Mac OS X 10.4 "Tiger"

What's New for UNIX Users?

General Highlights

* Pervasive Searching
* Automator
* VoiceOver
* Parental Controls
* SyncServices

New and Upgraded Apps

* Dashboard
* iChat AV conferencing
* Safari RSS
* QuickTime 7 with H.264
* Mail.app now uses SQLite

UNIX Highlights

* Filesystem fun (indexing and attributes)
* 64 bit libSystem
* Performance Performance Performance!
* Developer Tools update
* ASL "Apple System Logger"
* launchd "one daemon to rule them all"

Kernel

* fine grain locking SMP
* KPI work
  * FS locking is no longer per filesystem
* Improved Unix Conformance
* 64 bit userland support
* Performance

File Systems

* Extended attributes (POSIXy superset)
* EAs are emulated on non supporting FS types
* ACLs (favoring NT behavior)
* Higher level Spotlight search APIs
* UDF closer to 2.5
* HFS improved built-in de-fragmentation

File System commands

* cp, mv and friends are EA aware
* rsync requires the -E flag
* cvs is not EA aware

Networking

* Wide Area Bonjour using DNS updates
* Ethernet channel bonding/failover
* IPSec support for certificates
* Firewall logging, ipfw2 and IPv6 firewalling
* site to site VPN and support for Kerberos
* dummynet

Drivers

* Improved Power Management APIs
* 64 bit shimming for ABI reasons
* Adding a 802.11 family
* Support for persistent disk device nodes
* GPT support

Userland

* Perl 5.8.6
* Python 2.3.5
* Ruby 1.8.2
* Tcl 8.4
* Wait for the Q&A and I can check other tools.

Apple System Logger "ASL"

* A system database of log messages
* Easy searching
* Advanced pruning
* More flexable logging API
* Powerful "mixer" like filter control

Service Management in Mac OS X

Terminology

* Daemons
  A long running background processes
* Super-daemons
  A daemon that proxies some execution for other daemons
* Agents
  Daemons that operate during and only for a given login session
* Communication handle
  A Unix socket or Mach port

Assumptions

- Prior experience writing a daemon in the Mach or Unix world
- Familiarity with Mach IPC or Unix system calls

Introducing launchd

- launchd is all about background processes
  - Work directly on behalf of a user
  - Work indirectly on behalf of a user or users
  - You need to get your code running at some point in the system

What's Wrong With the Status Quo?

* Daemons deserve better treatment
* In both Unix and Mac OS, daemons were just processes which disassociated them from user input
* "Faceless background applications" in Mac OS 9 parlance

The Solution:

* A new super-daemon to manage them
  * Designed to do work for you
  * Designed to be flexible
  * Designed to support messaging and control

Launchd Is Open Source

* A critical Darwin component
  * We want all Unix daemons to adopt this technology
  * Open Sourcing it encourages adoption

What will be covered

* The issues that a modern daemon writer faces
* What launchd does
* What launchd doesn't do
* How to port an existing daemon to launchd
* How to write a savvy launchd daemon

Unix History

* inetd
  * Launches IP based daemons on demand at run-time
  * Assumes only one FD handle per daemon
* init
  * Maintains login daemons on ttys at run-time
* /etc/rc*
  * A shell script that runs to configure a Unix system. It often runs other scripts which in turn launch daemons
  * Poor support for run-time control
* cron/at/batch:
  * Time centric

Mach History

- mach_init
  - Launches daemons on demand based on Mach port IPC

Today's Problems

* Missing functionality:
  * Unix local domain socket support
  * File system based events to trigger a daemon launch
  * init and inetd don't support user supplied jobs
* Multiple event sources:
  * Networking daemons commonly listen on multiple ports these days
  * Some daemons use both Mach and Unix based IPC
  * Ultimately, time, file system, and IPC events need to be supported in the same "super-daemon"
* The ability to restart a daemon is important

The Future

- One daemon to rule them all
  - Support for transferable based event sources
    - Most file descriptors
    - Mach ports
  - Support for user supplied jobs

So What Does this Mean?

- Hopefully less work for you
  - No need to daemonize
    - fork() and have the parent exit()
    - setsid()
    - closing stray file descriptors
    - reopening stdio as /dev/null
    - etc.

Launch on Demand

* Helping you help us save system resources
  * We support keeping your communication handles alive even when you're not running
  * This saves system resources
  * It also improves the system boot-up speed

Parallel Load at Boot

- Making boot-up even more dynamic
  - We register all daemons configuration handles first
  - Then we let daemons run
  - This lets us remove the need for externally specified dependancies

User-Agents

* Users have their own special needs too!
  * Standardizes the way we launch user-agents
  * Allows us to launch them on demand too, thus improving login performance

Case Studies

- The real world is what matters
  - cupsd
    - Uses mach APIs to enable automatic restarting
    - with launchd, no extra code is needed
  - mDNSResponder
    - uses both Mach ports and Unix file descriptors
    - launchd handles both, nothing else does for launch-on-demand

More Case Studies

- User examples
  - ssh-agent
    - Complicated to automate the management of
    - Most users only want one per session
    - launchd makes this trivial with small modifications to ssh-agent

What Doesn't Launchd Do?

- Monitor non kernel fundamental event sources:
  - configd's database key/values
  - configd's events
  - NetInfo's database key/values
  - Bonjour service advertisements
  - IO Kit's namespace (which is built upon mach ports)
  - IO Kit events (e.g. power management)
  - etc.

Wait! Not XYZ?!?

* This is subject to change
* We have our own internal needs too

Porting

- The high level overview
  - A simple IPC API
  - A simple RTTI based object system to support message passing

The IPC API

- Kinda-sorta-CoreFoundation
  - So why not CF?
    - Portability
    - Mach port and file descriptor passing is not supported by CoreFoundation at the moment
  - All we need is RTTI, dictionaries and arrays

C APIs

```
#include <launch.h>

launch_data_t launch_msg(launch_data_t);

int launch_get_fd(void);
```

C API Semantics

* launch_data_t represents an object graph
* launch_msg() is a synchronous API for the common case
  * Returns NULL and sets errno on failure
* If you request asynchronous messages be sent back:
  * Call launch_msg(NULL) to get an asynchronous message
  * Keep calling until you get NULL back
  * If errno == 0, then no more asyncrhonous messages are available for reading

launch_data_t

* RTTI and container classes are fun!
  * Dictionaries
  * Arrays
  * File Descriptors
  * Mach Ports
  * Integers
  * Real numbers
  * Booleans
  * Strings
  * Opaque Data

launch_data_t APIs

- "Just enough for IPC, and no more"
  - Get/set operations for basic types
  - Dictionaries:
    - insert
    - lookup
    - remove
    - interate
  - Arrays:
    - set index
    - get index
    - get count

XML plist keys

* Label
* UserName
* GroupName
* ProgramName
* Root
* Umask
* WorkingDirectory
* ServiceDescription
* ProgramArguments
* EnvironmentVariables
* EventSources

What Are EventSources?

- Details, details, details...
  - How to setup a given mach port or socket
  - Who to connect to...
  - Where to listen...
  - etc.

XML plist → launch_data_t

* Data distillation
  * UserNames → UIDs
  * GroupNames → GIDs
  * "stuff" → file descriptors and mach ports

Example Messages

- Dictionaries where the key is the command
  - SubmitJob
  - RemoveJob
  - GetJobs
  - CheckIn
  - SetUserEnvironment
  - UnsetUserEnvironment
  - GetUserEnvironment

Rehash

- launchd is the future
  - Less work for you
    - pre-daemonized when main() is called
    - Just check-in and go
    - Automatic restarting
  - More flexibility in what event sources you react to
    - Multiple Unix file descriptors
    - Multiple Mach ports
  - User agents
    - A powerful concept for per session background processes

Demo

For More Information

* Apple's Open Source Web Site
  * http://developer.apple.com/darwin/

Q&A