functions below, `fopen`, `fclose` must bracket the function call!

File access 1.5: Opening and closing files

fopen

- `fid = fopen('filename', mode);`
Open a file, **do not discard fid!**
- mode is (>help fopen for full list):
 - 'r' – read (default)
 - 'w' – write (**overwrite** if file exists - **careful!**)
 - 'a' – append (append if file exists)

fclose

- `fid = fclose(fid);`
close file with identifier `fid`

Wherever **fid** is used as a parameter with functions below, `fopen`, `fclose` must bracket the function call!

File access 2: Text Files 1/3

`textread` (deprecated, will be removed)

- `[A, B, C, ...] = textread('filename', 'format', N);`
reads data from file 'filename' to **multiple outputs** A,B,C,... using specified `format` until **entire** file is read, or N times.

File access 2: Text Files 1/3

`textread` (deprecated, will be removed)

- `[A, B, C, ...] = textread('filename', 'format', N);`
reads data from file 'filename' to **multiple outputs** A,B,C,... using specified `format` until **entire** file is read, or N times.

`textscan`

- `C = textscan(fid, 'format', N);`
reads data from file `fid` **OR** a string to cell array `C` using specified `format` until **entire** file is read, or N times (resume from where left by calling `textscan` again later).

File access 2: Text Files 1/3

`textread` (deprecated, will be removed)

- `[A, B, C, ...] = textread('filename', 'format', N);`
reads data from file 'filename' to **multiple outputs** A,B,C,... using specified `format` until **entire** file is read, or N times.

`textscan`

- `C = textscan(fid, 'format', N);`
reads data from file `fid` **OR** a string to cell array `C` using specified `format` until **entire** file is read, or N times (resume from where left by calling `textscan` again later).

Use `textscan` if you want ...

- to read large files (better performance than `textread`)
- **one** cell array as opposed to many outputs
- read from any point in the file (use `fseek` on `fid` first)
- more options and choices in data conversion (see doc)

`fprintf`

- `count = fprintf(fid, 'format', A, ...);`
formats data in matrix `A` (and additional arguments) according to format string and writes to the file associated with `fid`
- `count` – number of bytes written

`fprintf`

- `count = fprintf(fid, 'format', A, ...);`
formats data in matrix `A` (and additional arguments) according to format string and writes to the file associated with `fid`
- `count` – number of bytes written

See also

- `dlmwrite`: Write matrix to ASCII delimited file
- `csvwrite`: Write matrix to comma-separated value file

fprintf example

```
clear all, clc, close all;  
  
% create data here (row vectors!)  
x = 1:10  
y = rand(1,10)  
z = rand(1,10)  
  
% open a file in write mode  
fout = fopen('random_numbers.txt', 'w');  
  
% write our data:  
% x is first column,  
% y is second column  
fprintf(fout, '%d\t%f\t%f\n', [x; y; z])  
  
% don't forget to close the file!  
fclose(fout)
```

fprintf example

```
clear all, clc, close all;  
  
% create data here (row vectors!)  
x = 1:10  
y = rand(1,10)  
z = rand(1,10)  
  
% open a file in write mode  
fout = fopen('random_numbers.txt', 'w');  
  
% write our data:  
% x is first column,  
% y is second column  
fprintf(fout, '%d\t%f\t%f\n', [x; y; z])  
  
% don't forget to close the file!  
fclose(fout)
```

output

```
1 0.438744 0.276025  
2 0.381558 0.679703  
3 0.765517 0.655098  
4 0.795200 0.162612  
5 0.186873 0.118998  
6 0.489764 0.498364  
7 0.445586 0.959744  
8 0.646313 0.340386  
9 0.709365 0.585268  
10 0.754687 0.223812
```

Outline

- 1 File access
- 2 Plotting Data**
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots**
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure**
- 5 Saving your Figure
- 6 Misc
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure**
- 6 Misc
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc**
- 7 Examples

Outline

- 1 File access
- 2 Plotting Data
- 3 Annotating Plots
- 4 Many Data - one Figure
- 5 Saving your Figure
- 6 Misc
- 7 Examples**