Data Sharing on Mac OS

Miro Jurišić <meeroh@mit.edu>

June 21, 2001

Why Data Sharing?

- Full user experience usually requires several components
- Implementation constraints might require helper components
- Components need to share data

Architecture of data sharing

- Data producers generate data
- Data consumers read and use the data
- Usually, data provider stores the data ("server")

Designing your data sharing

- Understand the data you need to share
- Create an API (even if a simple one)
- Choose an implementation to fit you runtime requirements

Obstacles

- Synchronization, preemption, and data coherency
- Calling Mach-O from CFM – Apple sample code
- Calling 68K from PPC (and vice versa) – Apple Technotes

Synchronization, preemption, and data coherency

- Preemptive scheduling vs. cooperative scheduling
- Cooperatively scheduled tasks yield to each other when they want to
- Preemptively scheduled tasks yield to each other when the system wants them to
- Preemptive tasks can introduce data coherency problems (cooperative tasks can too, but it's easier to fix)

Data (in)coherency

```
1:  Read x from memory
2:  Add 1 to x
3:  Write x to memory
```

Two tasks, A and B, run the code at the same time, preemptively scheduled. The initial value of x is 0.

A-1, A-2, $\langle$A is preempted$\rangle$, B-1, B-2, B-3, $\langle$B is preempted$\rangle$, A-3

Outcome: final value of x is 1!

Data coherency

- To achieve data coherency, you must avoid preemption
- Without preemption, many things don't work
- Therefore, avoid preemption as much as needed to avoid problems – and no more
- Understand scheduling in the code you are working with

Scheduling on Mac OS 9

- Applications cooperative to each other
- Iterrupt-time code preempts applications
- Deferred tasks do not preempt other deferred tasks
- Secondary interrupts do not preempt other secondary interrupts
- Other interrupt-time code preempts other interrupt-time code
- Multiprocessing threads preempt other MP threads, apps, and interrupt-time code

Scheduling on Mac OS X

- Applications preemt each other
- Threads (POSIX, Mach, MP) preempt each other
- Some Carbon callback preempt the main task *and* can preempt the main task

Synchronization APIs

- Driver Services
- Open Transport
- Multiprocessing
- POSIX semaphores

Sharing Mechanisms – Runtime Environments

- Interrupt time
- Standalone code
- 680x0 code
- InterfaceLib
- CarbonLib
- Mac OS X CFM
- Mac OS X Mach-O
- Mac OS X Classic

Gestalt

- Gestalt values vs. Gestalt callbacks
- Gestalt values: very simple, but also limited
- Gestalt callbacks: useful when data needs to be generated on demand
- Not very useful on Mac OS X

PPC Toolbox

- Harder to use than Gestalt (data provider runs at interrupt time)
- More powerful: can be used from interrupt time and standalone code
- Mac OS 8 and 9 only

CFM Shared Data

- All data consumers and producers link against a shared library
- Shared library has one copy of data shared among all apps
- Don't put code in the library with shared data
- Doesn't work on Mac OS X

Apple Events

- Extremely versatile
- Work on Mac OS 9 and X, can cross to Classic
- Can't use at interrupt time
- Hard to use in standalone code
- Performance can be problematic

Mac OS 9 File Mapping

- Map files to memory and access them directly
- Data persistent after data producers quit
- Only on 9.1, limited implementation

Mach IPC

- At the root of all inter-process communication on Mac OS X
- Very low-level
- Use higher-level APIs when possible
- Only on Mac OS X

Multiprocessing Queues

- Use them to communicate between different MP threads
- Only useful for inter-process communication on Mac OS 9
- Tricky to avoid deadlocks

Unix Pipes

- FIFO channel between two processes
- Use standard POSIX APIs to read and write
- Mac OS X only

Loopback Networking

- TCP/IP interface
- Data does not reach the network
- Use BSD, Carbon, or Cocoa networing APIs
- Can't cross Classic boundary

Component Manager

- Shared library architecture before CFM
- Not in Carbon, not on Mac OS X
- Similar to CFM shared data