Version of 1/14/94

605.724 Artificial Intelligence Programming
Spring 1994, Marty Hall

Emacs Quick Reference Guide

1. Nomenclature

^a	Hold CONTROL while hitting "a".
m-a	Hit ESCAPE then hit "a".

2. Basic Survival

^g	Cancel any unfinished command.
^x u	Undo previous command.
^l	Redisplay screen.
^x^f	Edit or create a new file.
^x s	Save current file.
^x^c	Exit from Emacs.
^x ^k†	Exit emacs, killing all buffers without any warnings or saving anything.
^h?	Gives you help options.
m-*N*	Repeats the following command *N* times.

3. Cursor Movement

^b	Move backward one character.
^f	Move forward one character.
^p	Move to the previous line.
^n	Move to the next line.
^a	Move to the beginning of the line.
^e	Move to the end of the line.
^v	Move one screen forward (down).
m-v	Move one screen backward (up).
m-<	Move to beginning of buffer.
m->	Move to end of buffer.
m-*N* m-g†	Go to line *N*.

4. Inserting and Deleting

^d	Delete character to the R of cursor.
DEL	Delete char to the L of cursor.
m-d	Delete word to the R of cursor.
m-DEL	Delete word to the L of cursor.
^k	Kill (delete) line.
^@	Set mark at current cursor position.
^w	Deletes (wipes) region (between cursor and mark). Set mark, move someplace, then cut.
m-w	Makes copy of region (between cursor and mark) for later pasting, but doesn't delete it.
^y	Pastes (yanks) back most recently deleted text from kill buffer. Anything deleted via ^w or ^k (or copied via m-w) goes in this buffer.
^x^x	Swap positions of cursor and mark. Useful for returning to a previously marked place in a long file.

†. Only if using my .emacs file; ~hall/.emacs

5. Search and Replace

^s (or ^])†	Search forward for string. ^s again looks for next occurrence.
^s^s (or ^]^]† **)**	Searches forward, looking for whatever string was in previous search.
^r	Reverse search (hitting ^r again during search looks for next occurrence).
m-%	Forward search and replace, with confirmation on each one. Type "?" when searching for a list of options.
%†	Jump cursor to matching paren if cursor is on a paren. Otherwise add a "%".

6. Buffers and Windows

^x 2	Split screen into 2 windows.
^x 1	Go back to 1 window.
^x o	Move cursor to other window (when in split-screen mode).
^x^w	Rename (write) buffer to new file.
^x^b	List all current buffers. Put cursor on buffer of interest and hit "q" to select.
^x b	Switch to different buffer (Type in buffer name, SPACE will do name completion).
m-r†	Switch to the most recent previous buffer.
^x k	Kill current buffer. Unlike ^x^k†, does not exit emacs.

7. LISP Specific commands

Assumes file extension of .lisp or .lsp. Matching parens will be blinked. There are also C, Pascal, and FORTRAN modes which do automatic indentation on TAB.

TAB	Indent current line to "proper" position. Cursor need not be at beginning of line.
LINEFEED/ENTER	Does RETURN and then TAB.
m-x mark-defun	Selects function (defun) containing cursor.
^x^d†	Copies current defun into LISP buffer, leaving you in original file.
^x d†	Same as above but leaves you in LISP.

8. Buffer Processes

m-l†	(or **m-x run-lisp**) Starts LISP in separate emacs buffer.
m-u†	(or **m-x shell**) Starts UNIX shell in separate buffer.
m-p†	Starts PROLOG in separate buffer.
m-r†	Switch to the most recent previous buffer.

For more detailed info, see "Using GNU Emacs" or send email to hall@aplcenmp.apl.jhu.edu.

Using Gnu Emacs

1. General

GNU Emacs is a powerful, extensible, full-screen editor written by Rich Stallman of the Free Software Foundation, formerly of the MIT AI Lab. It has many language-specific "modes" that support development and editing in various languages, including LISP, C, Pascal, FORTRAN, PROLOG, Modula-2, Scheme, and nroff. Most of these are entered automatically based on the file extension, but you can add new file types yourself. One of the most extensive language-specific modes is the LISP mode, which automatically balances your parens, indents your code, and more. This makes emacs an ideal editor for use in 605.724 (or 723). There are free versions available for almost every machine, and machines with the X-Window system can make use of even more advanced features using fonts and the mouse (especially with emacs 19.xx or lemacs).

Send questions or corrections to hall@aplcenmp.apl.jhu.edu.

2. Nomenclature

^*key* means to hold down CONTROL and hit *key*. It is not usually case sensitive; ^c = ^C.

m-*key* means to hit and release ESCAPE, then hit *key*. It is case sensitive; m-% ≠ m-5. Some keyboards have a "true" meta key (e.g. Suns), in which case m-*key* means to hold down META (or ALT or RIGHT) and hit *key*.

3. Getting Started from Unix

"%" refers to your Ultrix prompt. I.e. "% foo" means type "foo" at the Unix prompt.

% emacs	Starts emacs.
% emacs *file*	Starts emacs, editing or creating *file*.
^z	From within emacs, suspends emacs and returns you to Unix shell. In general in Unix, ^z suspends the current process.
% fg	Assuming that emacs was the last process you suspended, this returns you to emacs at the place you were when you hit ^z. In general, brings last suspended process to the foreground.
% %emacs	Assuming there is a suspended emacs process, this returns you there whether or not it was the most recently suspended process. In general, "%name" brings "name" to foreground. This may only work with Cshell (csh or tcsh); if you change your default shell to a Korn shell, you may need to learn a different method.

The above three commands let you switch back and forth among your LISP, Unix, and emacs processes, without losing the current state of any of them. For instance, from within LISP you could load several files, find an error in one (say foo.lisp), hit ^z to drop you back to Unix, and type "emacs foo.lisp" to edit file foo.lisp. You could then make the desired changes, save the file (but not exit emacs), hit ^z, and type "%lisp" to return to the existing Lisp process (**not** just "lisp", or you will get multiple lisp processes running), Then, from within LISP, reload just the changed file, with all other function definitions and variable bindings remaining unchanged.

There is actually a better way to do this, using multiple-buffers in emacs, and cutting and pasting between the buffers, but the above is the standard UNIX method of switching among processes. See "Buffer Processes" and "LISP Specific Commands".

4. Getting Help

^h t	Gives an interactive **t**utorial on using emacs.
^h i	Lets you browse through the emacs documentation (**i**nformation).
^h a	Finds all commands containing a substring, plus tells which keystrokes invoke that command. (**a**propos). Good way for finding new capabilities.
^h k	Tells what commands given **k**eystroke does.
^h ?	Gives you other **h**elp options. Many of these are very useful.

5. Basic Survival

^g	Cancel any unfinished command. If you get stuck, hit this once or twice.
^x u	**U**ndo previous command.
^l	Redisplay the screen (if it gets messed up somehow).
^x^f	Edit or create a new **f**ile. You will be prompted for the name.
^x s	**S**ave current file. If you have my .emacs, you will not be prompted.
^x^c	Exit from Emacs. You will be prompted if you have unsaved buffers.
^x ^k†	Exit emacs, **k**illing all buffers without any warnings or saving anything.
m-*N*	Repeats the following command *N* times. E.g. m-37 ^n moves down 37 lines, m-8 ^k deletes next 8 lines, m-4 ^y inserts 4 copies of the last thing deleted, etc.

6. Cursor Movement

^b	Move the cursor **b**ackward one character (left arrow also works on many keyboards)
^f	Move the cursor **f**orward one character (right arrow also works on many keyboards).
^p	Move the cursor to the **p**revious line (up arrow also works on many keyboards).
^n	Move the cursor to the **n**ext line (down arrow also works on many keyboards).
^a	Move the cursor to the beginning of the line.
^e	Move the cursor to the **e**nd of the line.
m-b	Move the cursor **b**ackward one word.
m-f	Move the cursor **f**orward one word.
^v	Move the cursor one screen forward (down).
m-v	Move the cursor one screen backward (up).
m-<	Move the cursor to the beginning of the buffer.
m->	Move the cursor to the end of the buffer.
m-*N* m-g†	**G**o to line *N*.

7. Inserting and Deleting

^d	**D**elete the character under the cursor.
DELETE	Delete the character to the left of the cursor.
m-d	**D**elete the word to the right of the cursor.
m-DELETE	Delete the word to the left of the cursor.
^k	**K**ill from cursor to the end of line.
^@	Set a mark **at** the current cursor position. Designates the beginning of a cut/paste region (cursor position is the other end). On some older terminals, you may have to use ^-shift-SPACE instead of ^@.
^w	Deletes (**w**ipes) region between cursor and mark. Set the mark someplace via ^@, move someplace else, and then ^w deletes the region in between.
m-w	Makes copy of region for later pasting, but doesn't delete it.
^y	Pastes (**y**anks) back most recently deleted text from the kill buffer. Anything deleted via ^w or ^k (or copied via m-w) goes in the kill buffer.
^x^x	Swap positions of cursor and mark. Useful for returning to a previously marked place in a long file.

†. Only if using my .emacs file; ~hall/.emacs

8. Search and Replace

^s (or ^]†) — **S**earch forward for string. ^s again looks for next occurrence.

^s^s (^]^]† — **S**earches forward, looking for whatever string was in previous search.

^r — **R**everse incremental search (^r^r reuses last string, hitting ^r again during search looks for next occurrence).

<u>Note:</u> since many terminals use ^s/^q as XON/XOFF, you may not be able to do a forward search. So to find a given string, do m-> to go to bottom of buffer, then do ^r to look backward for string. Of course, since emacs is extensible, you could redefine which key sequence invokes a search. For instance, I define ^] to do forward searches. See "Customizing Emacs".

m-% — Forward search and replace, with confirmation on each one. Type "?" when searching for a list of options to give for confirmation.

%† — If you are over a paren, jumps cursor to the matching paren, even if that is off the screen. If cursor is not over a paren, just inserts a %.

9. Buffers and Windows

^x 2 — Split screen into **2** windows.

^x 1 — Go back to **1** window. Some help options give you a split screen: this undoes that

^x o — Move cursor to **o**ther window (when in split-screen mode).

^x^w — **W**rite (rename) buffer to new file.

^x^b — List all current **b**uffers. Move cursor to line buffer is listed on and hit "q" to select it.

^x b — Switch to different **b**uffer. You will be prompted for name, and like in most of emacs, a space will make emacs try to complete the name.

^x k — **K**ill current buffer. You will be prompted to save it first, if necessary. Unlike ^x^k†, does not exit emacs.

m-r† — Switch to most **r**ecent buffer. Useful for switching back and forth.

10. LISP Specific commands

m-x lisp-mode — Tells emacs you are editing a LISP fil**e. This is the default when editing "xx.lisp" or "xx.lsp" files,** and is indicated by (Lisp) on the bottom emacs status line. *This will make emacs automatically "flash" the matching L-paren whenever you type a R-paren, will make TAB indent you to the "correct" level based on how many parens are open, and make LF do RETURN then TAB.*

TAB — Indent current line to "proper" position. Cursor need not be at beginning of line. TAB also does indentation in other modes (C, Pascal, PROLOG, etc.).

LINEFEED — (**ENTER** on many keyboards). Does RETURN and then TAB. A good habit is to always use LF instead of RETURN when editing LISP files, to guarantee "correct" (or at least good) indentation.

m-x mark-defun — Selects entire defun that cursor is inside. Makes it easy to copy (m-w), delete (^w) and move (^y in new location) entire functions.

m-a/m-e — Move to beginning/end of defun. Analogous to ^a/^e for moving to beginning/end of a line.

^x^d† — Copies current defun into LISP buffer, updating the definition of that function, and leaving you in original file. You must have an active LISP buffer for this to work; see "Buffer Processes" for more explanation. <u>This may be the single best timesaving feature of using emacs for LISP, and can save large amounts of debug time.</u>

^x d† — Same as above but leaves you in LISP rather than in your original source file.

m-=† — Insert a LISP-function separator (;;;===== ...) at current cursor location.

%† — See the description under "Search and Replace". The automatic paren-blinking lets you balance parens most of the time, but this is occasionally useful.

11. Buffer Processes

m-l† Lisp. Typing m-l† or “m-x run-lisp” will create a buffer with Common LISP running in it. If one already exists, this will just switch to it. Switch back to a file using ^x b (choose a buffer) or m-r† (most recent buffer). This is useful since general emacs commands work in this buffer, so that you can cut and paste in LISP, making it easy to repeat previous LISP commands, check earlier output or print the results of an interactive session. Even more valuable is the ability to incrementally update function definitions when you are debugging a program. You could select and copy a group of changed functions, then paste them into LISP to update their definitions. However, that is precisely what ^x^d† and ^x d† do. This can radically speed up coding and debugging time as compared to saving and reloading the entire file every time you make a change.

m-u† Unix. A Unix shell can be created in an EMACS buffer via m-u† or “m-x shell”. If one already exists this will just switch to it. Some people become so attached to emacs that they start it via their .login file, and only use the shell from within emacs, thus letting them see all their previous I/O and making it easy to edit command lines. Like all emacs buffers, this one can be saved and named via ^x^w. The only problem is that emacs traps control characters for use as commands, rather than passing them through to the shell. Use “m-x quote ^D” to pass a ^D through to the shell. ^Z gets passed through automatically.

m-p† PROLOG. Like m-l† and m-u†, if the buffer already exists, it will just switch to it, otherwise start it and then switch to it.

m-r† Recent. Switch to the most recent buffer (other than current). This is useful for switching back and forth between, for instance, the LISP buffer and a LISP source file.

12. Customizing Emacs

Whenever emacs is entered, it looks for a file named “.emacs” in your home directory. This allows you to redefine key bindings, define new functions, customize your display, etc. For people familiar with LISP syntax, you can quite easily write extensions and modifications to emacs. For more information see the section in the emacs documentation (invoked via ^h i) on the .emacs file. You can also use ^h a (apropos) to find out about various built-in capabilities that you might want to use/combine in customizations.

Alternatively, take a look at my .emacs file, which gives a bunch of examples. Feel free to copy and modify this file (~hall/.emacs). Students in my class should have a version similar to this automatically created in their account. The commands in this outline designated with “†” are defined in my .emacs and not available by default. Customizations not already described in this handout include commands to insert C or C++ function separators, recognizing .C and .cpp (C++ file extension) as C files, expanding memory limits to avoid garbage collection, and a few minor variable redefinitions.