19. Working with Data Files

Reading from a File
Writing to a File
fopen, fclose,
fgetl, feof,
fprintf,
sort

Storing a Cell Array of Strings in a Data File

Have: **{'abcd','123','xyz','3.14159'}**

Want:

```
abcd
123
xyz
3.14159
```

3 steps…

1. Create and Open the File

```
fid = fopen('MyFile.dat','w');
```

Each open file has an id.

'w' means that you can write to this file

The file name in quotes.

2. Write Each String to a Line in the File

```
C = {'Line1','Line2','Line3'}

for i=1:length(C)
     fprintf(fid,'%s\n',C{i});
end
```

"Print to file"

File id.

String Format

Carriage Return

3. Close the File

```
fclose(fid)
```

File id.

A Function to do This

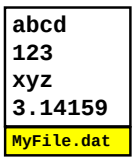
```
  function Cell2File(C,fname)
% C is a cell array of strings
% Creates .dat file with name
% specified by the string fname.
% The i-th line in the file is C{i}

  fid = fopen([fname '.dat'],'w');
  for i=1:length(C)
     fprintf(fid,'%s\n',C{i});
  end
  fclose(fid);
```

Example

```
C = {'abcd','123','xyz','3.14159'}

cell2File(C,'MyFile')
```



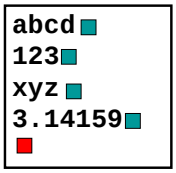
Reverse Problem

Read the data in a file line-by-line and store the results in a cell array.

How are lines separated?

How do we know when there are no more lines?

In a File there Are Hidden "Markers"



- Carriage return marks end of a line
- eof marks end of a file

Reading A File

1. Open the file.
2. Read it line-by-line until eof
3. Close the file.

1. Open the File

```
fid = fopen('MyFile.dat','r');
```

Each open file has an id.

'r' means that you can read from this file

The file name with suffix in quotes.

2. Read Each Line in the File

```
i=0;
while ~feof(fid)
   i=i+1;
   C{i} = fgetl(fid);
end
```

"False" until the end-of-file reached

Get the next line

File id "names" the file.

3. Close the File

```
fclose(fid)
```

File id.

A Function to Do This

```
  function C = File2Cell(fname)
% fname is a string that names a .dat file
% in the current directory.
% C is a cell array with C{i} being the
% i-th line in the file.

  fid = fopen([fname '.dat'],'r');
  i=0;
  while ~feof(fid)
     i=i+1;
     C{i} = fgetl(fid);
  end
  fclose(fid);
```

Example

abcd
123
xyz
3.14159
MyFile.dat

```
C = File2Cell('MyFile')
```

C: | 'abcd' | '123' | 'xyz' | '3.14159' |

A Detailed Read-File Example

From the protein database at

http://www.rcsb.org

we download the file **1bl8.dat** which encodes the amino acid information for the protein with the same name. We want the xyz coordinates of the protein's ``backbone".

The File has a Long "Header"

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0038)http://www.rcsb.org/pdb/files/1bl8.pdb -->
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.2722" name=GENERATOR></HEAD>
<BODY><PRE>HEADER    MEMBRANE PROTEIN                23-JUL-98   1BL8
TITLE     POTASSIUM CHANNEL (KCSA) FROM STREPTOMYCES LIVIDANS
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: POTASSIUM CHANNEL PROTEIN;
COMPND   3 CHAIN: A, B, C, D;
COMPND   4 ENGINEERED: YES;
COMPND   5 MUTATION: YES
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: STREPTOMYCES LIVIDANS;
```

Hundreds of lines—not relevant to us.

Eventually, the xyz Data is Reached…

```
MTRIX1   2 -0.736910 -0.010340  0.675910      112.17546    1
MTRIX2   2  0.004580 -0.999940 -0.010300       53.01701    1
MTRIX3   2  0.675980 -0.004490  0.736910      -43.35083    1
MTRIX1   3  0.137220 -0.931030  0.338160       80.28391    1
MTRIX2   3  0.929330  0.002860 -0.369240      -33.25713    1
MTRIX3   3  0.342800  0.364930  0.865630      -31.77395    1

ATOM      1  N   ALA A  23      65.191  22.037  48.576  1.00181.62        N
ATOM      2  CA  ALA A  23      66.434  22.838  48.377  1.00181.62        C
ATOM      3  C   ALA A  23      66.148  24.075  47.534  1.00181.62        C
```

Signal:Lines that begin with 'ATOM'

x y z

Location!

```
1-4   14-15 18-20   33-38  41-46 49-54
```

columns

```
ATOM     14  N   HIS A  25      68.656  24.973  44.142  1.00128.26           N
ATOM     15  CA  HIS A  25      69.416  24.678  42.939  1.00128.26           C
ATOM     16  C   HIS A  25      68.843  23.458  42.227  1.00128.26           C
ATOM     17  O   HIS A  25      68.911  23.354  41.007  1.00128.26           O
ATOM     18  CB  HIS A  25      70.881  24.416  43.300  1.00154.92           C
ATOM     19  CG  HIS A  25      71.188  22.977  43.573  1.00154.92           C
ATOM     20  ND1 HIS A  25      71.886  22.184  42.689  1.00154.92           N
ATOM     21  CD2 HIS A  25      70.877  22.182  44.625  1.00154.92           C
ATOM     22  CE1 HIS A  25      71.993  20.963  43.183  1.00154.92           C
ATOM     23  NE2 HIS A  25      71.388  20.935  44.356  1.00154.92           N
ATOM     24  N   TRP A  26      68.271  22.546  43.005  1.00 87.09           N
ATOM     25  CA  TRP A  26      67.702  21.311  42.475  1.00 87.09           C
ATOM     26  C   TRP A  26      66.187  21.378  42.339  1.00 87.09           C
ATOM     27  O   TRP A  26      65.577  20.508  41.718  1.00 87.09           O
```

x y z

Goal

Read past all the header information and then collect the xyz data from those lines whose 3rd column is 'CA'.

Theme: Just getting what you need From a data file.

Here is the solution…

```
fid = fopen('1bl8.dat','r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Open the file.

```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Initialize xyz arrays

```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Iterate Until End of File

```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Get the next line from file.

```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Make Sure It's a Backbone Amino Acid

```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x;str2double(s(33:38))];
         y = [y;str2double(s(41:46))];
         z = [z;str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Update the x, y, z arrays

Next Prob: Storing a Numeric 2D Array in a File

Have an array, e.g.,

```
>> A = rand(3,4)
A =
0.9218    0.4057    0.4103    0.3529
0.7382    0.9355    0.8936    0.8132
0.1763    0.9169    0.0579    0.0099
```

Storing a 2D Array in a File

```
0.9218    0.4057    0.4103    0.3529
0.7382    0.9355    0.8936    0.8132
0.1763    0.9169    0.0579    0.0099
```

MyMatrix.dat

Would like to specify the format, e.g., use %10.4f for each number.

Reason: Would make it easier to read the file

```
01234567890123456789012345678901234567890123456789
 0.9218   0.4057   0.4103   0.3529
 0.7382   0.9355   0.8936   0.8132
 0.1763   0.9169   0.0579   0.0099
```

MyMatrix.dat

How could we set up a 3-by-1 numeric array that house the 2nd column?

The formatting makes it easy.

```
01234567890123456789012345678901234567890123456789
 0.9218   0.4057   0.4103   0.3529
 0.7382   0.9355   0.8936   0.8132
 0.1763   0.9169   0.0579   0.0099
```

MyMatrix.dat

```
C = File2Cell('MyMatrix');
Col2 = [];
for i=1:3
   s = C{i};
   Col2 = [Col2;str2double(s(11:20))]
end
```

A Word About **sprintf**

Remember, **sprint** is a function that returns a string, e.g.

```
s = sprintf('h = %5d, x = %5.2f',h,x)
```

A Word About **sprintf**

Suppose x is a length-2 array. Then

```
s = sprint('%10.2f%10.2f',x(1),x(2))
```

is equivalent to

```
s = sprintf('%10.2f',x)
```

A Word About **sprintf**

Suppose x is a length-n array. Then

```
s = sprint('%10.2f',x);
```

is eqiuvalent to

```
s = [];
for i=1:length(x)
   s = [s sprintf('%10,2f',x(i))];
end
```

Storing a 2D Array in a File

```
01234567890123456789012345678901234567890123456789
```

```
0.9218    0.4057    0.4103    0.3529
0.7382    0.9355    0.8936    0.8132
0.1763    0.9169    0.0579    0.0099
```

MyMatrix.dat

```
fid = fopen('MyMatrix.dat','w');
for i=1:3
   str = sprintf('%10.4f',M(i,:));
   fprintf(fid,'%s\n',str);
end
```

2D Numeric Array M → File

```
01234567890123456789012345678901234567890123456789
```

```
0.92      0.40      0.41      0.35
0.73      0.93      0.89      0.81
0.17      0.91      0.05      0.00
```

MyMatrix.dat

```
fid = fopen('MyMatrix.dat','w');
for i=1:3
   str = sprintf('%10.2f',M(i,:));
   fprintf(fid,'%s\n',str);
end
```

2D Numeric Array → File

```
01234567890123456789012345678901234567890123456789
```

```
0.921829 0.405785 0.410653 0.352999
0.738214 0.935564 0.893678 0.813275
0.176322 0.916909 0.057998 0.009957
```

MyMatrix.dat

```
fid = fopen('MyMatrix.dat','w');
for i=1:3
   str = sprintf('%9.6f',M(i,:));
   fprintf(fid,'%s\n',str);
end
```

A Function to Do This

```
function Matrix2File(M,nbrFormat,fname)
% M is a 2D array of numbers
% Creates .dat file with name specified by the
% string fname.
% The ith line in the file is M(i,:) displayed with
% the format specified by the string nbrFormat

  [m,n] = size(M);
  fid = fopen([fname '.dat'],'w');
  for i=1:m
     str = sprintf(nbrFormat,M(i,:));
     fprintf(fid,'%s\n',str);
  end
  fclose(fid);
```

Examples

Suppose M is a real 2D array:

```
Matrix2File(M,'%10d','MyMat')
Matrix2File(M,'%9.2f','MyMat')
Matrix2File(M,'%10.3e','MyMat')
```

A Detailed Sort-A-File Example

Suppose each line in the file

StatePop.dat

is structured as follows:

Cols 1-14 State Name
Cols 16-24 Population (Millions)

Assume that in the file the states appear in alphabetical order.

```
Alabama          4557808
Alaska            663661
Arizona          5939292
Arkansas         2779154
California      36132147
Colorado         4665177
   :                :
   :                :
Texas           22859968
Utah             2469585
Vermont           623050
Virginia         7567465
Washington       6287759
West Virginia    1816856
Wisconsin        5536201
Wyoming           509294
```

A Detailed Sort-A-File Example

Create a new file

PopState.dat

that is structured the same as **StatePop.dat** except that the **states are ordered from smallest to largest according to population.**

First, Get the Populations into an Array

```
C = File2Cell('StatePop');
n = length(C);
Pop = zeros(n,1);
for i=1:n
  S = C{i};
  Pop(i) = str2double(S(16:24));
end
```

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(1) = x(3) = x(idx(1))`

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(2) = x(1) = x(idx(2))`

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(3) = x(5) = x(idx(3))`

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(4) = x(2) = x(idx(4))`

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(5) = x(4) = x(idx(5))`

Built-In Function **sort**

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(k) = x(idx(k))`

Sort from Little to Big

```
[s,rank] = sort(Pop);
Cnew = cell(n,1);
for i=1:n
    ithSmallest = rank(i);
    Cnew{i} = C{ithSmallest};
end

Cell2File(Cnew,'PopState')
```

```
Wyoming          509294
Vermont          623050
North Dakota     636677
Alaska           663661
South Dakota     775933
Delaware         843524
Montana          935670
   :                :
   :                :
Illinois       12763371
Florida        17789864
New York       19254630
Texas          22859968
California     36132147
```