

# Introduction to Linux

## Part 2a: additional items and editors

Anita Orendt and Wim Cardoen  
Center for High Performance Computing

# More on Environment

- Environment variable – variable with a name and associated value; used in shells, scripts
- Use `env` or `printenv` command to see environment variables
- Some important variables
  - `$USER`
  - `$PATH` – paths to search for commands
  - `$LD_LIBRARY_PATH` – paths to search for libraries when linking a program (more on that later)

# More about Processes

- Process := Running Linux program  
Each process has a PID (Process ID)

ps :: Report snapshot of the current processes  
ps [options]

ps x      Display ALL the processes to `whoami`

ps ax     Display ALL processes

ps aux    Display ALL processes (more detailed)

ps auxw   Display ALL processes (more detailed & unlimited width)

ps -eFwww

# Killing processes

- kill PID
- killall *processname*
- kill -9 PID

# Other Job Controls

- Ctrl+C (^C) terminate the currently running process
- Ctrl-Z (^Z) suspends the currently running process
- & runs the job in the background
- Jobs: lists all jobs, with their number
- bg %n: puts current or specified job (%n) in the background and
- fg %n: bring suspended program back to the foreground, e.g., so that it occupies the shell until done

# Monitoring processes/usage

- uptime
- free
- top
- atop
- htop
- sar

# Moving files to/from CHPC

- Windows – there are graphical tools such as WinSCP
- Mac, Windows, cloud options – cyberduck another graphical tool
- Linux
  - scp command (secure shell copy) – to copy files between linux systems
  - wget – to download from web with URL
  - curl
- For larger data sets – look into the Data Transfer Nodes (DTNs) and transfer tools such as globus

# Editors

There are many choices – a few are

- Nano
- Vi
- Emacs



# Nano editor

- Start with command  
nano

```
^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

# vi / vim graphical cheat sheet

**Esc**  
normal mode

~ toggle case	! external filter	@. play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	( begin sentence	) end sentence	"soft" bol down	+ next line
\ goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto <sup>3</sup> format
Q ex mode	W next word	E end word	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank <sup>1,3</sup>	u undo	i insert mode	o open below	p paste <sup>1</sup> after	[ misc	]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	.	ex cmd line	'' reg. <sup>1</sup> spec	bol/ goto col
a append	s subst char	d delete <sup>1,3</sup>	f find char	g extra <sup>6</sup> cmds	h ←	j ↓	k ↑	l →	.	repeat t/T/f/F	' goto mk. bol	\ not used!
Z quit <sup>4</sup>	X back-space	C change to eol	V visual lines	B prev word	N prev (find)	M screen mid'l	< un- <sup>3</sup> indent	> indent <sup>3</sup>	?	find (rev.)		
Z extra <sup>5</sup> cmds	X delete char	c change <sup>1,3</sup>	v visual mode	b prev word	n next (find)	m set mark	,	reverse t/T/f/F	.	repeat cmd	/	find

- motion** moves the cursor, or defines the range for an operator
- command** direct action command, if red, it enters insert mode
- operator** requires a motion afterwards, operates between cursor & destination
- extra** special functions, requires extra input

**q.** commands with a dot need a char argument afterwards  
 bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: `quux({foo, bar, baz});`  
 WORDs: `quux(foo, bar, baz);`

**Main command line commands ('ex'):**  
 :w (save), :q (quit), :q! (quit w/o saving)  
 :e f (open file f),  
 :%s/x/y/g (replace 'x' by 'y' filewide),  
 :h (help in vim), :new (new file in vim),

**Other important comands:**  
 CTRL-R: redo (vim),  
 CTRL-F/-B: page up/down,  
 CTRL-E/-Y: scroll line up/down,  
 CTRL-V: block-visual mode (vim only)

**Visual mode:**  
 Move around and type operator to act on selected region (vim only)

- Notes:**
- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,\*) (e.g.: "ay\$ to copy rest of line to reg 'a')
  - (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
  - (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
  - (4) ZZ to save & quit, ZQ to quit w/o saving
  - (5) zt: scroll cursor to top, zb: bottom, zz: center
  - (6) gg: top of file (vim only), gf: open file under cursor (vim only)

# Vi editor

- Editor provided with the OS
  - There are other choices such as nano (easy) and emacs (hard)
  - It is better to edit in a linux shell than moving file to your laptop for editing
- To start – command is **vi** (usage: `vi filename`)
  - If filename exists, vi will open it up in the editor
  - If filename does not exist, vi will create it and put you in insert mode
- Use arrow keys to move cursor to location
- Two modes – command, input
  - in command mode – the characters typed are interpreted as commands
    - There is also an external command mode – access with ‘:’
  - In insert mode – the characters typed are added to the file
    - The esc key exists the insert mode
- Many commands, we are just mentioning some basic ones to get you started

# Exiting the vi editor

- ❑ **ZZ** – will exit and save as same filename
- ❑ **:q** – quit; will let you know if you have unsaved changes
- ❑ **:q!** – quit discarding changes
- ❑ **:wq** – write and quit; same as doing **:w** followed by **:q**
  - ❑ saves as same filename
- ❑ To change name add new name after the **:w** or **:wq**
- ❑ Note – if you **ctrl Z** to suspend – vi saves a swap filename to save current status of the edits. File is named `.filename.swp`. Next time you try to edit this same file it will let you know the swp file exists and asks you how to proceed. If you do not want to keep changes, delete this file

# Moving around a file

- ❑ **down arrow** or **j** moves down one line
- ❑ **up arrow** or **k** moves up one line
- ❑ **right arrow** or **l** moves right by one character
- ❑ **left arrow** or **h** moves left by one character
- ❑ **0** or **^** moves to start of current line
- ❑ **\$** moves to end of current line
- ❑ **w** moves to beginning of next word
- ❑ **b** moves to beginning of previous word
- ❑ **:0** or **:1** or **1G** moves to start of first line
- ❑ **:n** or **nG** moves to start of nth line
- ❑ **:\$** or **G** moves to start of last line
- ❑ **cntl-f (^f)** moves forward one screen (cntl-d moves forward ½ screen)
- ❑ **cntrl-b (^b)** moves back one screen (cntrl-u moves back ½ screen)

# Inserting or Adding Text

- Remember – **esc** to exit to command mode
- i** – insert at position of cursor
- I** – insert at beginning of line
- a** – append after position of cursor
- A** – append at end of line
- o** – new line below current line
- O** – new line above current line

# Changing and Deleting Text

- ❑ **r** – replace single character
- ❑ **R** – replace characters, starting at cursor position, until **esc** hit
  
- ❑ **x** – delete character that cursor is on (**nx** for n characters starting with one cursor is on)
- ❑ **dd** – delete current line (can do n lines with **ndd**)
- ❑ **D** – delete from cursor to end of line
- ❑ **dw** – delete word

# Cutting and Pasting Text

- ❑ **Y** – “yanks” current line into buffer (can use **nY** for n lines, current plus following lines)
- ❑ **p** – paste lines in buffer after current line ( **P** – paste lines before current line)

## Other Useful Commands

- ❑ **/pattern** – searches for next occurrence of pattern, then n goes to next occurrence; can also use **/** and **?** To move to previous and next occurrence
- ❑ **?pattern** – searches for previous occurrence of pattern
- ❑ **u** – to undo results of last command (use multiple times to revert back through multiple commands)



# External commands

- ❑ already mentioned some of these in the moving around and exiting vi
- ❑ **:s/old\_text/new\_text/** – replaces next occurrence of old\_text in current line
- ❑ **:1,\$s/old\_text/new\_text/g** – replaces all occurrences of old\_text in file

# Exercise – Practice these commands!

- Try the “vimtutor” command
- Try writing some source code (C, python, fortran, etc)
- Try writing your grocery list