Python usage in Samba and OpenChange

Jelmer Vernooij

Samba OpenChange

PUN December 7, 2008

Agenda

What is Samba ?

- Free (GPLv3) implementation of the SMB protocol and others
  - DCE/RPC, NetBIOS, LDAP, CLDAP, Kerberos, . . .
- “Network Neighborhood” (and more) for POSIX
- Traditionally written in C
- Extremely portable
- About the same age as Python (’91)
- Developed by a team of 25 people
- Originally developed thru network analysis

What is Samba ? - Branches

- Samba 3
  - Proven, stable, codebase
- Samba 4
  - Our very own “Duke Nukem Forever”
  - Originally started in 2003, as an effort to improve the file server
  - Strong focus on the right infrastructure

What is OpenChange ?

- Free implementation of the MAPI protocol
  - As used by Microsoft Outlook/Exchange
  - Works on top of the DCE/RPC protocol
- Being build on top of Samba 4
- French project, started in 2003
- Evolution OpenChange plugin will ship with GNOME 2.26

Who am I?

- CS Student
- FOSS developer, working on:
  - Samba, mostly Samba 4
  - OpenChange
  - Bazaar

Samba and scripting

- We are all C programmers
- Scripting: awk, shell or perl
- Samba had Python bindings for a while
  - Removed after several years because they were unmaintained

Why scripting?

- Within Samba
  - Quicker development
  - Easier to debug
  - Easier to understand
    - Lower barrier for contributions?
- For users
  - Easier to customize
  - Easier to use (administrator scripts, etc)

And the winner is...

- Original choice: JavaScript
- Small, fast engine (small enough to include with Samba)
- Familiar to a lot of developers out there
- A lot like C (familiar to developers)

JavaScript: Problems

- A lot like C
- Sucks as a scripting language
  - No exceptions
  - Poor string manipulation functions
  - No keyword arguments
- No bindings for standard libraries
- The library we were using was different from the standard library
  - Contributors had to learn yet another language (dialect)
- No development tools
- Hard to write unit tests

Why Python?

- Comes “with batteries included”
  - No need to reimplement utility functions and bindings for Samba
- Easy to create bindings
- Most existing libraries already have Python bindings
  - GTK+, Qt, HTTP, .ini-parsers...
- Large existing developer base
  - Potential contributors
- Better scripting language
  - Nested functions
  - Modularity
- More development tools available
  - Debugger, profiler, code coverage analyser, ...

Why not something else?

- Several developers already knew (and liked) Python
- Alternatives
  - Perl: Hard to use C API, silly syntax
  - Ruby: Not well known enough
  - Lua: Not really well suited for application development, just small snippets
  - Scheme: well, . . .

Concerns

- Our first ever mandatory build-dependency other than libc...
- A lot of code to migrate, requiring effort that could be useful elsewhere
- Maintainability
  - Unit tests mandatory
  - Should be used by core code

SWIG

- Doesn't require a lot of effort to generate simple bindings
- Can generate bindings for multiple languages at the same time
- Generates portable code

However...

SWIG

- Doesn't require a lot of effort to generate simple bindings
- Can generate bindings for multiple languages at the same time
- Generates portable code

However...

- Tends to create very C-like Python bindings

SWIG

- Doesn't require a lot of effort to generate simple bindings
- Can generate bindings for multiple languages at the same time
- Generates portable code

However...

- Tends to create very C-like Python bindings
- Customization language is hard to grasp

SWIG

- Doesn't require a lot of effort to generate simple bindings
- Can generate bindings for multiple languages at the same time
- Generates portable code

However...

- Tends to create very C-like Python bindings
- Customization language is hard to grasp
- Unreadable generated C code

Pyrex/Cython

- Very Python-like

But...

Pyrex/Cython

- Very Python-like

But...

- Needs to be run on the developer machine (extra build-dependency)

Pyrex/Cython

- Very Python-like

But...

- Needs to be run on the developer machine (extra build-dependency)
- Doesn't support certain

Pyrex/Cython

- Very Python-like

But...

- Needs to be run on the developer machine (extra build-dependency)
- Doesn't support certain
- Unreadable generated C code

Manual bindings

- The Python C API really isn't that bad
- Allows close integration between our memory manager and Pythons
- Much more flexible than autogenerated Python bindings

Partially generated from IDL

IDL Code

```
NTSTATUS unixinfo_GetPWUid (
[in,out,ref,range(0,1023)] uint32 *count,
[in,size_is(*count)] hyper uids[],
[out,size_is(*count)] unixinfo_GetPWUidInfo infos[*]
);
```

Python API

S.GetPWUid(uids) $->$ infos

Current state of affairs

- Mostly used for administrative tools:
  - provisioning the databases after installation
  - web service? (wsgi compatible)
  - Server functionality and performance-dependent
  - Some nifty GUI tools based on GTK+
- Popular for writing tests
- Performance-dependent code is still all in C

And most importantly:

- Developers seem reasonably happy

More crazy Samba Python hacks

- Full Python coverage of our libraries
- More GNOME integration in Python
- win32com on Linux?
- Port to Samba 3?
- Server partially in Python?

OpenChange

- Provisioning already uses Python
- Most client tools will be in Python
- Bindings still to be done

Remaining and new concerns

- No good standard mechanism for asynchronous functions (yet?)
- Some users are running Python older than 2.4
- Python3000 will drop support for some of our platforms
- Supporting all combinations of platforms with Python installed turned out to be quite a challenge

Reading TDB files

```
1  import tdb, sys
2
3  db = tdb.Tdb(sys.argv[1])
4  for (k, v) in db.items():
5      print "{"
6      print "key(%d) = %r" % (len(k), k)
7      print "data(%d) = %r" % (len(v), v)
8      print "}"
```

Using LDB

```
1  #!/usr/bin/python
2
3  import ldb
4
5  conn = ldb.Ldb("msg.tdb")
6
7  conn.add({"dn": "dc=samba,dc=org", "attr1": "foo"})
8
9  for msg in conn.search("dc=samba,dc=org"):
10     print str(msg.dn)
```

Connecting to LDAP using LDB

```
1  #!/usr/bin/python
2
3  import ldb
4
5  # Connect to the LDAP server
6  conn = ldb.Ldb("ldap://ldap.abmas.org/")
7
8  for msg in conn.search("dc=samba,dc=org"):
9      print str(msg.dn)
```

Adding users

```
1  #!/usr/bin/python
2  import samr, lsa
3
4  # Connect to the local SAM
5  conn = samr.samr("ncalrpc:", "st/dc/etc/smb.conf")
6
7  # Get SAMR connect handle
8  samr_handle = conn.Connect(0, 0xffffffff)
9
10 domainname = lsa.String()
11 domainname.string = u"SAMBADOMAIN"
12
13 sid = conn.LookupDomain(samr_handle, domainname)
14 print "Found sid %s for SAMBADOMAIN" % sid
15
16 conn.Close(samr_handle)
```

Unit tests

```
1  import winreg
2  from samba.tests import RpcInterfaceTestCase
3
4  class WinregTests(RpcInterfaceTestCase):
5      def setUp(self):
6          self.conn = winreg.winreg("ncalrpc:", self.get_lo
7
8      def test_hklm(self):
9          handle = self.conn.OpenHKLM(None,
10                     winreg.KEY_QUERY_VALUE | winreg.KEY_ENU
11         self.conn.CloseKey(handle)
```

More information

- `http://www.samba.org/`
- `http://www.openchange.org/`
- IRC: *#samba-technical* / *#openchange* on Freenode

If you have ideas about asynchronous function usage, please let me know.