



ECE 391 Discussion

Week 4

Announcements & Reminders

- ▶ PS2 has been posted
 - ▶ Work in groups
 - ▶ Due next Tuesday (Sept 25), 2pm
 - ▶ Dropbox is at 3rd floor ECEB #54
- ▶ MP1 grades is posted
 - ▶ Let us know if your grade is not there
 - ▶ There are no regrades for MP1
- ▶ MP2 will be posted later this week
 - ▶ Start early!!!!
 - ▶ Read the documentation carefully

ECE 391
Computer Systems Engineering
Fall 2018

Announcements
[Piazza](#)
[Queue](#)

Syllabus
[Overview](#)
[Staff Directory](#)
[Office Hours](#)

Assignments
[Course Notes](#)
[Exams](#)
[Grades](#)

Tools, References, and Links

ECE 391
Computer Systems Engineering

General Announcements

This page may not be kept updated. Please check [Piazza](#) for announcements.

- 8/03 Welcome to ECE 391 Fall 2018!

ECE 391 Big Brother

44	43	42	41		
35	36	37	38	39	40
34	33	32	31	30	29
23	24	25	26	27	28
22	21	20	19	18	17
11	12	13	14	15	16
10	09	08	07	06	05
01	02	03	04		

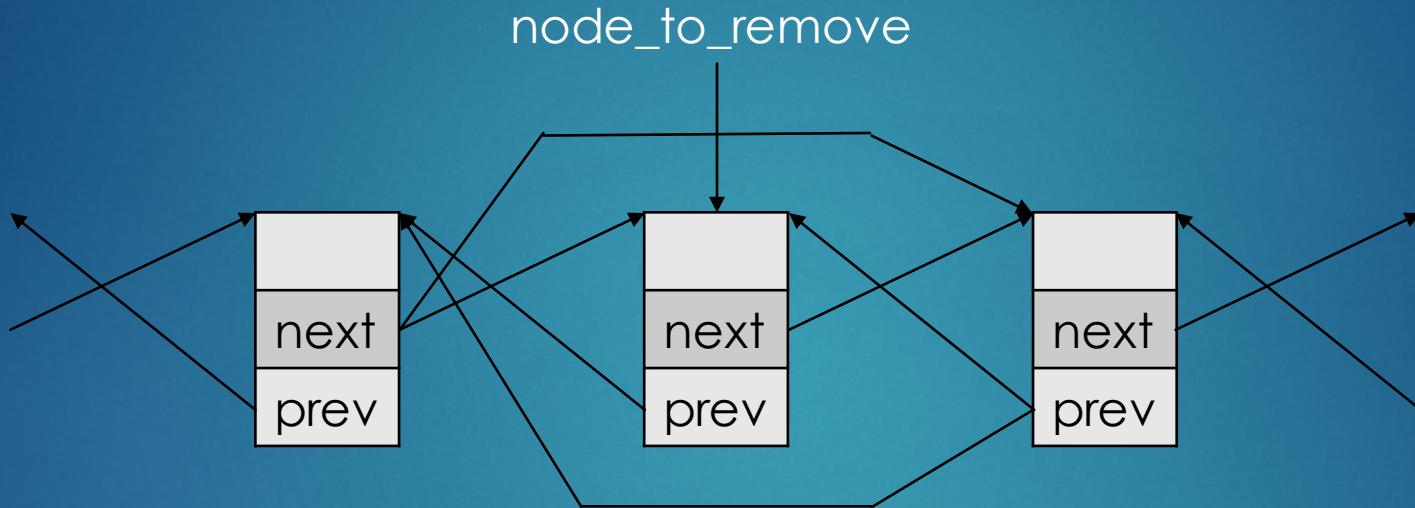
Problem Set 2

- ▶ Put everyone's name and NetID on one copy
- ▶ Don't be lazy, reading documentation will help you with your MP2
 - ▶ The website for VGA is related to checkpoint 1
 - ▶ The MTCP header file is related to checkpoint 2
- ▶ Don't over think in problem 3
 - ▶ It's not a full working program, just a primitive

Linked List

- ▶ Given a doubly linked list
 - ▶ Process 1 tries to traverse it (interrupt)
 - ▶ Process 2 tries to remove a node from it
 - ▶ Race condition

Linked List



```
CLI();  
item_to_remove->prev->next = item_to_remove->next;  
item_to_remove->next->prev = item_to_remove->prev;  
STI();
```

But CLI/STI is only good for one processor

Synchronization (Atomic Instructions)

broken_lock:

```
    movl 4(%esp), %eax
```

loop:

```
    movl (%eax), %edx
```

```
    jnz   loop
```

```
    movl $1, (%eax)
```

```
    ret
```



broken_lock:

```
    movl 4(%esp), %eax
```

loop:

```
    movl (%eax), %edx
```

```
    jnz   loop
```

```
    movl $1, %ecx
```

```
    xchgl %ecx, (%eax)
```

```
    ret
```

Spin Lock Implementation

```
typedef uint32_t spinlock_t;  
void spin_lock(spinlock_t *lock);
```

spin_lock:

```
    movl 4(%esp), %eax
```

loop:

```
    movl $1, %ecx
```

```
    xchgl %ecx, (%eax)
```

```
    cmpl $1, %ecx
```

```
    je loop
```

```
    ret
```

```
void spin_unlock(spinlock_t *lock);
```

spin_unlock:

```
    movl 4(%esp), %eax
```

```
    movl $0, (%eax)
```

```
    ret
```

Linked List Revisit

```
CLI();  
/* critical section code */  
STI();
```



```
CLI();  
spin_lock(spinlock_t *lock);  
/* critical section code */  
spin_unlock(spinlock_t *lock);  
STI();
```

When Using Locks

- ▶ Do not protect regions of memory from modification
- ▶ Do not mark certain data structures as locked
- ▶ Adding a lock to a struct does not magically protect that struct
- ▶ Does not matter if lock is in the struct or not
- ▶ It is up to the programmer to protect against race conditions

Synchronization Practice

- ▶ Ben Bitdiddle has a nasty problem with eating/drinking. In fact, he will eat anything placed in front of him immediately. Luckily for poor Ben, we ECE391 students have a tool to help him. This tool is called a lock. Ben is fed by a machine. The machine can dispense food and drink. To protect Ben, the machine must follow these rules:
 - ▶ • Ben can only consume one item at a time (i.e. one drink or one food).
 - ▶ • If food and drink are both ready for Ben, he must finish the drink first.
 - ▶ • The machine can only hold up to 10 drinks at a time. It returns -1 if it fails to produce new drinks, returns 0 otherwise.
 - ▶ • The machine can hold infinite amount of food (i.e. it will keep producing food no matter what).
 - ▶ • Ben isn't too picky about food. That means if food (a) and (b) are ready at the same time, he will happily eat either (priority of food does not need to be enforced).
 - ▶ • Ben isn't too picky about drinks. That means if drink (a) and (b) are ready at the same time, he will happily drink either (priority of drink does not need to be enforced).

Synchronization Practice

- ▶ Note: The only type of synchronization primitive you may use is spinlock t.
There is no limit to the amount of food the machine can hold, however there may be a restriction from your code. Please state this restriction in plain text.
- ▶ /* you may add up to 3 elements to this struct */
- ▶ typedef struct food_drink_lock {
 } fd_lock_t;
- ▶ void produce_food(fd_lock_t* fd) {}
- ▶ void consume_food(fd_lock_t* fd) {}