# Free Software Development

Andrew Tridgell
Samba Team

# The development model

- Most free software projects are meritocricies, where (at least in theory) the best idea wins
  - you must use technical arguments to make progress
  - you must understand the code and the problem deeply
  - the environment can be very harsh!
- There is a huge emphasis on testing, particularly automated testing
  - Often it is the best programmers who do the testing and write the test suites
  - Submitting untested code is a major mistake

# ... the development model

- There is also an emphasis on code quality
  - Saying “the code works” is not good enough. It must be “good” code, which means elegance, simplicity and above all ease of maintainence
  - Refactoring code to do the same thing but in a better way is encouraged, but the new code must be well tested. This means automated test suites are essential.
- Long term, but fast, development
  - Development is often very fast, but developers are usually involved for many years
  - You must be prepared to put a lot of time in

# How to be a systems programmer

- A “systems programmer” is someone who creates basic services, or interacts closely with the operating system. Samba developers are systems programmers.
- If you want to learn to be a systems programmer then you should follow these steps
  - learn the tools
  - build clones of common systems tools
  - read books on operating systems internals
  - learn C and at least one scripting language

# ... learn the tools

- There are a number of essential tools to systems programming
    - strace - to trace system calls
    - ltrace - to trace library calls
    - gdb - to debug and diagnose programs
    - valgrind - to find memory errors
- You should learn to run these tools as an expert and know what the output means
    - test them on common programs, plus on your own code

# ... read books on operating systems

- To be a good systems programmer you must understand the basics of operating systems
  - read a book on operating system design
    - even though much of the book might be wrong!
  - read the Linux kernel documentation, particular the description of the VM and filesystem interfaces
  - write simple programs to test your knowledge, and use tools like strace to watch the behaviour
  - read the POSIX or SUS documentation for important calls like open() and mmap()
  - you must understand the principles of locking and race conditions

# ... learn programming languages

- To be a systems programmer you must understand the C programming language in detail
- You should also learn at least one scripting language such as Perl or Python, and perhaps bourne shell scripting
- Learn what makes “good” and “bad” style in these languages
- Learn how to avoid common programming errors, such a buffer overruns

# How to join a project

- Joining a new project can be difficult!
  - pick a project that you like! You might be working on it for many years
  - read and experiment with the latest development code
  - read the mailing list, join the IRC channel
  - when you start to contribute, make sure your postings are accurate
  - read the answers you get carefully
  - expect harsh comments and initial rejection
  - consider starting as a “janitor”, working on trivial bugfixes and refactoring

# Why don't more chinese contribute to free software?

- Is there a problem? What is it?
  - maybe mailing lists are too harsh and rude?
  - maybe because of language difficulties?
  - maybe fear of criticism?
  - something else?
- What can be done to help?

# Random musings

- The mindcraft story
  - how the community responds to challenges
- The power of procrastination!
  - many free software projects get started as work avoidance
  - procrastination can lead to a new career!
- The barrier to entry curve
  - As a project ages, the quality requirements get much higher

# Questions?

- You can download these slides at:
  - http://samba.org/ftp/tridge/talks/development.pdf