Unit 7:
Input/Output Files

Objectives

- Understand different ways of storing and retrieving information from the hard drive.
- Understand the differences between an ASCII file an a binary file.
- Learn how to write and read ASCII and binary files.

Content

- Intro
- Manually: Import/Export data
- Programmatically
  - ASCII Files
  - Binary Files

Input and Output

- Sometimes we may need to permanently store data and thus data will persist after the execution of the program.
- We may also need to import or export data to other applications.

IMPORT / EXPORT

Import / Export

- Export:
  - *save:* Save workspace variables to a file with the extension .mat
- Import
  - *load:* Import previously exported MATLAB variables using the command *save*
  - *xlsread:* Import data from excel files
  - Using the import data assistant

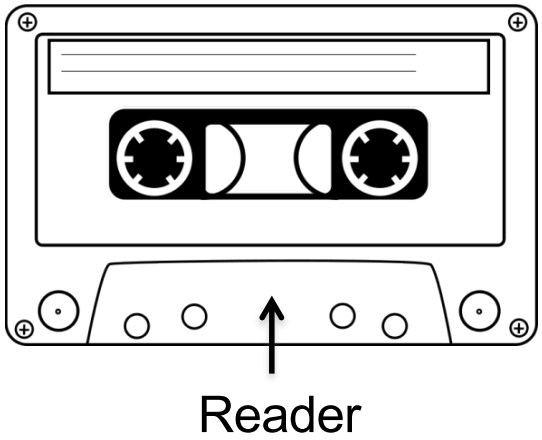
ASCII AND BINARY FILES

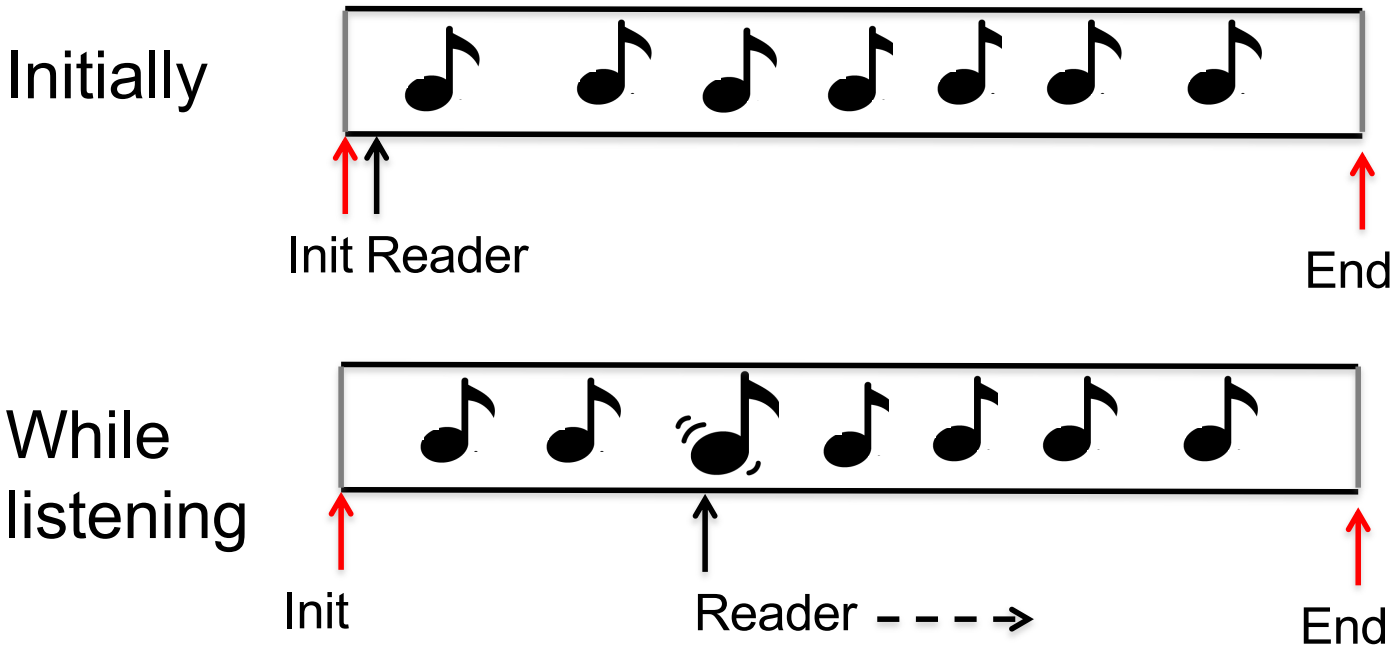
ASCII and Binary Files

- We use ASCII and Binary files to save the output data that our program generates.
  - These files are stored in our hardrive in the same way as any other data file like word files, excel files, music files..
  - **ASCII files** contain **text** (ASCII characters). We normally save them with the file extension ***.txt***
  - **Binary files** contain **binary information** (0s and 1s), which could represent numbers, images, sounds,….

ASCII and Binary Files

- The information in the file (ASCII or Binary) is read sequentially, such as on a music tape.





While listening the **reader** moves along the tape, transforming the information read into sound

ASCII and Binary Files

- We read and write the information in the files sequentally:
  - Initially, when you open the file, the “pointer” is on the first position of the file
  - When we read or write (text or binary) the pointer moves to the next position after the data read or writen.
  - Most of the times you will want to write at the end of the file, so you don’t overwrite what it’s already in the file.

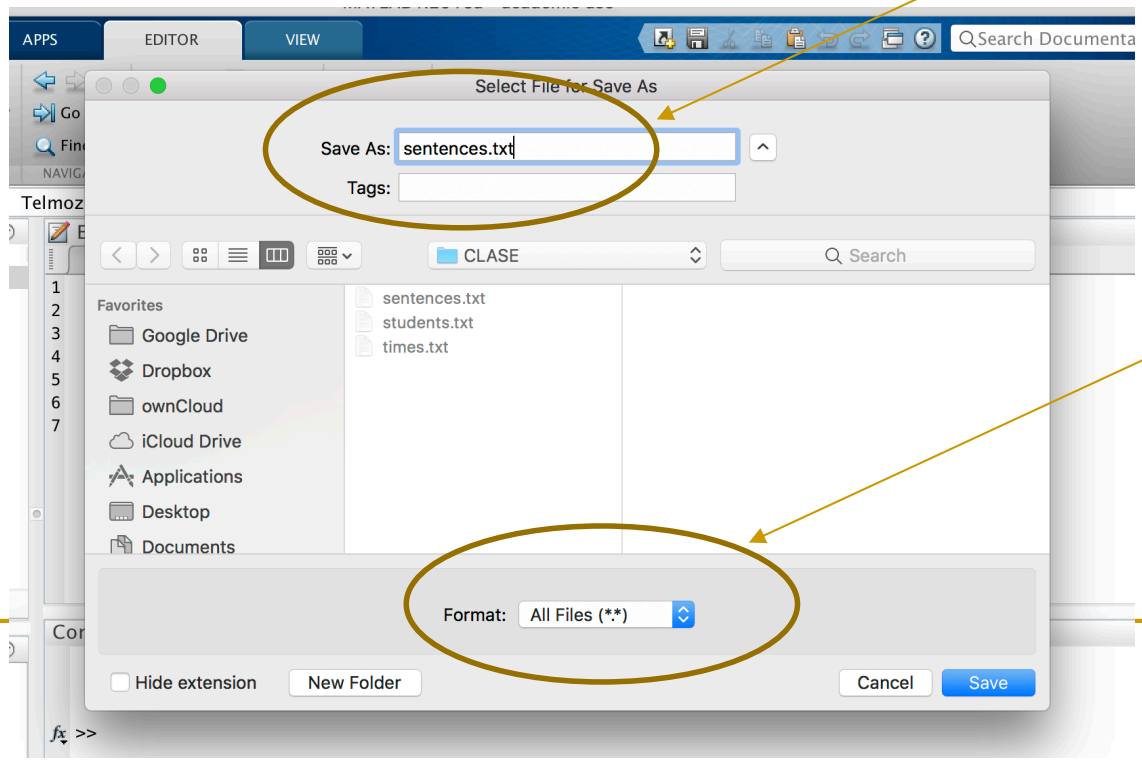
ASCII FILES

Read and Write ASCII Files

- An ASCII file contains ASCII characters
  - **Text files which can be read by a naked eye**
- ASCII files can be created by:
  - Our programs: instead of printing the output on screen we create an ASCII file and put print the results there
  - Text editors: ASCII text editors as notepad, textedit, the MATLAB editor,…
    - Remember: the Microsoft Word editor is not an ASCII text editor so don't use it to create ASCII files for your programs.

Creating an ASCII Files with the MATLAB editor

- To create an ASCII file:
  - Click on “New script”
  - Write the text you want to save in the file
  - Save the file using the “.txt” extension



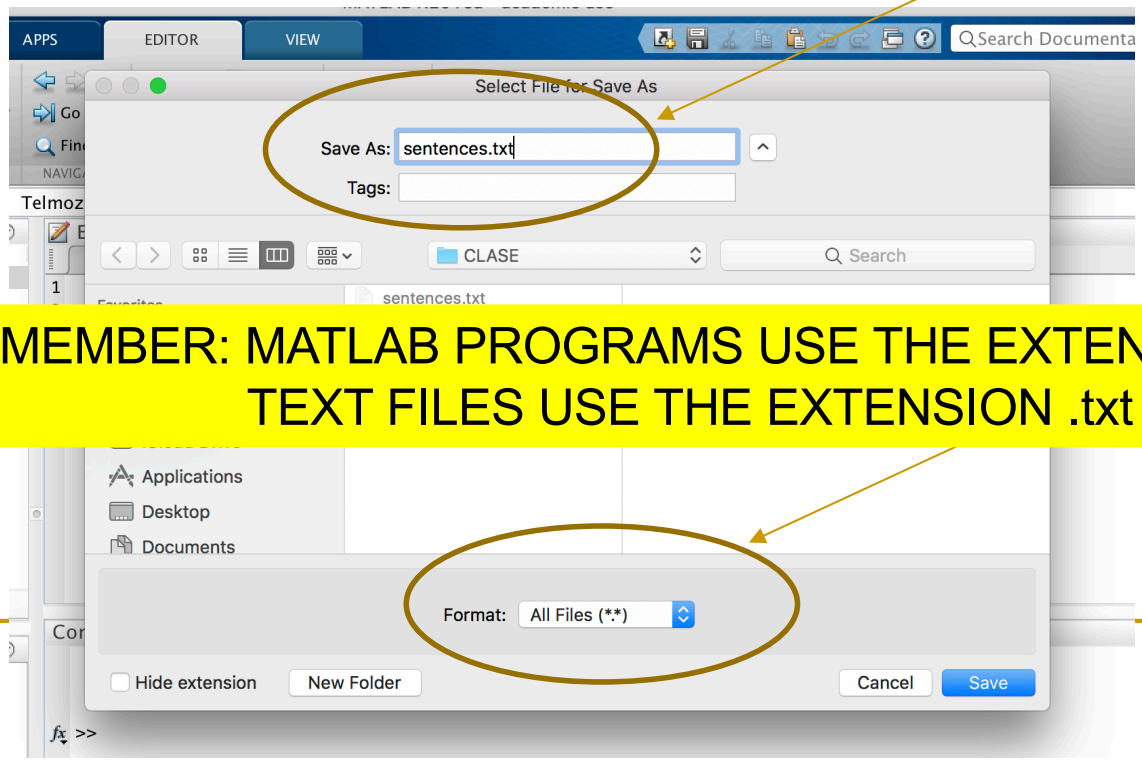
Write the name of your file including the .txt at the end

Make sure the file format is not .m

Creating an ASCII Files with the MATLAB editor

- ASCII files can be created by:
  - Click on “New script”
  - Write the text you want to save in the file
  - Save the file using the “.txt” extension

Write the name of your file including the .txt at the end



Make sure the file format is not .m

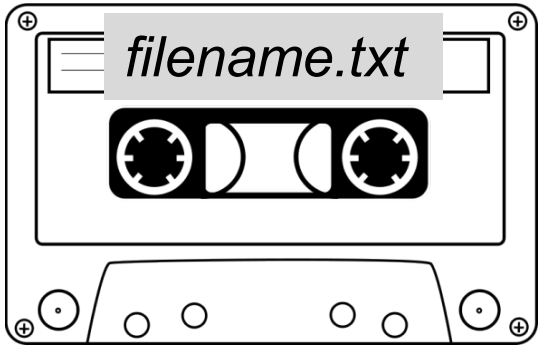
REMEMBER: MATLAB PROGRAMS USE THE EXTENSION .m
TEXT FILES USE THE EXTENSION .txt

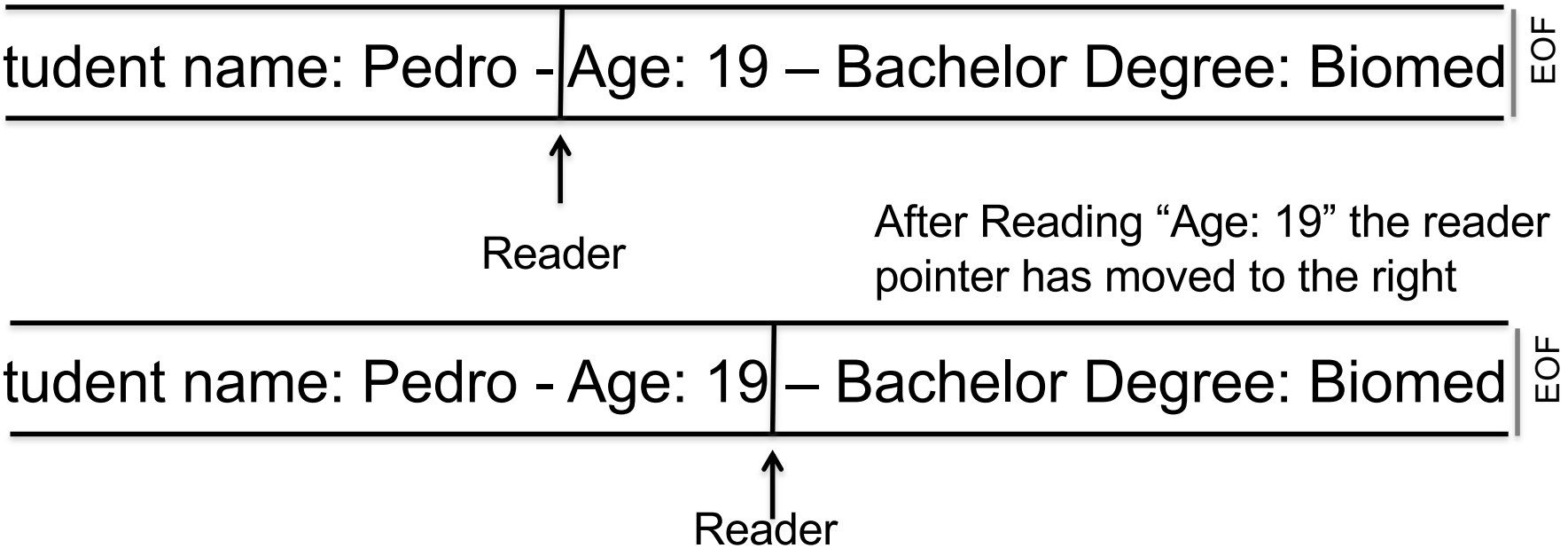
Read and Write ASCII Files

- In an analogous way to other programming languages as C, MATLAB provides functions to read and write data into an ASCII file.
- The process will be:
  - 1 step: Open file (*fopen*)
  - 2 step: Read or write information (*fscanf, textscan, fget, fprintf*)
  - 3 step: Close file (fclose)

Read and Write ASCII Files

- The information in the file is read sequentially, such as on a tape.
- The “open” operation leaves the reader pointer marking the beginning of the tape
- The “read” or “write” operations move the pointer to the right





Opening and closing files

- Opening a file

```
[fi, text]= fopen ('filename', '--')
```

Remember: we use the name to open the file and obtain an identifier. For the rest of the operations we will use this identifier, never the name.

File identifier. In case MATLAB fails to open the file its value is -1

OPTIONAL. In case MATLAB fails to open the file provides an error message here

Name of the file

Two or three **control characters** for indicating the operation to be performed:
'rt'
'wt'
'at'
…

- Closing a file

```
st = fclose (fi)
```

OPTIONAL. Value indicating possible errors

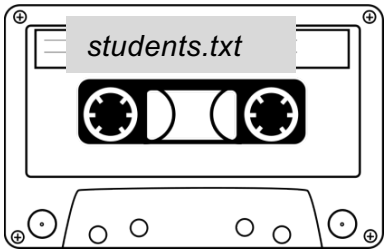
Opening and closing files

- Control characters for opening a text file

Characters	Operation
‘rt’	Open file for reading (default).
‘wt’	Open file, or create new file if it doesn’t exist, for writing; discard existing contents, if any
‘at’	Open file, or create new file if it doesn’t exists, for writing; append data to the end of the file.
‘rt+’	Open file for reading and writing.
‘wt+’	Open file, or create new file if it doesn’t exist, for reading and writing; discard existing contents, if any.
‘at+’	Open file, or create new file if it doesn’t exist, for reading and writing; append data to the end of the file

Opening files

```
fid = fopen ('students.txt', 'rt')
```



Reader

When we open the file with 'rt' or 'wt' the reader points at the beggining of the file

Student name: Pedro - Age: 19 – Bachelor Degree: Biom

```
fid = fopen ('students.txt', 'at')
```

When we "append" the reader starts at the end of the file

Reader

achelor Degree: Biomedical | EOF

Checking out if the pointer is at the end of the file

- feof(fi) Returns 1 if the file position indicator has been set by the previous operation in the end of the file, and 0 otherwise.

  Note: fi is a file identifier

feof is very usefull. In many problems you will have to read files until you reach the end of the file. For example, when counting the elements in a file that satisfy a given condition. In these cases we use 'while' loops that read until they reach the end of the file… and you will know that because at that point *feof(fid)* will return 1.

Writting ASCII files

- Writing to a text file

```
fprintf (fi, 'control characters', var1, var2,….)
```

File identifier

String for indicating the format of the information to be read.

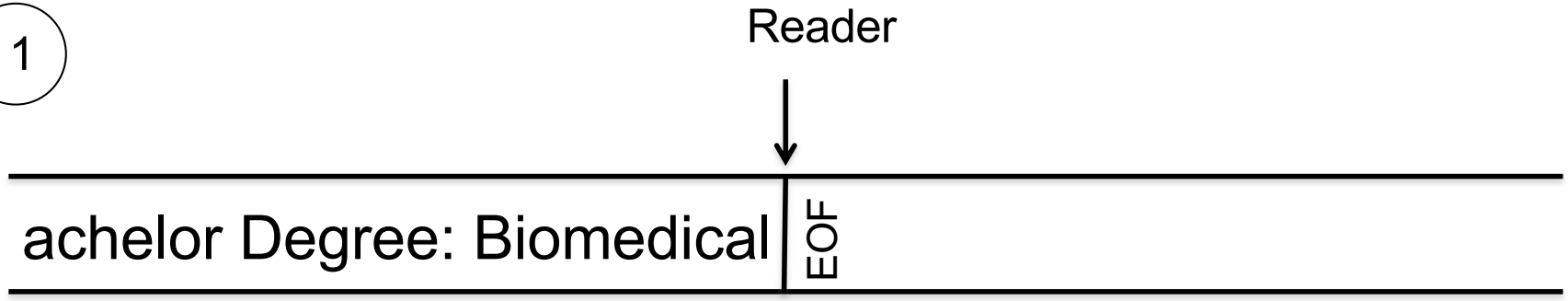
'%s'	strings
'%c'	characters
'%d'	integers
'%f'	float point

Variables to be written

We use **fprintf** in the same way as when printing in the screen. The only difference is that we **specify the id (file identifier)** of the file in which we want to print.

Writting ASCII files

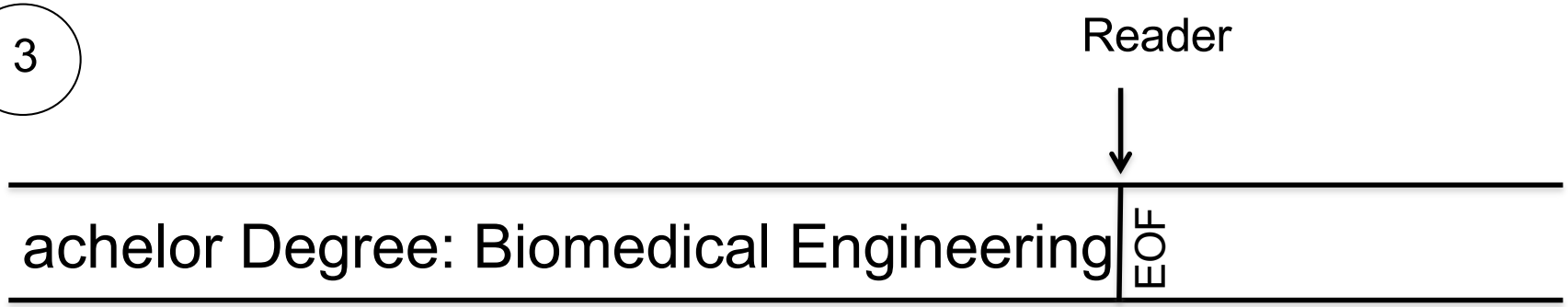
1



2

Execute the command: `fprintf(fi,'%s', 'Engineering');`

3



Example

- Write a script which asks the user to introduce a sentence, and store the sentence in a ASCII file called ‘sentence.txt’

Example

- Write a script which asks the user to introduce a sentence, and store the sentence in an ASCII file called ‘sentence.txt’

```
vfile = fopen('sentence.txt','wt');
if vfile == -1
     disp(‘Error when trying to open the file’);
else
     vsentence = input('Introduce a sentence: ',’s');
     fprintf(vfile,'%s',vsentence);
     fclose(vfile);
end;
```

Example

- Write a script which asks the user to introduce a sentence, and store the sentence in a ASCII file called ‘sentence.txt’

```
vfile = fopen('sentence.txt','wt');
if vfile == -1
     disp(‘Error when trying to open the file’);
else
     vsentence = input('Introduce a sentence: ',’s');
     fprintf(vfile,'%s',vsentence);
     fclose(vfile);
end;
```

What would you find in the ‘sentence.txt’ file if you executed this code twice?

Example

- Write a script which asks the user to introduce a sentence, and store the sentence in a ASCII file called ‘sentence.txt’

```
vfile = fopen('sentence.txt','wt');
if vfile == -1
     disp(‘Error when trying to open the file’);
else
     vsentence = input('Introduce a sentence: ',’s');
     fprintf(vfile,'%s',vsentence);
     fclose(vfile);
end;
```

What would you find in the ‘sentence.txt’ file if you executed this code twice?
Only the last sentenced introduced

Example

- Write a script which asks the user to introduce a sentence, and store the sentence in a ASCII file called ‘sentence.txt’ without deleting the content stored in it.

Example

- Write a script which asks the user to introduce a sentence, and store the sentence in a ASCII file called ‘sentence.txt’ without deleting the content stored in it.

```
vfile = fopen('sentence.txt','at');
if vfile == -1
     disp(‘Error when trying to open the file’);
else
     vsentence = input('Introduce a sentence: ','s');
     fprintf(vfile,'%s',vsentence);
     fclose(vfile);
end;
```

Reading ASCII files

- You can use three different commands to read from a text file:
  - fscanf => returns the data read in a vector or matrix
  - textscan => returns the data read in a cell array
  - fgets / fgetl => returns a string containing a whole line

Reading ASCII files: fgets

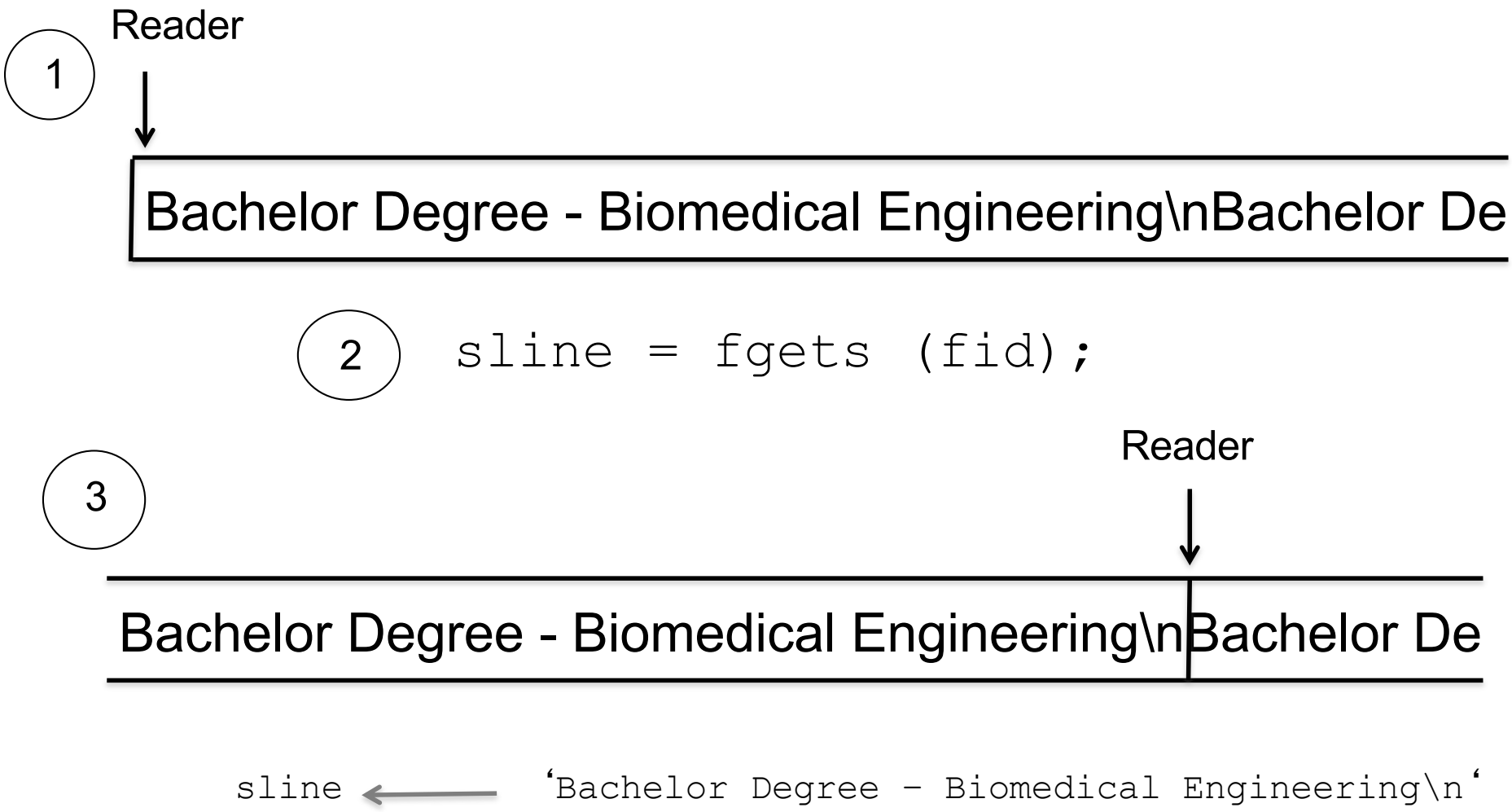
- Reading a whole line.

- sline = fgets(file_Id) — Reads the next line of the specified file and returns a string containing all the characters including the newline character at the end of the line ('\n')

- sline = fgetl(file_Id) — Reads the next line of the specified file and returns a string containing all the characters removing the newline character.

MATLAB marks the end of a line of text using the special character: \n
Note: this is a character you won't see if you open the file in a file editor.. but you'll know is there because the next character will be displayed in the a new line

Reading ASCII files: fgets



Reading ASCII files: fgets

Note: The special character ***\n*** *has the number 10 in the ASCII table. If in one of your programs you need to check if a character read from a file is the* ***\n*** *you should compare it with char(10).*

Example

```
if vcharacter == char(10)
        …
```

Never try something like this:

```
if vcharacter == '\n'
        …
```

MATLAB would think you are "trying to" compare vcharacter with a String that contains the characters '\' and 'n'

Reading ASCII files: fscanf and textscan

- Sometimes you don't want to read the whole line of a file.
- To only read some pieces of information you can use *fscanf* or *textscan*
  - You specify how many pieces of what type of information (integers, characters, double..) you want to read.
  - MATLAB reads the information in the file and (tries to) converts it into the type of data you asked for.

Reading ASCII files: fscanf

- Reading from a text file: *fscanf*

```
[var1, count]= fscanf (fi, 'control characters', size)
```

Vector or matrix in which the information is stored.
var1 is a char vector if only *c* or *s* control characters are included
Otherwise var1 is a numeric vector.

Optional. The amount of elements fscanf has read into var1

File identifier Previously obtaiined when opening the file

Same control characters as the ones used in fprint (%c,%s, %d…) They specify if you want to read a character, a string, a number…

Optional. Specifies how many elements are going to be read. When not specified it reads all the data in the file, so be carefull

REMEMBER: fscanf returns a vector or a matrix… and this kind of data structure can only store one single type of data

Reading ASCII files: fscanf

- Reading from a text file: *fscanf*

```
[var1, count]= fscanf (fi, 'control characters', size)
```

Vector or matrix in which the information is stored.
var1 is a char vector if only *c* or *s* control characters are included
Otherwise var1 is a numeric vector.

Optional. The amount of elements fscanf has read into var1

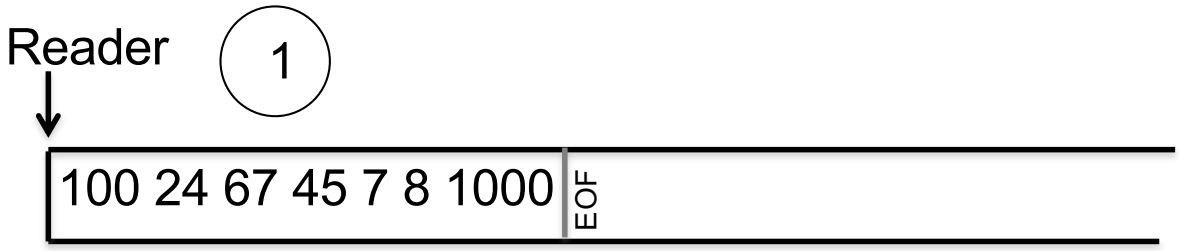
File identifier Previously obtaiined when opening the file

Same control characters as the ones used in fprint (%c,%s, %d…) They specify if you want to read a character, a string, a number…

Optional. Specifies how many elements are going to be read. When not specified it reads all the data in the file, so be carefull

If the content in the file doesn't match what you ask to read using the control characters.. you will get errors or unexpected results.

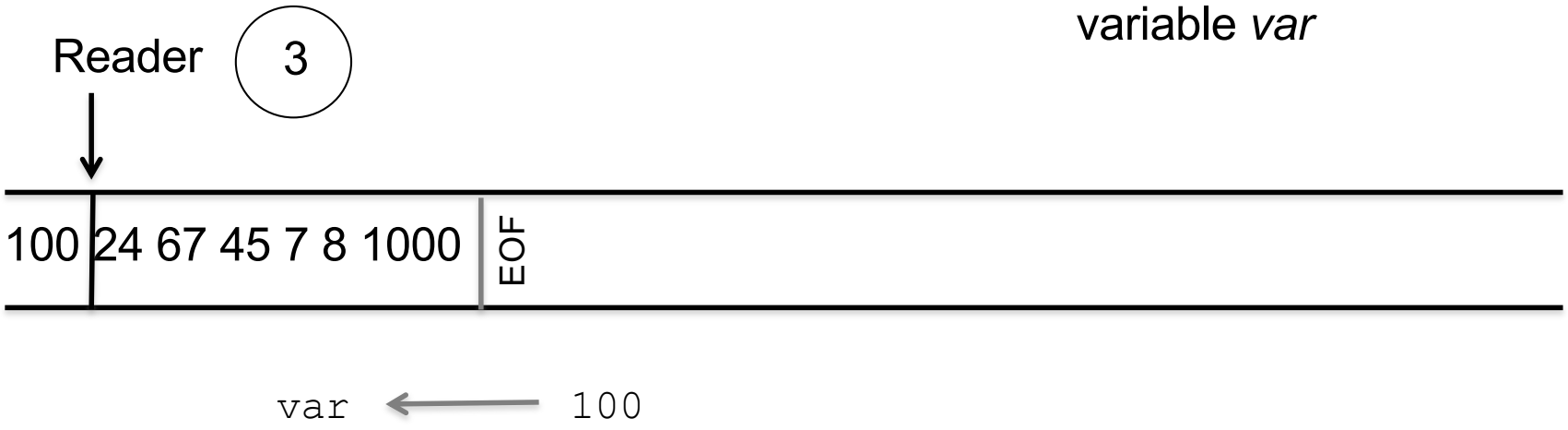
Reading ASCII files



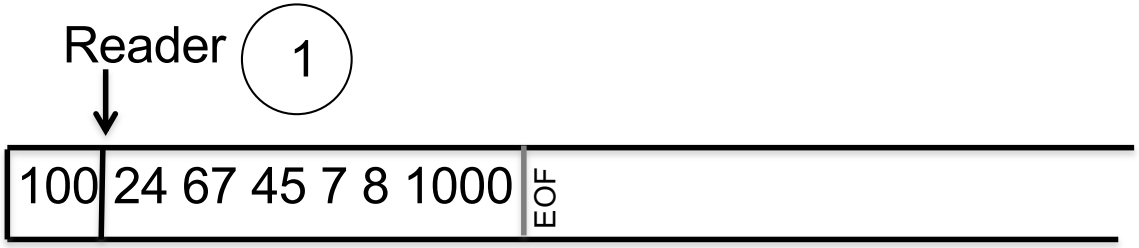
2

```
var = fscanf (fid, '%d', 1);
```

Reads 1 integer from the **file** Identified by the **identifier *fid*** and returns the value in the variable *var*



Reading ASCII files: fscanf

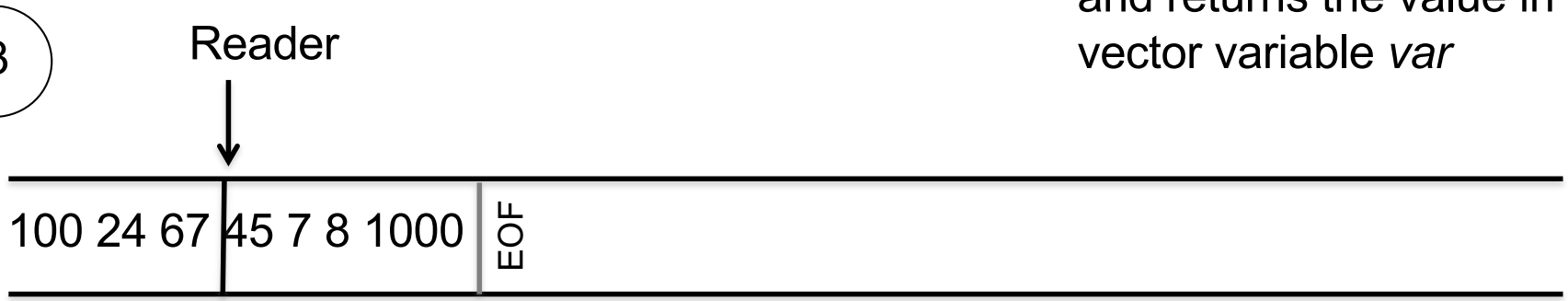


2

```
var = fscanf (fid, '%d', 2);
```

Reads 2 integers from the file Identified by the identifier *fid* and returns the value in the vector variable *var*

3




```
var(1) ← 24
var(2) ← 67
```

Reading ASCII files: fscanf

- Examples:

```
[var1, count]= fscanf (fi, '%s', 1);
```

Reads one string. **MATLAB considers that the strings in the file are delimited by blank spaces.**

```
[var1, count]= fscanf (fi, '%f', 2);
```

Read two floating-points numbers and stored them in the positions var1(1) and var1(2)

```
[var1, count]= fscanf (fi, '%10c', 1);
```

Reads one block of ten characters.

```
[var1, count]= fscanf (fi, '%f');
```

Read all the floating-points numbers and stores them in the vector var1. *Count* contains the number of floating-point numbers MATLAB could read.

```
[var1, count]= fscanf (fi, 'The password is: %s',1);
```

Skip the sentence 'The password is: ' in the file and return the contiguous string

Example

- Write a script which reads one word from a text file called ‘sentence.txt’ and prints it on screen. Use fscanf

Example

- Write a script which reads one word from a text file called ‘sentence.txt’ and prints it on screen. Use fscanf

```
vfile = fopen('sentence.txt', 'rt');
if vfile == -1
     disp('Error when opening the file');
else
      vword = fscanf (vfile,'%s',1);
      fprintf('The word is: %s', vword);
      fclose(vfile);
end;
```

Example

- Write a script which reads one word from a text file called ‘sentence.txt’ and prints it on screen. Use fscanf

```
vfile = fopen('sentence.txt', 'rt');
if vfile == -1
     disp('Error when opening the file');
else
      [vword, vcont] = fscanf (vfile,'%s',1);
      if vcont > 0    % with this we check that we have read
                      % something
            fprintf('The word is: %s', vword);
      end;
      fclose(vfile);
end;
```

Example

- Write a script which reads all the words of a text file called ‘sentence.txt’ and prints them on the screen. Use fscanf

Example

- Write a script which reads all the words of a text file called ‘sentence.txt’ and prints them on the screen. Use fscanf

```
vfile = fopen('sentence.txt','rt');
if vfile == -1
     disp(’Error when opening the file’);
else
   while feof(vfile) == 0
       vword = fscanf (vfile,'%s',1);
       fprintf('\nThe word is: %s', vword);
   end
   fclose(vfile);
end;
```

Example

- Write a script which asks the user to introduce the name of a file to read and prints on screen all the lines in the file.

Example

- Write a script which asks the user to introduce the name of a file to read and prints on screen all the lines in the file.

```
vname = input('Introduce the name of the file', 's');
vfile  = fopen(vname,'rt');
if vfile == -1
    disp('Error when opening the file');
else
   while (feof(vfile) == 0 )
     vline = fgets (vfile);
    fprintf('%s ', vline);
   end;
   fclose(vfile);
end;
```

The variable vline contains an string that ends up with the character '\n'

Example

- Write a script which asks the user to introduce the name of a file to read and prints on screen all the lines in the file.

```
vname = input('Introduce the name of the file', 's');
vfile  = fopen(vname,'rt');
if vfile == -1
    disp('Error when opening the file');
else
     while (feof(vfile) == 0 (
          vline = fgetl (vfile);
        fprintf('\n%s ', vline);
     end;
     fclose(vfile);
end;
```

In this case, when using fgetl, the strings will not contain the '\n' so we change of line ourselves in the fprintf.

Reading ASCII files: textscan

- Reading from a text file

```
[C, pos]= textscan (fi, 'control characters', N, 'param', value)
```

Cell array containing the results of the reading.

Optional. Indicates the position of the file in which textscan finished the reading.

File identifier. Previously obtained when opening the file

Similar to fprintf and scanf. The number of specifiers characters determines the number of cells in the array cell C

Number of times the control character sequence will be read from the file

Allow modifying default settings as the **characters used to delimit information**, etc.

The output of TEXTSCAN is a CELL ARRAY which itself contains arrays of the various data inputs.

Reading ASCII files: textscan

- Reading from a text file

```
[C, pos]= textscan (fi, 'control characters', N, 'param', value)
```

Cell array containing the results of the reading.

Optional. Indicates the position of the file in which textscan finished the reading.

File identifier. Previously obtaiined when opening the file

Similar to fprintf and scanf. The number of specifiers characters determines the number of cells in the array cell C

Number of times the control character sequence will be read from the file

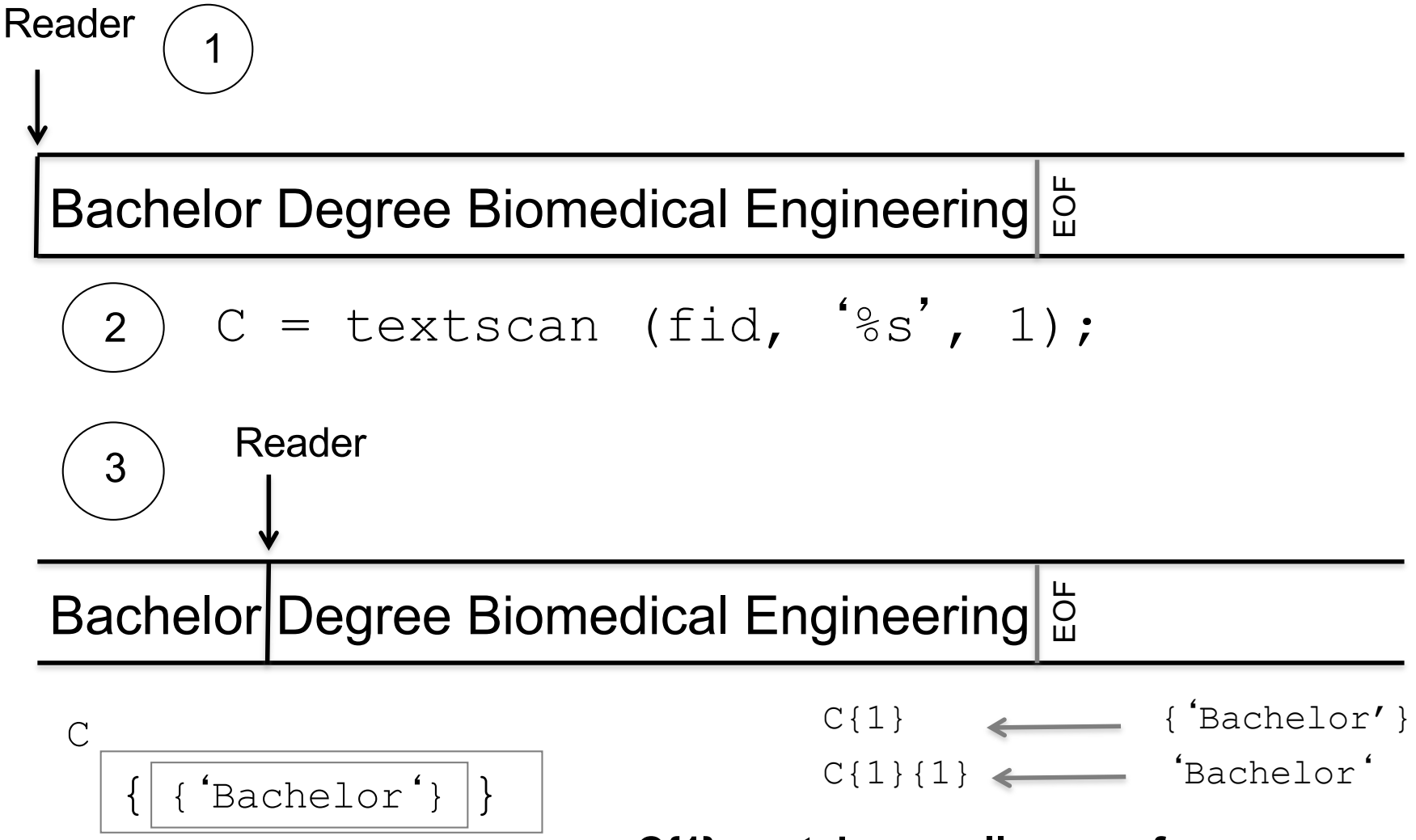
Allow modifying default settings as the **characters used to delimit information**, etc.

IMPORTANT:

When reading Strings TEXTSCAN returns **a cellarray whose cells are cellarrays that contain Strings or vectors of numbers**

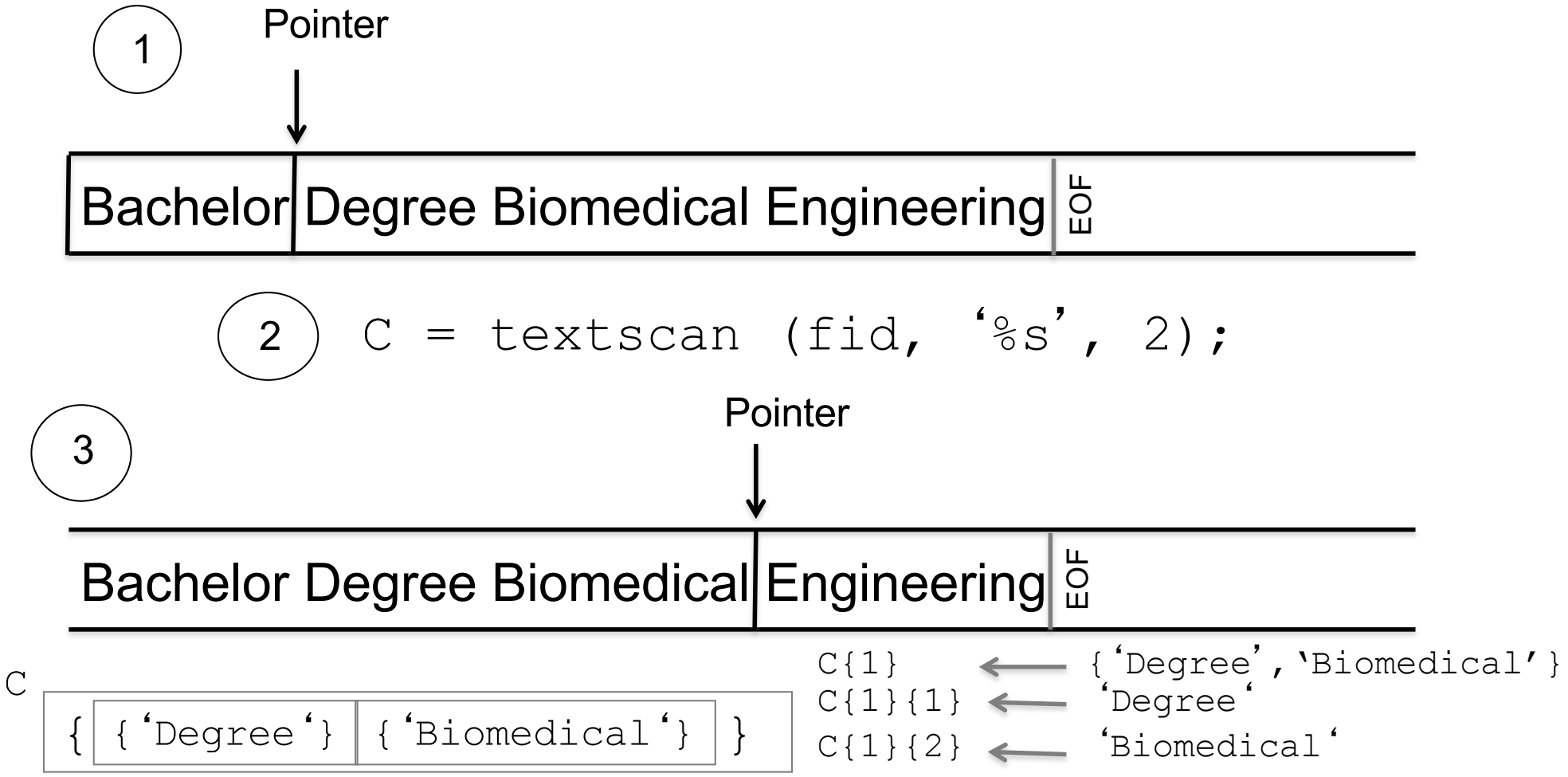
When reading numbers TEXTSCAN returns **a cellarray whose cells are vectors of numbers**

Reading ASCII files: textscan

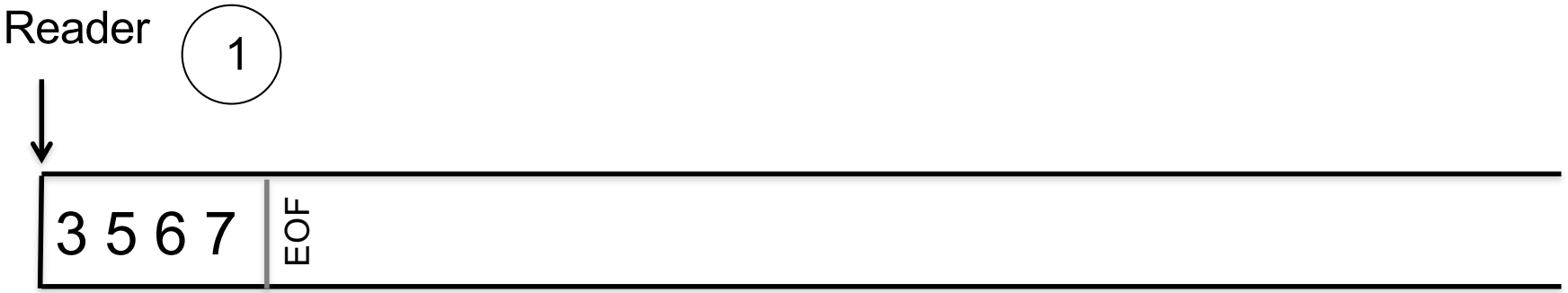


C{1} contains a cell array of Strings. If you want to retrieve the string you need to do C{1}{1}

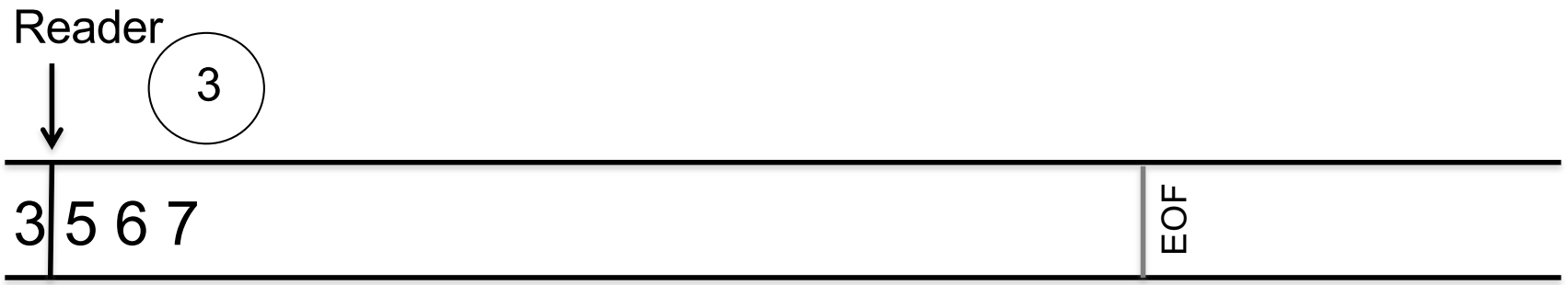
Reading ASCII files: textscan



Reading ASCII files: textscan



2 `C = textscan (fid, '%d', 1);`



C

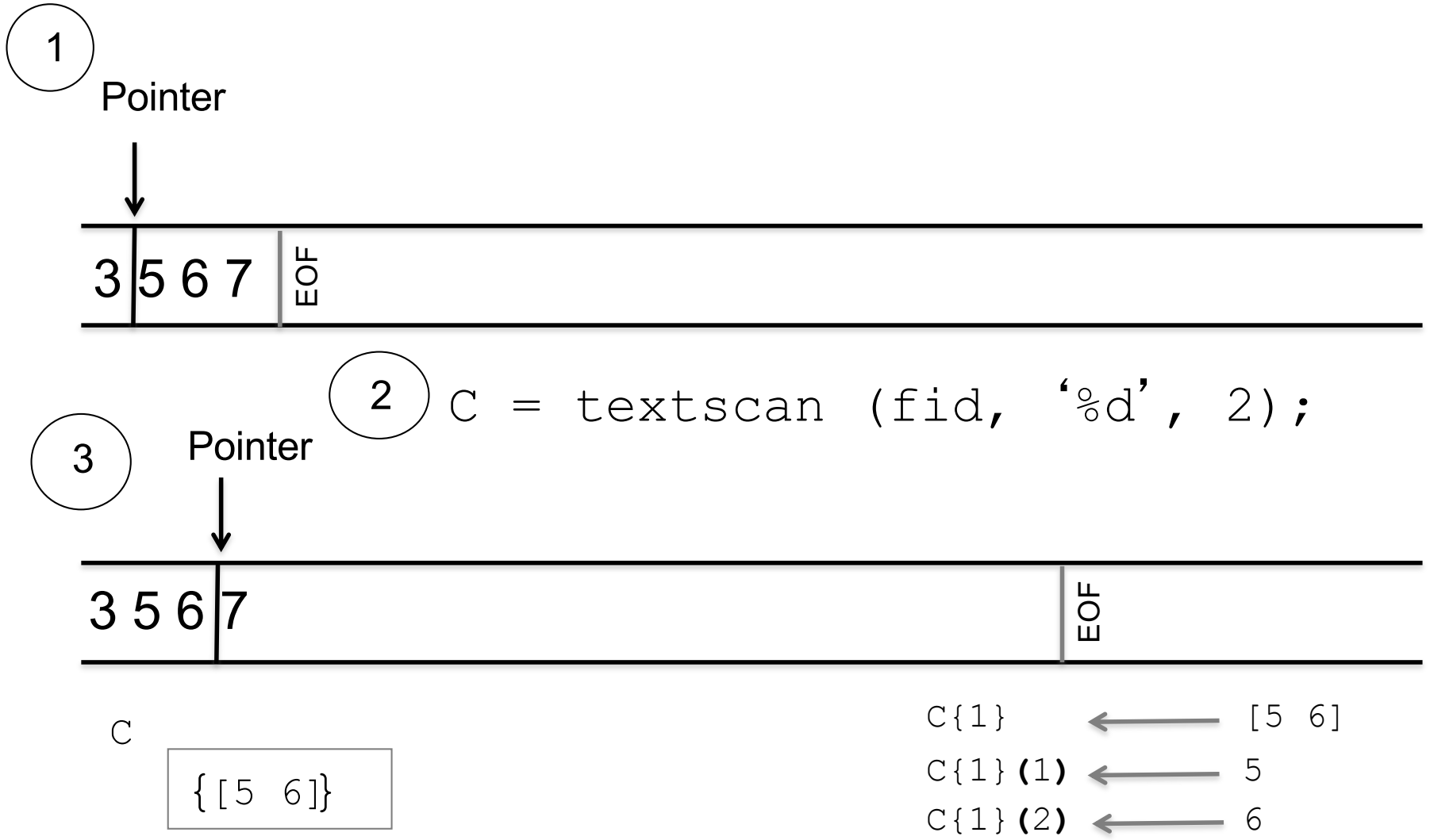
{ [3] }

C{1} ← [3]

C{1}(1) ← 3

C{1} contains a vector of numbers. If you want to retrieve the number you need to do C{1}(1)

Reading ASCII files: textscan



Reading ASCII files: textscan

- Examples:

```
C= textscan (fi, '%s', 1);
     Reads one string and places it in C{1}{1}.
C= textscan (fi, '%d', 1);
     Reads one number and places it in C{1}(1).
C= textscan (fi, '%s', 2);
     Reads two strings and place them in C{1}{1} and
     C{1}{2}
C= textscan (fi, '%s');
     Reads all the strings in the file and place them
     in C{1}{1}, C{1}{2}, C{1}{3}, C{1}{4}..
```

Other functions

- Other functions:

- varNum = str2num(varString) — Converts a string containing numbers into a number. Example:
  var = str2num('568')

- frewind(fi) — Sets the file position indicator to the beginning of a file.

Other functions

- Other functions:

- varNum = str2num(varString) Converts a string containing numbers into a number. Example:
  var = str2num('568')

- frewind(fi) Sets the file position indicator to the beginning of a file.

but we are never going to use this one!!!

Example

- Write a script which reads all the words of a text file called ‘sentence.txt’ and prints them on the screen. Use textscan

Example

- Write a script which reads all the words of a text file called ‘sentence.txt’ and prints them on the screen. Use textscan

```
vfile = fopen('sentence.txt','rt');
if vfile == -1
     disp(’Error when opening the file’);
else
     while (feof(vfile) == 0)
          cword = textscan(vfile,'%s',1);
          fprintf('\nThe word is: %s', cword{1}{1});
     end;
     fclose(vfile);
end;
```

Sumary: Reading ASCII files

- You can use three different commands to read from a text file:
  - fscanf => returns the data read in a vector or matrix

    ```
    Recommended for reading numbers, or when
    reading one single string at a time
    ```

  - textscan => returns the data read in a cell array

    ```
    Recommended for reading text files
    containing strings
    ```

  - fgets => returns a whole line (until \n) of text in a string

    ```
    Recommended when it is not necessary to
    process or split up the information in the
    line. For example: when copying files,
    counting lines, one single string per line,
    etc.
    ```