Stack and Heap

Sunday, October 4, 2020 7:24 PM

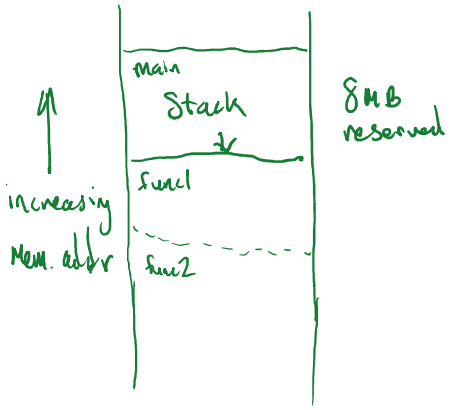
The Stack

Defn. The stack is where all local variables and parameters live for each function.

Properties A function's stack frame goes away when it returns.

Grows downwards when function called and shrinks upwards when it returns.

Each function call has its own stack frame.



Note - when the stack shrinks, stack frames are lazily deleted (marked as reusable) rather than deleted

stack overflow — when entirety of stack memory is used

The Heap

Defn. The heap is a part of memory that is managed by the program (rather than the runtime/OS)

Properties. Memory is only freed when it is manually deleted.

Grows upwards as more memory is allocated.

Malloc.

```
void* malloc(size_t size);
```

Takes in a size of bytes to allocate, then returns a pointer to an available block of memory with enough space.

void* - pointer to generic memory, do not need to cast to set equal to other type

NOTE: ONLY APPLIES TO C! C++ needs casting.

Memory not cleared before allocation - could contain garbage.

Can return NULL - meaning not enough memory for the request

In this case - use assert to guarantee non-nullness or crash.

Calloc.

```
void *calloc(size_t nmemb, size_t size);
```

Like malloc, but zeroes out memory.

Malloc's nmemb * size bytes and zeroes them out

Note - use sparingly as it is more expensive than malloc.

strdup.

```
char *strdup(char *s);
```

Convenience function that copies a string onto the heap and null-terminates it

More efficient than malloc + strcpy

Freeing memory

free. `void free(void *ptr);`

Need to delete heap-allocated memory

Use free command and pass in starting address of block to free

Notes.

Can only free a heap block once

Must free entire alloc at a time

Can reset pointer allocation after free

Memory leaks.

Allocating block but not freeing it - can run out of free memory on heap

Memory leaks don't cause crashes

⇒ Only free after program written

Use valgrind to find leaks

Resizing allocations.

realloc. `void *realloc(void *ptr, size_t size);`

Takes existing reallocation pointer and enlarges to new requested size, returning the new pointer.

If space exists in initial block, adds space

Else, moves memory to new location, frees memory in old location, returns new location

Note: ptr must be pointer to beginning of heap-allocated block.