The world needs another programming language!

1. Why
2. How

Why?

The following use case seems to appear very often and hasn't been solved satisfactorily yet:

Embedding */untrusted/* systems into ones' own system

Untrusted mobile code

Examples

- Embedding foreign code into your application
- Loading external resources into your website
- Processing untrusted inputs

Specific problem:

Accounting for and policing access to

- internal (instruction steps, memory) and
- external (files, networking etc.) resources

A short survey of current approaches

Specialized operating systems

- Genode
- SEL4

Operating system functionality

- cgroups
- Container solutions (Docker)

Virtual machines and Languages

- E, Tcl, Lua, EVM, Java
- Proof carrying code

Protocols

- CORBA

Why are these not good enough?

- most are platform dependent
- none of the above provide fine grained control
- separation into small, independent units not possible

Properties to look out for

- Universality: usefulness as a general platform
- Granularity: size of accountable units
- Persistence: size of serializable units

Persistence I

```
>>> def func():
...     busy_count = 0
...     while 1:
...         busy_count += 1
...         if busy_count % 10 == 0:
...             print busy_count
...
>>> stackless.tasklet(func)()
<stackless.tasklet object at 0x01BD16B0>
>>> t1 = stackless.run(100)
10
20
>>> s = pickle.dumps(t1)
>>> t1.kill()
>>> t2 = pickle.loads(s)
>>> t2.insert()
>>> stackless.run(100)
30
40
50
```

Persistence II

- Transparent (orthogonal) persistence

- system vs object images

Specific use cases

- Package managers
- Distributed computing (BOINC, distributed.net)
- Smart contract systems

Sandboxing and accounting enables resource sharing.

How

Recursion

A system sandboxing a system sandboxing a system…

- An external library using an external library
- Foreign code loading or generating code it can’t trust

Semantics

“How do resources behave?”

- Objective: resources finite, assignment=transfer
- Subjective: assignment=intention

Objective view: awkward

- Child can never have more resources than parent
- Every message to child must be wrapped in while loop checking if it has exhausted resources
- Manual refueling through the entire chain

Subjective view is more intuitive

- Process can give child more resources than it has itself
- If it runs out of resources while child is running, control returns to its parent
- If parent refuels process, child continues immediately as if nothing happened

Why a new language?

- this goes beyond syntax changes!
- requires low level support

Proposal

Step 1: Build virtual machine

- Low level enough to be fine grained, high level enough to be usable
- Can act as compilation target for other languages
- Level of abstraction: ~WebAssembly (WASM)

Step 2: Build a language on top of it

- Plan right now is to make it look like Python
- Though it should behave more like Self

Step 3: Create prototypes

- P2P computation network
- Consensus objects

Step 4: Convince people this makes sense

- Hard sell so far

Step 5: ???