

Instructor: Prof. Lee M. Thompson

E-mail: lee.thompson.1@louisville.edu

Office: CB 251

To edit a file on the CRC, you will use a program called “vi”. Using vi, you will be able to write computer code that you can then compile and run on the CRC. To use start, type “vi **whatever-filename-you-want.f03**”. The extension f03 is used to indicate that the file contains fortran 03 source code. To insert text, press “i” and then start typing or press the delete key to delete text. If you wish to paste text you must type i before pasting. To exit from typing mode back into view mode, press “esc”. You can use the arrow keys to move around your file. To save you must be in view mode (not insert) and type “:w” and then press enter. To exit the text editor, type “:q”. To simultaneously save and quit, type “:wq”, and to quit without saving changes type “:q!”.

Vim is famous for being an incredibly useful text editor, but with a steep learning curve. The instructions just provided are the minimum you require in order to use vi. However, there are many other key bindings that are very useful. The best way to learn is to use the program and try out some of the commands in the cheat sheet shown in fig. ??.

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	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto 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	u undo	i insert mode	o open below	p paste 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. spec	bol/ goto col	
a append	s subst char	d delete	f find char	g extra cmds	h ←	j ↓	k ↑	l →	. repeat t/T/f/F	' goto mk. bol	\ not used!	
Z quit	X back-space	C change to eol	V visual lines	B prev word	N prev (find)	M screen mid'l	< un-indent	> indent	? find (rev.)			
Z extra cmds	X delete char	c change	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, 5l, 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

Figure 1: Vi cheat sheet

Now we will try to write a basic computer program in fortran 03. In the following any bold text

should be replaced with your choice of text.

1. Open a file with the extension “.f03” in vi and on the first line enter the text “program **the-name-of-your-program**”
2. To link to the MQC library routines that you installed last week, add “use mqc_gaussian” to the next line of your code.
3. In the next lines, all variables that you use in the code must be declared along with their type. In fortran there are several intrinsic types (integer, real, complex, character, logical) that variables can take, along with several other properties, for example dimension(N) declares a vector of dimension N (where N is an integer number) and dimension(N,M) declares a $N \times M$ matrix. In addition, MQC provides a number of other types that we will use. On the next line add the text “implicit none” to force you to explicitly declare the type of each variable and not use the automatic typing based on the first character of each variable. Then on the next line write “type(mqc_matrix)::A,B,C” which declare three variables, A, B and C, of the type mqc_matrix.
4. Having declared our variables, we will now write the body of the code. On the next line write “A=reshape([2,1,1,0],[2,2])” and on the next line write “B=reshape([1,2,1,3],[2,2])”. The variables A and B were declared as a 4×1 vector and then the reshape function changed the vector to a 2×2 matrix. We will now multiply the the matrices together and put the result in variable C (we are doing the same matrix multiplication as question 1a in worksheet 2). On the next line write “C=matmul(A,B)” and then on the next lines write “call A%print(6,‘Matrix A’)”, “call B%print(6,‘Matrix B’)” and “call C%print(6,‘Result of A.B’)”. The number 6 in the previous command is a file number that fortran uses to determine where to write the output. The file number 6 is reserved for standard output, which is the terminal screen.
5. On the next line write end program “end program **the-name-of-your-program**” which marks the end of the program.

To summarize, the file you have should now look as follows:

```
program the-name-of-your-program

use mqc_gaussian

implicit none
type(mqc_matrix)::A,B,C

A=reshape([2,1,1,0],[2,2])
B=reshape([1,2,1,3],[2,2])
C=matmul(A,B)
call A%print(6,‘Matrix A’)
call B%print(6,‘Matrix B’)
call C%print(6,‘Result of A.B’)

end program the-name-of-your-program
```

Now we need to compile the code we have written into an executable. We could use a command

on the command line to compile the code (e.g. “gfortran **the-name-of-your-program.f03**”), but when you have more complicated requirements, such as linking with libraries, it is better to use a makefile which contains the necessary compilation commands. To write the makefile, exit out of the text file back to the command line. Type “vi makefile” and add the text below into the file:

```
mqcroot = “$(HOME)/mqc_install”
FC = gfortran
ifeq ($(FC),gfortran)
    FCFLAGS = -std=f2008 -fdefault-real-8 -fdefault-integer-8 -fopenmp
    MQCOBJS = -I$(mqcroot)/GNU/mod
    LIBS = -llapack -lblas $(mqcroot)/GNU/lib/libmqc.a
else ifeq ($(FC),pgfortran)
    FCFLAGS = -Mallocatable=03 -r8 -i8 -mp
    MQCOBJS = -module $(mqcroot)/PGI/mod
    LIBS = -llapack -lblas $(mqcroot)/PGI/lib/libmqc.a
endif

all: compile

compile: the-name-of-your-program.f03
    $(FC) $(FCFLAGS) $(MQCOBJS) -o the-name-of-your-program.exe the-name-of-your-program.f03 $(LIBS)

clean:
    rm -f the-name-of-your-program.exe the-name-of-your-program.o
```

Save and exit and on the command line, enter “module load gcc/9.2.0” and then type “make”. Your code should compile without errors. Once your code has compiled, run the program by typing “./**the-name-of-your-program.exe**”. Check that the program give the expected output for the matrix multiplication.