musings on software engineering

Andrew Tridgell
tridge@osdl.org

(Please ask questions during the talk)

An accidental linux box ...

NETGEAR®

This product includes software code developed by third parties, including software code subject to the GNU General Public License ("GPL") or GNU Lesser General Public License ("LGPL"). As applicable, the terms of the GPL and LGPL, and information on obtaining access to the GPL Code and LGPL Code used in this product, are available to you at http://kbserver.netgear.com/kb_web_files/open_src.asp. The GPL Code and LGPL Code used in this product is distributed WITHOUT ANY WARRANTY and is subject to the copyrights of one or more authors. For details, see the GPL Code and LGPL Code for this product and the terms of the GPL and LGPL.

203-10039-01

Software Engineering Progress

- What has been happening in practical software engieering?
  - 10 years ago, I think that most free software projects were behind most proprietary projects in terms of software engineering techniques
  - now, I think that free software is leading the way

New tools - new approaches

- The last few years has seen a new emphasis on good software engineering techniques
  - talked about for a long time, but now being widely adopted
  - a combination of tools and techniques
- Types of tools
  - static analysis
  - runtime analysis and simulation
  - memory management tools
  - code generation
  - much improved infrastructure libraries

Static Analysis

- Static code analysis has been around for a long time.
    - Lint has been around for well over 20 years!
- Now starting to be applied much more widely
    - advanced gcc warnings
    - 'sparse' analysis for kernel
    - stanford checker
    - findstatic.pl, minimal_includes.pl in Samba

Runtime analysis

- Runtime analysis also has a long history
    - when was the first profiler written? The first runtime heap checker?
- Recent advances have revolutionised runtime analysis
    - valgrind - perhaps the most important advance in recent years
    - tracing infrastructures

Code Generation

- Code generation has had a huge impact on the design of Samba4
  - A bit over 50% of code in Samba4 is now auto-generated
  - Mostly based on IDL, using pidl
  - some based on swig, for python bindings
- Could this be generalised?
  - more general code generators also become more unwieldy
  - different projects have quite different generator requirements
  - IDL compilers - Samba vs Wine vs Ethereal

Non-traditional IDL

- Initial motivation for IDL was for DCE/RPC
  - DCE/RPC is natually IDL based
  - code generation rules are quite simple
- Once we got the IDL bug ...
  - used for on-disk xattr format
  - used for NBT/WINS packets
  - used for datagram and mailslot code
- Lots of extension to IDL
  - non NDR formats, alignment rules, sub-contexts etc

The async problem

- Projects that implement network protocols tend to start out as “do one thing at a time” systems
  - very simple to code and understand
  - can suffer very badly from latency problems
  - some environments exacerbate this - like HSM
- There are three commonly used solutions
  - use threads
  - break up into separate processes
  - use a state machine

Threads are evil

processes are ugly

state machines send you mad

Samba4 - choose your own combination of evil, ugly and mad

A events/async framework

- We eventually settled on a system that provides flexibility while (hopefully) maintaining sanity
  - runtime chosen process models
  - a sane events framework
  - composite functions for taming state machines

Looking for an embedded solution?

- The end result is we can have a single non-blocking process that has
  - a ldap server
  - a CIFS server
  - a NBT server
  - a dgram server
  - a rpc server
  - and soon .... a web server

Composite Functions

- Composite functions keep us (relatively) sane
  - build higher level async functions out of lower level functions
  - a single coherent framework for state machine handling
  - sane error handling (thanks to talloc)
- Allows linkage between protocol subsystems
  - composite 'connect' does ...
    - DNS and NBT and /etc/hosts
    - SMB connection
    - spnego, NTLM etc
    - all in parallel!

Memory Management

- Memory management has always been a key problem is software engineering
- One solution - a new language?
    - Java, mono, python etc
- Another common solution has been pool based memory managers
    - apache runtime, old talloc
    - these help, but memory management is still painful

talloc - sane memory handling in C

- Pool based memory managers have been around for a long time
  - get a handle, alloc some chunks on that handle, destroy the handle
- Last year we noticed 'halloc' on freshmeat.net
  - hierarchical pool allocation? Could this be useful?
- Revolution!
  - new talloc implemented
  - destructors allow huge lumps of complex code to be removed
  - easy integration with existing resources

Code Coverage

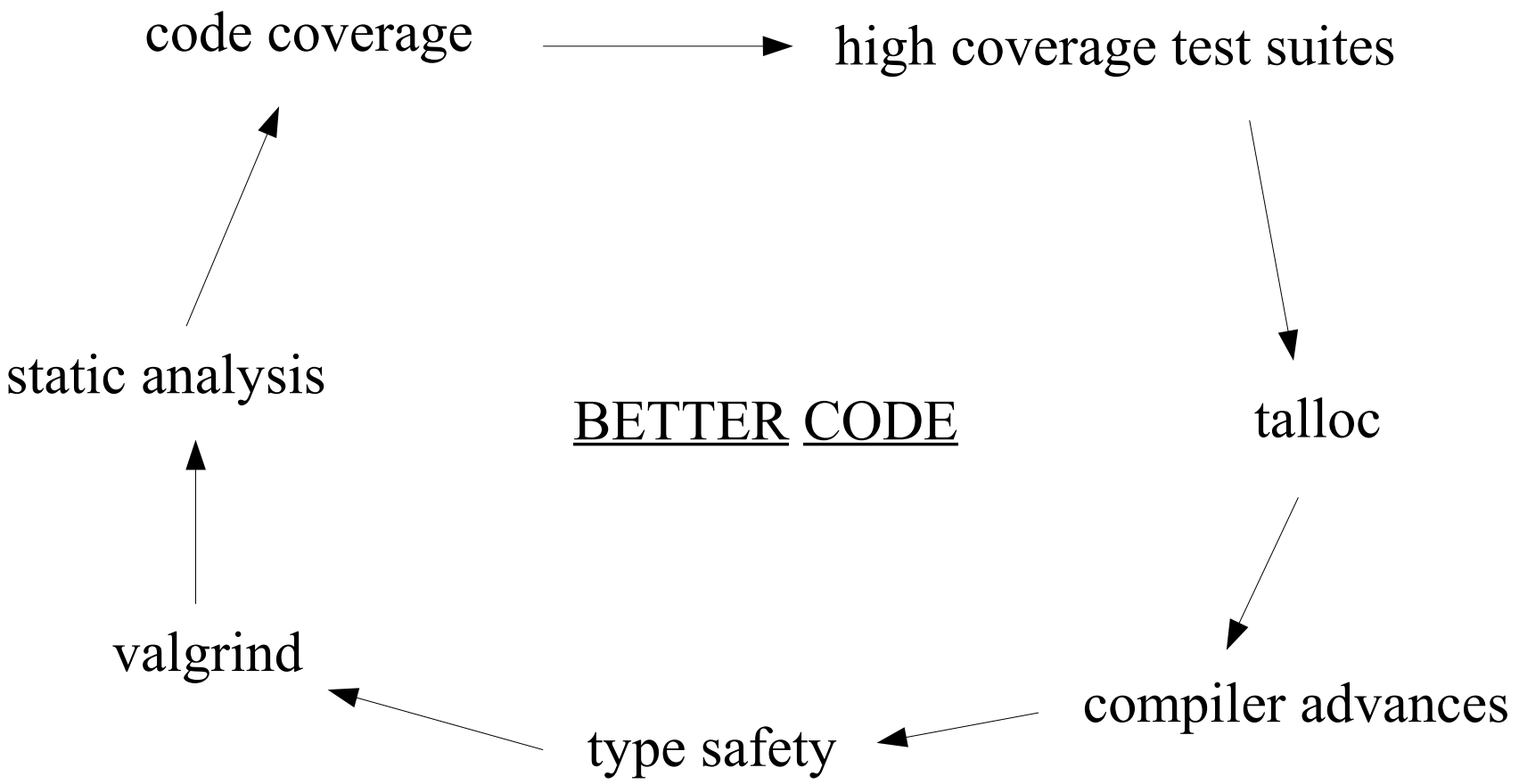
- Code coverage tools have been around for ages
  - like most people, I ignored them
  - started playing with gcov for small bits of code
- Eureka!
  - code coverage + good test suites + valgrind == wonderful!

The usability problem

- Oh the embarrassment!
  - ADS - Microsoft has taken traditional unix subsystems and integrated them better than the originals
    - krb5 + DNS + LDAP + RPC + CIFS
- For Samba, the problem was LDAP
  - OpenLDAP + SSL + SASL + krb5 == nightmare
  - How important is the standards focus of OpenLDAP for Samba?
- The solution was ldb

ldb - sane but minimal 'LDAP'

- Is LDAP without a schema useful?
  - Yes!
  - An empty file is a valid database
  - for embedded LDAP, trusting the application is OK
- ldb grows up
  - schema module added
  - ldap protocol server added
  - still allows for sane 'no schema' embedded usage


code coverage
high coverage test suites
talloc
compiler advances
type safety
valgrind
static analysis
BETTER CODE

Life at the bleeding edge

- A new filesystem, a new way to lose your data!
  - STFS - a filesystem for high decibel storage systems
- What happens if .....
  - we run smbtorture CHARSET test against Win32 STFS
  - ... and it tests all UCS2 characters
  - ... including character 0x12F
  - ... and we mount the same filesystem via STFS for Linux
  - ... and we want to cleanup the test using
    `rm -rf testdir`

When all else fails ... brute force

- The LSA session key problem
  - on ncacn_np transport session key is obvious
  - on ncacn_ip_tcp what is the key??
- re-cast problem as a public challenge
  - clarifies problem a lot, and sometimes gives a solution
  - noticed that encryption was the same on all servers!
  - a fixed key?!?
- How far do changes propogate?
  - input appears to be in 7 byte blocks. What could that be?

smart brute force

- bitslice DES
  - 32 DES calls in parallel
  - full search would still take years
- Could the key be human readable?
  - search weighted by character frequency
  - attack 51 bits at a time
  - 2 days of CPU later solution is “SystemLibraryDTC”

Questions?