THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

ENG 101 Module #4
MATLAB Input/Output

Dr. Emil Jovanov
Electrical and Computer Engineering
University of Alabama in Huntsville
emil.jovanov@uah.edu
http://www.ece.uah.edu/~jovanov

User Input/Output

- Asking user for input
  - Processing based on user input
  - Interactive change of parameters
- Formatting output
  - Describe the result
  - Prepare for reports and documentation

Prompting for Numerical Input

- *variable_name = input('message prompt ')*

- Example (go to MATLAB editor):
- byear = input('Enter your year of birth ')
- cyear = input('Enter the current year ')
- Your_Age = cyear-byear

Prompting for String Input

- *name = input('message prompt ', 's')*

- Example (go to MATLAB):
- name = input('Enter your name: ', 's')

Prompting for Publishing

- Regular Input statement generates error during publishing
- Use Input Dialog (`inputdlg`)

- `var = inputdlg(‘message prompt ’)`

- Pay attention to format!!!
  - Cell > String > Numeric input

- speed=str2num(cell2mat(inputdlg('Enter speed in [mph]: ')));

Displaying Output

1. We can simply type the variable and hit enter.
2. We can use the disp command:

```
disp(variable_name) or
disp(‘string’) or
disp(‘ ’)
```

3. We can use the fprintf command:

```
fprintf(‘string’) or
fprintf(‘some text %f more text’, var)
```

Using disp

num=10;

str='Pick a number';

disp(num)

disp(str)

disp(' ')

disp(4+num)

disp('the end')

- Notes on disp:
- *Automatically starts a new line*
- *Cannot combine multiple elements*

Using fprintf

- **%xy.z**C

- x can be a minus sign (left-justifies the number within the field), plus sign (prints a sign character in front of the number), or a zero (adds zeros to fill the field)
- y is the field width
- .z is the precision (number of digits to be displayed to the right of the decimal)
- C is the conversion character which determine the number format (f=fixed point notation, e/E is exponential notation, g/G is the shorter of e/E or f notations, s is for string data)

- %s - print a string
- %g – print a compact number
- %e - print a scientific number
- %f - print a floating point number
- \n - print a new line (go to the next line to continue printing)
- \t - print a tab
- \\ - print a slash
- %% - print a percent sign

fprintf

- Means file print formatted
  - *formatted text* is text that can be read by people
  - *unformatted text* looks random to people but computers can read it
- Can write to screen or to a file
- Can mix numbers and text in output
- Have full control of output display
- Complicated to use

Using fprintf

```
%Basketball stats
p_name='Sam';
games=100;
ppg=14.1738;
reb=10.9;
assists=4.359;

Fprintf('Over %g games, %s scored an average of %f points
per game.', games, p_name, ppg)
```

To make the next thing that MATLAB writes (after a use of `fprintf`) appear on the start of a new line, put the two characters "\n" at the end of the `fprintf` text

```
>> fprintf( ‘My text\n' )
My text
>>
```

Can also use `\n` in middle of text to make MATLAB display remainder of text on next line

```
>> fprintf(‘1st line\n2nd line\n3rd ...\n')
1st line
2nd line
3rd ...
>>
```

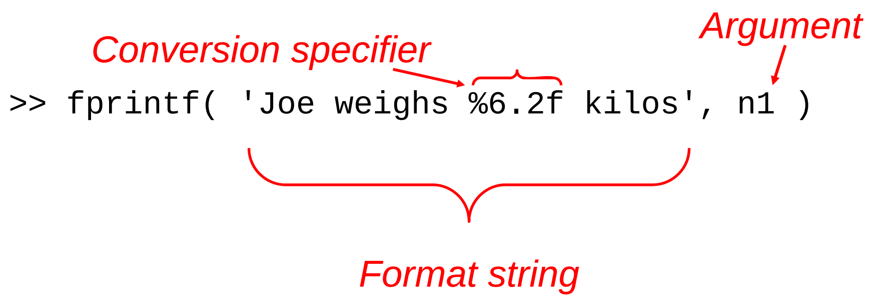
`\n` is an *escape character*, a special combination of two characters that makes `fprintf` do something instead of print the two characters

`\n` – makes following text come out at start of next line

`\t` – horizontal tab

There are a few more

fprintf(format, n1, n2, n3)



```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

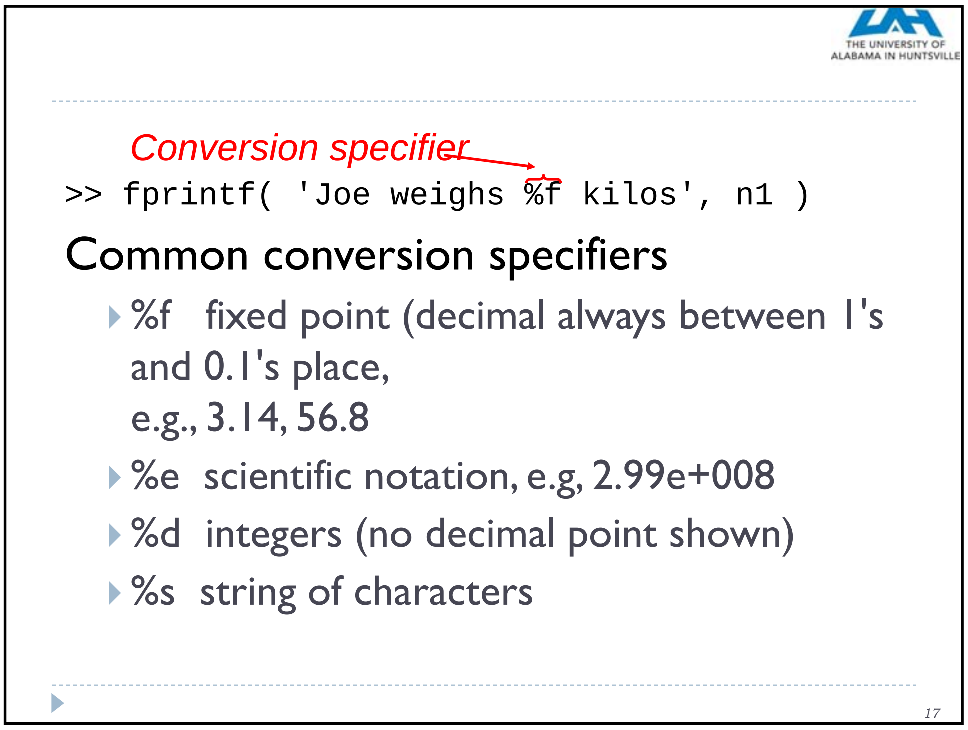
Format string

- May contain text and/or conversion specifiers
- Must be enclosed in SINGLE quotes, not double quotes, aka quotation marks (" ")

```
>> fprintf( 'Joe is %d weighs %f kilos', age, weight )
```

Arguments

- Number of arguments and conversion specifiers must be the same
- Leftmost conversion specifier formats leftmost argument, 2nd to left specifier formats 2nd to left argument, etc.

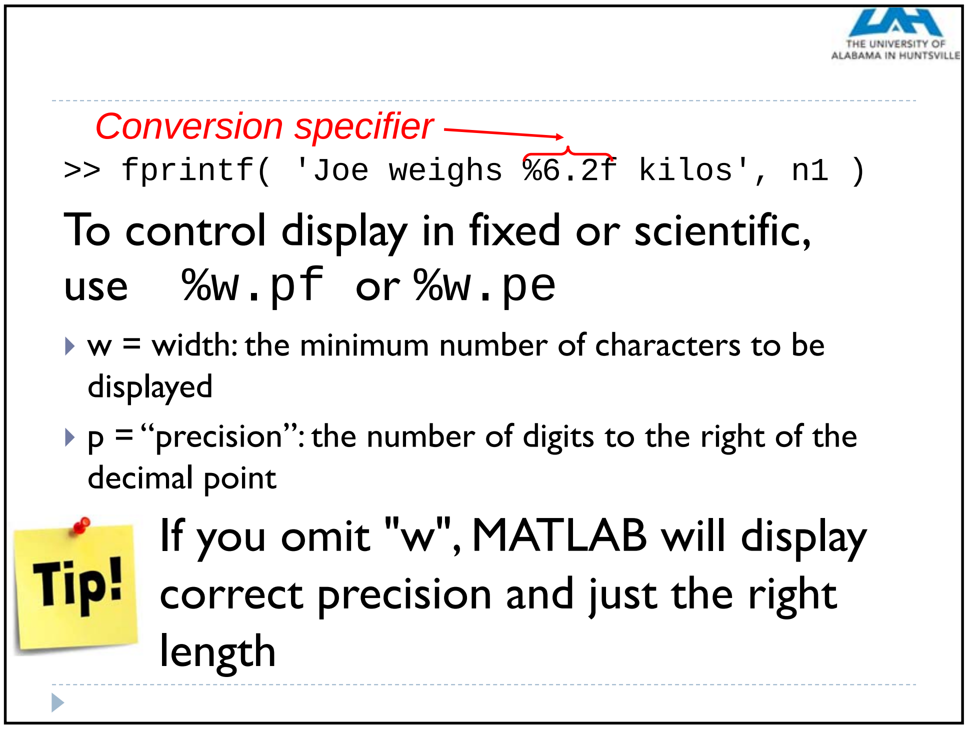


Conversion specifier

```
>> fprintf( 'Joe weighs %f kilos', n1 )
```

Common conversion specifiers

- %f fixed point (decimal always between 1's and 0.1's place, e.g., 3.14, 56.8
- %e scientific notation, e.g, 2.99e+008
- %d integers (no decimal point shown)
- %s string of characters



Conversion specifier

```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

To control display in fixed or scientific, use `%w.pf` or `%w.pe`

- w = width: the minimum number of characters to be displayed
- p = "precision": the number of digits to the right of the decimal point

Tip!

If you omit "w", MATLAB will display correct precision and just the right length

```
>> e = exp( 1 );
>> fprintf( 'e is about %4.1f\n', e )
e is about  2.7
>> fprintf( 'e is about %10.8f\n', e )
e is about 2.71828183
>> fprintf( 'e is about %10.8e', e )
e is about 2.71828183e+000
>> fprintf( 'e is about %10.2e', e )
e is about  2.72e+000
>> fprintf( 'e is about %f\n', e )
e is about 2.718282
```

Use escape characters to display characters used in conversion specifiers

- To display a percent sign, use %% in the text
- To display a single quote, use '' in the text (two sequential single quotes)
- To display a backslash, use \\ in the text (two sequential backslashes)

Make the following strings

- Mom's apple 3.14
- Mom's apple 3.1415926
- Mom's apple 3.1e+000

```
>> fprintf( 'Mom''s apple %.2f\n', pi )
Mom's apple 3.14
>> fprintf( 'Mom''s apple %.7f\n', pi )
Mom's apple 3.1415927
>> fprintf( 'Mom''s apple %.1e\n', pi )
Mom's apple 3.1e+000
```

Format strings are often long. Can break a string by

1. Put an open square bracket ([ ) in front of first single quote
2. Put a second single quote where you want to stop the line
3. Follow that quote with an ellipsis (three periods)
4. Press ENTER, which moves cursor to next line
5. Type in remaining text in single quotes
6. Put a close square bracket ( ])
7. Put in the rest of the `fprintf` command

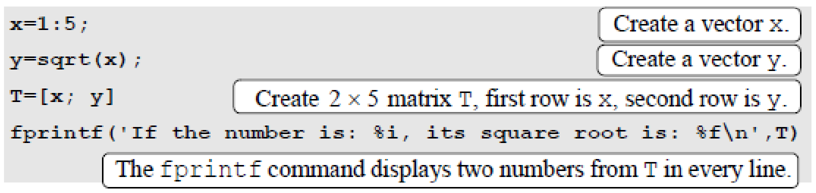
Example

```
>> weight = 178.3;
>> age = 17;
>> fprintf( ['Tim weighs %.1f lbs'...
' and is %d years old'], weight, age )

 Tim weighs 178.3 lbs and is 17 years old
```

fprintf is *vectorized*, i.e., when vector or matrix in arguments, command repeats until all elements displayed

- Uses matrix data column by column



```
x=1:5;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

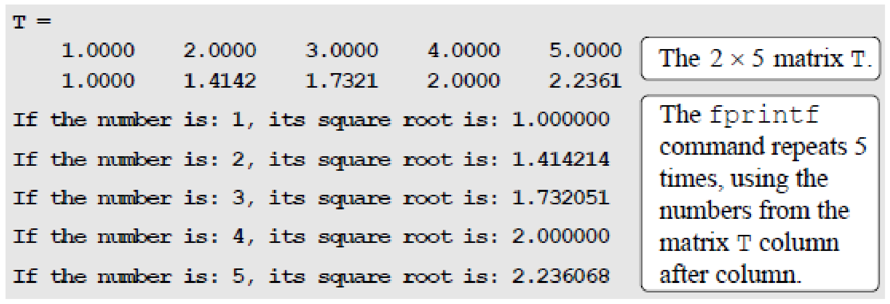
Create a vector x.

Create a vector y.

Create 2×5 matrix T, first row is x, second row is y.

The fprintf command displays two numbers from T in every line.

When this script file is executed the display in the Command Window is:



```
T =
    1.0000    2.0000    3.0000    4.0000    5.0000
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000
If the number is: 2, its square root is: 1.414214
If the number is: 3, its square root is: 1.732051
If the number is: 4, its square root is: 2.000000
If the number is: 5, its square root is: 2.236068
```

The 2 × 5 matrix `T`.

The `fprintf` command repeats 5 times, using the numbers from the matrix `T` column after column.

Using the fprintf command to save output to a file

Takes three steps to write to a file

<u>Step 1</u>: – open file

```
fid=fopen('file_name','permission')
```

`fid` – *file identifier*, lets `fprintf` know what file to write its output in

`permission` – tells how file will be used, e.g., for reading, writing, both, etc.

Some common permissions

- `r` - open file for reading
- `w` - open file for writing. If file exists, content deleted. If file doesn't exist, new file created
- `a` - same as `w` except if file exists the written data is appended to the end of the file
- If no permission code specified, `fopen` uses `r`

See Help on `fopen` for all permission codes

Step 2:

Write to file with `fprintf`. Use it exactly as before but insert `fid` before the format string, i.e.,

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

The passed `fid` is how `fprintf` knows to write to the file instead of display on the screen

Step 3:

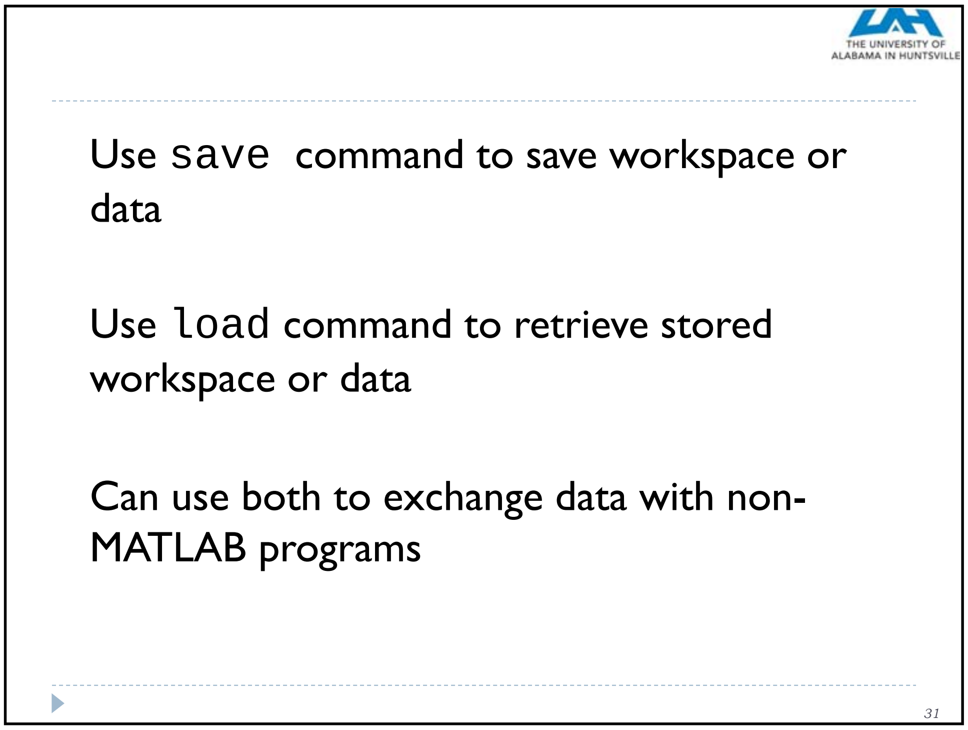
When you're done writing to the file, close it with the command `fclose(fid)`

- Once you close it, you can't use that `fid` anymore until you get a new one by calling `fopen`

Make sure to close every file you open. Too many open files makes problems for MATLAB

Miscellaneous

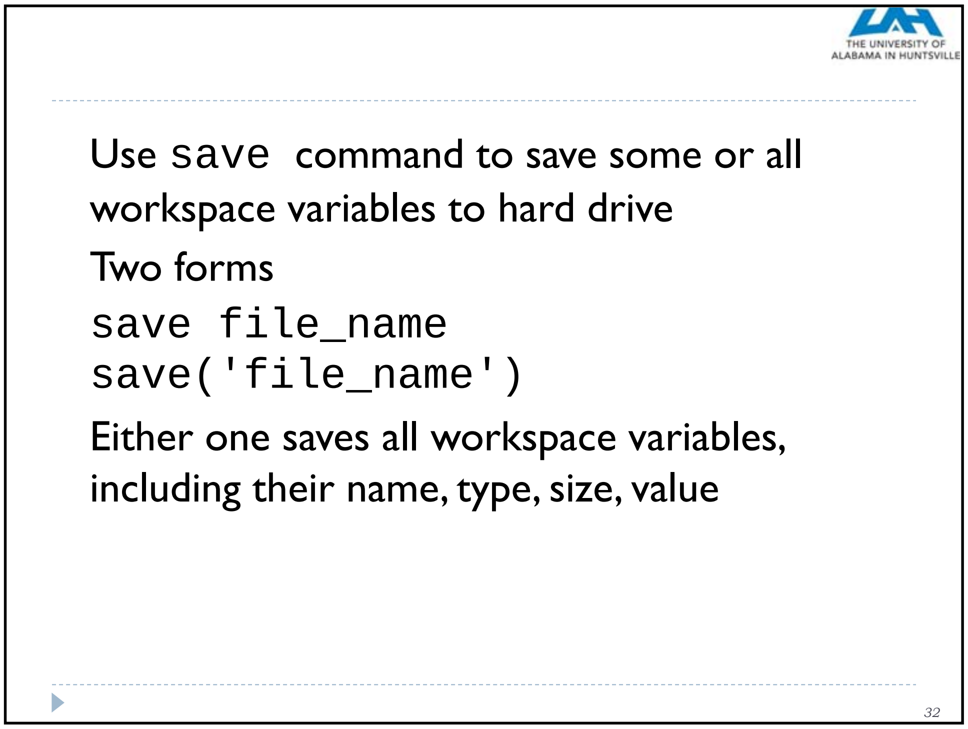
- If the file name you give to `fopen` has no path, MATLAB writes it to the current directory, also called **the working directory**
- You can have multiple files open simultaneously and use `fprintf` to write to all of them just by passing it different `fids`
- You can read the files you make with `fprintf` in any text editor, e.g., MATLAB's Editor window or Notepad


Use `save` command to save workspace or data

Use `load` command to retrieve stored workspace or data

Can use both to exchange data with non-MATLAB programs

31


Use `save` command to save some or all workspace variables to hard drive

Two forms

```
save file_name
save('file_name')
```

Either one saves all workspace variables, including their name, type, size, value

32

To only save specific variables, list variables after file name. For example, to save two variables named `var1` and `var2`

```
save file_name var1 var2
save('file_name','var1','var2')
```

All forms store variables in file called "file_name.mat"

- Called "mat" file
- Unformatted (binary) file
  - Only MATLAB can read mat file, not other programs
  - Can't read file in text editor, or MATLAB Editor Window

To save as formatted text (also called *ASCII text*)

```
save file_name -ascii
```

IMPORTANT – only saves <u>values</u> of variables, no other info, even their names!

- Can also just save certain variables, as before
- Usually just use to save value of one variable

To load data in a <u>mat</u> file into workspace

```
load file_name
load( 'file_name')
```

To load only specific variables from mat file, e.g., var1 and var2

```
load file_name var1 var2
load('file_name','var1','var2')
```

- If variable already exists in workspace, it is *overwritten* (its value is replaced by value in file)

To load data in a text file into workspace

```
load file_name
variable = load( 'file_name')
```

- In first form, creates variable called `file_name` and stores all file data in it
- If all rows in file don't have same number of columns, MATLAB displays an error
- Even if data created from multiple variables all with same number of columns, `load` still reads all data into one variable
  - Not very useful in this case

- MATLAB often used to analyze data collected by other programs
- Sometimes need to transfer MATLAB data to other programs
- In this section will only discuss numerical data
  - MATLAB has commands to load and save data from a number of other programs
  - Can also tell MATLAB what format data is in