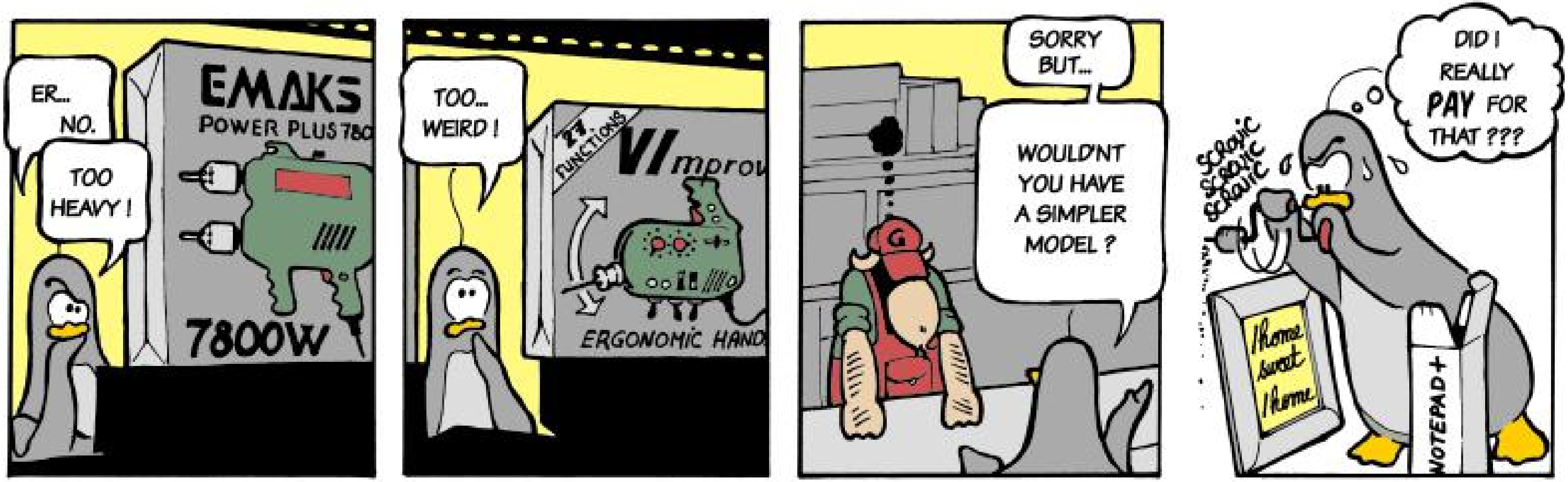
SISTEM OPERASI LANJUTAN

#2 Editor


ER... NO.
TOO HEAVY !
EMAKS
POWER PLUS 780
7800W
TOO... WEIRD !
27 FUNCTIONS
VImprov
ERGONOMIC HAND
G
SORRY BUT...
WOULD'NT YOU HAVE A SIMPLER MODEL ?
DID I REALLY PAY FOR THAT ???
scrouic scrouic scrouic
/home sweet /home
NOTEPAD+

NO MORE NOTEPAD!

- Notepad does not give you the power of a full-fledged programming editor
- Notepad lacks the tools need to write effective code (proper indention, macros, etc.)
- Notepad requires that you do everything locally and then FTP

Editor in Linux

- Vi/Vim
- gEdit
- Nano
- Emacs
- jEdit
- etc.

```
apt-get install [editor-name]
```

The vi/vim Editor

- Short for: visual editor
- Available on all UNIX systems
  - original vi part of BSD Unix
    - *written by Bill Joy in 1976*
  - many derived, improved versions available
  - open source vim (vi improved)
  - is part of GNU/Linux
- vi has multiple modes of operation:
  - command mode, input mode, last line mode

Why Use Vim?

- Good for programming
- Fast & Lightweight
- Ergonomic -> increase productivity
- Powerful
- Free

The Good, The Bad & The Ugly

- Can be challenging to learn
- Requires memorization
- Learning curve pays for itself in the end!

Command Mode

- Default mode
- Used to enter commands
  - Text manipulation
  - Change modes
  - Save/exit
- Most commands are just alpha characters, not control sequences
- Case sensitive!

Input Mode

- The mode that lets you edit and enter text
- Several sub-modes
  - Insert
  - Append
  - Open
  - Replace
  - Change
- You’ll spend most of your time here

Getting into Input Mode

i nsert before cursor

I nsert before first nonblank character on line

a fter cursor

A t end of line

o pen line below

O pen line above

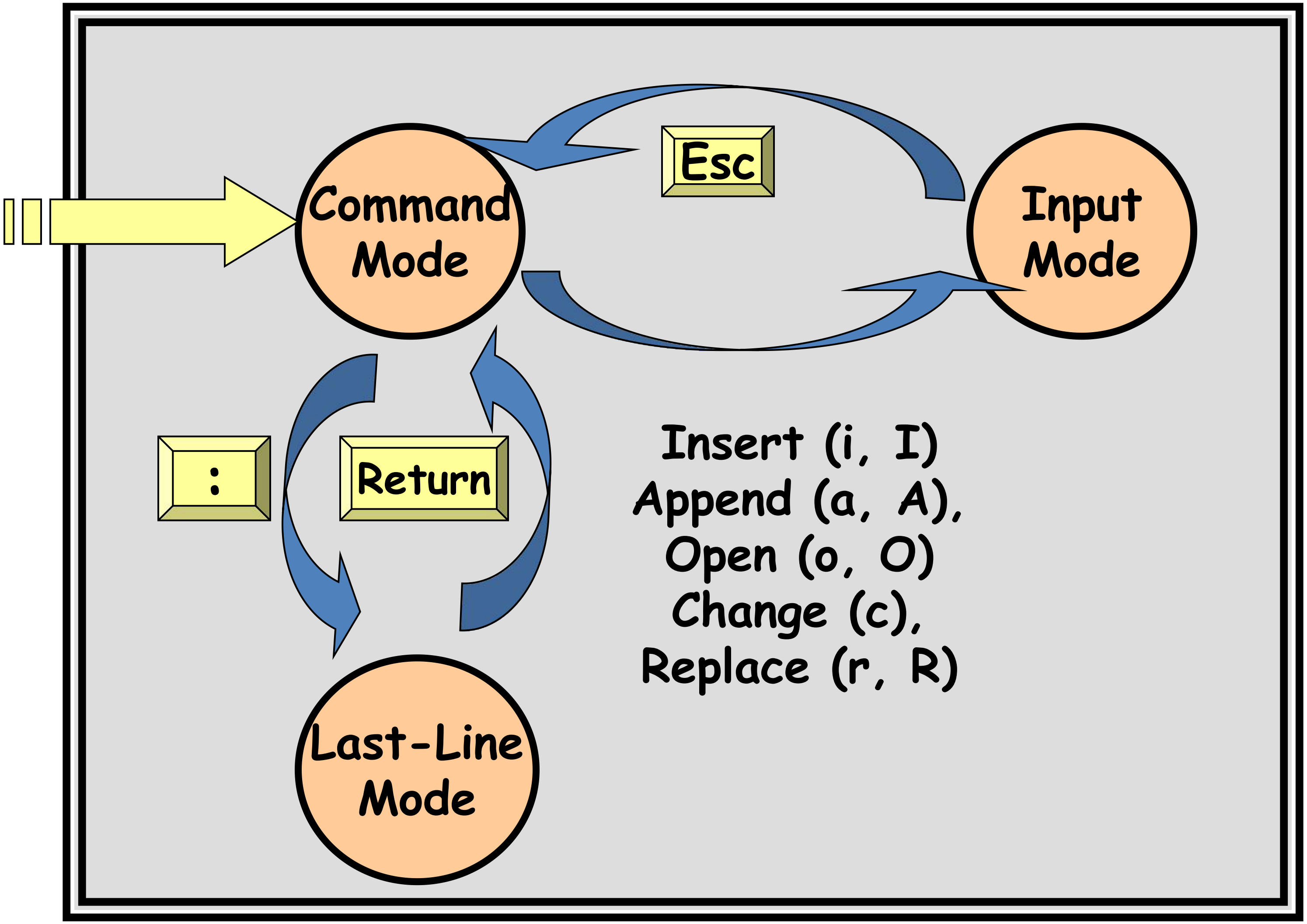
r eplace current character

R eplace characters

Last Line Mode

- From command mode press :
- Cursor jumps to the last line on the screen
- Here you can manage files, issue shell commands, change editor settings
- Also where you go to exit

vi Editing modes



VIM TUTORIAL

Never start something you can’t exit

- To end vi tutorial in the middle of the session, execute the command :q!
- :q! = quit without saving
- :wq = write out (save) and quit

F1 = help

- or :help
- :help <command>
- :q to exit help window

COMMANDS

Delete characters

- x deletes character under the cursor

Insert characters

- i converts to insert mode
- then type characters
- <esc> to exit insert mode

Motion in command mode:

- h,j,k,l: left,up,down,right
- 0,$: move to begin/end of current line

```
$ vim hello.c
$ cat hello.c
#include<stdio.h>

main() {
	printf("Hello STSN!\n");
}
$ gcc -o hello hello.c
$ ./hello

Hello STSN!
```

COMMANDS

Insert lines

- o = open line below cursor
- O = open line above cursor
- <esc> to exit insert mode

Append characters

- A converts to insert mode at end of a line
- then type characters
- <esc> to exit insert mode

COMMANDS

Deletion

- d$ deletes to end of line
- dw deletes to beginning of next word
- de deletes to end of current word
- d + motion

Using motions for movement

- Use any of the motions above
- Use count for repetition
- 2w = move cursor two words forward
- 0 = start of line

COMMANDS

Using repetition as part of deletion

- dw delete word
- 2dw deletes next two words

Deleting a line

- dd = delete line
- 2dd = delete two lines

Undo

- u = undo one command
- ctrl-R = redo a command

COMMANDS

p = put back the deleted text (in new place)

- one of the delete command above + put = cut-and-paste

More general cut-and-paste

- v = start visual mode (start block)
- move cursor to end of block
- y = yank (copy to buffer)
- then p = put in new place

COMMANDS

Location

- ctrl-g = show position in file
- G = go to bottom of file
- gg = go to top of file
- <number>g = go to line <number>

COMMANDS

Search

- /<phrase> = search
- /<phrase>\c = ignore case
- ?<phrase> = search backwards
- n = repeat search
- N = repeat search in the other direction

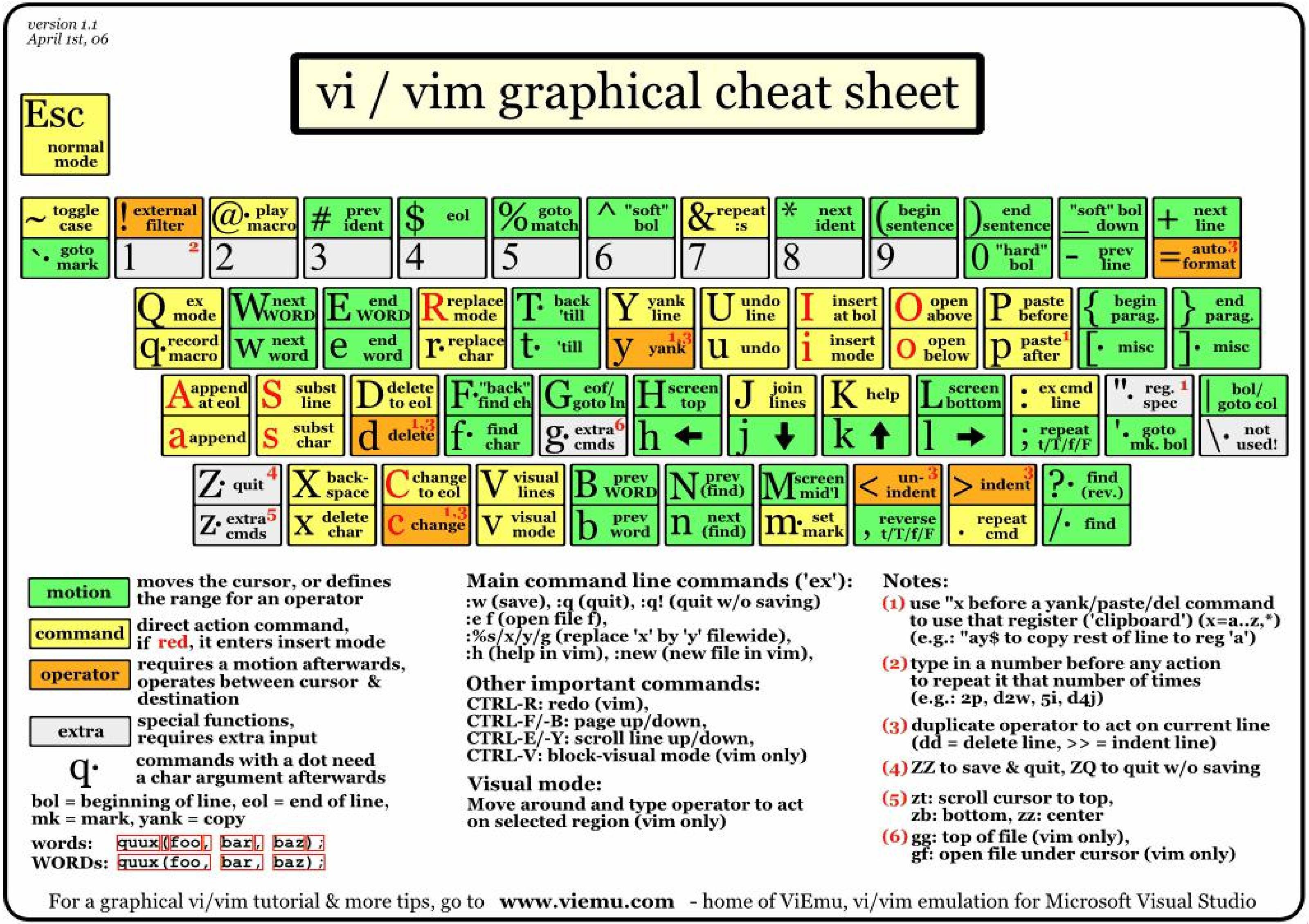
Search for matching parentheses

- Put cursor on (, [or {
- % = go to matching one
- % = go to first one again

COMMANDS

Substitute (replace)

- :s/thee/the = changes first one
- :s/thee/the/g = changes all (global change)
- :s/thee/the/gc = change all with query
- :#,#/thee/the/g = only change within that line range



version 1.1
April 1st, 06

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]	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto [3] 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 [1,3]	u undo	i insert mode	o open below	p paste [1] 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. [1] spec	| bol/ goto col
a append	s subst char	d delete [1,3]	f· find char	g· extra [6] cmds	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	'· goto mk. bol	\· not used!

Z· quit [4]	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- [3] indent	> indent [3]	?· find (rev.)
z· extra [5] cmds	x delete char	c change [1,3]	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 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)

For a graphical vi/vim tutorial & more tips, go to **www.viemu.com** - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

Customizing vim

Modify the ~/.vimrc file

Some common syntax

- set nu
- syntax on
- colorscheme desert

Vim plugins

Many online resources

Great color scheme, highlight keywords, etc.

https://github.com/amix/vimrc