The operating system: should there be one?

Stephen Kell
srkell@acm.org

“Oracle Labs”
What time is it?
$ grep −r ’pattern’ $d

■ what kind of thing is $d?
■ what is grep -r going to do with it?
■ what stops it from generalising further?
$ grep -r ’pattern’ $d

- $d$ is a tree node directory
- grep -r will
  - recursively explore readdir()
  - on leaf nodes, read characters read bytes
  - if match, output a reference filename
- what stops it from generalising further?
  - choosing arbitrary concretions
  - ... which become commitments (“binding”!)
What else might we recursively pattern-search?

$ grep −r ’pattern’ /usr/share/doc/grep
$ ls /usr/share/doc/grep
AUTHORS NEWS.gz README THANKS.gz TODO.gz copyright

$ grep −r ’pattern’ imap://stephen@xylophone/
grep: imap://stephen@xylophone/: No such file or directory

What happened to compositionality?

■ hitting a limit of Unix (semantics-free) byte-streams
■ hitting a limit of Unix object model (binding model)
■ point fixes don’t scale; want a “system”!
“An operating system is a collection of things that don’t fit into a language. There shouldn’t be one.”

Specific grumbles:

- “depart from an otherwise consistent framework”
- “very primitive”
- memory, storage, display, input, debugging…
Things I want to convince you of

- a key part of OSes’ value is communication abstractions
- … which **cut across** languages
- → there **should be** an OS (sorry to disappoint)
- Unix is tending towards Smalltalk!
- … but its meta-abstractions are **lurking**
- by evolving Unix-like interfaces, we can harness them
Things I want to convince you of

- A key part of OSes’ value is communication abstractions

- … which cut across languages

- → there should be an OS (sorry to disappoint)

- Unix is tending towards Smalltalk!

- … but its meta-abstractions are lurking

- by evolving Unix-like interfaces, we can harness them
Smalltalk: a partisan summary

- a programming abstraction
  - objects
  - messages
- a *descriptive* abstraction a.k.a. metasystem
- late binding, pervasively
- uniformity
- big, well-designed libraries
- (... never full enough to replace OS)
What do you mean by “metasystem”? 

“system”

- a collection of behaviours modelling a domain

“metasystem”

- … modelling the domain of systems

For this talk

- machine-readable specification (classes are one facet)
- not metacircular implementation (CLOS-style)

Metasystems enable “value added” orthogonal to the domain

- persistence, REPL, debugging…
“Since we are programmers, we naturally designed the system to make it easy to write, test, and run programs.”

- programming abstractions (multiple)
- primitive meta-abstraction for I/O
- late binding in some cases

Missing:

- semantic meta-abstraction (cf. classes)
- uniformity . . .
Fragmentation in Unix

Unix has too many binding mechanisms!

- file descriptor
- named file (early or late)
- numbered process (signals, nice, ...)
- (from BSD 4.2) sockets...
- in-process linkage
- shared memory ×2...

Processes cannot fully emulate files

- even with pipes!
“build a Unix out of little systems, not a system out of little Unixes”

- Unix-inspired research OS from Bell Labs
- use filesystem abstraction pervasively
- most applications provide and consume “file servers”
- simplifies binding $\rightarrow$ greater compositionality!
- still missing full metasystem, large/small unification,

...
Object = file

File(server)s and objects are

- some subdivision of a computation
- implementing a basic protocol...
- consisting of messages (→ network-transparent)
- … onto which additional meaning can be layered

Newly missing

- documented semantics for files!
- e.g. control files
- what can I grep? cp -r?
From file to object

The usefulness of a metasystem

Semantics vs. Smalltalk

The operating system... – p.15/23
The Lurking Smalltalk

Messages / objects are ubiquitous, but also metasystems:

- /proc filesystem, vmmmap, …
- other synthetic filesystems (/sys, sometime /dev)
- DWARF debugging
- {symbol, version, type} info in code representations
- /etc/services
- filename extensions, MIME…
- HTTP Content-Encoding
- “any sufficiently complex system” (pacmd demo)
“build a Smalltalk out of lots of fragments, not a big fragmented system out of little [insert PL here]s”

Smalltalk is one view among many!

- languages appear to be more ephemeral than OSes
- language impls should be “a view on the wider world”

Join the dots to join the metasystems…

- via minor expansions to (Unix-like) OS interfaces?
- unify the metasystem, not the system
- if you can’t unify mechanisms, “multiply”
Concrete suggestion #1: a metasystem for large and small alike

Work-in-progress: pmirror

$ LD_PRELOAD=libpmirror.so ./mycmd

- knows Linux /proc, ELF, DWARF (more wanted)
- exposes an API for in(tro)spection

Goal

- explain any value (/byte) in any process on the system
- describe every IPC channel too

Challenges

- coverage
- “explain” at what level(s)?
Concrete suggestion #2: IPC provide/require metasystem (1)

```c
fd = open(leaf_path, mode);
// then read bytes; hope they're chars

dirfd = open(branch_path, mode);
// then read dirents; fail if not dir!
```
Concrete suggestion #2: IPC provide/require metasystem (1)

```c
fd = open(leaf_path, mode);
// then read bytes; hope they're chars

dirfd = open(branch_path, mode);
// then read dirents; fail if not dir!

open(leaf_path, flags, TEXT_UTF8);
// read bytes; declare that they *will be treated as* UTF-8

open(branch_path, flags, DIRECTORY);
// read dirents, encoding whatever the tree *happens* to be
```

What is the structure of the rightmost arguments?

- something like DWARF, but for files too
Unification vs “multification”

one way to unify:

The operating system... – p.20/23
Unification vs “multification”

one way to unify:

another way to get almost-the-same effect
Multifying IPC

fopen()
connect()
popen()
dlopen()
shmat()...

creat()
bind(); listen()
ld -o ...
ld -shared -o ...
shmget()...

The operating system... – p.21/23
Multifying IPC

- fopen()
- connect()
- popen()
- dlopen()
- shmat()...

- creat()
- bind(); listen()
- ld -o ...
- ld -shared -o ...
- shmget()...

The operating system... – p.21/23
Multifying IPC

fopen() → creat()
connect() → bind(); listen()
popen() → ld -o ... dlopen() → ld -shared -o ... shmat()... → shmget()...
Concrete suggestion #3

File / socket / program / library namespaces...

- ... are large/infinite
- → can be extended transparently

Needn’t be $n^2$ underneath

- everything is really a `dlopen()` (ask me)

Sensible behaviour requires meta-info on files/sockets/...

The operating system... – p.22/23
Conclusions

We can think of Smalltalk not as a grand design

- ... but as something we aspire to (re)discover
- including in Unix

Compositionality means infrastructure for talking to aliens

- describe I/O as well as memory
- multify mechanisms
- integrate programmatically ask me!

Please ask questions!