CS 354 - Machine Organization & Programming
Thursday, October 3, 2019

Midterm Exam - Thursday, October 3rd, 7:15 - 9:15 pm

- **Lec 1 (2:30 pm):** room 3650 of Humanities
- **Lec 2 (4:00 pm):** room B10 of Ingraham Hall
- UW ID required
- #2 pencils required
- closed book, no notes, no electronic devices (e.g., calculators, phones, watches)
- see "Midterm Exam 1" on course site Assignments for topics

Project p2B (3%) DUE: 10 pm, Monday, October 7th

Last Time

C's Abstract Memory Model
Meet Globals and Static Locals
Where Do I Live?
Linux: Processes and Address Spaces
---------- END of Exam 1 Material ----------
Meet the Heap

Today

Exam Mechanics
C's Heap Allocator (`stdlib.h`)
Posix `brk` (`unistd.h`)
Allocator Design

Next Time

Heap Internal View, Block Placement, Splitting
Read: B&O 9.9.6 - 9.9.8

C's Heap Allocator (`stdlib.h`)

What? `stdlib.h` contains a collection of ~25 commonly used C functions.

- conversion: atoi, atod, atof
- execution: exit, abort
- Math: abs
- searching: bsearch (binary search)
- Sorting: qsort (quick sort)
- random nums: rand, srand
- allocator

C's Heap Allocator Functions

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

Allocates and returns generic ptr to block of heap memory of `size` bytes, or returns `NULL` if allocation fails.

```
void *calloc(size_t nItems, size_t size)
```

Allocates, clears to 0, and returns a block of heap memory of `nItems * size` bytes, or returns `NULL` if allocation fails.

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

Reallocates to `size` bytes a previously allocated block of heap memory pointed to by `ptr`, or returns `NULL` if reallocation fails.

```
if (ptr == NULL)  return malloc(size);
else if (ptr == 0) { free ptr; return malloc(size); }
else if  // Attend to reallocate.
```

```
void free(void *ptr)
```

Frees the heap memory pointed to by `ptr`. If `ptr` is `NULL` then does nothing.

undefined behavior: - free heap mem that's been freed.
- free stack mem.

❋ *For CS 354, if malloc/calloc/realloc returns NULL* just display an appropriate error message and exit the program.

Posix `brk` (`unistd.h`)

What?

- Posix (Portable OS Interface) standard for maintaining compatibility among Unix OS.
- `unistd.h` header file to access funcs in posix API.

unix standard.

DIY Heap via Posix Calls

- Funcs on this page are used by C' heap allocator
- Behavior is undefined if you use both these funcs and C's allocator

brk "Program Break" pointer to top of heap.

`int brk(void *addr)`
Sets the top of heap to the specified address `addr`.
Returns 0 if successful, else -1 and sets `errno`.
OS clears no pages of heap mem for security

`void *sbrk(intptr_t incr)`
Attempts to change the program's top of heap by `incr` bytes.
Returns the old brk if successful, else -1 and sets `errno`.

`errno` set by OS Funcs to communicate specific errors.

```
# include <errno.h>
:
printf("ERROR, %s \n", strerror(errno));
```

❋ *For most applications, it's best to use malloc/calloc/realloc/free*
Since the standard (Allocator is very well implemented and more portable)

Allocator Design

Goals

throughput Malloc & Free operation / sec.
higher is better.

memory utilization memory requested / heap allocated.
higher is better
minimize open head in heap blocks

Requirements

→ List the requirements of a heap allocator.

1. Alloc's use the heap.
2. Provide an immediate response
3. Must handle arbitory sequences of requests.
4. Don't move / change allocated blocks.
5. Follow Alignments Requirements.

Design Considerations

- Free block organization.
- Placement Policies
- Spitting free blocks to creat a better fit.
- coalescing to creeate longer free blocks.