SWINBURNE UNIVERSITY OF TECHNOLOGY

FreeBSD Development for Smarties

The quest for a better kernel development environment

Lawrence Stewart

lastewart@swin.edu.au

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology

Outline

Detailed outline (section 1 of 6)

1 Getting started

- Who is this guy?
- Managing expectations

Who is this guy (and who let him past security)?

- BEng (Telecomms and Internet Technologies) 1st class honours / BSci (Comp Sci and Software Eng) (2001-2006)
- Centre for Advanced Internet Architectures, Swinburne University (2003-2007)
  - Research assistant/engineer during/after studies
  - `http://caia.swin.edu.au/`
- Currently a PhD candidate in telecomms eng at CAIA (2007-)
  - Main focus on transport protocols
  - `http://caia.swin.edu.au/cv/lstewart/`
- FreeBSD user since 2003, developer since 2008
  - Experimental research, software development, home networking, servers and personal desktops

Managing expectations

- Focus is breadth, not depth
- Minimal to no personal experience with many tools and strategies
- Targeted at beginner to intermediate level
- Input from experienced developers appreciated and welcome
- Goal is to eventually incorporate into official documentation sources

Detailed outline (section 2 of 6)

2 Hardware

- Dev server
- Test server(s)
- Remote access & management
- FreeBSD cluster
- Virtualisation

Dev Server

- All workspace state and source is maintained here
- Services and builds for test server(s) are run here
- x86_64, AMD-V/VT-x capable SMP
- SSD, 10k+ RPM (Raptor), RAID-0
- 4-8GB RAM helps buffer cache and ZFS ARC shine
- Dual NICs
- Dual serial ports (USB-to-serial adapters work well)
- Firewire (dcons - 32 bit only?)
- Server motherboards are good (e.g. Intel S3200SH)

Test Server(s)

- Test code execution, remote KGDB target, crash dump storage
- Isolated from dev server, mostly stateless, fast bringup
- x86_64 capable SMP
- Regular 7.2K RPM small HDD
- 1GB+ RAM
- GigE NIC with PXE
- Dual serial ports (USB-to-serial are **NO** good here)
- Firewire (dcons - 32 bit only?)
- Regular cheap desktop motherboards work well

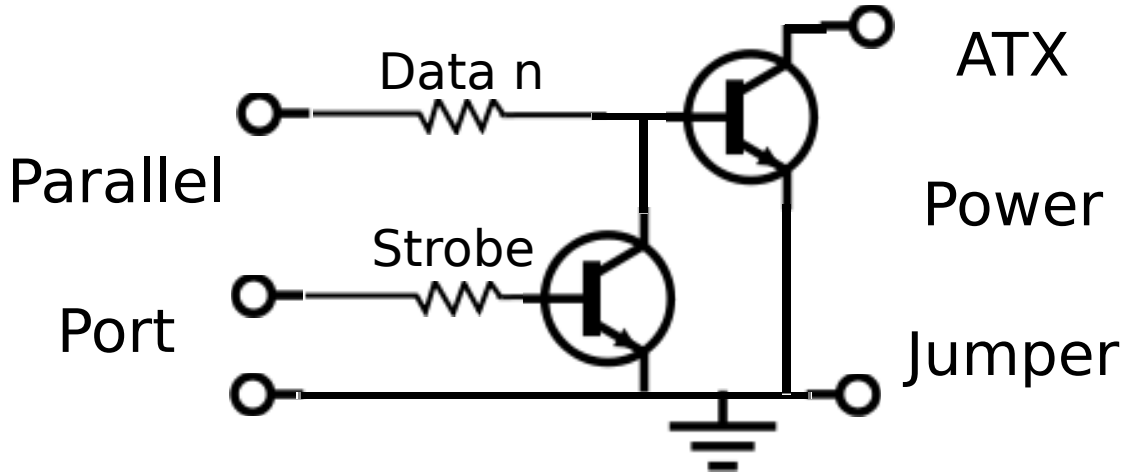
Remote Access and Management

- IPMI (ipmi.ko, sysutils/ipmi-tool port)
- Integrated Lights Out (ILO) et. al.
- IP KVM
- Power management strips
- Console servers
- All-in-one console power management (CPM)
- Homebrew LPT driven power management[1]

[1] Idea from Warren Harrop: `http://caia.swin.edu.au/cv/wharrop/`

Remote Access and Management

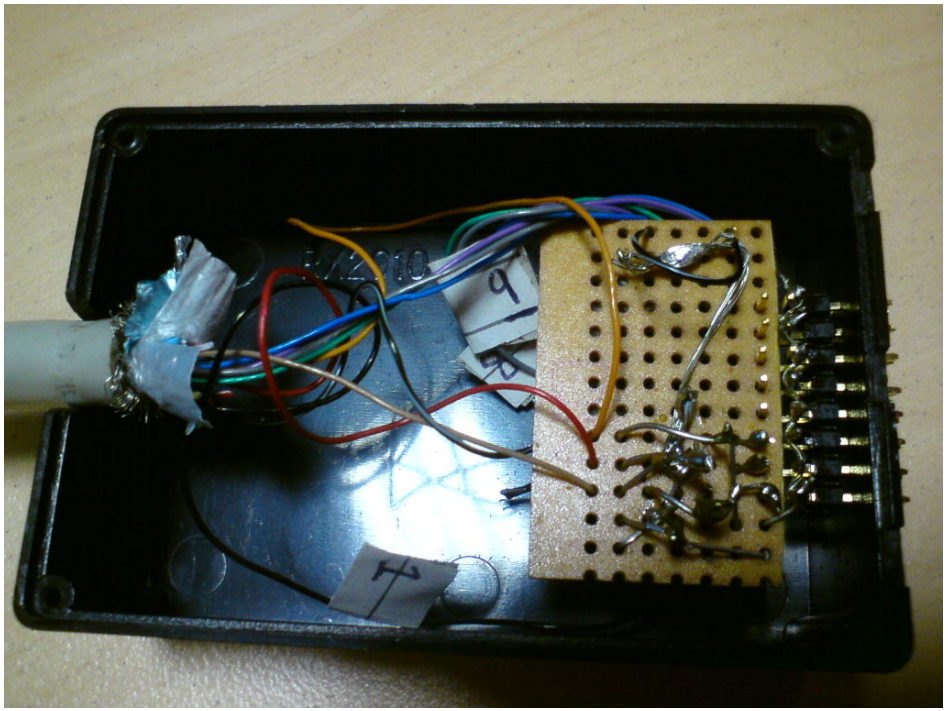
- Homebrew LPT driven power management[2]
  - Take a standard IEEE 1284 printer cable
  - Chop the female end off
  - Identify the 8 data line pairs
  - Wire like this:



  - Should use a capacitor to debounce the signal and photo diodes

[2] http://people.freebsd.org/~lstewart/misc/bsdcan2010/pushthebutton.c

Remote Access and Management



FreeBSD Cluster

- 10Gbps networking gear, range of hardware archs, large SMP systems
- Available for developers with commit bits
- `http://wiki.freebsd.org/NetperfClusterPointers`
- `http://wiki.freebsd.org/NetperfClusterReservations`

Virtualisation

- Hosted on FreeBSD: QEMU, VirtualBox, VIMAGE
- Hosted on other: VMware, Xen
- Useful features:
  - VM FS snapshots with rollback
  - Virtual serial ports for remote KGDB
  - pxeboot using host as dev server
- PITA issues:
  - Timing is bad except for VIMAGE
  - Host resources more contended, larger variances in service

Detailed outline (section 3 of 6)

3 Working with source code

- Revision Control
- Editors and Navigation

Revision Control

- Subversion[3]
  - Always use the devel/subversion-freebsd port
  - Set up autoprops in ~/.subversion/config correctly[4]
  - Committers will want to use ssh-agent[5]
  - setenv SVNBASE svn+ssh://<user>@svn.freebsd.org/base
  - Use branches in /projects for work destined for head
    e.g. svn cp $SVNBASE/head $SVNBASE/projects/tcp_cc_head
  - Use branches in /user/<username> for more speculative work
    e.g. svn mkdir $SVNBASE/user/<username> ; svn cp $SVNBASE/head $SVNBASE/user/lstewart/misc_head
  - Problematic for non-committers, external patch maintainers and long lived project branches

[3] http://wiki.freebsd.org/SubversionPrimer
[4] http://people.freebsd.org/~lstewart/misc/bsdcan2010/config
[5] http://people.freebsd.org/~lstewart/misc/bsdcan2010/sshagent.txt

Revision Control

- Perforce
  - Better merging capabilities for long lived project branches
  - Long-term contributors who are not yet committers can be granted access
  - Low visibility, centralised, minimal benefit over Subversion
- My current personal thoughts on DVCS
  - Should augment, not replace use of Subversion
  - Should run along side Perforce and probably replace it eventually
  - Greatly simplifies life for non-committers
  - Offline commit and log access
  - Lowers barrier to entry for getting involved
  - Would be useful for managing the ports tree
  - Boils down to Git vs Mercurial (I chose Mercurial[6])

[6] http://people.freebsd.org/~lstewart/misc/bsdcan2010/hg_notes.txt

Editors and Navigation

- VIM or EMACS: choose one, learn it well
- fxr.watson.org
- CSCOPE
  - cd /path/to/src/sys ; make cscope
  - VIM example[7]:
  - :cs add /path/to/src/sys/cscope.out /path/to/src/sys
  - :cs f g tcpcb
  - CTRL-t goes back through search sequence
- Glimpse
  - cd /path/to/src/sys ; make glimpse
  - Supposedly faster than CSCOPE for text searches

[7] http://cscope.sourceforge.net/cscope_vim_tutorial.html

Detailed outline (section 4 of 6)

Build System

- Useful top-level make targets: buildworld, buildkernel, universe, distribution, cleandir, buildenv, delete-old, delete-old-libs[8]
- Useful make options: DESTDIR, NO_CLEAN, KERNFAST, TARGET_ARCH, MAKEOBJDIRPREFIX, KERNCONF, NO_MODULES

[8] man build

Build System

- Typical pxebase bootstrap:
  - cd /path/to/tcp_cc_head/src
  - make -s -j<ncpus> KERNCONF=MYKERNEL buildworld buildkernel
  - mkdir /path/to/pxebase/tcp_cc_head
  - make -s DESTDIR=/path/to/pxebase/tcp_cc_head installworld installkernel distribution
  - mergemaster -iFD /path/to/pxebase/tcp_cc_head -m /path/to/tcp_cc_head/src
  - hint.uart.0.baud="115200", hint.uart.1.flags="0x80" in boot/device.hints
  - autoboot_delay="2", console="comconsole,vidconsole" in boot/loader.conf
  - Enable ttyu0 using vt100, std.115200 in etc/ttys
  - Mix of MFS, NFS and local HDD in etc/fstab [9]

[9] http://people.freebsd.org/~lstewart/misc/bsdcan2010/fstab

Build System

- Typical kernel development cycle (no dependency changes):
  - cd /path/to/tcp_cc_head/src
  - make -j<ncpu> -DKERNFAST KERNCONF=MYKERNEL buildkernel
  - make -s DESTDIR=/path/to/pxebase/tcp_cc_head installkernel
- Typical kernel development cycle (dependency changes):
  - cd /path/to/tcp_cc_head/src
  - make -j<ncpu> -DNO_CLEAN KERNCONF=MYKERNEL buildkernel
  - make -s DESTDIR=/path/to/pxebase/tcp_cc_head installkernel

Netbooting

- Simplifies and quickens development cycle
- Test server PXE boots and NFS loads kernel from dev server
- Optionally mounts 1 or more filesystems over NFS too
- Can cross compile for different test server architectures
- Ensures majority of state remains on dev server

Netbooting - Dev server

- TFTP
  - Enable tftp in /etc/inetd.conf
  - Enable inetd in /etc/rc.conf
  - Build pxe boot with serial console support[10]
  - Copy /path/to/obj/.../sys/boot/i386/pxeboot/pxeboot /path/to/tftp-root
  - tftp localhost and get pxeboot to test
- DHCP
  - Install a DHCP server (/usr/ports/net/isc-dhcp31-server)
  - Enable dhcpd in /etc/rc.conf
  - Create basic config and set “next-server” IP, “filename” boot-file and “root-path” in /usr/local/etc/dhcpd.conf

[10] http://jdc.parodius.com/freebsd/pxeboot_serial_install.html

Netbooting - Dev server

- NFS[11]
  - Export your NFS accessible directories with “-alldirs” in /etc/exports
  - Enable rpcbind, mountd (with -e flag), nfsv4_server, nfs_server, nfsuserd in /etc/rc.conf
  - mount_nfs localhost:/path/to/pxe/root /mnt to test

[11] man nfsv4

Netbooting - Test server

- BIOS
  - Enable boot from NIC
- Boot loader
  - If only loading kernel via PXE, set vfs.root.mountfrom="ufs:adXsYa"

Detailed outline (section 5 of 6)

1 Getting started

2 Hardware

3 Working with source code

4 Configuration

5 Testing & Debugging

6 Wrapping Up

5 Testing & Debugging

- Kernel debugging options
- Debugger basics
- Crash dumps
- Profiling & benchmarking

Kernel debugging options

- INVARIANTS, INVARIANT_SUPPORT (KASSERTs)
- WITNESS, WITNESS_SKIPSPIN (LORs)
- DEADLKRES
- KDB, DDB, GDB
- KTR, KTR_ALQ
- ALQ
- LOCK_PROFILING
- BREAK_TO_DEBUGGER, ALT_BREAK_TO_DEBUGGER
- makeoptions DEBUG=-g
- Few more...

Debuggers

- DDB vs KGDB[12]
- DDB:
  - Integrated into kernel, easily extensible
  - NMI (via ipmi tool or physical button on motherboard)
  - (ALT_)BREAK_TO_DEBUGGER
  - sysctl debug.kdb e.g. debug.kdb.panic=1
  - Switch to kgdb using “gdb” followed by “step”
- KGDB:
  - “remotebaud 115200” in /.gdbinit
  - gdb protocol over serial to test server
  - kgdb -r /dev/cuaU0 /path/to/debug/kernel
  - Can map frames to source lines

[12] http://www.bsdcan.org/2008/schedule/events/70.en.html

Crash dumps

- Set dumpdev="AUTO" in /etc/rc.conf or use “dumpon” to manually set the crashdump device
- From DDB: “call doadump”, “reset”
- Inspect a crash dump: kgdb /path/to/kernel /path/to/vmcore.X
- Many tools work on crash dumps too e.g ps, netstat
- Textdumps[13]
- `http://www.freebsd.org/doc/en/books/developers-handbook/kerneldebug.html`

[13] man textdump

Profiling & benchmarking

- Profiling and analysis tools: HWPMC, pmcannotate, DTRACE
- Stess testing: regression test suite, stress2, buildworld (!)
- Benchmarking is HARD to do well[14]
  - Identify and characterise all sources of error
  - Deeply understand your assumptions and assertions
  - ministat, killing unnecessary PIDs[15]
  - Draw careful conclusions based on statistically significant evidence

[14] http://www.eecs.harvard.edu/~margo/papers/freenix03/

[15] http://www.freebsd.org/doc/en/books/developers-handbook/testing.html

Detailed outline (section 6 of 6)

6 Wrapping Up

- Share the love
- Useful links
- Acknowledgements
- Questions

Share the love

- Did I miss something useful/important?
- Got a useful tip/trick that's not common knowledge?
- Let me know and I'll continue to collate all the information

lastewart@swin.edu.au, lstewart@freebsd.org

Useful links

- http://lists.freebsd.org/mailman/listinfo
- http://wiki.freebsd.org/
- http://forums.freebsd.org/
- http://caia.swin.edu.au/urp/newtcp/papers.html
- http://www.lemis.com/grog/Papers/Debug-tutorial/tutorial.pdf
- http://people.freebsd.org/~lstewart/articles/cpumemory.pdf

Acknowledgements

- The FreeBSD Foundation

- Dan Langille, et. al.
- FreeBSD community

Fin

Questions?