Memory Management

Program

Language runtime (Pointers, memory allocation, garbage collection)

Operating system (Virtual memory, page tables, swapping, ...)

Hardware (Memory controller, Direct Memory Access (DMA), Translation Lookaside Buffer (TLB), ...)

```
let f n =
  let l = Base.List.range 0 n in
  fun i -> List.nth l i
```

This call to f allocates a list l

```
─( 16:13:20 )─< command 5 >──────────────────────────────{ counter: 0 }─
utop # let g = f 10;;
val g : int -> int = <fun>
─( 16:13:44 )─< command 6 >──────────────────────────────{ counter: 0 }─
utop # g 3;;
- : int = 3
─( 16:13:57 )─< command 7 >──────────────────────────────{ counter: 0 }─
utop # g 5;;
- : int = 5
─( 16:14:13 )─< command 8 >──────────────────────────────{ counter: 0 }─
utop # g 7;;
- : int = 7
```

All calls Refer to the same l.

```
let g2 = f 100

g2 73
```

Creating g_2 does not invalidate g

There must be two independent l-s in memory.

Jobs of the language runtime

- Memory management
- What things are in memory?
- What memory is used?
- What memory is available?

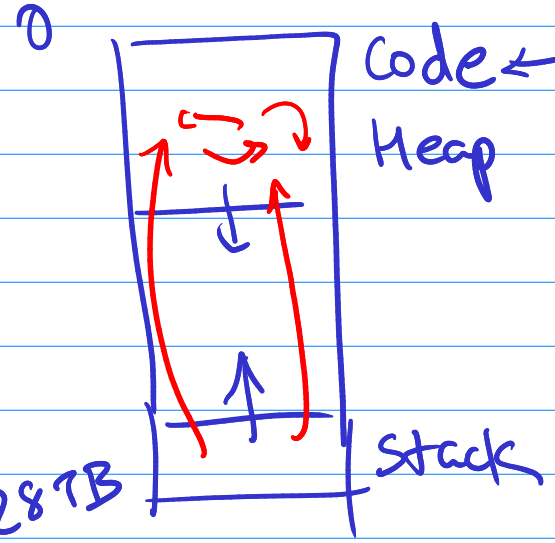
```
let rl = ref (Base.List.range 0 10)

rl := (List.map Int.succ (Base.List.range 0 200))
```

Old list not reachable any more.

Memory can be reclaimed!

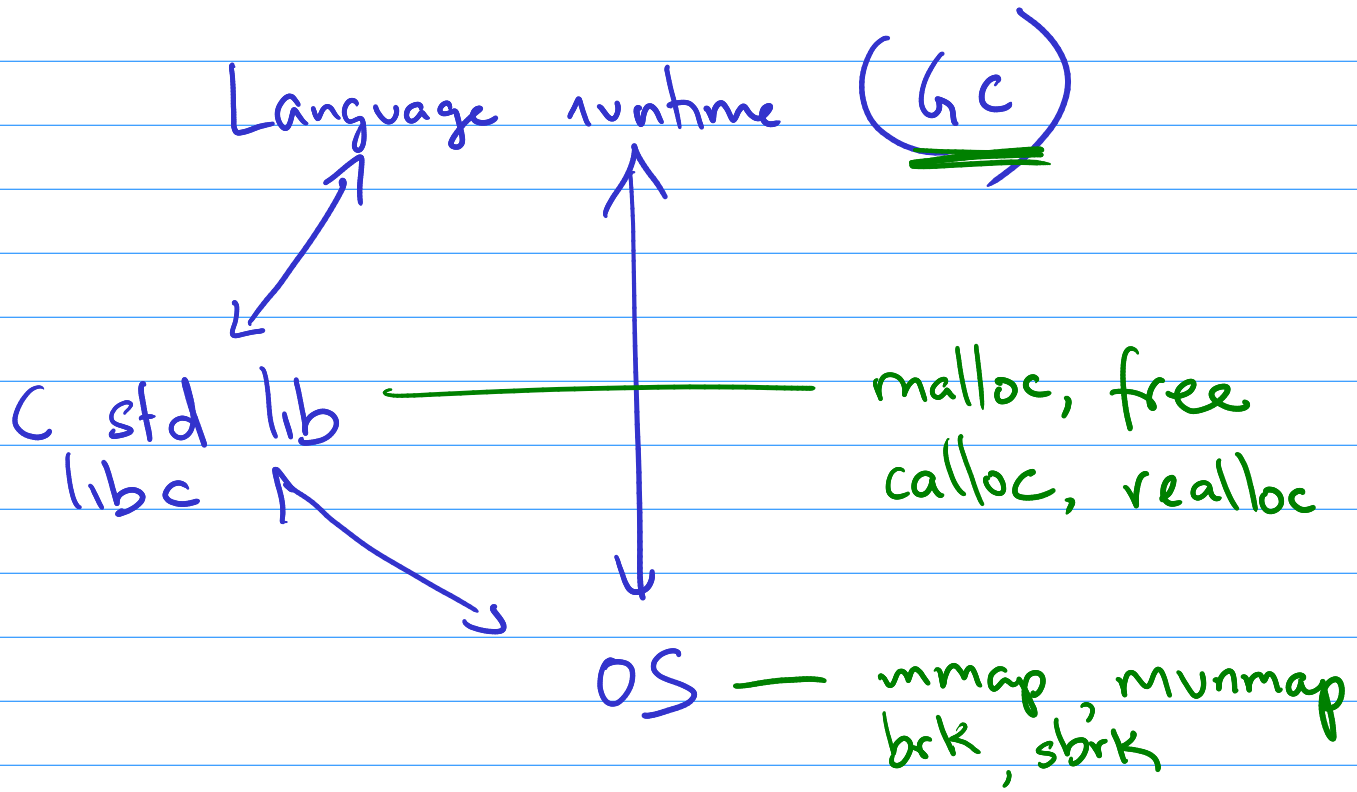
Virtual address space

2^{64}



$2^{47} = 128TB$

All currently accessable values must be reachable from the stack.


Language runtime
(GC)
C std lib
libc
malloc, free
calloc, realloc
OS
mmap, munmap
brk, sbrk