Introduction to the Use of Computers

Text editing and encoding

Friday 19th October 2012

This lab session is about editing text, and the ways in which text is encoded for storage to a file system or for transmission to processes.

1. This part of the lab introduces you to a programmer's text editor, Emacs.

   (a) using Mac OS X as the operating system, start the Emacs application. It should be present in the dock (the series of icons at the bottom of the screen); if it is not, there should be an application launcher for Emacs in the Applications section of the Finder (the file manager).

   (b) Emacs comes with comprehensive built-in help, including an introductory tutorial. Firstly, launch the tutorial by issuing the following keystrokes: `C-h t` (meaning: first, hold down the control key and, while still holding it down, hit '`h`'; then, let go of the control key (and the '`h`') and hit `t`). The tutorial should replace the initial startup screen; reading carefully, follow the tutorial's instructions from beginning to end.

2. This part of the lab introduces some more of Emacs' functionality. Emacs comes with functionality not just for text editing but for all sorts of tasks which programmers and other computer professionals routinely perform. One such task is investigation and editing of binary files.

   (a) from the course website, download the three '`hello`' files used in lecture 1 to introduce the ASCII encoding.

   (b) open the simple text file `hello.txt` in Emacs (use `File`→`Visit New File` from the menu bar, or `C-x C-f`). Verify that the buffer's contents actually contain the text you expect.

   (c) activate the binary mode by typing the following keystrokes: `M-x hexl-mode RET` (That is, the `M-x` or `ESC x` extended command keystroke covered in the tutorial, followed by `hexl-mode` typed normally, followed by the return key). At this point you may be prompted about discarding undo information, in which case you should accept (by hitting '`y`'). The display of the file should change to one where (side-by-side) the file contents are displayed first in hexadecimal, then as characters. Investigate whether the ASCII encoding given in lecture 1 is correct.

   (d) exit the binary mode by typing `M-x hexl-mode-exit RET`; again, you may be prompted about discarding undo information, in which case you should accept (by hitting '`y`').

   (e) repeat the previous steps with the html file referred to in lecture 1. When in the normal (text-viewing) mode, you should be able to type `C-c C-v` to display the html in a web browser.

   (f) repeat the previous steps with the postscript file referred to in lecture 1. When in the normal (text-viewing) mode, you should be able to type `C-c C-c` to display the rendered postscript directly within Emacs.

3. This part of the lab is a series of questions to reinforce the material covered in lectures. If you wish, you may use Emacs (introduced in the previous part) or other text editors or similar tools to help you answer the questions.

   (a) For each of the following statements, state whether it is true, false or arguable (and if 'arguable', explain why):

      i. ASCII is a character set;
      true

      ii. ASCII is a character encoding system;
      true

      iii. ASCII is a 256-character repertoire;
      false: it is 128 characters

      iv. ASCII is normally encoded in 7-bit bytes;
      false: normally it is encoded in 8-bit bytes with the top bit being zero; packed encodings are relatively rare, because of the inconvenience to the programmer.

      v. ASCII is a fixed-width encoding;
      true

      vi. ISO-8859 repertoires allow multiple-language texts;
      arguable: they permit some multiple-language combinations, but not arbitrary ones.

      vii. ISO-8859-1 is a 256-character repertoire;
      true

      viii. ISO-8859-1 is a good solution for Western European languages;
      arguable: it is not any more, but it was a decent solution 20 years ago

      ix. the string `£10` is encoded in ISO-8859-1 as `a3 32 31`;
      false: should be `a3 31 30`

      x. ISO-8859-1 is a fixed-width encoding;
      true

      xi. Unicode is a character set;
      true

      xii. Unicode is a character encoding system;
      false

      xiii. Unicode covers all known natural languages;
      arguable: there are some rare or historic scripts which are not covered, as well as some pseudo-natural languages (perhaps the most famous of those rejected being Klingon from Star Trek or Tengwar from Tolkien). Egyptian hieroglyphics were only added to Unicode in 2009; there are almost certainly less-well-known historic scripts which are not covered.

      xiv. Unicode allows multiple-language texts;
      true

      xv. UCS-2 is a fixed-width encoding;
      true

      xvi. UCS-2 allows encoding of texts made from arbitrary Unicode characters;
      false: only the first 65536 Unicode code points can be encoded in UCS-2

      xvii. UTF-16 is a fixed-width encoding;
      false

xviii. UTF-16 allows encoding of texts made from arbitrary Unicode characters;
true

xix. the string `£10` is encoded in UTF-16 as `00 a3 00 31 00 30`;
arguable: the character encodings have been done correctly, but in there's no byte order mark present, and so the endianness of the encoding is debatable (this could in the other endianness be an encoding of the Unicode characters `0xa300`, `0x3100` and `0x3000`).

xx. UCS-4 is a fixed-width encoding;
true

xxi. UCS-4 allows encoding of texts made from arbitrary Unicode characters;
true

xxii. UTF-8 is a fixed-width encoding;
false

xxiii. UTF-8 allows encoding of texts made from arbitrary Unicode characters;
true

xxiv. the string `£10` is encoded in UTF-8 as `c2 a3 31 30`;
true