# CSCI 136- Emacs Hints

## Cursor Motion Commands:

| | |
|---|---|
| C-a | Beginning of line |
| C-e | End of line |
| C-v | Next screenful |
| M-v | Previous screenful |
| M-< | Beginning of buffer |
| M-> | End of buffer |
| C-s | Search forward incrementally |
| C-r | Reverse search incrementally |
| C-x C-x | Swap mark and cursor |

## Editing Commands

| | |
|---|---|
| C-d | Delete next character |
| Backspace | Delete previous character |
| M-x replace-string | Global string replacement |
| M-% | Query string replacement |
| M-d | Delete next word |
| C-k | Kill to end of line (delete to end of line) |
| C-w | Cut region |
| M-w | Copy region |
| C-y | Yank most recent cut/copy (paste command) |
| M-y | Replace yanked text with previously cut/copy text (only works immediatly after C-y or another M-y) |

## File Commands

| | |
|---|---|
| C-x C-f | Open a file |
| C-x C-s | Save buffer to file |
| M-x revert-buffer | Throw out all changes and revert to the last saved version of the file. |

## Buffer Commands

| | |
|---|---|

| | |
|---|---|
| name> | Replace the current buffer with the one with the given name |
| C-x 1 | Make the current buffer be the only buffer |
| C-x 2 | Split the current buffer into two buffers |

## Help Commands

| | |
|---|---|
| C-h i | Info |
| C-h a | Apropos |
| C-h b | Key bindings |
| C-h m | Mode help |

## Java Mode Commands

| | |
|---|---|
| C-j | Insert a newline and indent the next line. |
| Tab | Fix indentation of current function |

## Miscellaneous Commands

| | |
|---|---|
| C-x C-c | Exit Emacs |
| C-g | Cancel command |
| M-x goto-line | Go to specific line number |
| C-SPACE | Set mark |
| C-x u | Undo |
| M-x shell | Start a shell in a new buffer |
| M-x print-buffer | Send the contents of the current buffer to the printer |
| M-x compile | Compile a program |
| M-x set-variable | Change the value of an Emacs variable to customize Emacs |

## Emacs Tips

Here are some more commands you will find useful in Emacs. Practice them to learn them.

### Searching

It is typically useful to be able to search for strings in Emacs. Emacs provides an incremental search command that is very useful. You begin the search using C-s. In the message window at the bottom of the screen, you will see:

```
I-search:
```

The cursor will be following this prompt. Here you can type in the string that you are looking for. The search is done incrementally as you enter characters in the string. So, for example, suppose I wanted to search for the word "regions". I would type in "r" with no carriage return and Emacs will immediately move the cursor in the main buffer to the next r in the buffer. Now, when I type an "e", Emacs will move to the next location of "re". After typing characters one of 3 things will happen:

- You will find the occurrence of the string you are looking for. In this case, type C-SPACE (the control and space keys). This will exit the search leaving you where the character was found.
- You will find an occurrence of the string you are looking for, but not the occurrence that you want. In this case, type C-s again. You do not need to enter the string again.
- The string will not be found. In this case, you will get the message:

  ```
  Failing I-search
  ```

  What this means is that string did not occur between the point in the buffer where you started the search and the end of the buffer. It still might be in the buffer prior to where you started the search. If you believe that is true, type C-s again. The search will now wrap around to the beginning of the buffer. The prompt will change to

  ```
  Wrapped I-search:
  ```

  If you keep searching beyond your original search point, the prompt will change to

  ```
  Overwrapped I-search
  ```

  This is an indication that you are searching through text that you have already searched through once.

If at any point you want to abort the search, use the cancel command (C-g). This will put you back in the buffer where you started the search. Also, if you type lowercase characters in your search string, they will match both uppercase and lowercase characters in the text. If you enter uppercase characters, however, it will assume the case is significant and will only match uppercase characters.

There is a similar search that goes up from your current location in the buffer. This is bound to C-r and uses the prompt

```
I-search backward
```

## Replacing

Replacement is related to searching. There are two replacement commands. The first is global, while the second is incremental. For each command, you give the string you want to replace, hit carriage return, and then give the string to replace it with. The global replacement will replace all occurrences of the first string with the second string and then tell you how many replacements were done. The incremental replacement will show you each occurrence and ask you if you want to change that occurrence or not.

To use global replacement, type "M-x replace-string". You will get the prompt:

```
Replace string:
```

Type the existing string to replace followed by carriage return. Now you will get the prompt:

```
Replace string <old string> with:
```

where <old-string> is the first string you typed. Enter the new string followed by carriage return. You will then see the message

```
Replaced <num> occurrences.
```

where <num> is the number of strings replaced.

Incremental replace is actually called query replace and can be executed using M-%. The first prompt is:

```
Query replace:
```

Type the existing string to replace followed by carriage return. Now you will get the prompt:

```
Query replace <old string> with:
```

where <old-string> is the first string you typed. Enter the new string followed by carriage return. Emacs will move the cursor in the buffer to the next occurrence of the first string. You will then see the prompt:

```
Query replacing <old-string> with <new-string>: (? for help)
```

Type y to replace that occurrence and move to the next occurrence. Type n to leave that occurrence unchanged but move to the next. Use the cancel command (C-g) to quit query replace and go back to the point in the buffer where you started the replace. Canceling will not undo any replacements already done and will leave you at the last occurrence Emacs visited, not where you started the replacement. If you do not cancel, when you reach the end of the buffer, you will be told how many occurrences were changed as before:

```
Replaced <num> occurrences.
```

where <num> is the number of strings replaced.

For both global and query replacement, only those occurrences between your starting point in the buffer and the end of the buffer are changed. If you want to change all occurrences in the buffer, you must first go to the start of the buffer. Also, the global replacement is somewhat hazardous to use, especially for short strings, as it also matches parts of words. So you need to think carefully when using global replacement about whether you will accidentally change some text you don't want to change.

## Regions

The cursor is the dark rectangle visible when editing in a buffer. It is possible to set a mark at the current cursor position. When you move the cursor, the mark stays at the old cursor location. The area between the mark and the cursor is called a region. There are numerous operations that can be performed on a region. The most important of these are described in this section.

One thing you can use this for is to quickly move between two points in your buffer. Set the mark at one point,using C-SPACE then go to the place you want to edit. When you want to jump back to the saved point, type C-x C-x. This will swap the mark and the cursor. Your cursor will be at the previously marked point and the mark will be at your recent cursor position. In this way, typing C-x C-x a second time will take you back to where you jumped from. It is often a good idea to confirm that the region includes what you expect before applying a region command. To do this, just type C-x C-x once to confirm what one boundary is and then C-x C-x again to confirm where the second boundary is.

One thing that regions are very useful for is cut-and-paste and copy-and-paste. To cut a region, type C-w. To copy a region. type M-w. To paste in the cut/copied text, type C-y (yank).

Note that some commands set the mark in addition to performing their main action. These commands will print

```
Mark set.
```

in the message buffer. That is why it is a good idea to confirm region boundaries before applying region commands.

## More on cutting and pasting

On previous days, we have seen C-d and Backspace as ways of deleting individual characters. In the previous section, we saw C-w as a way of cutting an entire region. There are some other useful cut commands:

| | |
|---|---|
| M-d | Cut next word |
| C-k | Cut rest of the line |

The cut commands put the text cut into a kill ring. The kill ring contains the last 30 chunks of text that were cut. The paste command (C-y) pastes back in the most recently cut text. If that is not the one you want use M-y as your next command. That replaces the pasted text with the next most recently cut text. It is called a kill ring because it is circular. Eventually, you might get the 30th most recently cut text. If you type M-y again, you will wrap around to the most recently cut text.

Note that C-d and Backspace, which delete individual characters, do not update the kill ring. These deletions are small and are therefore not worth having each character take up a slot in the kill ring. It would be just as easy to type in that single character. Only commands that normally excise multiple characters are saved in the kill ring.

Also, note that if you use multiple cut commands consecutively, they cut text will be grouped together into one entry in the kill ring. This is very convenient so that you can use C-k to cut multiple consecutive lines and then paste them as a single unit elsewhere in the text.

## Undo

I am sure you are all familiar with Undo from using Macintosh and Windows applications. Undo in Emacs is bound to "C-x u". You can issue it multiple times in a row to undo multiple commands. Typing any command other than undo will stop the undo. Now, an interesting thing happens. Your command history now has several undo commands in it. If you start undoing, you will essentially undo the undo commands. Think of this as a redo command but it does not have a separate key binding.

If you want to undo all the editing changes since the last time you saved the file use the revert-buffer command (M-x revert-buffer).

If you start a command and then decide you don't want to complete it, type C-g to cancel it.