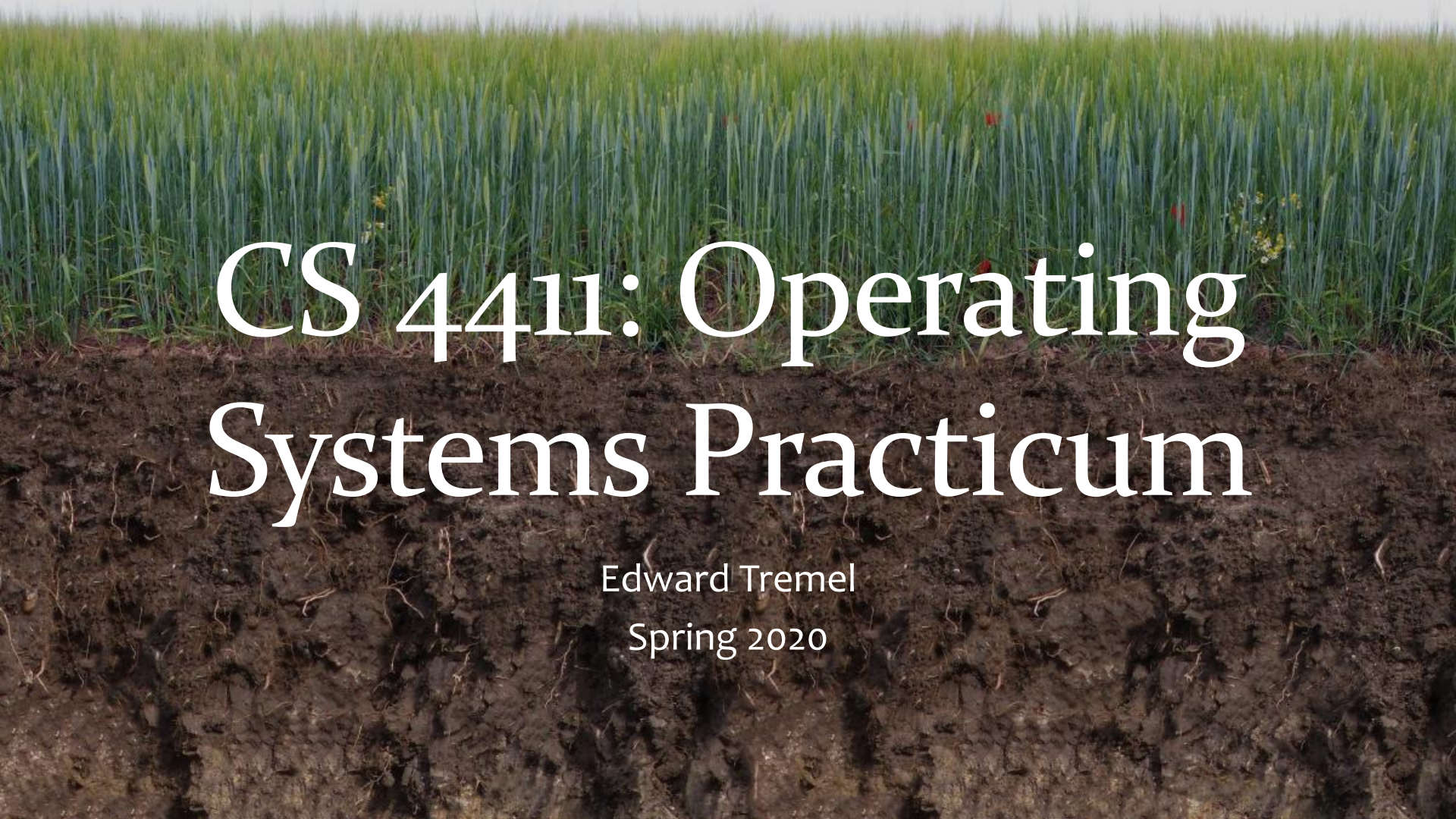


CS 4411: Operating Systems Practicum

Edward Tremel

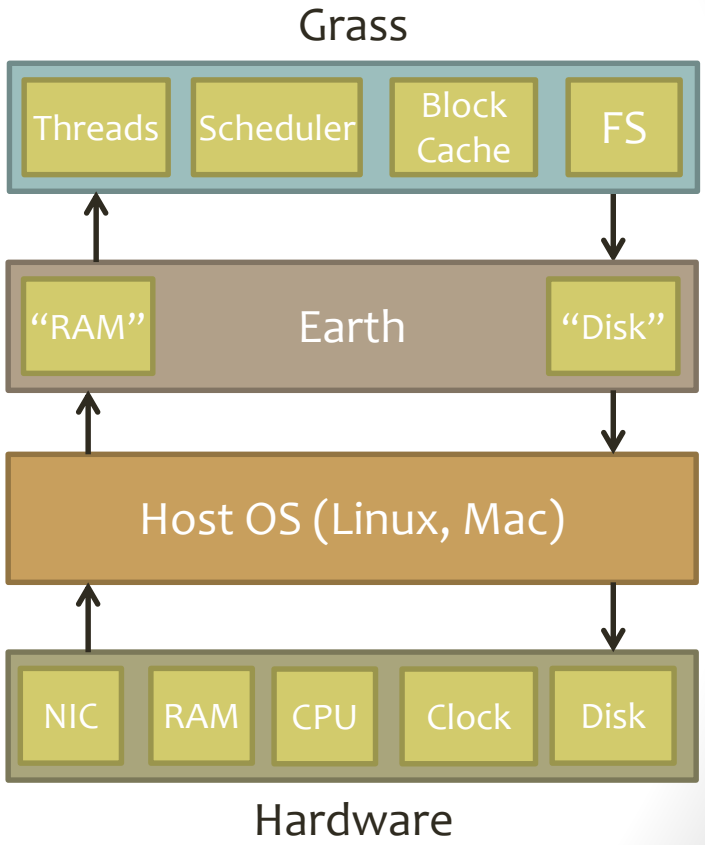
Spring 2020

Outline for Today

- Course Overview
- Introductions
- Course Goals
- Warm-up Activity
- Logistics and Administrivia
- C and Pointers

What is 4411?

- Hands-on experience developing an operating system
- Write code to implement the concepts you'll learn in 4410
- All projects add components and features to the same operating system, EGOS (Earth and Grass OS)



Why Take This Class?

- Lots of practical C programming experience
- Learn the “nuts and bolts” of how an OS works
- Practice working on a large software project, not just self-contained homework assignments

About Me

Edward Tremel

AUGUSTA UNIVERSITY

- Research: Distributed systems
  - Replicated services over RDMA networks
  - Data collection from IoT devices
- Teaching: Mostly OS
  - Summer 2019's 4410
  - Many semesters of TAing 4410/11
- Ph.D. at Cornell – graduating this semester
- Non-CS interests: Board games, running, Victorian literature, Pokémon

Learning Goals

- **Threads:** Design and implement a library that enables applications to create multiple threads, scheduled by the OS
- **Scheduling:** Understand the multilevel-feedback-queue scheduling algorithm and create an implementation of it
- **Caches:** Compare and contrast caching strategies; design and implement a block cache for disk storage that uses the CLOCK algorithm
- **Filesystems:** Design and implement a FAT-based filesystem that allows the OS to store and retrieve files from block storage (disk)

Activity: C Syntax

What are all the differences between these two variables?

```
int x = 100;
int *y = malloc(sizeof(int));
```

~~What is `x + y`?~~

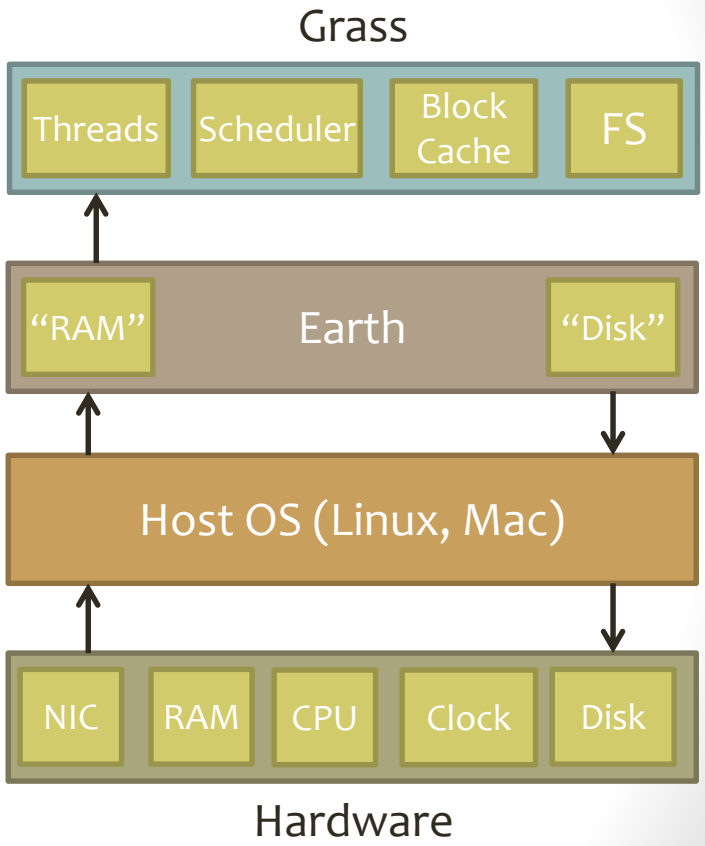
What is `x + *y`?

Outline for Today

- Course Overview
- Introductions
- Course Goals
- Warm-up Activity
- **Logistics and Administrivia**
- C and Pointers

Content Overview

- Biweekly project assignments:
  - User-Level Threading
  - Multi-level Scheduling Queue
  - Filesystems I – Block Cache
  - Tracing and Debugging
  - Filesystems II – FAT filesystem
- Plus initial “mini” assignment: Queue
- All projects done in teams of 2 students
- No exams



Logistics Overview

- Lecture: Fridays at 2:30
  - Goal is to prepare you for each project
  - Some weeks have no lecture, class meeting time is extra office hours
- Class rules: Just like 4410
  - No cell phones
  - No laptops
  - Asking questions is good,
    private discussions are disruptive
- Textbook: Same as 4410

Resources

- Course website: http://www.cs.cornell.edu/courses/cs4411/2020sp/
  - Schedule and lecture slides
  - Office hours information
- Piazza: http://piazza.com/cornell/spring2020/cs4411
  - Questions, announcements, and finding teammates
- **CMS** for assignments
- GitHub for code: https://github.coecis.cornell.edu

GitHub Logistics

- EGOS code will be distributed via a GitHub repository
- https://github.coecis.cornell.edu/cs4411-2020sp/egos
- To access this repository, you will need to join the **cs4411-2020sp** organization and the **students** team within it
- Submit your Cornell GitHub username via a Google form (coming soon) so that we can add you
  - Details on your Cornell GitHub username on the course website
- Suggestion: Fork the EGOS repo and share it with your partner

Office Hours

- Instructor :
  - Wednesdays 9:45-11:45 am
  - Rhodes 402
  - Also during class period when there is no lecture
- PhD TA:
  - Mondays 1:00-2:45 pm
  - Rhodes 597
- Course staff:
  - See website

Grading

- All grades come from projects
- All projects equally weighted
- Late Policy:
  - Each **team** has a total of 3 “slip days”
  - You can use at most 1 per project
  - After 24 hours, or with no slip days remaining, no credit

Academic Integrity

All submitted work must be your own

- First project must be entirely your own work
- Other projects must represent solely the work of the two members of the team

Do not put project code in a public GitHub repository

- This is equivalent to letting someone else look at your code

Outline for Today

- Course Overview
- Introductions
- Course Goals
- Warm-up Activity
- Logistics and Administrivia
- **C and Pointers**

Pointers Again

- What is the difference between these function declarations?

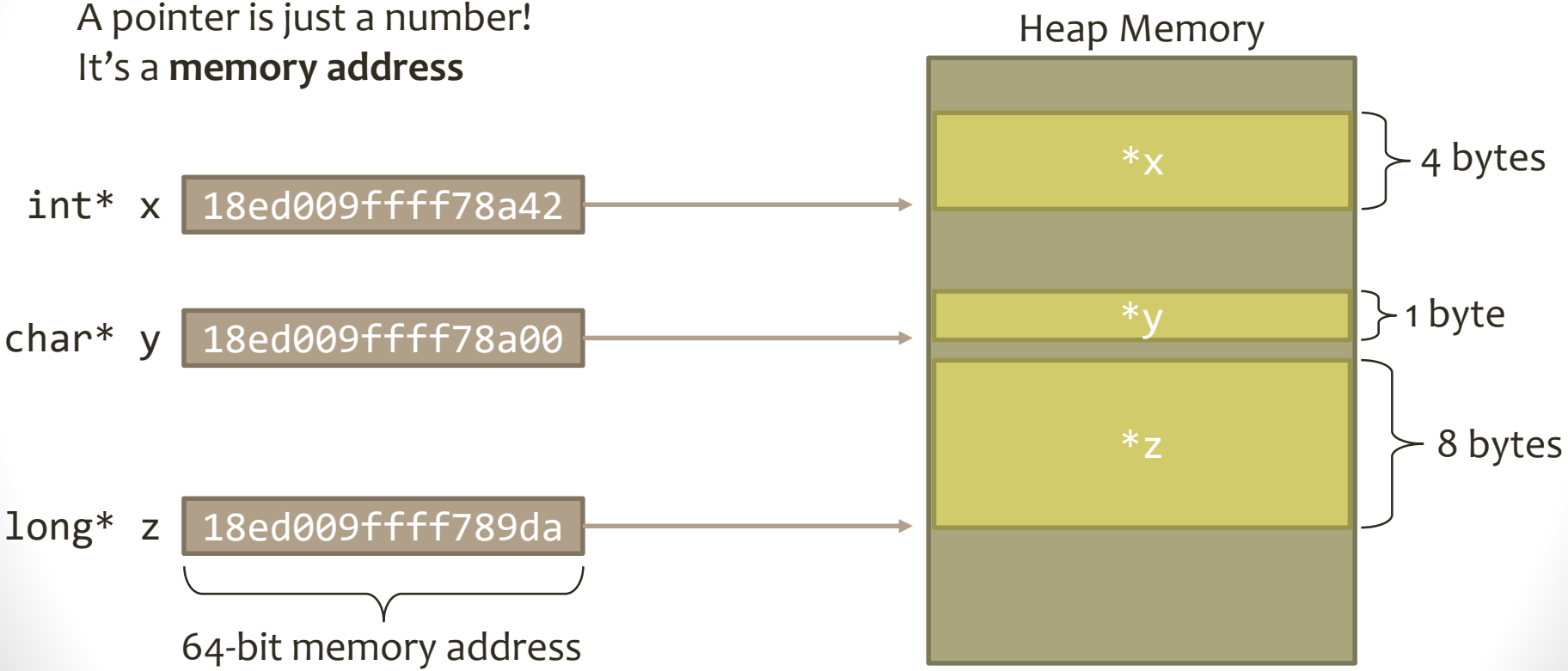
```
int queue_append(struct queue q, void *item);
int queue_append(struct queue *q, void *item);
```

- Which one is better?

- Style I prefer:
  ```
  int queue_append(struct queue* q, void* item);
  ```

Pointers and Memory

A pointer is just a number!
It's a **memory address**



Pointer Types

- The **type** of a pointer determines how much memory it points to

Pointer Type	Size of Memory
`char*`	1 byte
`int*`	4 bytes
`long*`	8 bytes
`float*`	4 bytes
`double*`	8 bytes
`struct foo*`	`sizeof(struct foo)`
`void*`	?????????

What is a void*?

```
int queue_append(struct queue* q, void* item);
```

- "Generic Pointer"

item `18ed009ffff789da`

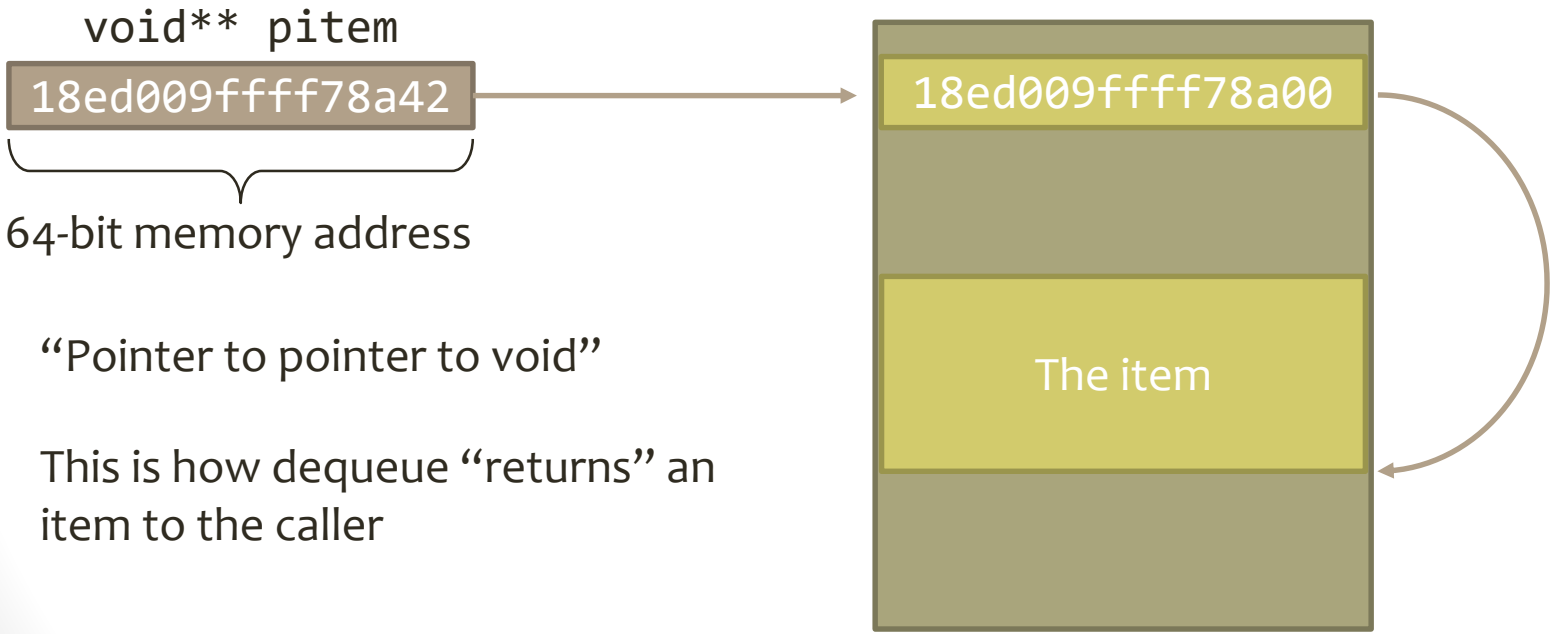
- Contains a memory address
- Cannot be dereferenced – you don't know what type it points to
- Must be cast or assigned to its "real" type to dereference:

```
long x = 100 + *(long*)(item);
long* z = item;
long x2 = 200 + *z;
```

- Casting to the wrong type is dangerous!

Adding Another Asterisk

```
int queue_dequeue(struct queue* q, void** pitem);
```



"Pointer to pointer to void"

This is how dequeue "returns" an item to the caller

Typedefs and Hiding Pointers

- Ordinary typedef: `typedef unsigned int msg_id_t;`
- Common "struct defining" typedef:

```
typedef struct message {
  msg_id_t id;
  char body[256];
} message_t;
```

`message_t` is now an alias for `struct message`

- Pointer-hiding typedef:

```
typedef struct queue* queue_t;
```

`queue_t` is now an alias for `struct queue*`