Unix Philosophy, Text Editors, IDEs

Comp Sci 1585
Data Structures Lab: Tools for Computer Scientists

Missouri S&T Computer Science
1. Unix philosophy

2. What is text?

3. X-forwarding

4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom

5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
   - Writing code
   - Qt creator
   - KDevelop
Excerpts from the Unix philosophy

- Write programs to handle text streams, because that is a universal interface.
- Combine “small, sharp tools” and the use of “common underlying format the line-oriented, plain text file” to accomplish larger tasks.
- Store data in flat text files

Another major tenet of the philosophy is to use plain text (i.e., human readable alphanumeric characters) rather than binary files (which are not fully human readable) to the extent possible for the inputs and outputs of programs and for configuration files. This is because plain text is a universal interface; that is, it can allow programs to easily interact with each other in the form of text outputs and inputs, in contrast to the difficulty that they would have if each used mutually incompatible binary formats and because such files can be easily interfaced with humans. The latter means that it is easy for humans to study, correct, improve and extend such files as well as to port (i.e., modify) them to new platforms (i.e., other combinations of operating systems and hardware).
Excerpts from the UNIX philosophy

Unix tradition strongly encourages writing programs that read and write simple, textual, stream-oriented, device-independent formats. Under classic Unix, as many programs as possible are written as simple filters, which take a simple text stream on input and process it into another simple text stream on output. Despite popular mythology, this practice is favored not because Unix programmers hate graphical user interfaces. It’s because if you don’t write programs that accept and emit simple text streams, it’s much more difficult to hook the programs together. Text streams are to Unix tools as messages are to objects in an object-oriented setting. The simplicity of the text-stream interface enforces the encapsulation of the tools. More elaborate forms of inter-process communication, such as remote procedure calls, show a tendency to involve programs with each others’ internals too much. To make programs composable, make them independent. A program on one end of a text stream should care as little as possible about the program on the other end. It should be made easy to replace one end with a completely different implementation without disturbing the other.
1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
     - Writing code
   - Qt creator
   - KDevelop
• ‘Plain text’ is ASCII-encoded data. (Or UTF-8 encoded data, or EBCDIC encoded data...) 
• Most programming languages and network protocols use plain text because it is easy for developers to understand. 
• Text editors are just tools for editing plain text files. You’ll be using one a lot, so you should know how to use your editor of choice.
## ASCII TABLE

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[NULL]</td>
<td>32</td>
<td>20</td>
<td>[SPACE]</td>
<td>64</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BELL]</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[BACKSPACE]</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[HORIZONTAL TAB]</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>[LINE FEED]</td>
<td>42</td>
<td>2A</td>
<td>+</td>
<td>74</td>
<td>4A</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[VERTICAL TAB]</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
<td>45</td>
<td>2D</td>
<td>.</td>
<td>77</td>
<td>4D</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
<td>46</td>
<td>2E</td>
<td>,</td>
<td>78</td>
<td>4E</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS Idle]</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[ENG OF TRANS. BLOCK]</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>[SUBSTITUTE]</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>[ESCAPE]</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>[FILE SEPARATOR]</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>[GROUP SEPARATOR]</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>[RECORD SEPARATOR]</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>[UNIT SEPARATOR]</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Unix philosophy

What is text?

### X-forwarding

Text editors

nano

Vim

Emacs

Kate

Atom

### Integrated Development Environments

Code::Blocks

Building / Running

Writing code

Qt creator

KDevelop

---

### ASCII

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BELL]</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[BACKSPACE]</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[HORIZONTAL TAB]</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>[LINE FEED]</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[VERTICAL TAB]</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS Idle]</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[ENG OF TRANS. BLOCK]</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>[SUBSTITUTE]</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>[ESCAPE]</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>[FILE SEPARATOR]</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>[GROUP SEPARATOR]</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>[RECORD SEPARATOR]</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>[UNIT SEPARATOR]</td>
</tr>
</tbody>
</table>

---

### Building / Running

Writing code

Qt creator

KDevelop

---

### Integrated Development Environments

Code::Blocks

Building / Running

Writing code

Qt creator

KDevelop

---

### Unix philosophy

What is text?
You are responsible for submitting all text and source files you will submit in the entire class encoded UTF-8, Unix delimited.
1 Unix philosophy
2 What is text?
3 X-forwarding
4 Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5 Integrated Development Environments
   - Code::Blocks
     - Building / Running
     - Writing code
   - Qt creator
   - KDevelop
X-forwarding

• X-windows is Linux’s system for displaying graphical programs.

• X server: Program that manages what is displayed on the screen.

• X client: Program that wants to display something.

• Windows: Run Xming to start the server, then use PuTTY like normal.

• Mac: Install X11.app, then use `$ ssh -X <hostname>`

• Linux: You are already running an X server! Just use `$ ssh -X <hostname>`

• Use `xeyes` to check if X forwarding is set up correctly.
1. Unix philosophy

2. What is text?

3. X-forwarding

4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom

5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
     - Writing code
   - Qt creator
   - KDevelop
1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
   - Building / Running
   - Writing code
   - Qt creator
   - KDevelop
First part of the command:

- ^- is **Ctrl**
- M- is **Alt**

Full command examples:

- “Write out” (save): ^- O
- Quit: ^- X
- Undo: M-U
- Redo: M-E

Most importantly:

- Help: ^- G
• **Buffer**: Opened file
• **Window**: Visual section displaying a buffer
• *(Atom borrowed this terminology from vim!)*
• **Composability**: You can connect simple commands together into complex ones
• $ vimtutor can teach you some basic vim stuff

**Tip**: $ :set mouse=a
Modes: Keys do different things in different modes

- **Normal**: Navigation/commands. \[\text{Esc}, \text{Ctrl} + c\]
- **Insert**: Writing text.
  - \(i\): Insert at cursor
  - \(I\): Insert at beginning of line
  - \(a\): Insert after cursor (append at cursor)
  - \(A\): Insert at end of line (Append to line)
  - \(o\): Insert on new line below cursor
  - \(O\): Insert on new line above cursor
  - \(c\): Change text at cursor
- **Replace**: Overwriting text. \(R\)
- **Visual**: Selecting text.
  - \(v\): Character select
  - \(V\): Line select
  - \(\text{Ctrl} + v\): Block select
Moving Around

Unix philosophy
What is text?
X-forwarding

Text editors
nano
Vim
Emacs
Kate
Atom

Integrated Development Environments
Code::Blocks
Building / Running
Writing code
Qt creator
KDevelop

Moving Around

• j/k/h/l: up/down/left/right
• ^/$: Beginning/end of line
• w/e/b: Next word/Next word end/Back one word
• %: Matching brace or parenthesis
• gg/G: Top/bottom of document

Tip: Repeat commands by specifying a number first: $ 2w
Tip: Compose with insert modes: $ c4w
• `:w` Save
• `:q` Quit
• `:wq/ZZ` Save and quit
• `u/Ctrl+r`: undo/redo

**Clipboards**
• `y/yy`: Copy/‘yank’ (line)
• `d/dd`: Cut/‘delete’ (line)
• `p/P`: Paste after cursor/Paste before cursor
• `"+y`: Yank to system clipboard

• `>/<`: Indent/Outdent
• `=`: Autoformat
Configuration:

- `$ .vimrc` in your home directory
- Example config on the website

Plugins:

- Airline: Pretty status bar
- Fugitive: Git integration
- Syntastic: Syntax checking

Links:

- Vim Cheat Sheet
- Another cheat sheet
1 Unix philosophy
2 What is text?
3 X-forwarding
4 Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5 Integrated Development Environments
   - Code::Blocks
   - Building / Running
   - Writing code
   - Qt creator
   - KDevelop
Emacs: GNU Emacs

Use `$ emacs -nw` to launch in the terminal (instead of a GUI)

- Frame: Everything you can see (you can have multiple frames!)
- Buffer, Window: As with vim and atom
- Instead of `Ctrl+n`, emacs writes this as `$ C-n`
- Meta key (written `$ M-x`): Use `Alt+x` or `Esc x`
- In Emacs: `$ C-h t` starts a tutorial
Unix
philosophy
What is text?
X-forwarding
Text editors
nano
Vim
Emacs
Kate
Atom
Integrated Development Environments
Code::Blocks
Building / Running
Writing code
Qt creator
KDevelop

Moving

- $ C-n / $ C-p  next/previous line
- $ C-f / $ C-b  forward/back one character
- $ M-f / $ M-b  forward/back one word
- $ C-a / $ C-e  Beginning/end of line
- $ C-v / $ M-v  Down/Up one page
Unix philosophy

What is text?

X-forwarding

Text editors
nano
Vim
Emacs
Kate
Atom

Integrated Development Environments
Code::Blocks
Building / Running
Writing code
Qt creator
KDevelop

---

Cut/Paste

- Deleting: Backspace(\$<DEL>/\$C-d
- Kill (cut) a word: \$M-<DEL>/\$M-d
- Kill to the end of a line: \$C-k
- Unkill (paste): \$C-y
- \$M-y to cycle through kills
• Saving a file:  $ C-x C-s
• Quitting:  $ C-x C-c
• Stop a command:  $ C-g or hit Esc three times
• Undo:  $ C-_-
• Windows:
  • Make 2 windows:  $ C-x 2
  • Jump between windows:  $ C-x o
• Links:
  • Reference Card
  • Emacs Tour
1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
   - Writing code
   - Qt creator
   - KDevelop
KDE Advanced Text Editor (Kate)

- Linux-only
- Can be used as a simple text editor with no learning curve
- Can also be used in Vi/m input mode
- Has many advanced features
- Like other KDE applications, is highly configurable
**Terminology**
- **Buffer**: Contents of an open file.
- **Pane**: A visual section of the editor. Holds buffers, status bars, etc.

**Shortcuts**
- \(\text{Ctrl} + \uparrow + o\): Show a directory in the tree view
- \(\text{Ctrl} + \uparrow + p\): Command palette you can search through
- Searching for files:
  - \(\text{Ctrl} + t\), \(\text{Ctrl} + p\): Find a file in the current open directories
  - \(\text{Ctrl} + b\): Jump to an open buffer

**Packages**
- **minimap**: Shows a scrollable overview of your file
- **autocomplete-clang**: C++ Autocompletion
- **vim-mode**: Some vim shortcuts

**Link (click following): Documentation**
Outline

1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
   - Building / Running
   - Writing code
   - Qt creator
   - KDevelop
What are Integrated Development Environments?

- Commonly referred to as an IDE
- Combines editor, project management, compiler, debugger, etc. into one program.
- Commonly used on Windows (and OS X, to a lesser extent).
- Usually feature code completion.
- Provide a build management system for complicated projects.
- Some come with graphical tools for building GUIs.
1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
     - Writing code
   - Qt creator
   - KDevelop
Building with CodeBlocks

- f9 Build and run
- Ctrl + f9 Build
- Ctrl + f10 Run
- Project » Build Options » Enable $ -Wall
Writing code with Code::Blocks

- **Ctrl + .** Go to function implementation.
- **Ctrl + ↑ + .** Go to function declaration.
- **Ctrl + space** Show completions.
- Right-click on a file and choose ‘Format this file’ to autoformat.
• Excellent general purpose C++ IDE that is also used for Qt-GUI development.

• Requires that you create a “project” rather than just running .cpp files quickly.

• Excellent debugger (GDB) integration and visualization (more to come later).
1. Unix philosophy
2. What is text?
3. X-forwarding
4. Text editors
   - nano
   - Vim
   - Emacs
   - Kate
   - Atom
5. Integrated Development Environments
   - Code::Blocks
     - Building / Running
     - Writing code
   - Qt creator
   - KDevelop
• IDE natively intended for C++/C, but that can be used for other languages like Python as well.
• Requires that you create a “project” rather than just running .cpp files quickly.
• Pretty good debugger (GDB) integration and visualization.