Porting ANSAware/RT to Windows NT

Ian Macmillan

iam@ansa.co.uk

ANSAware/RT

- *Provides applications with access to real-time OS facilities*
- *Facilities include:*
  - Extended tasking system using POSIX threads (p-threads)
  - Lightweight RPC protocol (TREX)
  - API extensions (scheduling entries, QoS objects)
  - PREPC extensions for explicit binding with QoS
- *Currently available for DEC Alpha running OSF/1*
  - Other ports include HP/RT and LynxOS
- *NT port started on 3.1 base, now 3.51 with Visual C++ 1.1*

NT Port - Design Decisions

- *Produce a p-thread emulation module*
- *Retain existing build procedure (imake and make)*
- *Use dynamic linking for capsule library (DLL)*
- *Use portable interfaces to OS (e.g. C library rather than WIN32)*
- *Build applications as ‘console’ processes rather than GUI*
- *Use windows socket library (winsock)*

P-thread Emulation

- *P-threads provide three scheduling policies:*
  - **Normal, FIFO and Round Robin**
- *and two priorities:*
  - **background and foreground**
- *NT provides four process classes*
  - **idle, normal, high and real-time**
- *and five thread priorities*
- *NT mutexes are recursive (unlike p-threads)*
- *NT mutexes are of signal or broadcast type (p-threads distinguish by operations)*

P-thread Emulation

- ***Implemented by C module using WIN32 interfaces***
- ***Mapping of scheduling policies is not obvious, initial mapping:***
  - **Normal -> Normal, FIFO -> read-time, RR -> high priority**
- ***Initial thread priority mapping:***
  - **Normal -> Normal, background -> idle, foreground -> high**
- ***Mutexes are currently recursive (different from p-threads)***
  - **this is handled by capsule library**
- ***Only mutex broadcast is supported (not signal)***
  - **transparent due to mode of use but may be performance implications**

Build Procedure

- *ANSAware/RT uses high level makefiles (imakefiles)*
  - pre-processed (by ‘imake’) before running ‘make’
- *NT has has a broadly compatible ‘make’ utility (nmake)*
- *‘Imake’ is supplied with ANSAware/RT and was ported to NT*
- *Difficulties included:*
  - Different tools (e.g. library building)
  - Embedded Unix shell script
  - No symbolic links on NT (used for debug versions)

Dynamically Linked Capsule Library

- ***Motivation:***
  - **Much reduced runfile size (e.g. trader 1.5Mb -> 435Kb)**
  - **Easier updating of libraries (no application recompilation)**
- ***Difficulties due to global and static data:***
  - **Need to explicitly identify routines and data to be exported**
  - **Cannot use address of exported variable in static initialisation**
  - **ANSAware/RT allows nucleus configuration through statics (e.g. number of tasks)**

Interfaces to the OS

- *In principle, use high level interfaces (e.g. stdio, string, etc.)*
  - rather than platform specific (WIN32)
- *NT does not provide Unix compatible directory routines*
- *C library insufficiently flexibile for capsule manipulation*
- *NT differs in handling of signals:*
  - Has subset of Unix signals (e.g. SIGINT but not SIGTERM)
  - Cannot use ‘kill’ to raise signal in another process
  - Capsule termination implemented through WIN32 generated ‘console event’

Conclusion

- *NT kernel facilities not significantly different from Unix*
  - Majority of source code needed no modifications
  - Most differences accomodated by existing platform dependent files
- *P-thread emulation approach successful*
- *Considerable build changes due to different tools and scripting*
- *DLL conversion made difficult by global and static data*
  - and lack of any coding conventions for identifying them