

Allan Kelly

Porting

( the interactive version )

August 2001

[www.allankelly.net](http://www.allankelly.net)

# Who is Allan Kelly?

- Yet another Englishman in Silicon Valley
- Over 10 years of software development
- Independent contractor in the UK for 5 years
- Worked for Sema Group, BZW (Barclays Capital), Reuters, DIP, Nixdorf and more....
- Currently with PostX Corp of Cupertino working on secure e-mail systems

# What is Porting?

- Moving an application from one platform to another

# What is a Platform?

- OS
- Database
- API
- Third party library
- Compiler
- Language (computer)
- Language (human)
- Anything else....

# Ports seldom involves one parameter

- Don't have the luxury of *All other things being equal*
- Changing OS often means changing compiler e.g. Forte on Sun, GCC on Linux
- Change Database means an API change
- (One of our greatest advantages is keeping these the same e.g. GNU C++)

# So, Platform is vague....

- Deliberately leave the definition of platform vague so we can encompass more
- But.....
- OS ports and Database ports are the two big ones

# Break : why do we port?

- 3 types of software development houses:
  - Mass producer : 1 to many e.g. Microsoft
  - Few customers : 1 to few e.g. Dodge
  - Bespoke : 1 to 1 often inhouse e.g. Goldman Sachs, or bespoke developers e.g. Accenture
  - (Also body shops/consultancies but these work for one of the above)
- Libraries : unusual 1 to many case

# OS porting : Unices...

SunOS	Interactive Unix	Linux : RedHat, SUSE, Mandrake, Caldera, etc.
Solaris (Intel & Sparc)	SCO Unix	AUX (Apple)
Irix	UnixWare	OS-X (Apple again)
HP-UX	System V	Apollo
AIX (IBM & Bull)	BSD – original, Free, Net, Open	Sinix (Siemens Nixdorf)
Digital Unix / TruUnix	Ultrix	Data General, ICL, Acorn, Olivetti ....
VX-Works	NextStep / Mach	

# Microsoft & Windows.....

Windows (1)	NT 3.1 / 3.5	PC-DOS
Windows 286	NT 4.0	MS-DOS
Windows 3.x	Win2000	DR-DOS
Windows 95/98/ME	XP	Free DOS
PenWindows	CE 1 / 2 / 3	CP/M 86
	OS/2	GEM

# And the others....

PalmOS

BeOS

VMS

MacOS

OS-X

MVS

EPOC

OS/360

(Symbian)

Geo

OS/400

interactive

AmigaOS

NewOS

Multics

Plan-9

Spring

# Databases...

Oracle 7/8/9

PostGres

xBase

Sybase SQL  
Server

MySql

ISAM

Microsoft  
SQL Server

Informix (rip)

Versant

Ingres (rip)

Poet

Db2

InterBase

Access

# Microsoft $\Leftrightarrow$ Unix

- The most common type of port
- The one we are most likely to encounter
- Focus on this for the rest of the talk
- But not exclusively

# Cultural differences

## Microsoft world

- GUI tools
- IDE
- Client
- Crashes a lot
- Suites
- Get the job done
- Bills Gates & Steve Ballmer

## Unix world

- Command line tools
- Vi
- Server
- Runs forever
- Beards and sandals
- Obscure diversions
- Bill Joy & Richard Stallman

- Vast generalisations but some element of truth
- Lead to very different approaches to problems
- Technically the differences are much less, e.g. all platforms aren't really Posix compliant!

# C++ and porting

- C++ is portable
- But we could lapse back to C
- Or even K&R C
- ISO standard not implemented so what subset do we use?
- Comes down to compiler....

# Tools

- Tools play an important part in porting
- The three big ones
  - Source Code control
  - Make
  - Compiler

# Choice of Compiler

## Native compiler

- **It is available**
- Optimised
- Lowest common denominator
- What standards?

## Common compiler

- Is it available?
- Same support on all platforms
- Are all elements supported on all platforms?

# If you go for Native compiler....

- How low do you go?
- What libraries are you going to use?
- What tools can you use?
- No exceptions, no namespace, no STL, .....

# Mozilla coding guidelines for portability

See <http://www.mozilla.org/hacking/portable-cpp.html>

- Don't use C++ templates.
  - Don't use static constructors.
  - Don't use exceptions.
  - Don't use Run-time Type Information.
  - Don't use namespace facility.
  - Put a new line at end-of-file.
  - Always have a default constructor.
  - Don't put constructors in header files.
  - Don't use return statements that have an inline function in the return expression.
  - Don't use mutable.
- ... and lots more.....

# Common compiler....

- Will it support all your platforms?
- Common compiler usually means GNU
- Tools? Libraries?
- Still have differences in platforms e.g.  
endian

# Macros control

## **One macro does all**

- New platforms require revisiting all `#ifdef`'s
- Simpler to understand, test, debug
- Best for applications with only few platforms

## **One macro per feature**

- Complex make files
- Complex interactions
- Easier to add new platforms
- Harder to maintain
- Best for libraries

# Force the differences down....

- Higher levels should deal with application logic
- Lower levels with platform/compiler logic
- Keep the program flow obvious
- Keep the program readable
- Abstract away the differences
- Application code is just that: application code

# Physical design of application

- Classic three tier model....

Presentation

Application logic

Storage, utilities and interfacing

- Model appears again and again
- Usually see more layers today
- The importance of layering.....

# Layering....

- A layer is one or more modules
- Layer should be self contained
- Depends on layers below
- Is depended on from above
- Does not depend on layers above or equal
- Package modules as libraries: .dll, .so, .a, .lib
- Exposes a number of header files

# Layering is a way of looking at dependencies

- Dependencies must be controlled in any system
- Few dependencies the more solid the code - unlike a brick building
- Especially important when porting
- Good layering is modular code, high cohesion, low coupling

# Dependencies

- Shortest route to the top has the fewest dependencies
- What are dependencies?
  - Include files
  - Lower layers
  - Libraries

# Include files is where dependencies begin

- #include is evil
- #include is essential
- Only include what you need
- Include no more, no less
- Forward declarations can reduce includes

# Includes....

- Three types of include
  - System
  - Project
  - Local
- Can further split Project
- Include ordering
  - most specific first
  - least specific first

# Inline functions just say no!

- Inlines are for:
  - optimised code
  - libraries
  - templates (which usually libraries and when we have **export** things will change!)
- Premature optimisation can damage code
- Inlines increase dependencies
- Too many getter/setters indicate poor class design

# Shortest route to the top

- Applications are a series of modules
- Deliverables are usually multiple applications (i.e. a set of programs)
- Port the minimum set of modules to get something - anything! - running
- Proof of concept
- Find out what you face

# Technical differences

- Shared libraries : differ on compilers and platforms
  - Initialisation
  - Memory models
  - Linking models
- Endian-ness
  - Not really an issue in porting
  - Is an issue in data transfer: why not use XML?
- Configuration
  - Registry, files, XML

# More technical differences

- Resources
- Multi-threading
- Installation
- User access rights
- Scripts
- Process models
- Inter process communication.....

# Technical differences : inter-process communication

- IPC looks different at first but actually has more in common
- NT & Unix both have:
  - shared memory
  - pipes
  - TCP
  - semaphores

# Using C++ features

- Overloading functions
- Template code generation
- Standard library : increases abstraction
- Exception handling
  - helps force down differences
  - not all errors exist on all platforms so why check for them?
  - let the errors come to you!
  - difficult to retro-fit

# Keep doing it!

- Porting is an on going process
- You must now support it on all platforms
- Build early, build often, build always!
- Developers must be cross-platform
- May have platform experts

# Alternative approaches

- Separate source code trees : now you have two project!
- Branch and forget : assume you are never going back
- Emulation : OK in the short term or where performance is not an issue
- IBM “Affinity”
- Outsourcing : make it some elses problem!

# What I haven't mentioned...

- GUI porting
- Wide character issues
- Project structuring (see article 1)

Remember to put these slides on  
my web site.....

[www.allankelly.net](http://www.allankelly.net)