

# Introdução ao GNU/Linux

Curso de introdução ao GNU/Linux  
Abordagem Prática para HPC



# Aula 03

- Programa vim
- Execução de Programas
  - Foreground e Background
  - Execução sequencial
- Programas úteis
  - Gerenciamento de processos
  - Manipulação de diretórios
  - Manipulação de arquivos
  - Comandos Diversos

# Editor de texto vim

- Vim vi Improvised ( vim – 1991 / vi – 1976 )
- Editor de rico em recursos
- Simples de utilizar ( comandos padronizados )
- Disponível em todos os sistemas operacionais
- Indicado para usuários, programadores e administradores de sistemas
- Pode ser programado com funções ou expansões

Insert Mode

Command Mode

Visual Mode

<Esc>

i

a

I

A

v

V

<Ctrl>-v

<Esc>

# Editor de Texto vim

- Comando para inserção
  - i Inserção antes do caractere atual
  - a Inserção depois do caractere atual
  - I Inserção no início da linha
  - A Inserção no fim da linha
- Comando para visual
  - v visual por caractere
  - V visual por linha
  - Ctrl+v Bloco
- [http://blog.interlinked.org/tutorials/vim\\_tutorial.html](http://blog.interlinked.org/tutorials/vim_tutorial.html)

# Comandos Vim

- “:” Acessar comandos
- Ctrl+C Cancela comando
- Ctrl+D completa comando
- Cima e Baixo histórico de comandos
- :help <expressão>
- :help quit
- :help copy

# Comandos Vim

- `:[range]co[py] {address}`
- `:help :s`
- `:help /s`
- `:help`
- `[tópico] Ctrl+]`
- Voltar com `Ctrl+T`
- `vimtutor`
- `man vim`

Referencia <http://vimdoc.sourceforge.net/html/doc/>

# Comandos Vim

- “d” apaga e copia – “dd” apaga linha
- “c” modifica – “cc” modifica linha
- “y” copia – “yy” copia linha
- “p” e “P” cola
- Repetindo n+ação
- ESC comando ESC
- “.” repetir último comando



# Comandos Vim

- Movimentos
  - w próxima palavra
  - b palavra anterior
  - e final da palavra
  - 0 início da linha
  - ^ início da frase
  - \$ final da linha
- Comando + Movimento
- Quantidade + Comando + Movimento

# 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			

<b>motion</b>	moves the cursor, or defines the range for an operator
<b>command</b>	direct action command, if <b>red</b> , it enters insert mode
<b>operator</b>	requires a motion afterwards, operates between cursor & destination
<b>extra</b>	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 commands:

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)

# Execução de Programas

- No GNU/Linux podemos executar programas:
  - Foreground execução interativa dependentes da sessão.
  - Background execução de tarefas independentes da sessão.
- Execução background
  - comando &
  - comando Ctrl+Z → bg
  - jobs comando para listar as execuções em background
  - fg voltar para execução foreground

# Execução sequencial

- Podemos executar comandos um depois do outro com <enter>
  - ls
  - pwd
  - cd GNULINUX
- Podemos colocar todos em uma linha separando com “;”
  - ls ; pwd ; cd GNULINUX ( sem relação entre os comandos)
- Podemos relacionar a execução dos comandos com o operador “AND” - “&&”
  - ls && pwd && cd GNULINUX ( relação entre os comandos)

# Execução em paralelo

- Para executar comandos em paralelo, podemos colocar eles em background
  - `ls &`
  - `pwd &`
  - `find &`
- Execução de programas mesmo sem sessão
  - `nohup programa > saida.txt 2>&1 &`

# Gerenciamento de Processo

- Informações de processos
  - ps
  - pidof
  - pstree
- Matando um processo
  - Ctrl+C
  - kill
  - kill -l
  - kill --signal
  - man 7 signal

# Gerenciamento de diretórios

- ls - lista
- cd – troca diretório atual
- pwd – exibe diretório atual
- mkdir – cria diretório
- rmdir – remove diretório vazio
- rm – apaga arquivos e diretórios

# Manipulação de arquivos

- cat – concatena arquivo e imprime ( tac – inverso )
- rm – apaga arquivos e diretórios
- touch – altera data de modificação, cria arquivo
- cp – copia arquivo
- mv – move arquivo, renomear
- head – exibir início de arquivo



# Manipulação de arquivos

- tail exibir final de arquivo
- grep busca em arquivo
- more and less
- sort – ordenar
- wc – contar letras, palavras e linhas
- cut – exibir partes
- diff – diferença entre arquivos

# Comandos diversos

- date – data do Linux
- find – procura arquivos/ diretórios
- time – tempo de execução
- seq – gerador de sequência
- whereis – localizador de comandos e manuais
- which – localizador de comandos

# Comandos diversos

- cal - calendário
- file – detecta tipo de arquivo
- Gerenciamento
  - free – memória ram
  - df - disco
  - top - processos
  - ldd - bibliotecas
  - Passwd - senhas

# Exercícios

Preparar o ambiente:

Utilizaremos o arquivo ~/GNULINUX/345.txt

Crie um arquivo resultado.txt na pasta ~/GNULINUX/aula03/exercicios para colocar com suas palavras o resultado dos exercícios da aula 03.

# Exercícios

1) Utilizando o vim entre na ajuda do copy.

copie o primeiro parágrafo do 1o capítulo para depois do último parágrafo do mesmo capítulo.

Escreva no resultado.txt o comando.

# Exercícios

2) Utilizando o modo visual, faça a cópia do primeiro paragrafo do segundo capitulo para depois do ultimo

paragrafo do mesmo capitulo.

Escreva no resultado.txt os comandos.

# Exercícios

3) Escolha um outro comando do vim.

Veja no manual como utilizar, elabore um comando completo.

Escreva no resultado.txt o comando e o resultado com suas palavras.

Pode utilizar o arquivo 345.txt

( Indicação para case, sort e delete )

# Exercícios

1. 4) Tente utilizar os comandos de edição no arquivo 345.txt

d - apaga e c - modifica

com algum movimento:

w - palavra

O - início da linha

\$ - final da linha

2. Descreva os resultados no arquivo resultado.txt



# Exercícios

5) Escolha 3 comandos no slide comandos diversos.

Execute o man <comando> e escolha uma opção de comando.

Descreva com suas palavras a parte DESCRIPTION

Salve em resultado.txt

# Exercícios

6) Descreva no arquivo resultado.txt comandos para

Execução dos comandos sleep 1, sleep 2, sleep 3 – sequencial ( executar sem dependência)

Execução dos comandos sleep 1, sleep 2, sleep 3 – paralelo

Execução dos comandos sleep 1, sleep 2, sleep 3 – sequencial ( com dependência entre os comandos )