Pharo 64bits
by Esteban Lorenzano
Why we need 64bits Pharo?

• Because most systems nowadays are 64bits, and while they offer 32bits compatibility, it is not the best (special with linux, where compatibility may not be trivial).

• Because applications need to allocate more than 2G memory.

• Because not all libraries can be compiled/accessed on 32bits versions.

• … and because otherwise we are behind history ;)

The OpenSmalltalk-VM

• All **vm-dev** collaborators re-united under the “benevolent dictatorship” of Eliot (even if he would reject that appellation).

• Eliot made most of 64bits JIT for Linux and macOS

• Nicolas Cellier is working on the JIT for Windows

• Not trivial because in Windows sizeof(long) != 8
Bootstrap 64bits

- Pharo 6.0: Integration happens in 32bits and then a process is executed to create a 64bits image.

- Pharo 7.0: Both images will be bootstrapped separately.
FFI 64bits

- FFI backend was moved early this year.
- Is different between platforms
  - SysV: Linux, macOS
  - Win64: Windows
- But both are done :)
UFFI 64bits

- UFFI was designed from scratch to be “64bits compatible” (waiting for the right time)

- But we still need to adapt some things:
  - Structures and offsets
  - Structures and long sizes
UFFI and offsets

- Different sizes:
  - 32bits: `FFIExternalType sizeOf: #FFITestStructure "28"
  - 64bits: `FFIExternalType sizeOf: #FFITestStructure "40"

- We need to calculate structure sizes each time you start the image (on first execution), then each structure also holds some OFFSET_FIELDNAME variables.
FFIExternalStructure subclass: #FFITestStructure
  instanceVariableNames: ''
  classVariableNames: 'OFFSET_BYTE OFFSET_DOUBLE OFFSET_FLOAT OFFSET_INT64 OFFSET_LONG OFFSET_SHORT'
  package: 'UnifiedFFI-Tests'

fieldsDesc
" self rebuildFieldAccessors "
  ^ #(  
    byte byte;  
    short short;  
    long long;  
    float float;  
    double double;  
    int64 int64;  
  )

double
"This method was automatically generated"
  ^ handle doubleAt: OFFSET_DOUBLE

double: anObject
"This method was automatically generated"
  handle doubleAt: OFFSET_DOUBLE put: anObject
UFFI and longs

- Windows `sizeof(long)` != Rest-of-the-world `sizeof(long)`

- Real problem of this is that FFI backend does not acknowledge this difference (and in fact, atomic types on FFI are a bit “old”) when dealing with structures.

- But since UFFI is a layer that happens before a FFI call, we can do some nice stuff.

```
platformLongAt: byteOffset
  ^ self
  integerAt: byteOffset
  size: FFIArchitecture forCurrentArchitecture longTypeSize
  signed: true
```
Migration

• Almost transparent (most things works “out of the box”)

• Possible problems:
  • If you used long instead void*, this will not be right when Windows VM 64bits will arrive.
  • Some libraries have different structure definitions for each platform. You will need to use a strategy to solve this.
  • Some people/tools prefers to use #FFIInt32 instead plain “int” or “long”. You need to be as close to the C definition as possible.
Libraries currently supported

- All FFI functions in Pharo “just works”
- Athens
- SDL2
- libgit2: partially, to be finished soon™
Demo?

```
wget -0 get.pharo.org/64/60+vm | bash
wget -0 get.pharo.org/64/60+vmT | bash
```

just play with it!
Thanks!