CNRS ANF PYTHON

Memory works

Marc Poinot
Numerical Simulation Dept.
marc.poinot@onera.fr
Outline

- Memory management & Python
  - Basic memory concepts
  - Python
    - allocation concerns
    - reference count
    - numpy
  - workflow and memory ownership
Questions

- What is memory and how does it work with Python?
- How to share arrays of data from Fortran,C,C++ to Python?

Answers

- An overview of who, what, where, when (and maybe why)
- Simple recipes to make your life more comfortable

Outline

- Fast survey of memory concepts
- Memory management with Python/Numpy
- A strategy for Fortran/C/C++/Python/Numpy assembly
About memory

The memory is the part of a computer where you store data

From the user point of view
- Store the program and its permanent data
- Store temporary data and states of the program
- Exchange data with other programs

From the computer point of view
- Physical devices (primary is RAM, secondary is disk)
- Internal services (caching, paging, segmenting, swapping...)
- Services for the user (allocate, deallocate, lock...)

Now the computation platforms hardware and operating systems are more and more complex. We draw here large approximations to make concepts easier to understand, but reality is... complex.
Actors

script.py

python

C++  C  Fortran

OS

virtual memory

RAM  CPU  Other
Concurrency

- Memory is allocated for processes
  - The Operating System is your interface to this allocation
  - The OS is a process (more or less one per processor)
  - OS returns an address and reserve the memory up to the size
  - A memory can be reserved for a process
  - A shared memory can be reserved for several processes
  - The scope of the address is the OS process itself

- Each time you allocate memory you may stop your process
  - malloc(3) calls brk(2)
Addressing memory

**Logical View**
- **Word**: Represents the size of memory addresses.
- **Address**: Specifies the location in memory.
- **Base**: Indicates the starting point of memory allocation.
- **Offset**: Shows the distance from the base address.

**Internal Management**
- The diagram illustrates memory allocation with different colors.
- The code space ranges from 410 to 430, with specific areas highlighted.

The image provides a visual representation of how memory addresses are organized and managed, emphasizing the concepts of word size, address, base, and offset, along with the internal management of allocated memory blocks.
Variable/ Object/ Class

- Each reference to an object is tracked
Reference counting - 2

```python
>>> l1=[1,2,3]
>>> l2=[l1,4,5]
>>> l2
[[1, 2, 3], 4, 5]
>>> l1[2]=7
>>> l2
[[1, 2, 7], 4, 5]
>>> l3=l2[:]
>>> del l1
>>> l1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'l1' is not defined
>>> l2
[[1, 2, 7], 4, 5]
>>> l3
[[1, 2, 7], 4, 5]
>>> l3[2]=0
>>> l3
[[1, 2, 7], 4, 0]
>>> l2
[[1, 2, 7], 4, 5]
```
Garbage collector

 Günther König

 GC

 ▶ Find objects without reference
 ▶ Release the object memory
 ▶ Not very useful for memory leak finding
Weak references

- A garbage-able reference
  - Actual reference
  - Not taken into account for refcount
  - Not available on all Python objects

```python
>>> import weakref
>>> a=set([1,2,3])
>>> b=weakref.ref(a)
>>> b()
set([1, 2, 3])
>>> a
set([1, 2, 3])
>>> del a
>>> b()
set([1, 2, 3])
>>> b
<weakref at 0x7fad2e777100; dead>
>>> print b()
None
```
Memory profiling

▶ External modules
  ▶ memory_profile
    - displays memory use per function
  ▶ objgraph
    - displays relationships between objects
Memory leaks

- obmalloc.c
  - Manages arena of fixed size block
  - No way to find back object using this memory
  - No way to move the pointers
  - Arena memory is released only when all objects are released in the arena
  - This can lead to memory leak
Memory leaks

▶ obmalloc.c
  ▶ Manages arena of fixed size block
  ▶ No way to find back object using this memory
  ▶ No way to move the pointers
  ▶ Arena memory is released only when all objects are released in the arena
  ▶ This can lead to memory leak
Memory ownership

- **Mixed allocation/ release**
  - Python
    - PyMem_Malloc, PyMem_Free
  - C
    - malloc/free
  - C++
    - new/delete
  - Fortran
    - OS

- Owner acquire/release memory
- ndarray NPY_OWNDATA
- float32 NPY_F_CONTIGUOUS
- C/C++ float*
- Fortran real
- python object
- extension data
Practical Training

- import sys
- sys.getrefcount(o)

  - Create lists
  - Add references
  - Check counts
  - Use del