COMP1511 Programming Fundamentals

LINKED LIST LECTURE PROGRAM!
(TO WRAP THINGS UP)
WEEK 9 LECTURE 1

with Tammy
(Sasha is back for next lecture :))

Course Admin
Tammy Zhong
She/Her

Announcements

ASSIGNMENT 1 MARKS

RELEASE EARLY TO MID THIS WEEK :)

(ANY ISSUES PLEASE EMAIL CS1511)

MYEXPERIENCE SURVEY

KEEP AN EYE ON YOUR UNSW EMAIL FOR IT

WE WOULD LOVE YOUR FEEDBACK ❤️

Announcements

FINAL EXAM INFO

SASHA WILL GO THROUGH ON WEDNESDAY :)

WEEK 10

PRACTICE EXAM

WILL BE HELD IN LABS

IF YOU ARE ENROLLED IN ONLINE TUT-LAB, YOU WILL BE ABLE TO SIGN UP TO AN IN-PERSON LAB VIA LINK* ON FORUM

WEEK 11

REVISION SESSIONS

(LAST SET OF REVISION SESSIONS)

LOOK OUT FOR SIGN UPS FOR THOSE ON THE FORUM THIS/NEXT WEEK

*Coming soon!

LIVE CODE HERE:

https://cgi.cse.unsw.edu.au/~cs1511/24T1/live/week_9/

LAST FEW LECTURES:

- Linked Lists
  - insert at head
  - traverse (and print) a linked list
  - insert at tail
  - insert anywhere
  - deletion of specific nodes
    - at head
    - at tail
    - in between two nodes

THIS WEEK:

- Today:
  - Wrapping up linked list
  - Linked list lecture Program
- Wednesday:
  - Exam Information

LINKED LIST CODE WRITING CHECKLIST :)

For any linked list operations you try and code up:

- ☑ *Are you drawing diagrams as you code - Draw, Code, Repeat! (It's so much easier to debug this way!)*

- ☑ *Have you considered all the possible cases we can operate in? Here are <u>some</u> we mentioned that might apply:*
  - *How many nodes do we have in the list?*
    - *Empty list?*
      - *then there's nothing to delete!*
    - *Only one node in the list?*
    - *More than one / many nodes in the list?*

  - *Which node/where in the list do we want to operate on:*
    - *at the head?*
    - *between two nodes?*
    - *at the tail?*

!!

FAQ :)

When do we use malloc(...)?

- *when we have "new" data to be inserted into a list*
  - *working with existing data doesn't count e.g. printing the list, we don't need malloc(...)*

When do we use free(...)?

- *when we are trying to "remove" any node(s)*
- *whenever we use malloc(...) in our programs, there should be a corresponding free(...) for each piece of memory malloc-ed*

WE SPOKE ABOUT MEMORY LEAKS BUT...

HOW DO I GO ABOUT CHECKING FOR MEMORY LEAKS IN MY CODE?

(OTHER THAN MANUALLY LOOKING AT IT)

WE SPOKE ABOUT MEMORY LEAKS BUT...

HOW DO I GO ABOUT CHECKING FOR MEMORY LEAKS IN MY CODE?
(OTHER THAN MANUALLY LOOKING AT IT)

```
~$ dcc program.c -o program --leak-check
```

UH OH! - BUG IN OUR `INSERT_AT_POSITION` FUNCTION PREVIOUSLY CODED?!

WHAT HAPPENS WHEN YOU TRY AND INSERT A NODE AT POSITION LENGTH + 1?

E.G. INSERTING A NODE AT POSITION 6 IN A LINKED LIST OF 5 NODES (WHICH IS INVALID)

Let's look back at our linked_list.c code from last week...

FEI

(FREQUENTLY ENCOUNTERED ISSUES)

- *Accessing NULL pointer variable*
- *Uninitialised pointer*
- *Memory leak*

FEI

(FREQUENTLY ENCOUNTERED ISSUES)

! *Accessing NULL pointer variable*

```
Runtime error: accessing a field via a NULL pointer
dcc explanation: You are using a pointer which is NULL
  A common error is  using p->field when p == NULL.

Execution stopped in insert_at_position(data=0, position=6, head=0x602000000070) in linked_list.c at line 126:

                current = current->next;
        }
        // current == NULL --- reached the end of the list
        // OR ---- BUGGGG MISSED -- IT's not OR but AND/OR
        // counter == position --- we have reached the position we want to insert
        if (counter == position) { // && current != NULL
-->             new_node->next = current->next;
                current->next = new_node;
                return head;
        }

Values when execution stopped:

counter = 6
current = NULL
position = 6
new_node->next = NULL
```

(Screenshot of example output you might get from dcc)

FEI

(FREQUENTLY ENCOUNTERED ISSUES)

! Uninitialised pointer

```
struct node *some_ptr;
struct node *head = create_node(7, some_ptr);
```

```
● z5163340@nw-k17-login1:~/public_html/24T1_1511/test$ dcc linked_list.c -o linked_list
linked_list.c:32:37: warning: variable 'some_ptr' is uninitialized when used here [-Wuninitialized]
        struct node *head = create_node(7, some_ptr);
                                           ^~~~~~~~
linked_list.c:31:23: note: initialize the variable 'some_ptr' to silence this warning
        struct node *some_ptr;
                            ^
                             = NULL
dcc explanation: you are using variable 'some_ptr' before it has been assigned a value.
Be sure to assign a value to 'some_ptr' before trying to use its value.
Don't understand?  Get AI-generated help by running: dcc-help
⊗ z5163340@nw-k17-login1:~/public_html/24T1_1511/test$ ./linked_list

Runtime error: member access within misaligned address 0x000000000001 for type 'struct node', which requires 8 byte al
Execution stopped in print_list(head=0x602000000070) in linked_list.c at line 72:

void print_list(struct node *head) {
        struct node *current = head;
        while (current != NULL) { // while we have not reached the end of the list
-->             printf("%d ", current->data);
                current = current->next;
        }
        printf("\n");

Values when execution stopped:

current = 0x1

Function call traceback:

print_list(head=0x602000000070) called at line 37 of linked_list.c
main()
```

(Screenshot of example output you might get from dcc)

FEI

(FREQUENTLY ENCOUNTERED ISSUES)

! *Memory Leak*

```
z5163340@nw-k17-login1:~/public_html/24T1_1511/wk8_code$ dcc linked_list.c -o linked_list --leak-check
z5163340@nw-k17-login1:~/public_html/24T1_1511/wk8_code$ ./linked_list
11 8 7
11 8 7 6
11 8 5 7 6
11 8 7 6

Error: free not called for memory allocated with malloc in function create_node in linked_list.c at line 55.
```

A Bigger Linked List Program

(Bigger than what we have done so far, smaller than ASSN 2)

(Starter code is in the live lecture code url!)

- *Working with a larger prorgam*
- *Putting linked list in a context*
- *Understanding provided code in multiple files*
- *More variations of linked list operations*

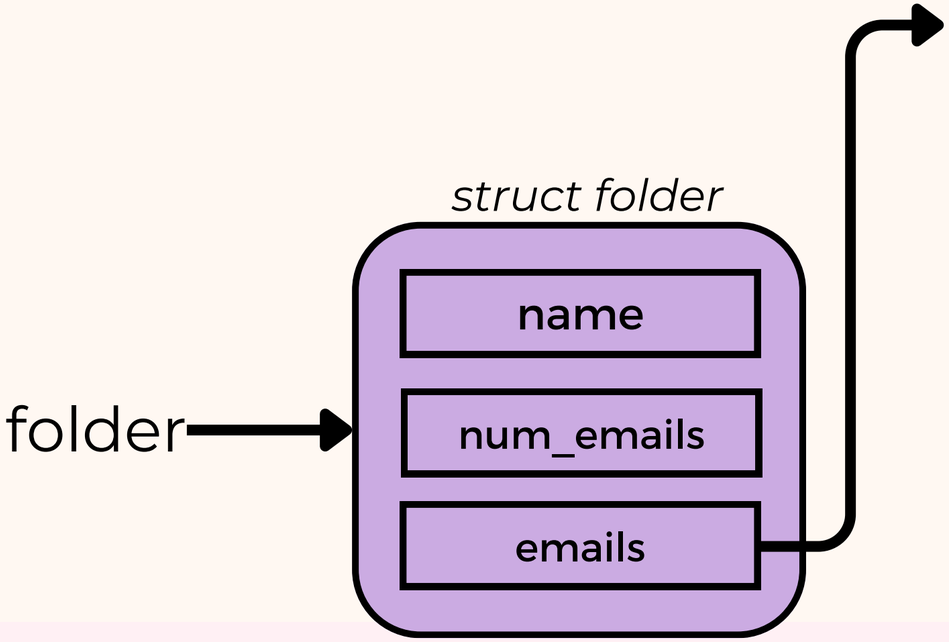
LECTURE LINKED LIST PROGRAM

- Context: **Email Management System**
  - managing emails using linked list
- Files/code provided (3 files):
  - `*email_management_system.c*` (TODO)
  - `*email_management_system.h*` (PROVIDED)
  - `*main.c*` (PROVIDED)
- Task:
  - Complete all `TODO` function definitions in `*email_management_system.c*`
- *Assumption: all emails ever created in a program have unique subjects (to simplify for demo)*

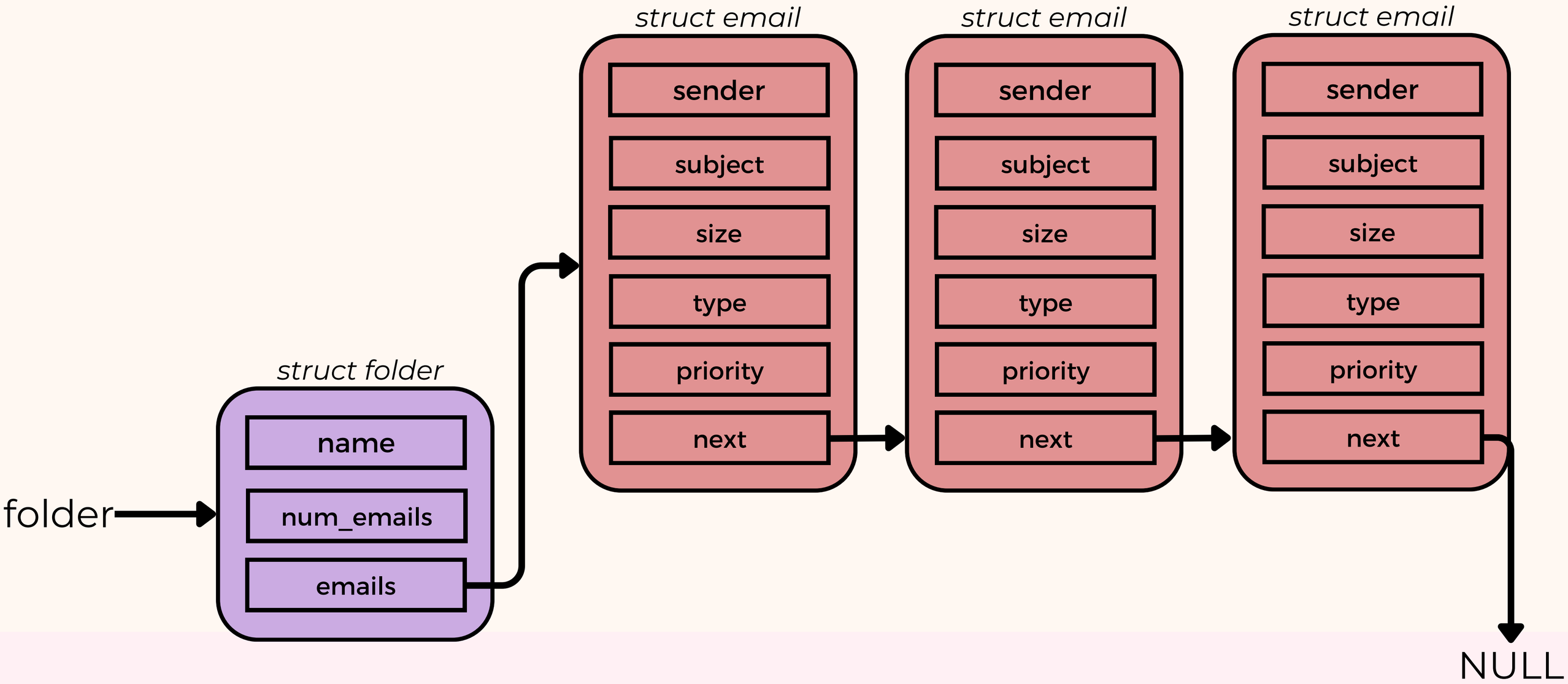
LET' S FIRST UNDERSTAND WHAT THE PROVIDED CODE IS DOING + HOW THEY CONNECT TOGETHER

A VISUAL REPRESENTATION OF WHAT THE LINKED LIST FOR THIS PROGRAM CAN LOOK LIKE...

A VISUAL REPRESENTATION OF WHAT THE LINKED LIST FOR THIS PROGRAM CAN LOOK LIKE...



A VISUAL REPRESENTATION OF WHAT THE LINKED LIST FOR THIS PROGRAM CAN LOOK LIKE...



CODING TIME!

Email Management System

BREAK TIME!

MORE CODING TIME!

Email Management System

SUMMARY OF TODAY

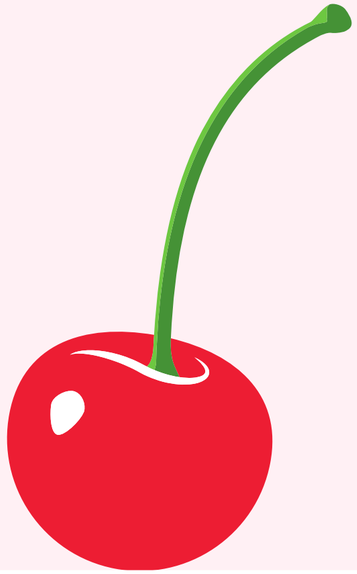
- Wrapped up linked list :)
- Lecture Program on linked list

NEXT LECTURE

Sasha is back :D

Exam Info time!

EXAM!



FEEDBACK (PRETTY PLEASE WITH A CHERRY ON TOP)

Thank you for taking time to give me feedback!

https://forms.office.com/r/Cn8FgdFPhu

If you have any questions:

COURSE RELATED

COURSE FORUM + HELP SESSIONS!

ADMIN RELATED

CS1511@UNSW.EDU.AU

THANK YOU

Come say hi if you see me around on campus :D

GOODBYE

GOOD LUCK!