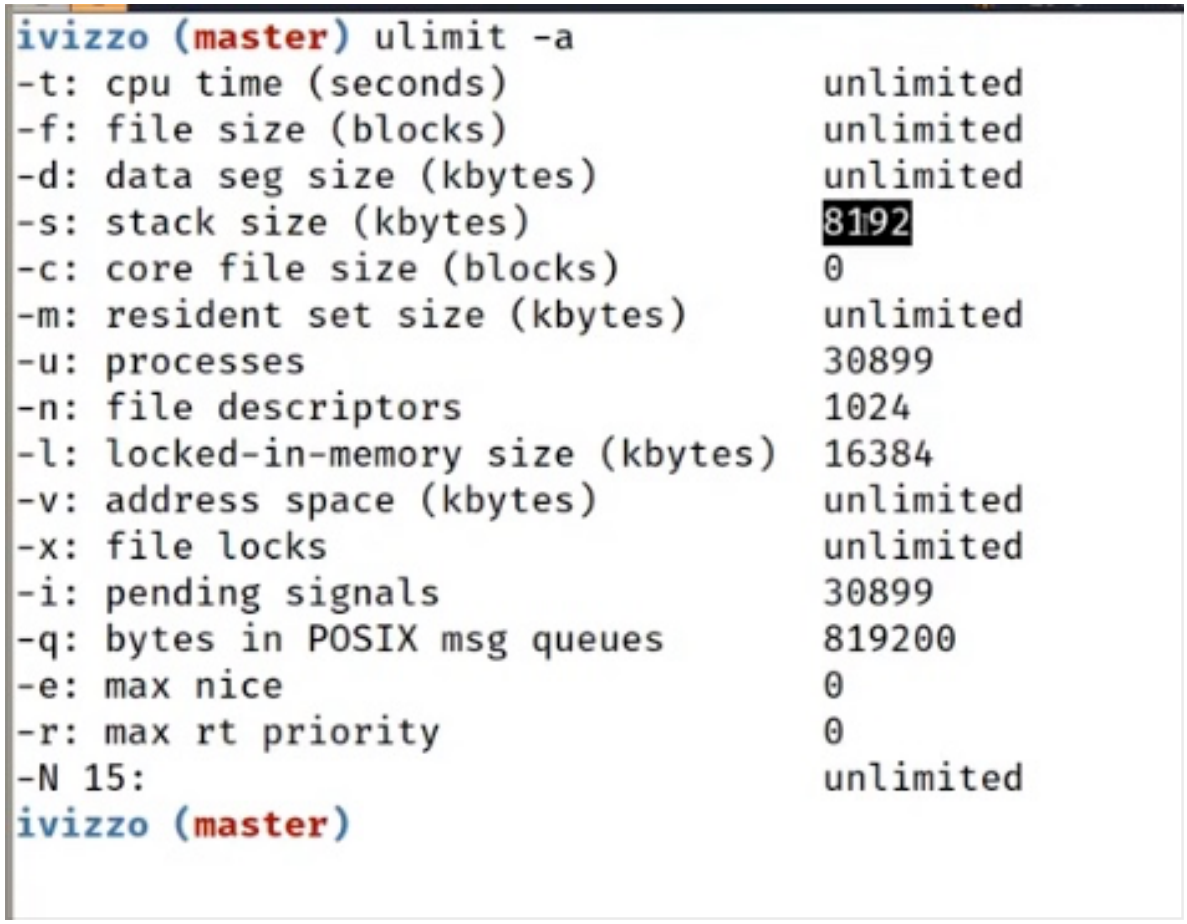
Stack and Heap

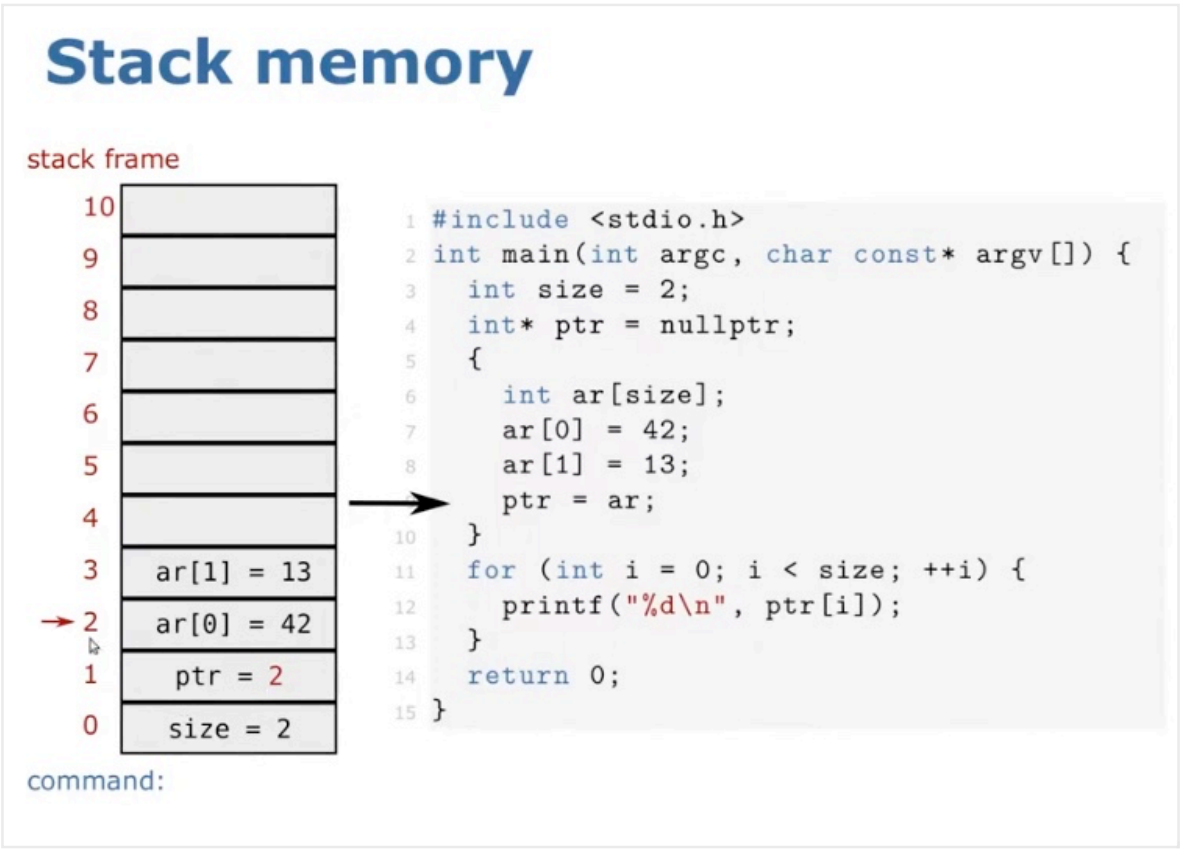
Memory Management Structures

- Working memory is divided into two parts:
  - Stack
  - Heap

Stack memory

- Static memory
- Available for short term storage (scope)
- Small / limited (6 MB linux typically)
- Memory allocation is fast
- LIFO (Last In First Out) structure





Heap Memory

- Dynamic memory
- Available for long time (program runtime)
- Raw modifications possible with new and delete (usually encapsulated within a class)
- Allocation is slower than stack allocations.

Operators `new` and `new[]`

- User controls memory allocation (unsafe)
- Use `new` to allocate data:

```
1 // pointer variable stored on stack
2 int* int_ptr = nullptr;
3 // 'new' returns a pointer to memory in heap
4 int_ptr = new int;
5
6 // also works for arrays
7 float* float_ptr = nullptr;
8 // 'new' returns a pointer to an array on heap
9 float_ptr = new float[number];
```

- `new` returns an address of the variable on the heap
- **Prefer using smart pointers!**

Operators `delete` and `delete[]`

- **Memory is not freed automatically!**
- User must remember to free the memory
- Use `delete` or `delete[]` to free memory:

```
1 int* int_ptr = nullptr;
2 int_ptr = new int;
3 // delete frees memory to which the pointer points
4 delete int_ptr;
5
6 // also works for arrays
7 float* float_ptr = nullptr;
8 float_ptr = new float[number];
9 // make sure to use 'delete[]' for arrays
10 delete[] float_ptr;
```

- **Prefer using smart pointers!**