

# *olpcfs*

*Redesigning the datastore  
from the ground up  
and top down*

C. Scott Ananian  
cscott@laptop.org



# *End-user goals*

- Support **journal** and **bulletin board** abstractions
- Provide **Bitfrost** P\_SF\_RUN and P\_SF\_CORE protections



# Journal: objects & actions

- Action view

The screenshot shows a journal application interface. At the top, there is a search bar labeled "Search my journal" and three filter dropdowns: "Anything", "Anyone", and "Anytime". Below the search bar is a "Type a title" input field and an "Add new entry" button. The main content area displays a list of journal entries, each starting with a right-pointing triangle icon. The entries are as follows:

- ▶ Added 8 objects to **My Homework** ▶ 5 minutes ago
- ▶ Took 4 photos of **Rain Forest** ▶ with yesterday
- ▶ Painted a picture of **Our school** ▶ with 2 days ago
- ▶ Read **History of Thailand** ▶ with 3 days ago
- ▶ Wrote **History of Thailand** ▶ 1 week ago
- ▶ Painted a picture of **My Cat** ▶ with 1 week ago
- ▶ Looked at **A Photo of my Cat** ▶ with 1 week, 2 days ago
- ▶ Wrote **History of Thailand** ▶ 1 week ago, 3 days ago
- ▶ Painted a picture of **Our school** ▶ with 1 week ago, 3 days ago
- ▶ Looked at **Uruguay - Wikipedia, the fr...pedia** ▶ with 1 week, 2 days ago



# Journal: objects & actions

- Object view

<input checked="" type="checkbox"/>	★	What	Who	When	
<input type="checkbox"/>	☆	<b>The Tortoise and the Hare</b>		1 week, 5 days ago	
<input type="checkbox"/>	☆	<b>Russian Tortoise - Wikipedi...edia</b>		1 week, 5 days ago	
<input type="checkbox"/>	★	<b>A photo of my cat</b>		1 week, 6 days ago	
<input type="checkbox"/>	★	<b>This fantastic story abou...wrote</b>	6	2 weeks ago	
<input type="checkbox"/>	☆	<b>image clipping</b>		2 weeks ago	
<input type="checkbox"/>	★	<b>Our school</b>		2 weeks ago	
<input type="checkbox"/>	★	<b>A movie of my family</b>		2 weeks, 1 day ago	
<input type="checkbox"/>	★	<b>Uruguay - Wikipedia, the fr...edia</b>		2 weeks, 3 days ago	
<input type="checkbox"/>	☆	<b>History of Uruguay</b>		2 weeks, 3 days ago	
<input type="checkbox"/>	☆	<b>My homework assignment</b>		3 weeks ago	



# Bitfrost

- P\_SF\_CORE: no system files may be modified
- P\_SF\_RUN: the “working copies” of the system can't be modified



# *Bitfrost: it gets interesting!*

- When #P\_SF\_RUN is disengaged...  
[i]nstead of loading the stored files directly, a COW (copy on write) image is constructed from them, and system files from that image are initialized as the running system. ... These modifications persist between boots, but only apply to the COW copies: the underlying system files remain untouched.



# *Bitfrost: turning P\_SF\_RUN back on.*

- If #P\_SF\_RUN is re-engaged after being disabled, the boot-time loading of system files changes again; the system files are loaded into memory directly with no intermediate COW image, and marked read-only.
- ***End result:***
  - Hacking is safe again!



# Design goals

- Filesystem w/ POSIX semantics
  - This is the codeword for “standard filesystem”. Windows, UNIX, and MacOS in various flavors have (more-or-less) POSIX-compliant filesystems.
  - Our first generation design was a “simple” proprietary wrapper: let's move forward!
  - Aim to provide **best possible** support for legacy applications





# Design goals

- Content-addressable
  - Lots of attempts out there to create global distributed filesystems with unified namespaces – *let's not try this!*
  - Local arrangement & organization of documents is up to the individual user; all we need is an opaque tag to call it by.
  - Commercial support: XAM/Honeycomb (Sun)/Jackrabbit (Apache), etc, etc.



# Design goals

- Versioned

- Support exploratory learning by always allowing user to undo his most recent mistake.
  - “Continuous” versioning.
  - Snapshots don't work for this.
- Groups of files may have independently modifiable *tagged versions* (“full persistence”)
  - Gives us our P\_SF\_CORE/P\_SF\_RUN support
  - Also very useful when importing collaborative work



# *The olpcfs filesystem*

- **Transparent versioning**
  - Reach back and study the past – then change it!
- **Rich metadata via POSIX xattrs**
  - Enhanced by mechanisms to treat metadata as 1st-class files
- **Integrated metadata indexing**
  - Unifies “Journal” and “files & folders” views

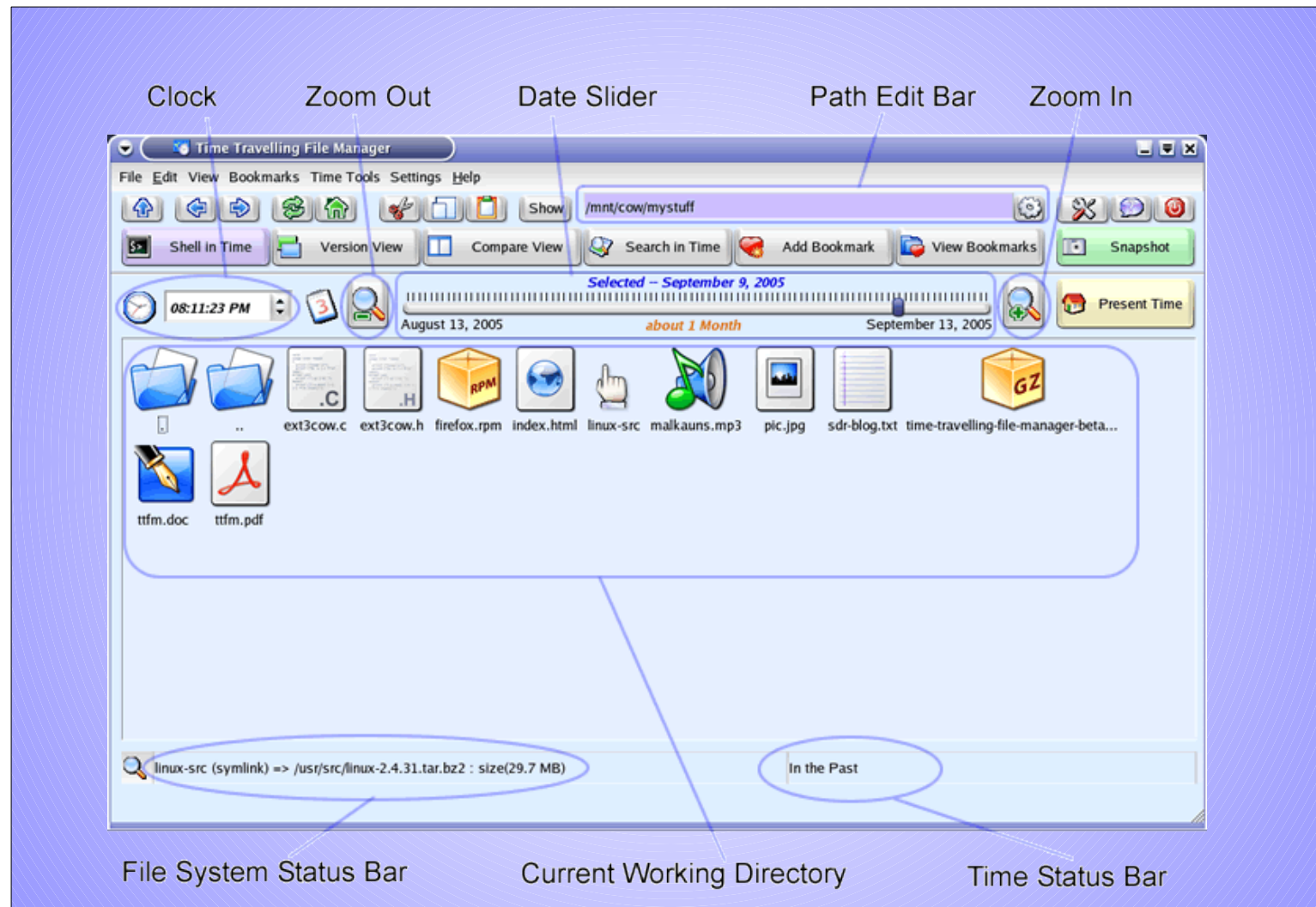


# Demo

- <http://wiki.laptop.org/go/Olpcfs>  
has pointers to the source
- 2,500 lines of Python
  - Bdb and python-fuse packages
- First impressions:
  - I prefer managing directory objects, rather than being given full pathnames



# Time-travelling file manager (an aside)



# *Journal integration*

- All documents live in  
~o1pc/Documents
- Sugar-aware activities add  
action\_id xattrs for file  
grouping
- Add'l journal properties are  
directly implemented as xattrs



# *Journal, cont.*

- Journal search built on native indexing
- Journal versions built on native versions
  - But additional attributes may be used for richer merge semantics, etc.
  - “Keep stars” in Journal correspond to landmark versions



# *Sync'ing & sharing*

- All objects have “XUID”
  - Content-addressable
- Distributed indexes of various scopes on top of local index
- Not all local objects may appear in filesystem tree!
  - Some may be imported into index only





# *Sync'ing & sharing*

- XUID encapsulates *object plus metadata*
  - “Who's got this XUID?”
  - “I'll tell you which XUIDs I don't have if you'll tell me your XUIDs.”
- Independently-modified documents may result in tagged versions in filesystem after import



# Implementation scope

- Lots of fallback/alternative implementations possible
  - Currently writing proof-of-concept to test APIs and unblock journal & other work
  - Non-versioned implementations with look-aside metadata easy using FUSE, lufs, 9P, etc.
- Flash filesystems are hard.
  - But the datastructures used here are very flash-friendly!



# Conclusions

- Even if it's not as ambitious as this, our datastore should look like a filesystem!
- Lots to learn from BeOS & the BSDs; they rock!
  - Even NTFS



# Questions?

