

Operating Systems Practicum

Lecture 2 – The **vi** Text Editor

What is **vi**?

- **vi** is the most widely used full-screen text editor for UNIX and Linux system.
- **vi** is short for visual extension (of the line-oriented editor **ex**) developed by Bill Joy, co-founder of Sun Microsystems.
- The UNIX/Linux editor succession is
ed → **ex** → **vi**
- The main alternate editor, EMACS, was developed at MIT.

Before Starting **vi**

- Before starting to edit using **vi**, we need to let **vi** know what our terminal type is.
- On many systems, it can be changed by typing:
`set TERM=vt100 or vt300`
`export TERM`
- If you don't know the initial terminal type, you can find out by typing
`echo $TERM`

On panther

- On panther, this is necessary because users are normally set with a terminal type that is compatible with **vi**.
`[SIEGFRIE@panther ~]$ echo $TERM`
`xterm`

User Profile

- When a user logs in, there is a profile that is used to initialize the terminal session:

```
[SIEGFRIE@panther ~]$ more .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
unset USERNAME
[SIEGFRIE@panther ~]$
```

Starting vi

- To invoke the vi editor, type
vi <filename>
- If the file is new, you will get a nearly-blank screen.
- You are initially in command mode. By typing, **i** or **a**, you can start to enter text.

Entering Input Mode

a	Append after current <u>cursor position</u> .
A	Append after current <u>line</u> .
i	<u>Insert</u> before current cursor position.
I	<u>Insert</u> before current line.
o	Open line <u>below</u> current line.
O	Open line <u>above</u> current.

ex Commands in **vi**

- **ex** commands are always available for use in vi command mode if you first type ":"
- Examples
 - :**s/x/y/** will substitute an **y** for **x**.
 - :**., +1j** will join the current line with the following line.
- Finish the command by pressing ↵ just as you would in ex.

Working in Command Mode

- The command never appears on the screen (except for ex commands).

```
This is new
text enter using "d"
~
~
~
~
~
~
~
~
~
~
:s/"a"/"d"/          2,1      All
```

Leaving Input Mode

- To get back to Command Mode, press the "Escape" key.
- When in doubt about which mode you are in , press "Escape."
- If you press "Escape" in Command Mode, **vi** will beep.

Exiting Input Mode

Enter "a" in Command Mode

The screenshot shows a terminal window with the following content:

```
Hello, world  
My name is Sy.  
~  
~  
  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

In the bottom right corner of the window, the text "2, 14" and "All" is displayed. An arrow from the text "Enter 'a' in Command Mode" points to the first tilde character (~). Another arrow from the text "Press ESC in Input Mode" points to the bottom right corner of the window.

Press **ESC** in Input Mode

Exiting vi

- To write and/or quit your vi session:

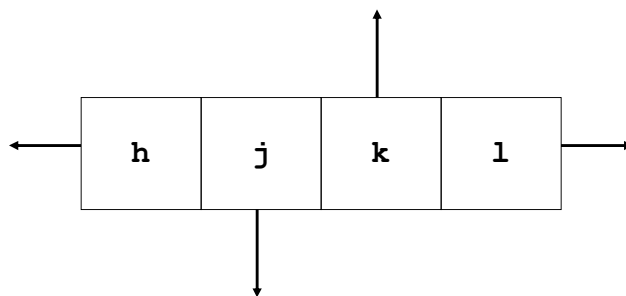
:w	Writes the files
:wq	Writes and quits
:x	Same as :wq
ZZ	Same as :wq
:q	Quits without writing (if unmodified)
:q!	Quits without writing

vi tells you the Number of Lines

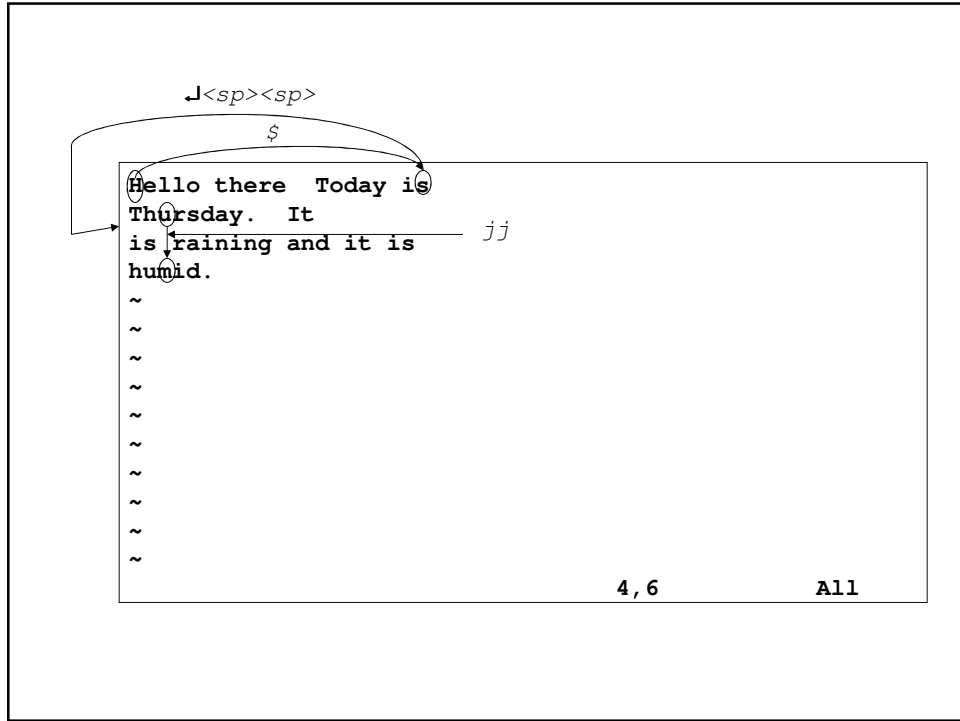
```
#include      <stdio.h>

int    main(void)
{
    printf("Hello, world\n");
    return(0);
}
~
~
~
~
~
~
~
~
~
~
~
"hello.c" 7L, 78C                6, 2-9      All
```

Moving Around the Screen



0	Beginning of current line
\$	End of current line
↵	Beginning of next line
+	Beginning of next line
-	Beginning of previous line



Other Move Commands

w	move to the next word or punctuation
e	move to the end of the next word of punctuation
b	move to the beginning of the previous word
)	move to the beginning of the next sentence
(move to the beginning of the current sentence
}	move to the beginning of the next paragraph
{	move to the beginning of the current paragraph

vi Words and Paragraphs

- **vi** sentences end with ".", "?" or "!"
- **vi** paragraphs begin after a blank line.

Diagram illustrating vi word and paragraph boundaries:

```
Hello world again.This is a mysterious  
file.  
  
Help  
~  
~
```

The diagram shows the text "Hello world again.This is a mysterious file." with arrows indicating word boundaries (w) and paragraph boundaries (}b). A blank line follows, and the word "Help" is shown with a paragraph boundary arrow (}w) pointing to it.

Moving Around Faster

^f	Writes the files
^d	Move forward 1/2 screen
^b	Move backward one full screen
^u	Move backward 1/2 screen
G	Moving to the end of the file

A number in front changes the command

3^f – move ahead 3 screens

99G -go to line 99

Modifying Text

- **r** *char* – replace the current character with *char*.
- **R** *string* <esc> - Overwrite text with this string.
- **J** – joins current and next line into one line.
- **~** - Switches upper and lower case.

Modifying Text – An Example

Now **i**s the time

for all good men to

- Typing **R**t is<Esc> changes it to

Now it **i**se time

for all good men to

- and **J** changes it to:

Now it ise time **f**or all good men to

- **~****~****~** changes it to

Now it ise time **FOR** all good men to

Deleting Text

x	deletes a single character
<i>num</i> x	delete <i>num</i> characters
dw	deletes the rest of the current word
<i>num</i> dw	deletes <i>num</i> words from the current position
d\$	deletes the rest of the line
d)	deletes to the beginning of the next sentence
dd	deletes current line
dnumd	deletes <i>num</i> lines from the current line

Deleting Text – An Example

Date: 12/31/09

Room: 110HHE

Time: 10:00:00AM

- 5x:

Date: /09

- d) changes it to

Date:

After d2d

Time: 10:00:00AM

- Typing u will undo the most recent change.

Moving Text

- Lines last deleted (or yanked) are placed in a buffer. You can "put" it anywhere in a file.

p – put it to the right or below the current position.

P – put it to the left or above the current position.

Moving Text – An Example

Line 1		}	Line 3
Line 2			Line 4
Line 3	After d2djp →		Line 1
Line 4			Line 2

Searching For Text

- `/string` – search forward for *string*
- `?string` – search backward for *string*

Copying and Moving Text

- To copy text, use **yy** (to yank a line) or **yw** (to yank a word) or **y\$** (to yank until the end of the line) or **y)** (to yank until the next sentence) followed by **p** or **P**.
- To move text, use **dd**, **dw**, **d\$** or **d)** followed by **p** or **P**.

Using **ex** Commands in **vi**

- All **ex** commands in **vi** are preceded by a colon:

:wq

:q!

Some Useful **ex** Commands

:address s/oldpattern/newpattern/ - replace the old pattern in the text with the new pattern on these lines

:address d - deletes these lines

:g/opattern/s//npattern - globally searches for *opattern* and replaces it with *npattern*.

:r file - reads in *file*

!: cmd - perform UNIX shell command *cmd*.

Addresses in **ex** Commands

- Address in these commands can be:
 - 1, 5 – lines 1 through 5
 - ., 30 – current line through line 30
 - 30, \$ - line 30 through the end
 - ., +5 – current line through 5 lines down
 - :-3, +1 – 3 lines above until 1 line down

Miscellany and Metacharacters

- ^1 – refreshes screen
- Metacharacters
 - ^ - beginning of line
 - \$ - end of line
 - . – matches any single character
 - * - matches preceding character any number of times
 - [*string*] – matches any character in *string*
 - [*^string*] – matches any character ***NOT*** in *string*.

Metacharacters – An Example

- `/^xyz` – search for **xyz** at beginning
- `/line.$` - search for this at the end of the line
- `/^$` - search for empty line
- `s/x.*$/Hello` – finds **x** followed by any character any number of times at end of line and replaces it with **Hello**.
- `:1, $s/[0-9]/-/g`
- `:1, $s/[^a-zA-Z]/0/g`
- `:s/United .*ica/USA/`