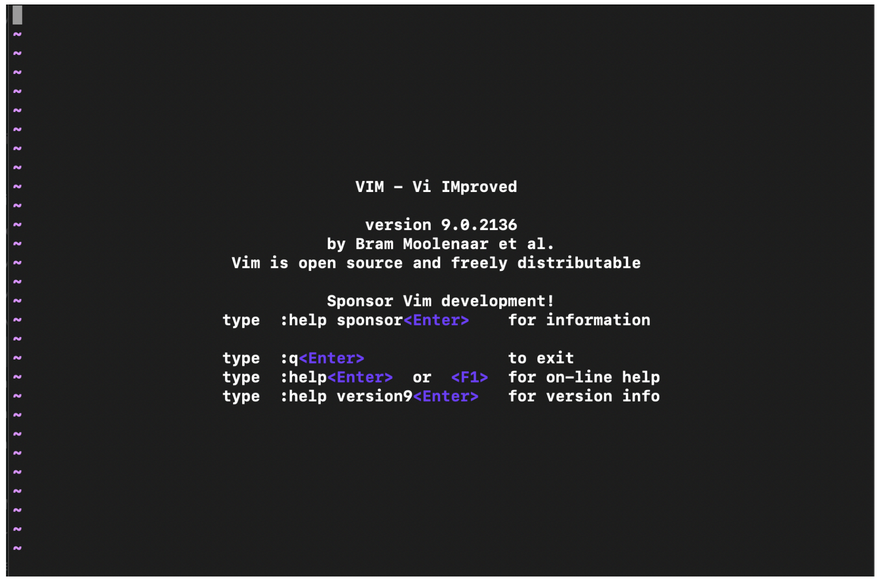
Adding Support of Persian/ Arabic Languages to vi.iv

CS 846 Course Project

Amir Seyhani, July 2024

vi.iv

- vi.iv is a bi-directional revision of Vi
- Currently, Vim is the standard, full-screen editor available on UNIX systems
- Vim is the improved version of Vi



```
                      VIM - Vi IMproved

                       version 9.0.2136
                   by Bram Moolenaar et al.
          Vim is open source and freely distributable

                  Sponsor Vim development!
type  :help sponsor<Enter>    for information

type  :q<Enter>               to exit
type  :help<Enter>  or  <F1>  for on-line help
type  :help version9<Enter>   for version info
```

Vim

```
sh-3.2$
```

History of Vi: ed

- ed was one of the first three key elements of the Unix (assembler, shell)
- Developed by Ken Thompson in 1969
- Text can be edited by using commands

```
a
ed is the standard Unix text
editor
This is line number two.
.
2i

.
,l
ed is the standard Unix text
editor$
$
This is line number two.$
w cs846.txt
62
3s/two/three/
,l
ed is the standard Unix text
editor$
$
This is line number three.$
w cs846.txt
64
q
```

History of Vi: ex

- Bill Joy improved the ed to be less demanding on the processor, 1978
- Providing a more user-friendly interface
- It is similar to Vi's command mode

```
:1
ed is the standard Unix text
editor
:3
This is line number three.
:1,3
ed is the standard Unix text
editor

This is line number three.
:a
Here is the new line!

:1,4
ed is the standard Unix text
editor

This is line number three.
Here is the new line!
:3d
Here is the new line!
:1,3
ed is the standard Unix text
editor

Here is the new line!
```

History of Vi

- Bill Joy added visual mode to ex
- Releasing it as Vi in 1979
- The name Vi is derived from the ex command for visual mode

Vim

- Vim: Vi Improved
- Released in 1991
- It used the original source code of the ed, not Vi
- Features:
    - syntax highlighting
    - mouse support
    - graphical versions
    - large amount of extension in the area of ex commands

vi.iv

- A need for a full complement of tools for bi-directional text processing
- There already exist two batch-oriented bi-directional formatters:
    - ditroff/ffortid
    - TEX/XET
- A terminal-independent, full-screen bi-directional editor was needed
- Originally developed to support Hebrew

vi.iv Goals

- Ability to work with bi-drectional files
- The extension into bi-directional be as orthogonal as possible
- Be language independent: works unchanged with any reasonable terminal for any right-to-left language
- Can be built as a slight modification to an existing implementation of Vi

Time Order vs Visual Order

- RL should be displayed from right to left and LR should be displayed from left to right
- Human being would prefer to enter all text in what is called time order

He said “שלום” to me. He said “سلام” to me.

Time Order vs Visual Order

- Streak: a maximal length string of text within a single line all of whose characters are in languages of the same direction
- There are three streaks
- It is the job of the displaying software to construct the visual ordering of a file from its time ordering

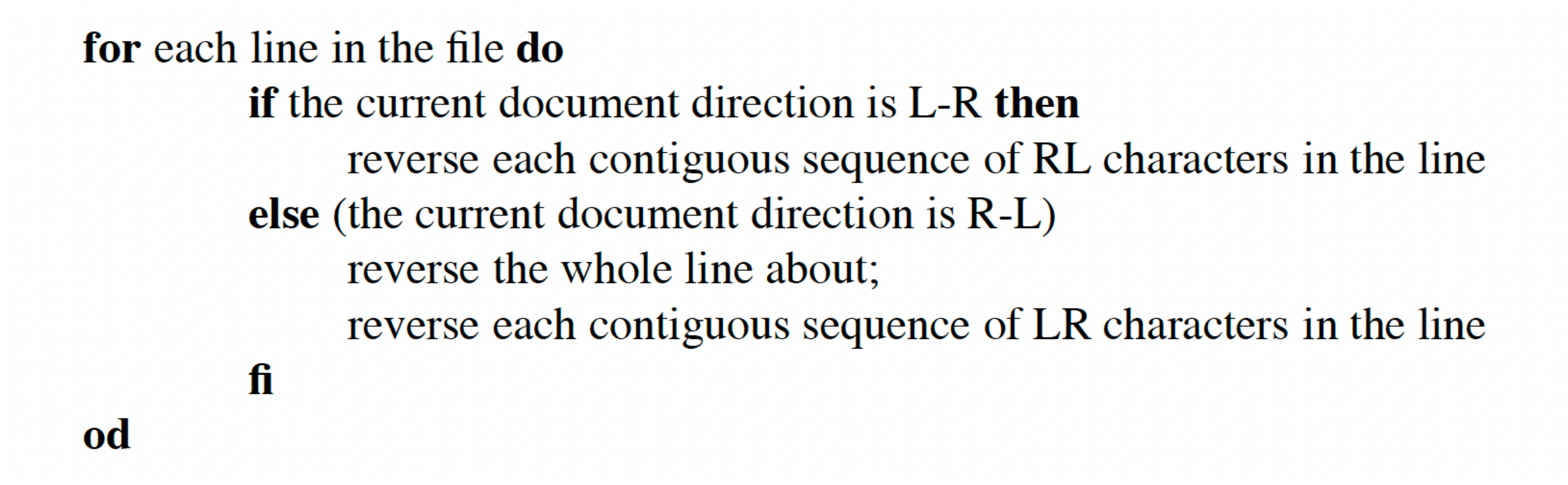
He said “

سلام

” to me.

Layout Algorithm

- The process of converting text from visual to time order is called **layout**



```
for each line in the file do
    if the current document direction is L-R then
        reverse each contiguous sequence of RL characters in the line
    else (the current document direction is R-L)
        reverse the whole line about;
        reverse each contiguous sequence of LR characters in the line
    fi
od
```

Layout Algorithm

- The process of converting text from visual to time order is called **layout**
- For lines longer than the physical line length, time-ordered line is folded into pieces that fit the physical line length
- Then, each piece is subjected to layout as if each were a line itself
- Pieces are interpreted in the same document direction as the original line

Layout Algorithm

- When layout should be performed?
  - As the text is entered
  - As the text is printed
- Drawback of first option: appearance is a function of the line length
  - Need to reconstruct the original input to calculate the new apperance

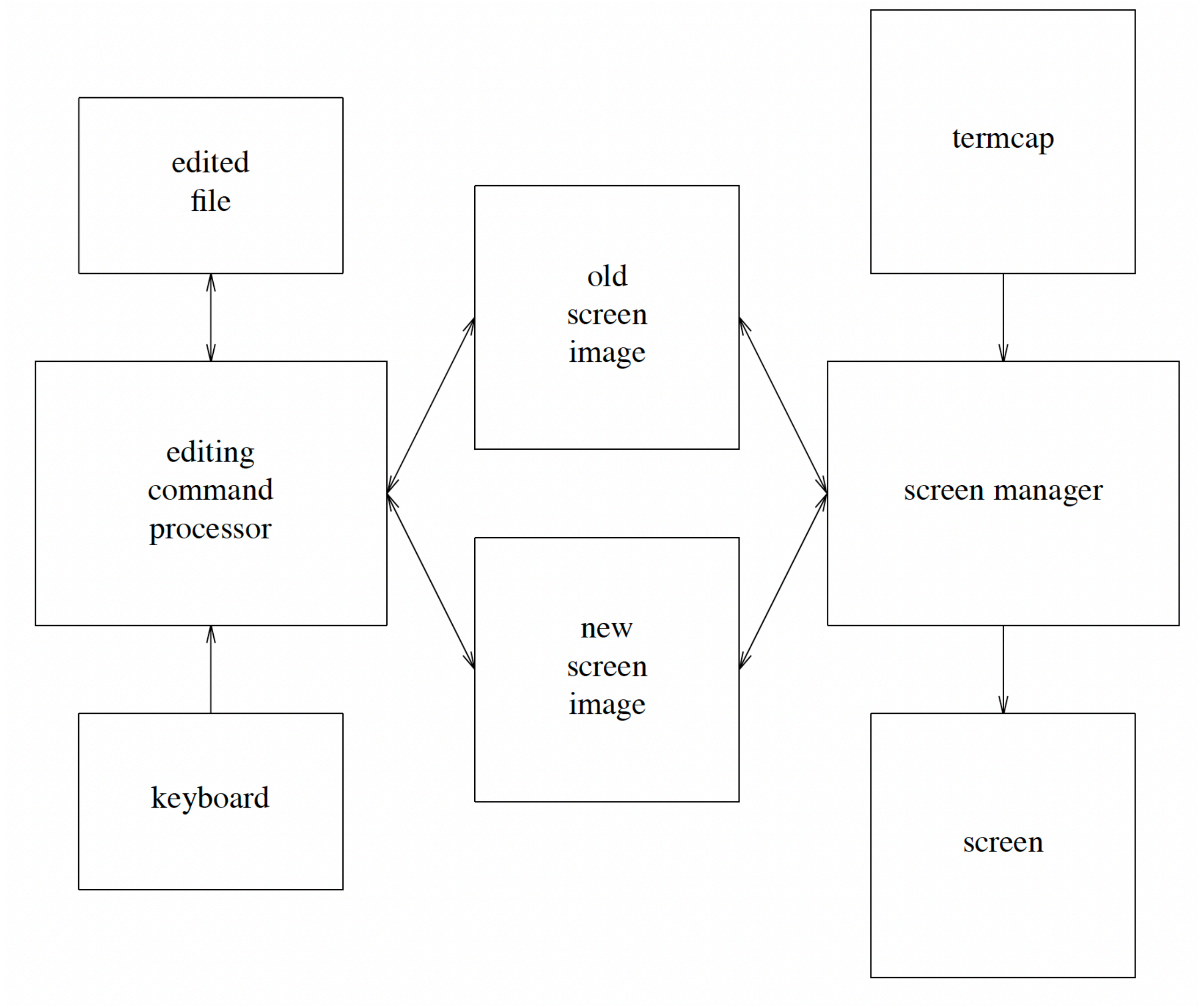
Layout while Printing

- Layout-while-printing and storing the files in the input order is more general
- Drawback: the time spent to lay the file out each time the file is printed
- The cost wasn’t considered burdensome back then!
- Layout-while-printing is more general and is not too much more expensive

Encoding Characters

- Need to distinguish between LR and RL letters
- Using the proper binary code for each letter
- In fact, at least Latin, Arabic, Farsi, and Hebrew have standard 7-bit codes
- Using the eight-bit to distinguish LR and RL letters
- The eighth-bit method of distinguishing alphabets is satisfactory when two languages with small alphabets are involved

Vi Structure



Project Goals

- Properly showing the connected letter forms of Persian/ Arabic
- Add support for standalone mode
- Add support for timeorder mode

Vi Code Challenges

- ~25k lines of code!
- Code was written in C
- Spaghetti due to the high orthogonality of vi commands!
- Communication is done using global variables!
- No proper documention, some comments exists

Layout Function in vi.iv

- 1. Break the line into pieces according to line width
- 2. Apply the layout algorithm to each line
- The function name is “changseclan”
- After each change to a line this function is called

Properly Showing Connected Letters

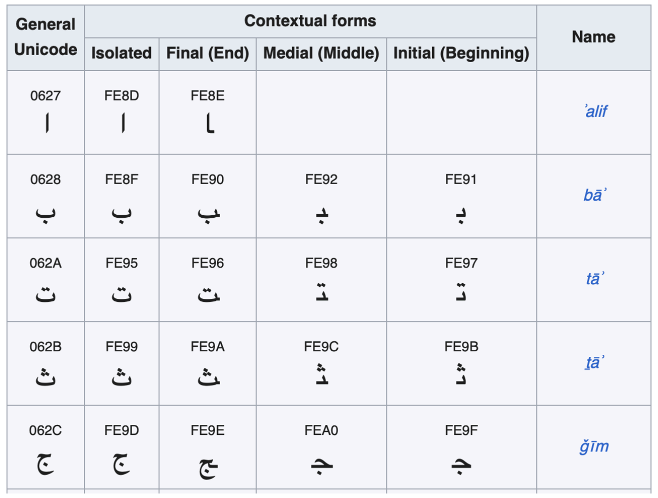
- Position identification
- Printing the proper form based on position

Non-Letter	Standalone	Non-Letter
Non-Letter	Initial	Letter
Letter	Medial	Letter
Letter	Final	Non-Letter

Properly Showing Connected Letters

- Position identification
- Printing the proper form based on position

Non-Letter	Standalone	Non-Letter
Non-Letter	Initial	Letter
Letter	Medial	Letter
Letter	Final	Non-Letter



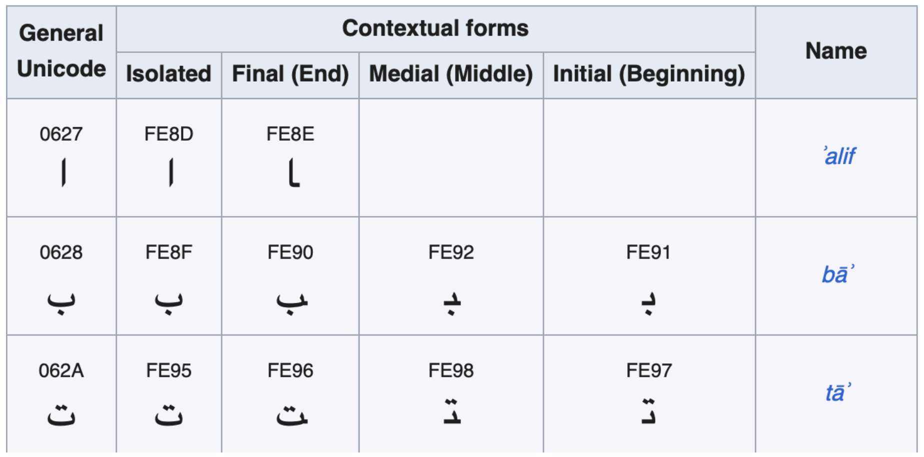
General Unicode	Contextual forms				Name
	Isolated	Final (End)	Medial (Middle)	Initial (Beginning)	
0627 ا	FE8D ﺍ	FE8E ﺎ			ʾalif
0628 ب	FE8F ﺏ	FE90 ﺐ	FE92 ﺒ	FE91 ﺑ	bāʾ
062A ت	FE95 ﺕ	FE96 ﺖ	FE98 ﺘ	FE97 ﺗ	tāʾ
062B ث	FE99 ﺙ	FE9A ﺚ	FE9C ﺜ	FE9B ﺛ	ṯāʾ
062C ج	FE9D ﺝ	FE9E ﺞ	FEA0 ﺠ	FE9F ﺟ	ǧīm

Layout Function in vi.iv

- 1. Break the line into pieces according to line width
- 2. Apply the layout algorithm to each line
- The function name was “changseclan”
- After each line this function is called
- Alternative option: format the letter inside put char functions
- We need to look at the previous and next characters

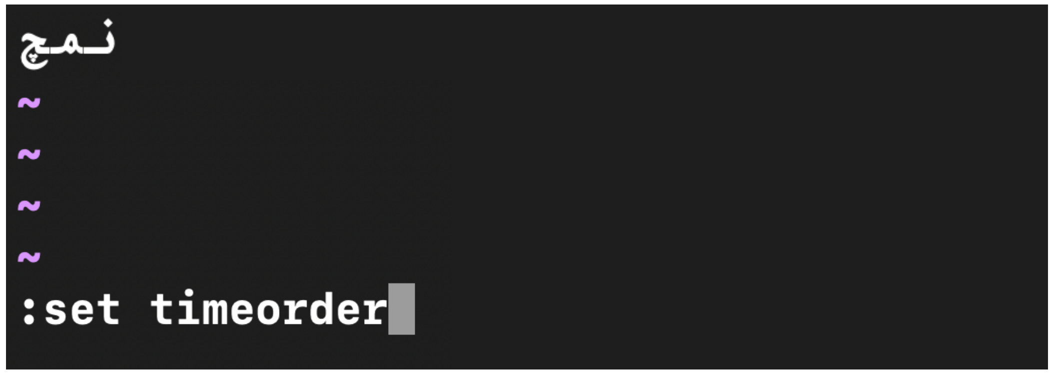
ASCII to Unicode

- Use the contextual form unicode table for that language
- Use the proper unicode based on the position

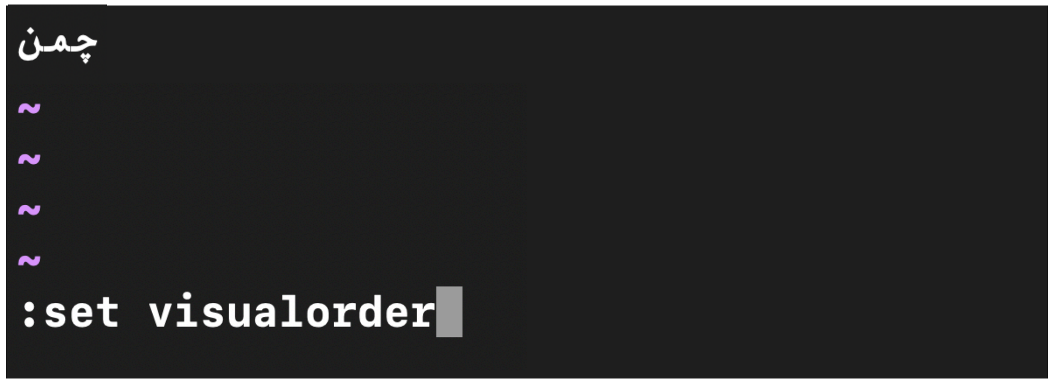


General Unicode	Contextual forms				Name
	Isolated	Final (End)	Medial (Middle)	Initial (Beginning)	
0627 ا	FE8D ﺍ	FE8E ﺎ			*ʾalif*
0628 ب	FE8F ﺏ	FE90 ﺐ	FE92 ﺒ	FE91 ﺑ	*bāʾ*
062A ت	FE95 ﺕ	FE96 ﺖ	FE98 ﺘ	FE97 ﺗ	*tāʾ*

Supporting Timeorder Mode

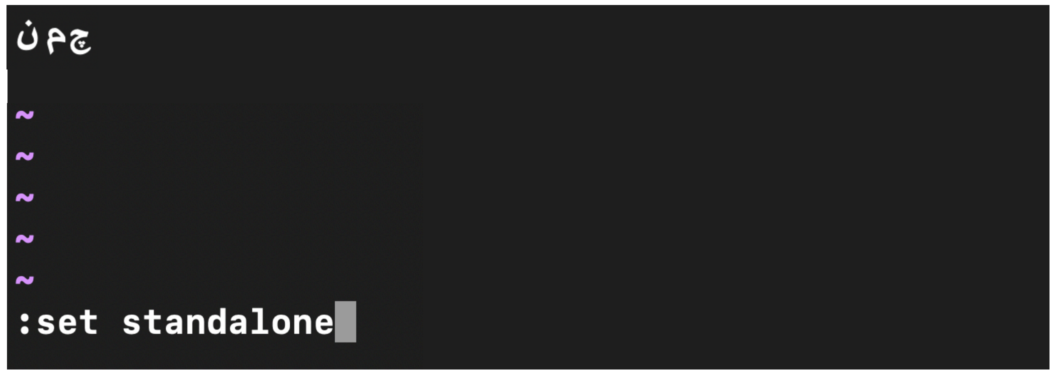


```
نـمـچ
~
~
~
~
:set timeorder
```

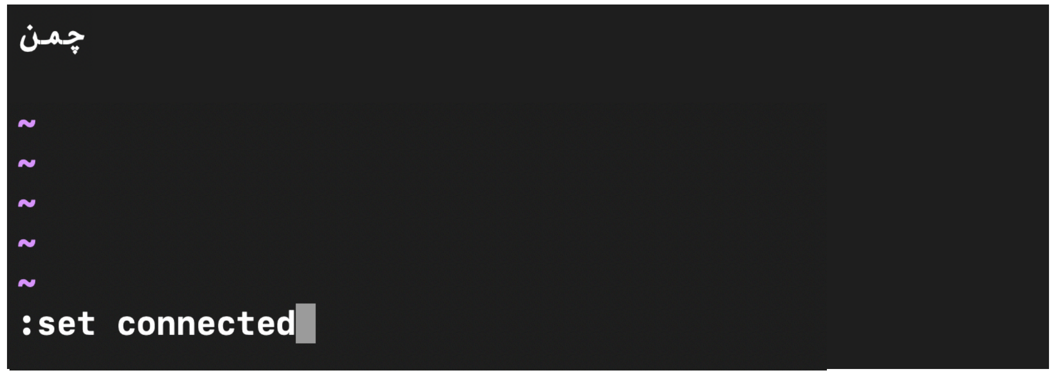


```
چمن
~
~
~
~
:set visualorder
```

Supporting Standalone Mode

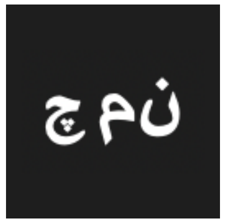
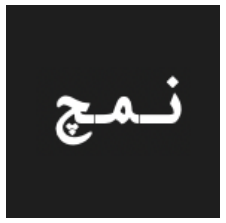
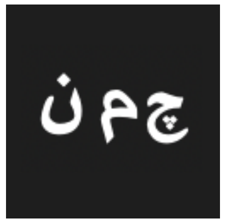
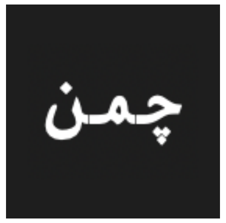


```
چ م ن

~
~
~
~
~
:set standalone
```



```
چمن

~
~
~
~
~
:set connected
```

Orthogonal Options

	Standalone	Connected
Time Order		
Visual Order		

Summary of Code Changes/Additions

- changseclan function
- Change char to wchar_t
- format_char_with_pos
- lookup_unicode
- Code changes required for adding the timeorder, visualorder, standalone, and connected options
- If conditions to skip RL functions in timeorder mode
- If conditions to skip connected formatting functions in standalone mode