

COPYING GARBAGE COLLECTOR IN C

KYPIAKH MHNIAΔΟΥ & AIKATERINH MARIA ΓΕΡΑΚΙΑΝΑΚΗ
CSD4220 & CSDP1263
csd4220@csd.uoc.gr & agerakianaki@csd.uoc.gr

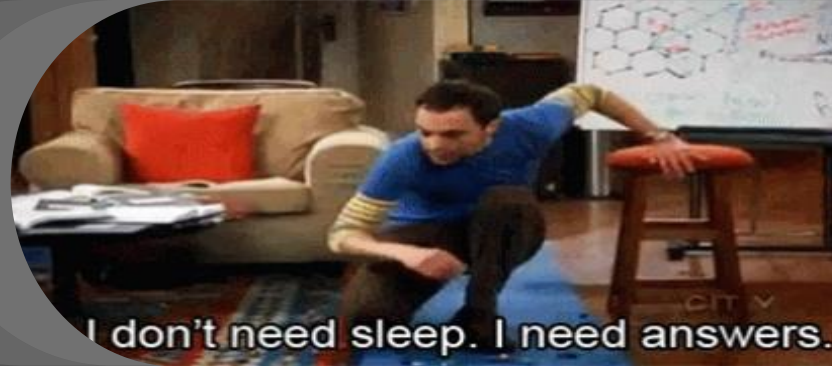
COPYING GARBAGE COLLECTOR IN C

- Copying collector starts from a set of roots.
- Traverse all of the reachable memory-allocated objects.
- Copies them from one half of memory into the other half.
- **Old space** is the area of memory in which we ***copy from***.
- **New space** is the area of memory we ***copy to***.
- Reachable data are copied and compacted in contiguous chunk.
- After compaction, we end up with a copy in the new space data and a large contiguous area of memory in new space.
- Goal: quickly and easy allocate new objects.
- Next garbage collection \rightsquigarrow the roles of old space and new space are reversed

Example

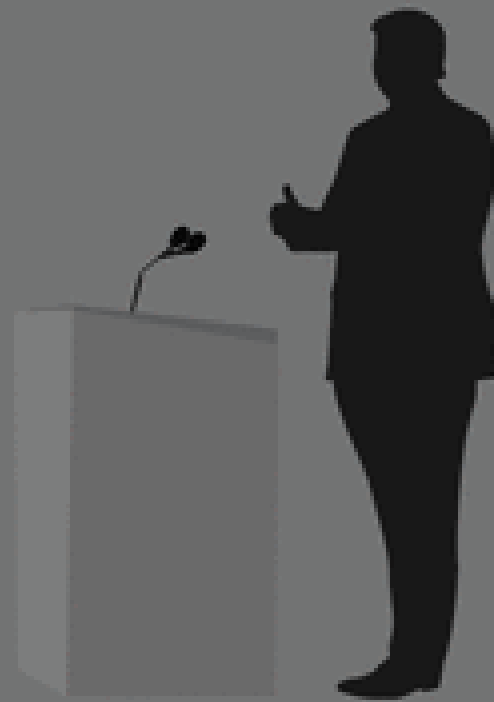
Obj1	Obj2	Obj3	Obj4	Obj5					
Obj1	Obj2	Obj3	Obj4	Obj5	Obj2'	Obj4'			
					Obj2	Obj4			
					Obj2	Obj4	Obj6	Obj7	Obj8
Obj4	Obj6	Obj8							

Okay... Why did we choose this project?



- How do we copy live objects in memory spaces?
- Pointers....our favourites.
- We wanted to refresh our knowledge in C.
- Why does C not use Garbage collectors?!
 - We actually want to fight...

DEBATE





Culture

VS

**Garbage
Collector**

**C is *perfect*
as it is...**

**V
S**

**Just a
reminder ...
Segmentation
fault ... Needs
a garbage
collector**

C was designed to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support.

V S

Bruh ...Data Structures in C are a headache...It needs a Garbage Collector!

The culture of C is to leave storage management to the programmer.

VS

If you lose track of an object, you have 'memory leak'. If C had Garbage Collector that couldn't be an issue!

**Doing this would
involve things that
made the language
implementation
slow and
compromise real-
time performance.**

VS

**Ok, are you sure that
this is true? Special
cases do exist! We do
not care for real time
in all applications!**

**It would be
technically difficult
and expensive to
implement a
garbage collector
for C.**

VS

**Ok, but what if I told
you I already made
one
?**



**Where is the
implementation
then?**

Implementation

- We will request page size memory chunks from the kernel.
- Our malloc() will do first fit.
- The GC algorithm is going to start from a set of roots (initialized data, BSS, stack)
- We will have to write assembly 🙄
- Two-Space Collection


Implementation issues ... so far

- C programs require a linker so we will need to find the final location of our roots in memory.
- Most modern Unix linkers provide the users with *etext* and *end* that will help us find the start and end of the data segment.
- The Stack top is easy and can be accessed by the `%esp` register.
- The bottom of the Stack is trickier.
 - We are still examining this part.
- There is no function in C that returns a hash map to all the variables that have been stack-allocated.

How are we going to evaluate our project?

- Write a program with data structures using our Garbage Collector and compare it to the same program written using the library `<stdlib.h>`.
- We may search for open source code programs that have memory leaks and rewrite them with our implementation.
- Performance time and memory usage.
- Any Idea? Help us!



A middle-aged man with a friendly expression is the central figure. He wears a dark baseball cap and blue denim overalls over a plaid shirt. He stands in a vast, green field under a clear sky. In the distance, rolling hills and a few buildings are visible. A semi-transparent white box is overlaid on the left side of the image, containing the text 'Thank you for your time!'. At the bottom of the image, a large line of text reads 'It ain't much, but it's honest work' in a stylized, outlined font.

**Thank you for your
time!**

It ain't much, but it's honest work