Software Development Using Open Source Development Tools

John F. Davis
davisjf@gmail.com
johndavi@us.ibm.com

Overview

This presentation demonstrates building software using open source tools. The goal is to give a brief overview of each tool, how to use it and helpful customizations settings. A linux device driver which manipulates LED's via the parallel port is used as an example software development project. User code to test the driver is written in Java using eclipse and netbeans.

So what tools are necessary for building software?

- Editor
- Diagramming Tool
- Compiler
- Build Utility
- Debugger
- Configuration Management Tool
- Index and Searching
- Automated Testing Tools

That’s a big list. Are we going to cover them all?

Proposed list of tools for discussion

- blackbox/screen/vim - A custom IDE!
- man pages
- ctags - an indexer
- id-utils - another indexer
- cscope/cxref - other indexer
- expect - an automated testing tool
- Eclipse and Netbeans

What's not covered (but we can if you wish)

- gdb/ddd
- cvs/subversion
- diff/patch
- Make/Ant
- Cruise Control

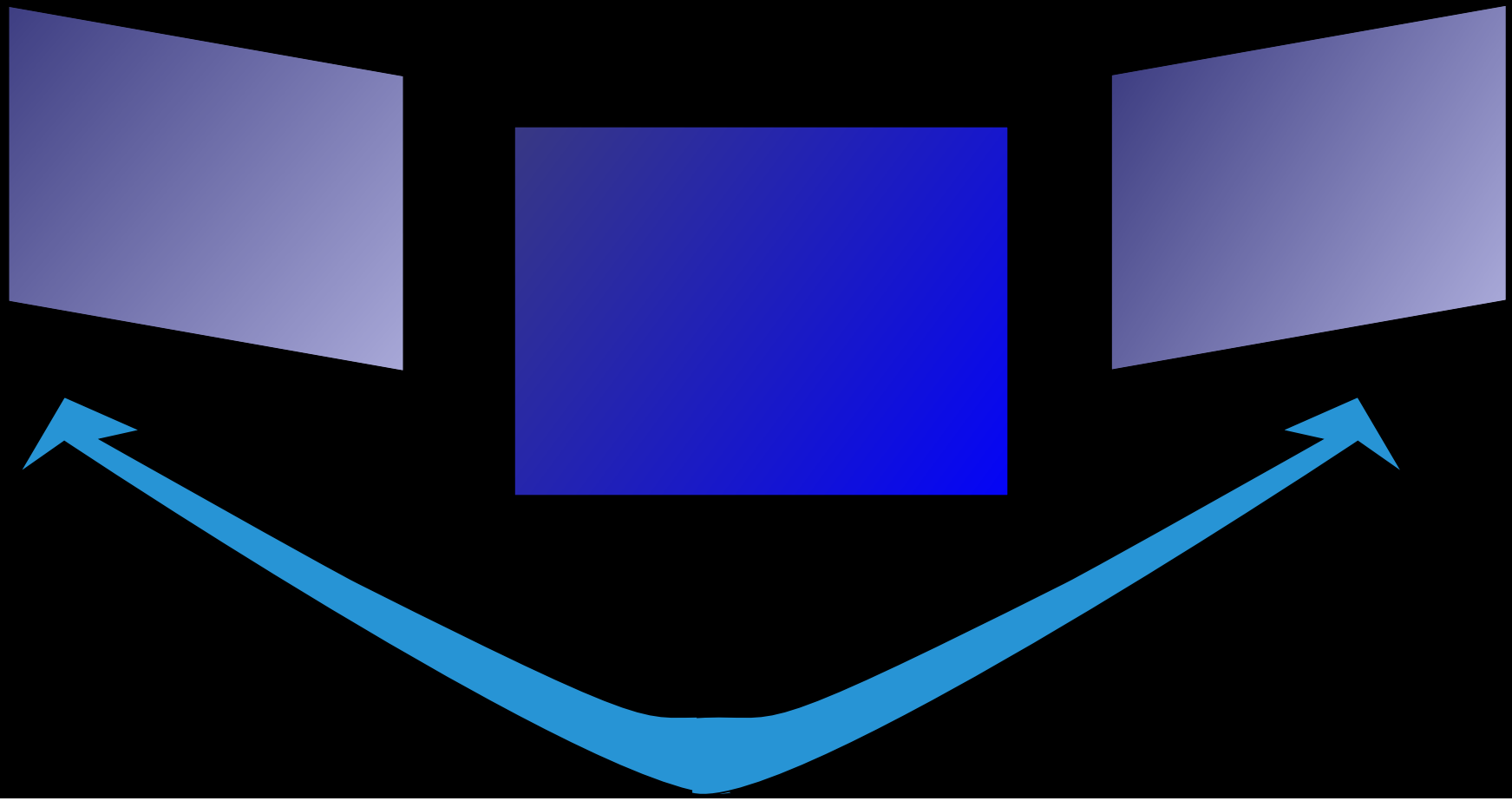
So are we just going to look at tools?

No. I'll introduce some of the tools in order to put them in context. Then I'll show how to use them while we add functionality to some sample code.

Part I - The tools

blackbox/screen/vim

- blackbox window manager (PROs)
  - lightweight on resources
  - maximal screen real estate
  - key bindings for rolling desktop (alt-arrow)
  - key bindings for switching applications (alt-tab)

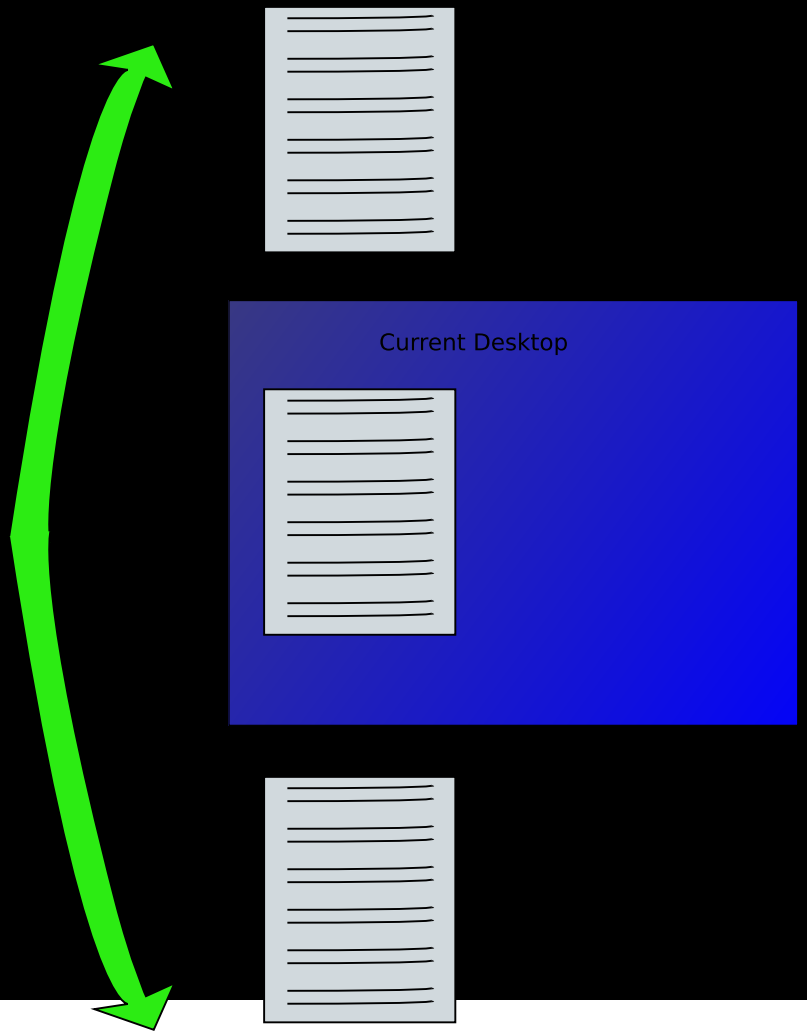


blackbox/screen/vim

- blackbox window manager (CONs)
  - Not as popular as other window managers.
  - Uses multiple config files which don’t necessarily inter-operate with each other. Case in point, switching desktops with keys.
    - ~/.blackboxrc
    - ~/.bbkeysrc
    - ~/.bbtools
    - ~/.blackbox
    - ~/.xsession
    - normal debian /usr/share/menu mods

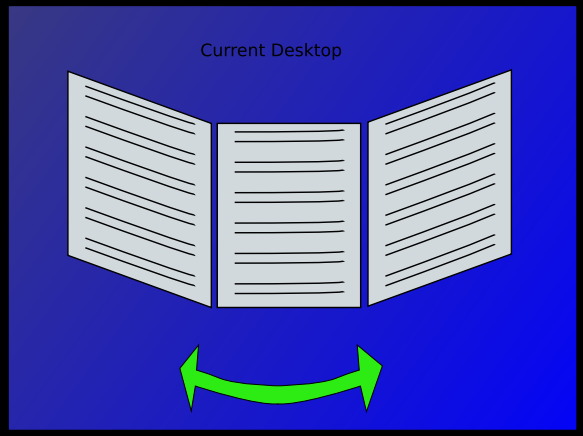
blackbox/screen/vim

- screen
  - key bindings for rolling terminal (ctrl-alt p/n)
  - key bindings for scrolling screen (ctrl-alt esc)
  - ability to suspend and resume sessions.



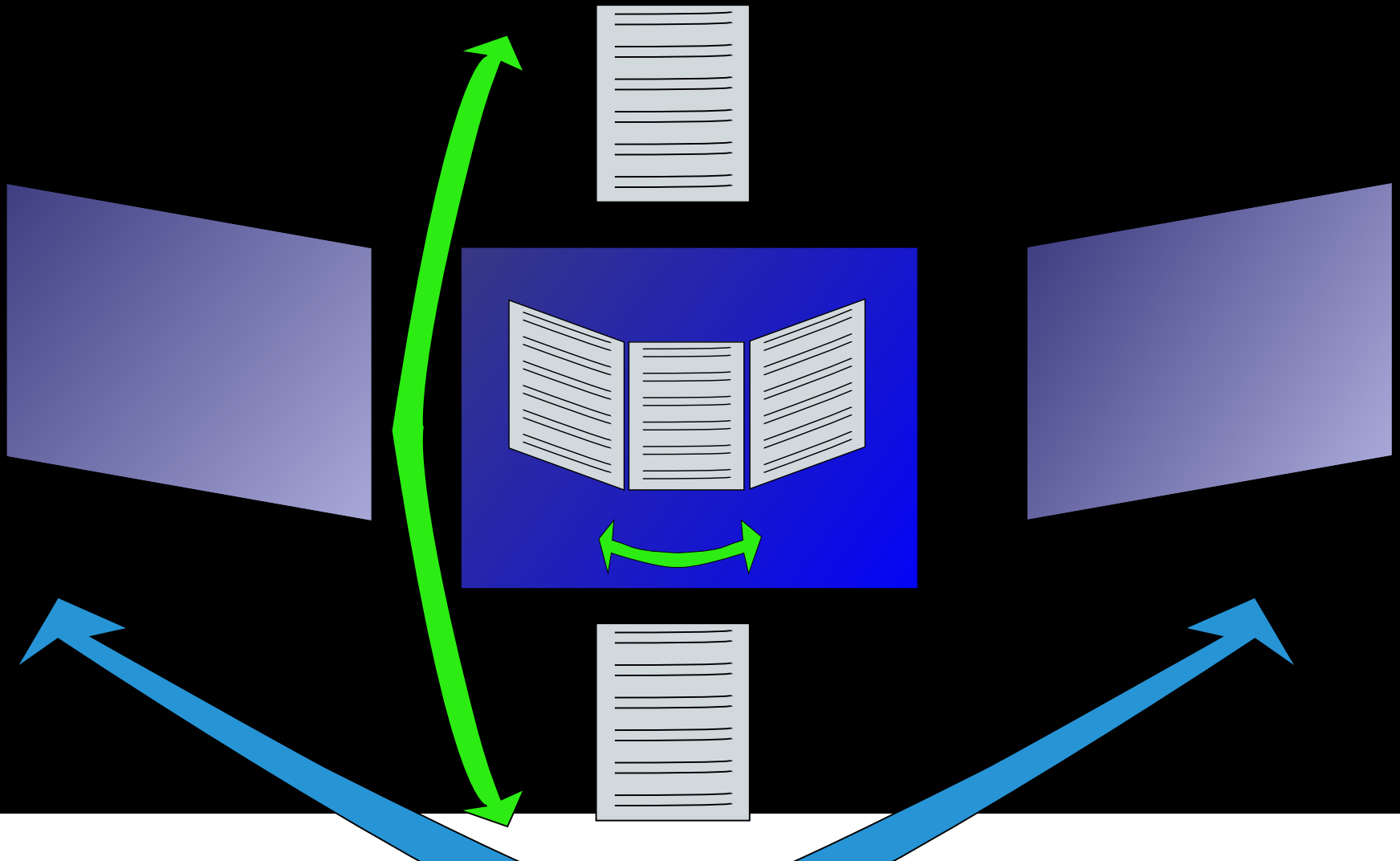
blackbox/screen/vim

- vim
  - key bindings for rolling sessions (ctrl-l/h)
  - syntax highlighting for all files
  - visual diff



blackbox/screen/vim

- All together now.
- Using only a few keys you can slide various views of work into place.



Manpages

- Everyone is familiar with man pages for unix tools, but are you aware that man pages exist for libc as well?

$ man 2 open

Ctags

- Works with VI or Emacs to lookup references in code
- Can specify #DEFINES when building Tag database
- Keybindings for vi (ctrl-] and ctrl-t)
- Vi also allows :3 tag foo (to select the 3rd instance of tag)

Ctags

- Example setup for linux kernel

```
ctags -V -R --exclude='*.o' --exclude='linux/include/asm-ppc/*' linux
```

Id-utils

- Fast lookup of symbols at command line
- Use id-utils instead of
  - grep -r or
  - find . -name "*.c" | xargs grep
- Has built-in tools for custom grep regular expressions

Id-utils

- Example setup for linux kernel

```
ctags -V -R --exclude='*.o' --exclude='linux/include/asm-ppc/*' linux
```

Id-utils

- Example usage
  - gid - like grep
  - defid - grep for symbol definition instead of all references
  - aid - gives list of symbols containing substring and the files where used.

Compare results of the following:

```
davis@bic:/usr/src$ gid request_region
davis@bic:/usr/src$ defid request_region
davis@bic:/usr/src$ aid request_region
```

Cscope and Cxref

- cscope is an interactive analyzer. Some people love it. KDE even builds an app around it. I don’t use it much, but its worth mentioning. I find it slow and the output to be dull.

- cxref builds a web page from your code. I used it on a project a long time ago. At that time it was ported to dos. Seems to be better at smaller projects. I don’t know how to generate a cxref dump for the linux kernel.

Expect

- A TCL based tool for automating user input. Great for adding automated testing to your builds. Especially if you use a host and target methodology and testing at system level.

- Demonstrate trilug.exp
- Demonstrate $? with a good username and a bad one.

Eclipse

- A uber IDE. Its "opensource enough" that its a Debian package.
- QNX uses it for their C bsp development. Windriver does likewise.
- It is possible to use external ant or makefiles to build your projects.
- It lacks a GUI builder though.

Netbeans

- Similar to eclipse. The GUI builder rocks. Not sure if it works with C or C++.

Part II - Using the tools

- Examine the driver source
- Show how to build it, install it and test it.
- demo building a quick java program with eclipse to test the parallel port.
- demo building a quick java gui with netbeans to test the parallel port.

backup

- Add read capability to the driver.
- Add ioctl control for blinking to the driver.
- Demonstrate/discuss additional tools.

Bonus Tool

This document was created using magicpoint.

http://www.mew.org/mgp