Code and Poetry – Pandoc and Markdown Lab 9-17-14

* Technology is a way to frame the world around you (Heidegger and Winner) – up to you what instruments you use
  + Tools are extensions of your brain – affect the way you think about things, extending memory
* The way you organize files influences the way your files are published/consumed/etc.
* Superstructural problems have their roots in the base phenomenon (production of knowledge)
* Important to trace transfer of texts from your finger to your computer, from your computer to your audience
* Useful to start thinking of documents not as points, but as lines/vectors from your brain to another brain – travels in space and time
* Documents as moving/changing/indeterminate
  + Very difficult to know when the poem/text is finished – multiple versions
  + Continually moving/morphing
* **Lab**
* How do you store/backup/share documents?
  + Different ecosystems
    - Encryption – prevent people from snooping/tapping into the vector
    - Different vectors pulled in different directions – Google and Amazon take care of it for you, handle backup/storage
      * Problem: tend to use a proprietary format at the birth of a vector – because you already chose a particular ecosystem/vector (had it chosen for you), you may not get to change it later on (ie encrypt it)
      * Take the origin point of all the separate vectors with common textual problems (accessing/searching/sending/backing up/encrypting) and totalize under something (Google)
        + Uncomfortable to completely cede control
  + Computer is a universal machine (Turing)
    - Computer is a machine that can emulate any other machine
  + Free and open software movement
    - Evan Moglin – we should own the hardware/make our own servers (fairly extreme), have the physical layer under our control
  + Somewhere, there’s an opportunity to make all files (ebooks, email, documents, blog posts) plaintext files, unifying all vectors at the point of origin
    - Nonproprietary, long lasting, easy to search, easy to vectorize
    - At some point will be forced to vectorize in a way that’s not of your choosing, but the source material will be under your control (and malleable)
      * Documents born in plaintext, but sometimes that plaintext will become a pdf, word doc, text message, email
  + Encryption is considered military technology – treason to teach people it – dangerous for people to have access to codes
    - Turing – Enigma Project
    - To encrypt something you should start with plaintext
* **Pandoc and Markdown**
* Any plaintext editor will do
* Microsoft word – lots of gunk injected into document, but not done maliciously – why do they do that? (see reading for next week)
  + “form and content” – is there such a thing as just content?
    - Separating form and content gives you certain freedoms in programming
    - In MS word form and content are the same – we write and format at the same time – what you see is what you get
      * Underneath there is some markup to make italic/some other command – hiding another format layer under the surface – only MS word and printer will understand this layer
        + Printer flattens form and content
      * Typesetting – stripping formatting, putting in own formatting, laying out on page and sending to publishers
      * Smart thing to do would be to separate content from format – you don’t know where the paper will end up – completely different formats, so why are we so concerned with formatting on the printed page
        + Freeing from initial formatting saves you time
        + Markdown lets you do this while also seeing the page nicely formatted/laid out
* HTML
  + Human readable underneath (making it preferable to MS word)
    - .doc -> .docx tries to address this complaint – x = extensible markup language
      * xml
      * rml – making ml tags – took the chaotic world outside and separated into little markups
        + identify features that are important to you
    - HTML already had this idea
    - Paragraph is semantic structure – can be formed in different ways (blank space btwn them, indented)
      * Let programs worry about it – what paragraphs look like is established on a different layer – layout is expressed in a different layer, where you define how you want paragraphs formed
  + Problem with HTML – it’s complicated, doesn’t look good – you don’t want a paper in markup language
  + Form, content, and formatting are different
    - Formatting – style, italics, bold
    - Form – semantic unit (paragraph)
    - Content
    - HTML – not a particular format for emphasis (bold? Italic?), etc.
  + Problems: not good at separating form, content, and style; not very human readable
* Markdown tries to address this
* Who decides what a paragraph is? A footnote? HTML?
  + Very political governing body
    - Whoever holds the key to the base layer has huge implications on their business model
* Homework: Look up on daring fireball website Markdown
  + Reddit, Wikipedia, almost all web interface is in Markdown