Editing with *vim*

- Two tool environments exist for developing embedded software
  - IDE: (Integrated Development Environments), AVRStudio, Codewarrior, Elicpse
  - Command line: gcc, gas, ld, avr-objcopy, avr-objconv, avrdude
  - Both have their advantages depending upon the programming environment.
- We will be using command line tools for several reasons
  - You can understand the details (nothing is hidden)
  - The tools are free (freedom and beer)
  - You own the tools on your laptop
  - You get valuable exposure to Linux and programming editors

Editing with *vim*

- *vi* is pronounced "vee eye".
- When I say *vi* I mean *vim*.
- *vim* is available on all *nix systems
- *vim* is fast, stays out of your way
  - no menus
  - short commands
  - no *mouseing* around
- High beginning threshold (12 commands) to become productive
- Many *vim* help sites on the web such as `http://www.rru.com/~meo/useful/vi/vi.intro.html`
- A good, more advanced book on *vim*:
  "Practical Vim - Edit Text at the Speed of Thought" by Drew Neil

Editing with *vim*

- You will do more programming than you think.
- Even HW designers mostly use Verilog, VHDL, shell scripts, Python.
- Learn one editor really well. Use it for all your editing tasks.
- The editor should:
  - Be configurable, extensible and programmable
  - Have language sensitive syntax highlighting
  - Have auto completion, and indention
  - Have seamless compiler support
  - Be available everywhere
- An editor needs to stay out of the way of thinking. It becomes an extension of your hand that needs no cognitive attention. You think, and code appears on the screen.
- If you can't edit code well, the structure begins to suffer that causes readability problems followed by bugs. Comments go stale and become misleading.

Editing with *vim*

Entry and Exit

`vim`	opens *vim*
`vim file_name`	opens or creates `file_name` for editing
`view file_name`	opens *vim* in read only mode on texttfile_name
`:q`	quit with query if changes were made
`:q!`	quit, don't save changes
`:x`	save and exit
`<shift>ZZ`	save and exit
`:w`	save the present file
`:w new_file`	save present file as `new_name`
`:wq`	write and quit (like `<shift>ZZ`)
`:r file2`	read `file2` into current file at cursor
`:e new_file`	edit `new_name` discarding current file

Editing with *vim*

Moving around in the file

`h`	move one character left
`l`	move one character right
`k`	move one character up
`j`	move one character down
`<ctrl>u`	move up one half page
`<ctrl>d`	move down one half page
`w`	move forward 1 word (cursor at first character)
`W`	move forward 1 word like "w" but ignore punctuation)
`b`	move backwards 1 word (cursor at first character)
`B`	move backwards 1 word like "b" but ignore punctuation)
`e`	move forwards 1 word (cursor at last character)
`E`	move forwards 1 word like "e" but ignore punctuation)

Editing with *vim*

Moving around in the file

`$`	move to end of the line
`0`	move to beginning of the line
`gg`	move to top of file, column 1 (HOME)
`G`	move to bottom of of file, column 1 (HOME)
`:n`	move to line "n" of file
`nG`	move to line "n" of file
`<ctrl>G`	tell me what file I'm in)

Editing with *vim*

Adding or Replacing Text

`i`	insert text in front of cursor
`a`	append test after cursor
`o`	open a new line below the current one
`O`	open a new line above the current one
`r`	replace the character under the cursor and exit input mode
`R`	replace characters continuously
`<esc>`	exit the input mode

Editing with *vim*

Editing commands

`yy`	yank current line
`p`	paste a line previously yanked
`8yy`	yank 8 lines
`x`	delect character under cursor
`cw`	change the word beginning at the current cursor position
`dw`	delete the word beginning at the current cursor position
`dd`	delete current line
`10dd`	delete ten lines
`J`	Join current line and the one below
`u`	undo the last action
`<ctrl>r`	redo the last action (undoes an undo)
`.`	repeat the last insert or append type command *powerful!*

Editing with *vim*

Search and Replace

`/pattern`	search forwards for pattern
`?pattern`	search backwards for pattern
`n`	repeat the last search
`:%s/old/new/g`	find old and replace with new everywhere
`:%s/old/new/gc`	find old and replace with new pending confirmation
`%`	show matching { }, ( ), [ ]
`*`	find next occurence of word under cursor
`#`	find previous occurence of word under cursor

Editing with *vim*

Shell and Make

`:make`	compile file as specified in local Makefile
`:make program`	compile and program target as specified in local Makefil
`:make clean`	clean up compilation as specified in local Makefile
`:! <command>`	execute shell command without leaving vi
`:! ls`	list files in current directory,
`:! date`	print date and time

Editing with *vim*

Multiple Screens

`:sp` OR `<ctrl>wn`	split screen with existing file
`:sp new_file`	open split screen with new_file displayed
`:vsp` OR `<ctrl>wv`	vertically split screen with existing file
`:vsp new_file`	open vertically split screen with new_file displayed
`<ctrl>ww`	moves cursor between split screens