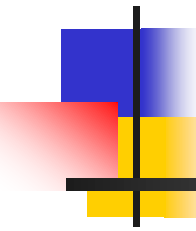
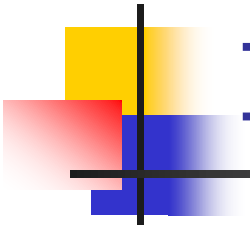


Chapter 8: Input/Output Functions

Yunong Zhang (张雨浓)

Email: zhynong@mail.sysu.edu.cn



Introduction to Input/Output Fun.

- Data to be used in the future
- Communicate with other languages

Save and load Commands

- save

  save all data in the current workspace to a file named matlab.mat in the current directory

- save *filename* [list of variables] [options]

options

-mat save data in MAT-file format

-ascii save data in ASCII format

-append add the specified variables to an existing MAT-file

Save and load Commands (Cont.)

```
>> a=[1 2 3;4 5 6;7 8 9];
>>b=1;
>> save data_a_b  a b

>> c=[10 11;12 13];
>> save data_a_b  c  -append
```

Save and load Commands (Cont.)

- load

  load all data in file matlab.mat into the current workspace

- load filename

  load data_a_b

Save and load Commands (Cont.)

>> save data_a_b -ascii

"save –ascii" command can NOT save cell array or structure array data, and it converts string data to numbers before saving it

Save and load Commands (Cont.)

```
>> a=cell(2,1)
>> a{1,1}='string'
>> a{2,1}=[1 2]
>> celldisp(a)
 a{1} =
      string
a{2} =
      1     2
```

Save and load Commands (Cont.)

>> save cell_a a -ascii

Warning: Attempt to write an unsupported data type to an ASCII file.

a not written to file.

```
>> clear
>> load cell_a
>> whos
  Name          Size                Bytes  Class

  cell_a        0x0                     0  double array

Grand total is 0 elements using 0 bytes
```

Other functions and commands are needed!

textread Command

- read ASCII files into columns of data

  [a b c...]=textread(filename, format, n)

  filename: name of the file to be opened

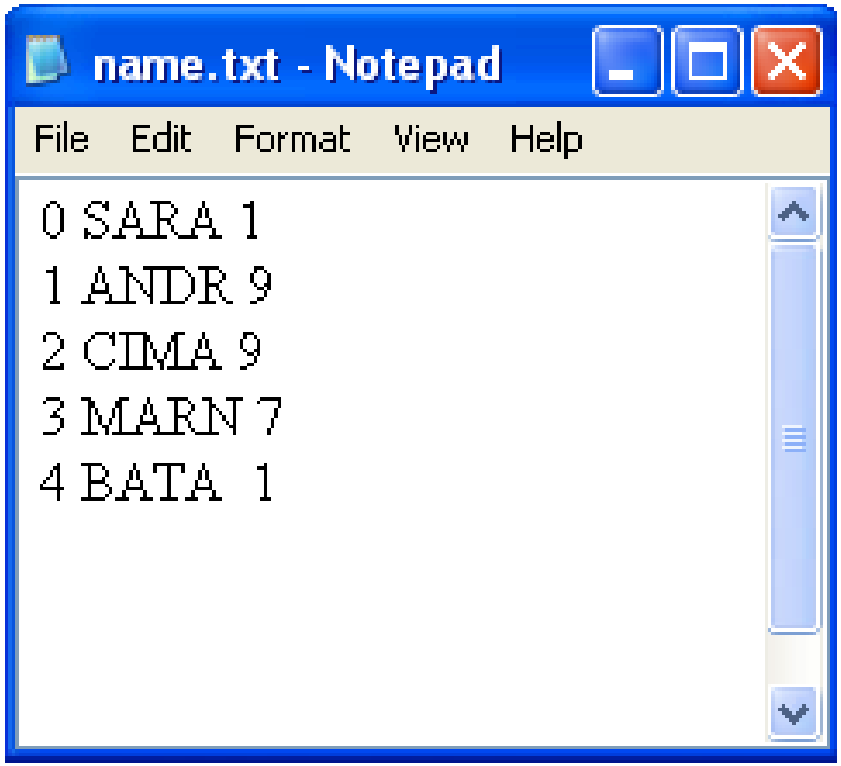
  format: a string containing a description of the type of data in each column

  n: number of lines to read (if n is missing, the command reads to the end of the file)

textread Command (Cont.)

```
>> [a b c]=textread('name.txt','%d %s %d')
```

a =	b =	c =
0	'SARA'	1
1	'ANDR'	9
2	'CIMA'	9
3	'MARN'	7
4	'BATA'	1



cell

textread Command (Cont.)

Supported conversion specifications:

%n - read a number - float or integer (returns double array)

%5n reads up to 5 digits or until next delimiter

%d - read a signed integer value (returns double array)

%5d reads up to 5 digits or until next delimiter

%u - read an integer value (returns double array)

%5u reads up to 5 digits or until next delimiter

%f - read a floating point value (returns double array)

%5f reads up to 5 digits or until next delimiter

%s - read a whitespace separated string (returns cellstr)

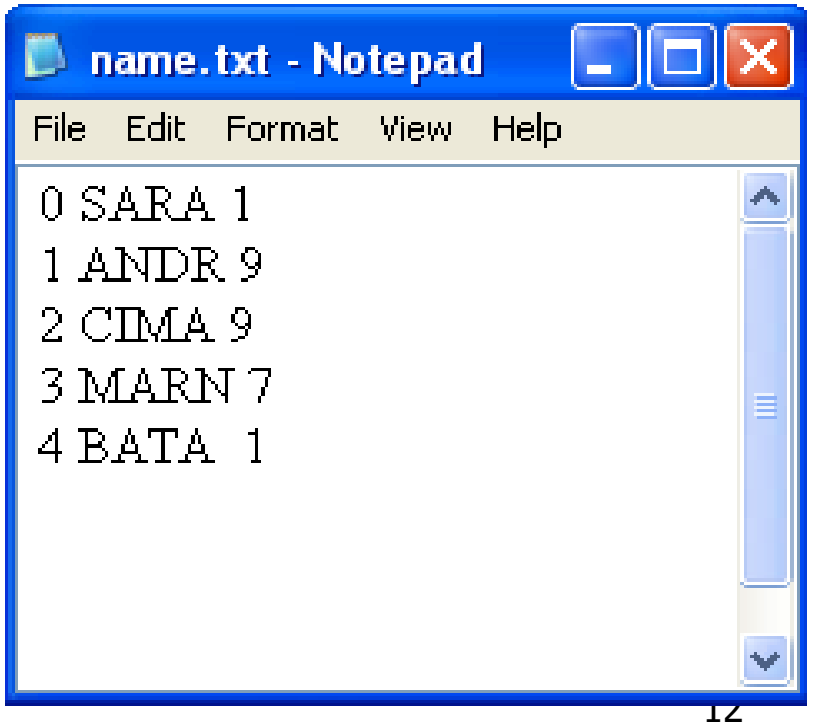
%5s reads up to 5 characters or until whitespace

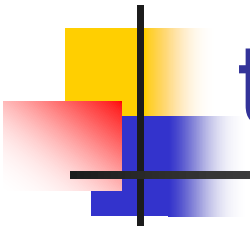
textread Command (Cont.)

- Skip selected columns by adding an asterisk to the corresponding format descriptor

```
>> [a c]=textread('name.txt','%d %*s %d')
```

```
a =          c =

    0            1
    1            9
    2            9
    3            7
    4            1
```





textread Command (Cont.)

- format

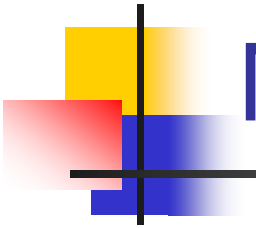
%c single character

%s string of characters

%d signed integer

%f floating-point

%n - a number - float or integer (returns double array)



MATLAB File Processing

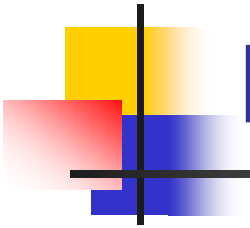
- fopen()
- read from/write to file:

  binary/formatted character
- fclose()

File Opening

- fid=fopen(filename, permission)
- [fid message]=fopen(filename, permission)

'fid' is a scalar MATLAB integer, called a file identifier. You use the fid as the first argument to other file input/output routines. If 'fopen' can not open the file, it returns -1.



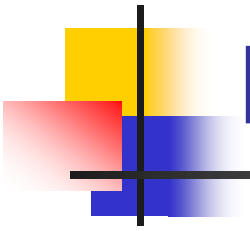
File Opening (Cont.)

- filename: files stored on disk, magnetic tape, or some other storage devices

- message:

  If the specified file is opened successfully, the message will be an empty string

  If the file fails to be opened, the message will be a string explaining the error



File Opening (Cont.)

- permission:

'r' open an existing file for reading only

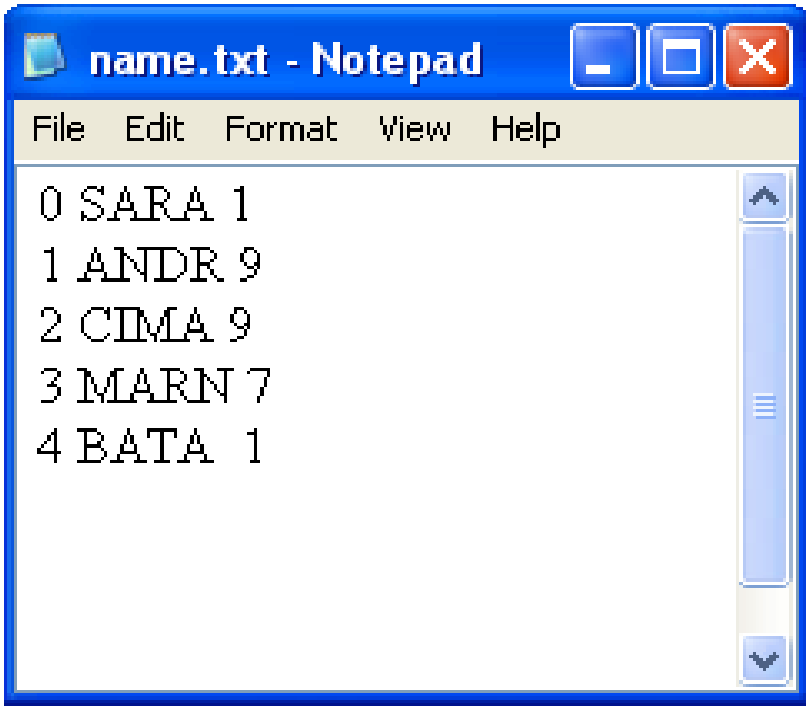
'r+' open an existing file for reading and writing

'w' open an existing file for writing only

'a' append data to the end of the opened file

File Opening (Cont.)

- fid=fopen('name.txt','r')
- [fid message]=fopen('name.txt','r')



fread and fscanf functions

- fread: Read binary data in a specified format

  array=fread(fid, size, precision)

--

size: specify the amount of data to be read, =

n: read exactly n values. array will be a column vector containing n values

Inf: read until the end of the file

[n m]: read exactly n*m values, and format the data as an n*m array

fread and fscanf functions (Cont.)

- precision:

'char'	8-bit characters
'int8'	8-bit integer
'int16'	16-bit integer
'int32'	32-bit integer
'float32'	32-bit floating point
'float64'	64-bit floating point

C, Fortran, and/or JAVA equivalent?

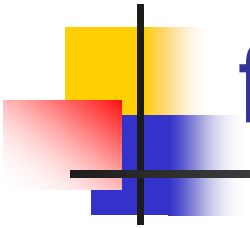
Any of the following strings, either the MATLAB version, or their C or Fortran equivalent, may be used. If not specified, the default is 'uchar'.

MATLAB	C or Fortran	Description
'uchar'	'unsigned char'	unsigned character, 8 bits.
'schar'	'signed char'	signed character, 8 bits.
'int8'	'integer*1'	integer, 8 bits.
'int16'	'integer*2'	integer, 16 bits.
'int32'	'integer*4'	integer, 32 bits.
'int64'	'integer*8'	integer, 64 bits.
'uint8'	'integer*1'	unsigned integer, 8 bits.
'uint16'	'integer*2'	unsigned integer, 16 bits.
'uint32'	'integer*4'	unsigned integer, 32 bits.
'uint64'	'integer*8'	unsigned integer, 64 bits.
'single'	'real*4'	floating point, 32 bits.
'float32'	'real*4'	floating point, 32 bits.
'double'	'real*8'	floating point, 64 bits.
'float64'	'real*8'	floating point, 64 bits.

Platform-dependent formats

The following platform-dependent formats are also supported by Matlab but they are not guaranteed to be the same on all platforms.

MATLAB	C or Fortran	Description
'char'	'char*1'	character, 8 bits (signed/unsigned)
'short'	'short'	integer, 16 bits.
'int'	'int'	integer, 32 bits.
'long'	'long'	integer, 32 or 64 bits.
'ushort'	'unsigned short'	unsigned integer, 16 bits.
'uint'	'unsigned int'	unsigned integer, 32 bits.
'ulong'	'unsigned long'	unsigned integer, 32 bits or 64 bits
'float'	'float'	floating point, 32 bits.



fread and fscanf functions (Cont.)

- array=fread(fid,[100 50],'float64');

fread and fscanf functions (Cont.)

- fscanf: read formatted data in a specified format from a file

array=fscanf(fid,format)

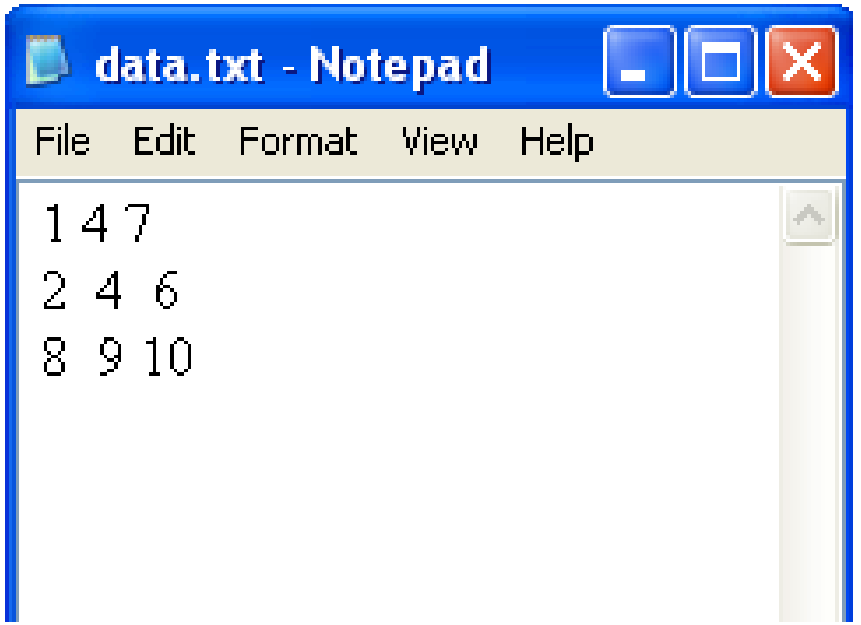
array=fscanf(fid,format,size)

fread and fscanf functions (Cont.)

- size: specify the amount of data to be read
  - n: read exactly n values. array will be a column vector containing n values
  - Inf: read until the end of the file
  - [n m]: read exactly n*m values, and format the data as an n*m array

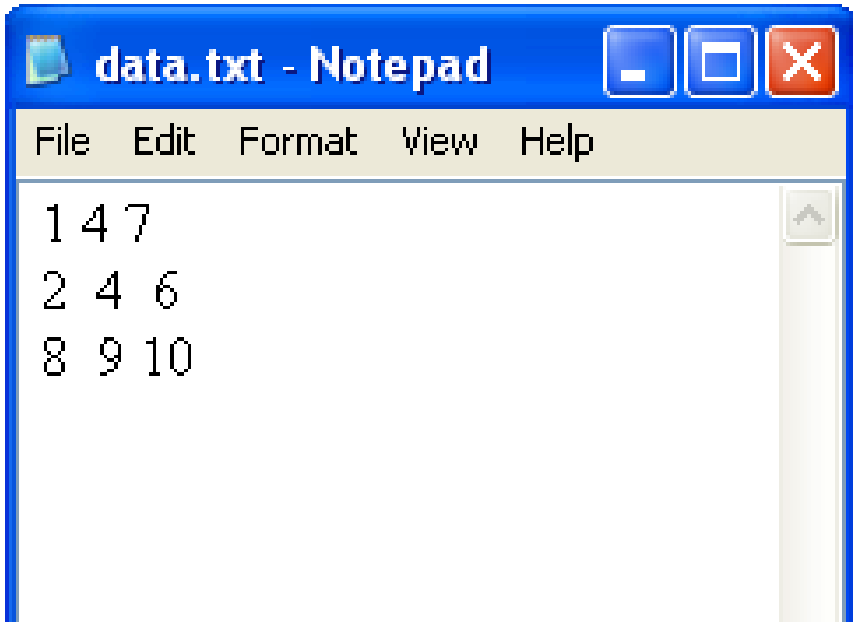
fread and fscanf functions (Cont.)

```
>> a=fscanf(fid,'%d')

a =

     1
     4
     7
     2
     4
     6
     8
     9
    10
```



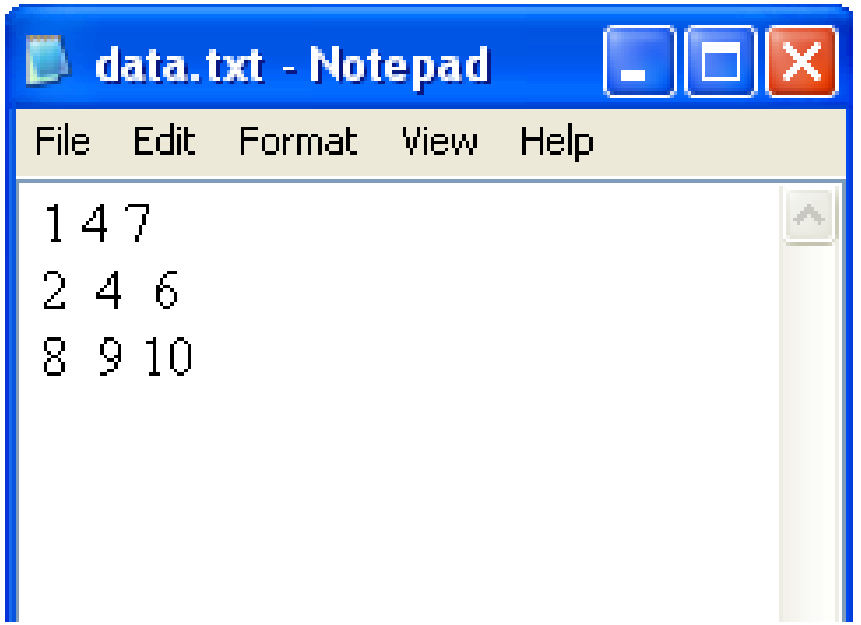
fread and fscanf functions (Cont.)

```
>> a=fscanf(fid,'%d', [1 9])

a =
     1     4     7     2     4     6     8     9    10
```



fread and fscanf functions (Cont.)

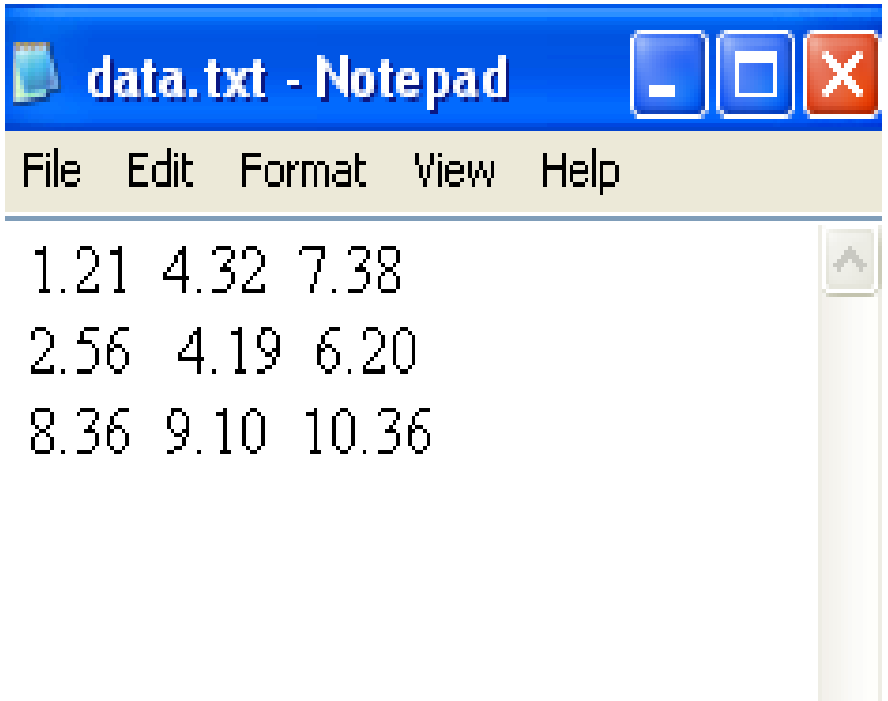
```
>> a=fscanf(fid,'%d',[3 3])

a =

     1     2     8
     4     4     9
     7     6    10
```



fread and fscanf functions (Cont.)

```
a=fscanf(fid,'%f');
a=
```

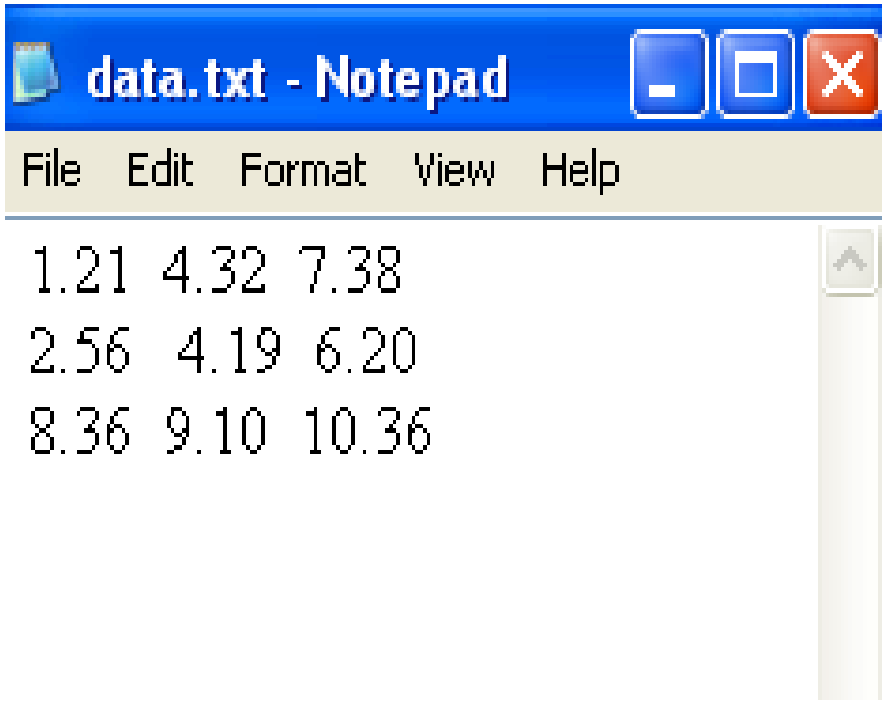
1.21
4.32
7.38
2.56
4.19
6.2
8.36
9.1
10.36



fread and fscanf functions (Cont.)

```
a=fscanf(fid,'%f',[3 3]);
a =
```

1.21	2.56	8.36
4.32	4.19	9.1
7.38	6.2	10.36

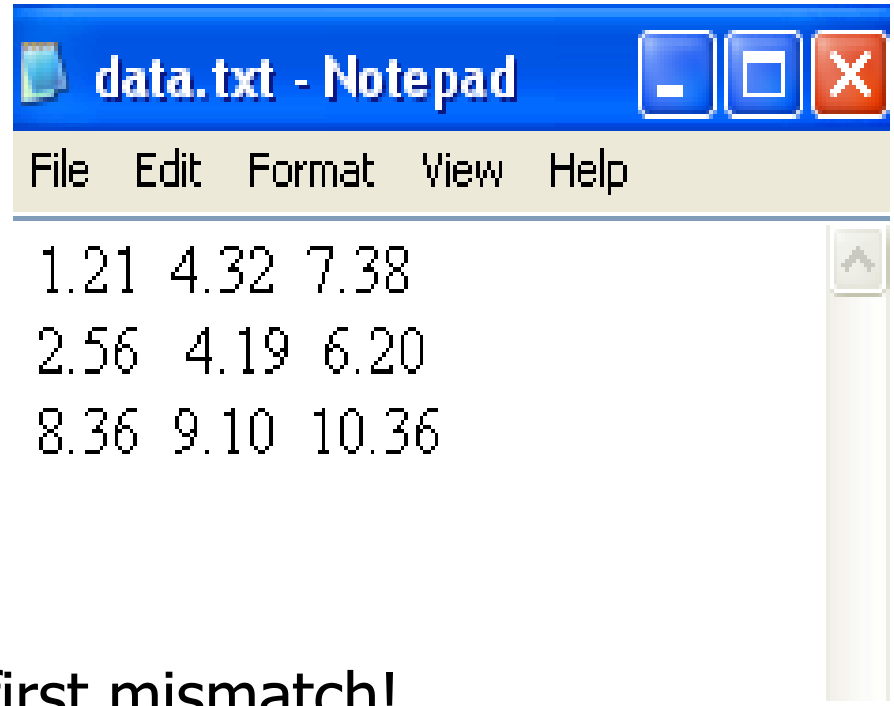


Read: column-by-column（行读列放）

fread and fscanf functions (Cont.)

```
a=fscanf(fid,'%d');
```

```
a =

     1
```

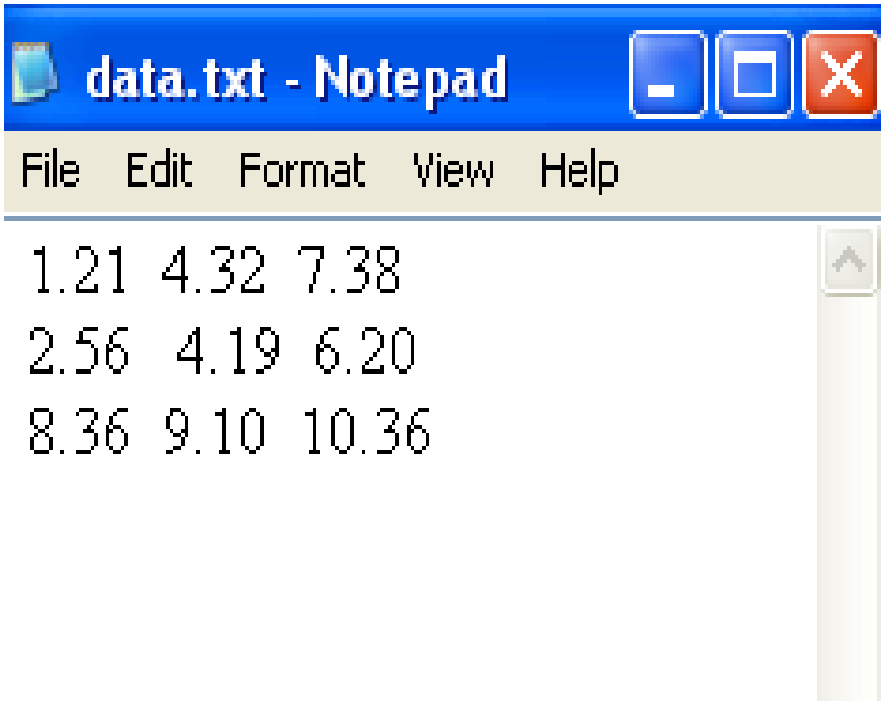


Because fscanf stops at the first mismatch!

fread and fscanf functions (Cont.)

```
a =
     1
    21
     4
    32
     7
    38
     2
    56
     4
    19
     6
    20
     8
    36
     9
    10
    10
    36
```

```
a=fscanf(fid,'%d.%d');
```



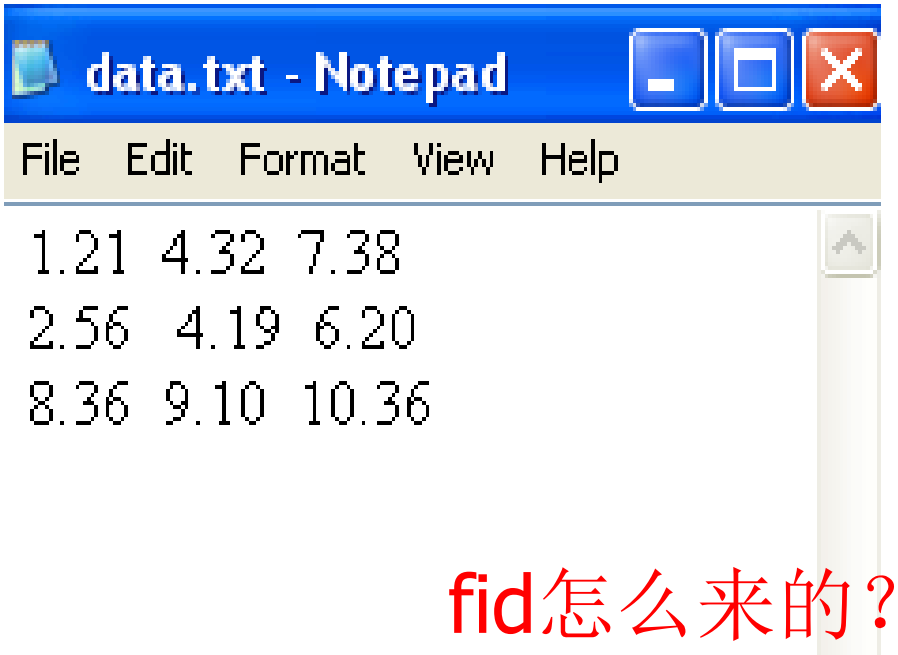
Delimiter [计] 定界符, 分隔符

fread and fscanf functions (Cont.)

a=fscanf(fid,'%s');

1.214.327.382.564.196.208.369.1010.36

The string specifier ignores white-space.



fid怎么来的？

fid=fopen('data.txt','r')

fwrite and fprintf functions

- fwrite: writes binary data to a file

fwrite(fid, array, precision);

array: array values to be written out;

Matlab writes out data in column order, which means that the entire first column is written out, followed by the entire second colum, and so forth.

fwrite and fprintf functions (Cont.)

- for example, array=[1 2 3;4 5 6]
- data will be written out in the order:

  1 4 2 5 3 6

fread行读列放。fwrite列读行放。See back Page 30：

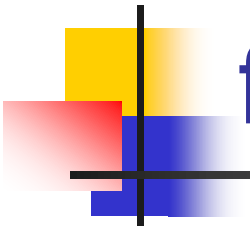
column-by-column read-and-write!

fwrite and fprintf functions (Cont.)

- fprintf function: writes formatted data in a user-specified format to a file

fprintf(format,var1,var2,…)

fwrite and fprintf functions (Cont.)



```
var=10;
fprintf(fid,'%d',var);

var=10.256;
fprintf(fid,'%.2f',var);        10.26

fprintf(fid,'%f',var);          10.256000
```

display 6 digits after the decimal place

fwrite and fprintf functions (Cont.)

```
var='matlab';
fprintf(fid,'%s',var)
```

```
var=10.256;
fprintf(fid,'%d',var);
```

The specifier will be ignored and the number will be displayed in exponential format

```
 1.025600e+001
```

File Closing

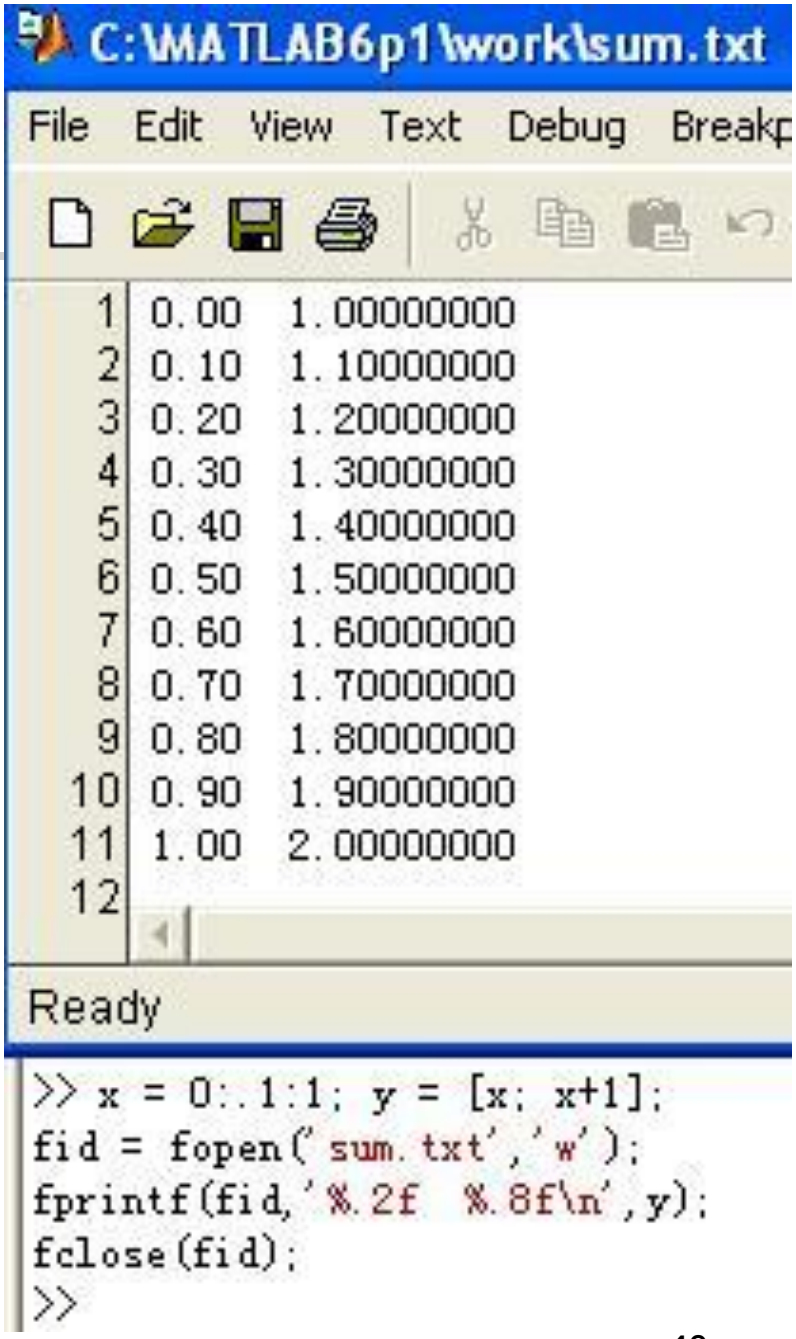
- fclose(fid)

fid怎么来的？

fid=fopen(‘data.txt’,’r’)

Example 1

```
x = 0:.1:1; y = [x; x+1];
fid = fopen(  'sum.txt','w');
fprintf(fid,'%.2f  %.8f\n',y);
fclose(fid);
```



Example 1 (Cont.)

x: 1*11

0 0.1 0.2 0.3 … 1

y: 2*11

0 0.1 0.2 0.3 … 1

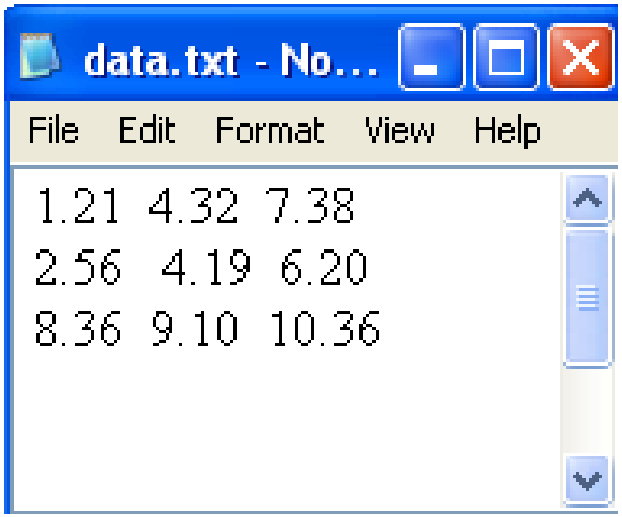
1 1.1 1.2 1.3 … 2

sum.txt

0.00 1.00000000
0.10 1.10000000
0.20 1.20000000
0.30 1.30000000
0.40 1.40000000
0.50 1.50000000
0.60 1.60000000
0.70 1.70000000
0.80 1.80000000
0.90 1.90000000
1.00 2.00000000

Example 2

```
fid = fopen('data.txt','r');
A=fscanf(fid,'%f',[3 3]);
fclose(fid);
```



1.21	2.56	8.36
4.32	4.19	9.1
7.38	6.2	10.36

See back Pages 30 & 35: column-by-column read-&-write!

Example 3

```
fid = fopen('data.txt','r');
A=fscanf(fid,'%f');
fclose(fid);

A=A+1;

fid = fopen('new_data.txt','w');
fprintf(fid,'%.6f\n',A);
fclose(fid);
```

data.txt

```
1.21  4.32  7.38
```

```
A =
    1.21
    4.32
    7.38
```

```
A =
    2.21
    5.32
    8.38
```

new_data.txt

```
2.210000
5.320000
8.380000
```

fscanf行读列放。fprintf列读行放

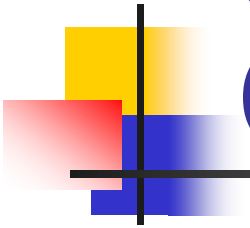
Comparing formatted and binary I/O

	Formatted file	Binary file
display data on output devices	Y	N
Easily transport data between different computers	Y	N

Comparing formatted and binary I/O (Cont.)

	Formatted file	Binary file
Require a relatively large amount of disk space	Y	N
Require a lot of computer time	Y slow	N fast
Truncation or rounding errors	Y large	N small

Comparing formatted and binary I/O (Cont.)



- **Formatted file**: data that must be readable by humans or that must be transferable between programs

- **Binary file**: data that do not have to be directly examined and will remain only on one type of computer

Comparing formatted and binary I/O (Cont.)

```
clear;
array=randn(1,100000);
tic;

fid=fopen('binary.dat','w');
fwrite(fid,array,'float64');
fclose(fid);

time=toc;

fprintf('Write time for binary file=%.10f\n',time);
```

Comparing formatted and binary I/O (Cont.)

```
clear;
array=randn(1,100000);
tic;
fid=fopen('formatted.dat','wt');
fprintf(fid,'%.15f\n',array);
fclose(fid);
time=toc;

fprintf('Write time for formatted file=%.10f\n',time);
```

To open files in text mode, attach the letter `'t'` to the `permission` argument, such as `'rt'` or `'wt+'`.

Comparing formatted and binary I/O (Cont.)

- Write time for binary file=0.0200000000
- Write time for formatted file=0.4910000000

0.491/0.02=24.55

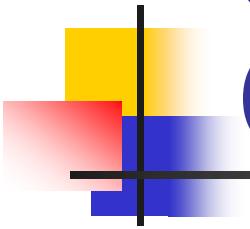
Comparing formatted and binary I/O (Cont.)

```
clear;
tic;

fid=fopen('binary.dat','r');
in_array=fread(fid,Inf,'float64');
fclose(fid);

time=toc;

fprintf('Read time for binary file=%.10f\n',time);
```

Comparing formatted and binary I/O (Cont.)

```
clear;
tic;

fid=fopen('formatted.dat','r');
in_array=fscanf(fid,'%f',Inf);
fclose(fid);

time=toc;

fprintf('Read time for formatted file=%.10f\n',time);
```

Comparing formatted and binary I/O (Cont.)



Read time for binary file=0.0000000000

Read time for formatted file=0.5710000000

读写之时间&内存的优化 vs 人类可理解程度

The existing Function

- exist: Check for the existence of *a variable in workspace*, *a built-in function*, or *a file in the Matlab search path*

```
ident=exist('item');
ident=exist('item','type');

 id=exist('formatted.dat');
 id=exist('formatted.dat','file');
```

in search of 寻找

The exist Function (Cont.)

- type:
  'var' , 'builtin' , 'file' , 'dir'
- ident:

value	meaning
0	item not found
1	item is a variable in the current workspace
2	item is an m-file or a file of unknown type
3	item is a MEX file
4	item is a MDL file
5	item is a built-in function
6	item is a pcode file
7	item is a directory

The feof Function

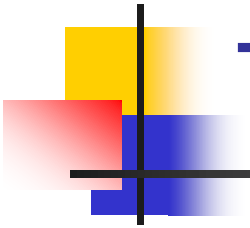
- feof: test to see if the current file position is at the end of the file

```
state=feof(fid);
```

1: the current file position is at the end of file

0: otherwise

F-E-o-F



The fgetl Function

- fgetl: reads the next line excluding the end-of-line characters from a file as a character string.

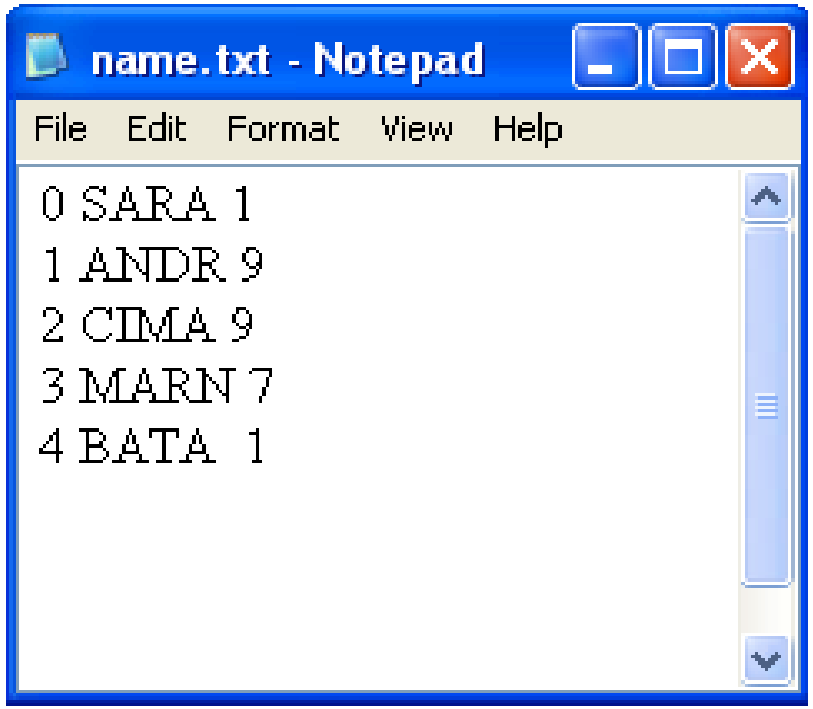
```
line=fgetl(fid);
```

If fgetl encounters the end of file,

then the value of line is -1.

The fgetl Function (Cont.)

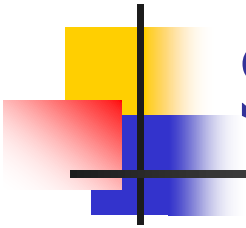
```
fid=fopen('name.txt');
while (~feof(fid))
    tline = fgetl(fid);
    disp(tline);
end
fclose(fid);
```



The frewind Function

- frewind: reset a file's position to the beginning of the file

```
frewind(fid);
```



Sincere Thanks!

- Using this group of PPTs, please read

- [1] Yunong Zhang, Weimu Ma, Xiao-Dong Li, Hong-Zhou Tan, Ke Chen, MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs, Neurocomputing 72 (2009) 1679-1687

- [2] Yunong Zhang, Chenfu Yi, Weimu Ma, Simulation and verification of Zhang neural network for online time-varying matrix inversion, Simulation Modelling Practice and Theory 17 (2009) 1603-1617