

EECS 22: Advanced C Programming

Lecture 15 (TuTh)

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Part 1: Overview

- Dynamic Data Structures
 - Double-linked List
 - Append and prepend operations
 - Remove first and last entries
 - Example: List of student records (continued)
 - `Student.h`
 - `Student.c`
 - `StudentList.h`
 - `StudentList.c`
 - `Makefile`

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

Length	0
First	●
Last	●

1. Empty list

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
3

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

Length	1
First	●
Last	●

List	●
Next	●
Prev	●
Student	●

ID	1002
Name	"John Doe"
Grade	'C'

1. Empty list
2. Add a student

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
4

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

The diagram shows a header node with fields Length (2), First, and Last. The First pointer points to the first node, and the Last pointer points to the second node. The first node has List, Next, Prev, and Student pointers. Its Next pointer points to the second node, and its Prev pointer points to the header's First pointer. The second node has List, Next, Prev, and Student pointers. Its Prev pointer points to the first node, and its Next pointer points to the header's Last pointer. Both nodes point to their respective student records.

ID	1002	1003
Name	"John Doe"	"Jim Doe"
Grade	'C'	'B'

1. Empty list
2. Add a student
3. Append a student

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
5

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

The diagram shows a header node with fields Length (3), First, and Last. The First pointer points to the first node, and the Last pointer points to the third node. The first node has List, Next, Prev, and Student pointers. Its Next pointer points to the second node, and its Prev pointer points to the header's First pointer. The second node has List, Next, Prev, and Student pointers. Its Next pointer points to the third node, and its Prev pointer points to the first node. The third node has List, Next, Prev, and Student pointers. Its Prev pointer points to the second node, and its Next pointer points to the header's Last pointer. All three nodes point to their respective student records.

ID	1001	1002	1003
Name	"Jane Doe"	"John Doe"	"Jim Doe"
Grade	'A'	'C'	'B'

1. Empty list
2. Add a student
3. Append a student
4. Prepend a student

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
6

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

Length	42
First	•
Last	•

List	•	•	•	•
Next	•	•	•	•
Prev	•	•	•	•
Student	•	•	•	•

ID	1001	1002	1003	...	1042
Name	"Jane Doe"	"John Doe"	"Jim Doe"	...	"Z End"
Grade	'A'	'C'	'B'	...	'A'

1. Empty list
2. Add a student
3. Append a student
4. Prepend a student
5. ...

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
7

Dynamic Data Structures

- Double-Linked List (with header)
 - Example: List of Student Records

Length	42
First	•
Last	•

List	•	•	•	•
Next	•	•	•	•
Prev	•	•	•	•
Student	•	•	•	•

ID	1001	1002	1003	...	1042
Name	"Jane Doe"	"John Doe"	"Jim Doe"	...	"Z End"
Grade	'A'	'C'	'B'	...	'A'

```

typedef struct Student
    STUDENT;
typedef struct StudentList
    SLIST;
typedef struct StudentEntry
    SENTRY;

struct StudentList
{
    int Length;
    SENTRY *First;
    SENTRY *Last;
};

struct StudentEntry
{
    SLIST *List;
    SENTRY *Next;
    SENTRY *Prev;
    STUDENT *Student;
};

struct Student
{
    int ID;
    char Name[SLEN+1];
    char Grade;
};
                
```

EECS22: Advanced C Programming, Lecture 20
(c) 2017 R. Doemer
8

Dynamic Data Structures

- Example `student.h`

```

/* Student.h: header file for student records */
#ifndef STUDENT_H
#define STUDENT_H

#define SLEN 40

struct Student
{
    int ID;
    char Name[SLEN+1];
    char Grade;
};
typedef struct Student STUDENT;

/* allocate a new student record */
STUDENT *NewStudent(int ID, char *Name, char Grade);

/* delete a student record */
void DeleteStudent(STUDENT *s);

/* print a student record */
void PrintStudent(STUDENT *s);

#endif /* STUDENT_H */

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

9

Dynamic Data Structures

- Example `studentList.h` (part 1/2)

```

/* StudentList.h: header file for lists of student records */
#ifndef STUDENT_LIST_H
#define STUDENT_LIST_H

#include "Student.h"

typedef struct StudentList SLIST;
typedef struct StudentEntry SENTRY;

struct StudentList
{
    int Length;
    SENTRY *First;
    SENTRY *Last;
};

struct StudentEntry
{
    SLIST *List;
    SENTRY *Next;
    SENTRY *Prev;
    STUDENT *Student;
};

...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

10

Dynamic Data Structures

- Example `studentList.h` (part 2/2)

```

...
/* allocate a new student list */
SLIST *NewStudentList(void);

/* delete a student list (and all entries) */
void DeleteStudentList(SLIST *l);

/* append a student at end of list */
void AppendStudent(SLIST *l, STUDENT *s);

/* prepend a student at beginning of list */
void PrependStudent(SLIST *l, STUDENT *s);

/* remove the first student from the list */
STUDENT *RemoveFirstStudent(SLIST *l);

/* remove the last student from the list */
STUDENT *RemoveLastStudent(SLIST *l);

/* print a student list */
void PrintStudentList(SLIST *l);

#endif /* STUDENT_LIST_H */

/* EOF */

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

11

Dynamic Data Structures

- Example `studentList.c` (part 1/9)

```

/* StudentList.c: maintaining lists of student records */

#include "StudentList.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

/* allocate a new student entry */
static SENTRY *NewStudentEntry(STUDENT *s)
{
    SENTRY *e;
    e = malloc(sizeof(SENTRY));
    if (! e)
        { perror("Out of memory! Aborting...");
          exit(10);
        } /* fi */
    e->List = NULL;
    e->Next = NULL;
    e->Prev = NULL;
    e->Student = s;
    return e;
} /* end of NewStudentEntry */

...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

12

Dynamic Data Structures

- Example `studentList.c` (part 2/9)

```

.../* delete a student entry */
static STUDENT *DeleteStudentEntry(SENTRY *e)
{
    STUDENT *s;
    assert(e);
    s = e->Student;
    free(e);
    return s;
} /* end of DeleteStudentEntry */

/* allocate a new student list */
SLIST *NewStudentList(void)
{
    SLIST *l;
    l = malloc(sizeof(SLIST));
    if (! l)
        { perror("Out of memory! Aborting...");
          exit(10);
        } /* fi */
    l->Length = 0;
    l->First = NULL;
    l->Last = NULL;
    return l;
} /* end of NewStudentList */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

13

Dynamic Data Structures

- Example `studentList.c` (part 3/9)

```

...
/* delete a student list (and all entries) */
void DeleteStudentList(SLIST *l)
{
    SENTRY *e, *n;
    STUDENT *s;
    ...

    assert(l);
    e = l->First;
    while(e)
        { n = e->Next;
          s = DeleteStudentEntry(e);
          DeleteStudent(s);
          e = n;
        }
    free(l);
} /* end of DeleteStudentList */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

14

Dynamic Data Structures

- Example `studentList.c` (part 4/9)

```

.../* append a student at end of list */
void AppendStudent(SLIST *l, STUDENT *s)
{
    SENTRY *e = NULL;
    assert(l);
    assert(s);
    e = NewStudentEntry(s);
    if (l->Last)
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = l->Last;
        l->Last->Next = e;
        l->Last = e;
    }
    else
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = NULL;
        l->First = e;
        l->Last = e;
    }
    l->Length++;
} /* end of AppendStudent */

```

EECS ...

Dynamic Data Structures

- Example `studentList.c` (part 5/9)

```

.../* prepend a student at beginning of list */
void PrependStudent(SLIST *l, STUDENT *s)
{
    SENTRY *e = NULL;
    assert(l);
    assert(s);
    e = NewStudentEntry(s);
    if (l->First)
    {
        e->List = l;
        e->Next = l->First;
        e->Prev = NULL;
        l->First->Prev = e;
        l->First = e;
    }
    else
    {
        e->List = l;
        e->Next = NULL;
        e->Prev = NULL;
        l->First = e;
        l->Last = e;
    }
    l->Length++;
} /* end of PrependStudent */

```

EECS ...

Dynamic Data Structures

- Example `studentList.c` (part 6/9)

```

.../* remove the first student from the list */
STUDENT *RemoveFirstStudent(SLIST *l)
{
    SENTRY *e = NULL;
    assert(l);
    if (l->First)
    {
        e = l->First;
        l->First = e->Next;
        if (l->First)
        {
            l->First->Prev = NULL;
        }
        else
        {
            assert(l->Last == e);
            l->Last = NULL;
        }
        l->Length--;
        return DeleteStudentEntry(e);
    }
    else
    {
        return(NULL);
    }
} /* end of RemoveFirstStudent */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

17

Dynamic Data Structures

- Example `studentList.c` (part 7/9)

```

.../* remove the last student from the list */
STUDENT *RemoveLastStudent(SLIST *l)
{
    SENTRY *e = NULL;
    assert(l);
    if (l->Last)
    {
        e = l->Last;
        l->Last = e->Prev;
        if (l->Last)
        {
            l->Last->Next = NULL;
        }
        else
        {
            assert(l->First == e);
            l->First = NULL;
        }
        l->Length--;
        return DeleteStudentEntry(e);
    }
    else
    {
        return(NULL);
    }
} /* end of RemoveLastStudent */
...

```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

18

Dynamic Data Structures

- Example `studentList.c` (part 8/9)

```

...

/* print a student list */
void PrintStudentList(SLIST *l)
{
    SENTRY *e;
    assert(l);
    e = l->First;
    while(e)
    { PrintStudent(e->Student);
      e = e->Next;
    }
} /* end of PrintStudentList */

...

```

Dynamic Data Structures

- Example `studentList.c` (part 9/9)

```

#ifdef MAIN
int main(void)
{
    STUDENT *s = NULL;
    SLIST *l = NULL;
    l = NewStudentList();
    s = NewStudent(1001, "Jane Doe", 'A');
    AppendStudent(l, s);
    s = NewStudent(1002, "John Doe", 'C');
    AppendStudent(l, s);
    s = NewStudent(1000, "New Kid", 'F');
    PrependStudent(l, s);
    PrintStudentList(l);
    s = RemoveFirstStudent(l);
    AppendStudent(l, s);
    s = RemoveLastStudent(l);
    PrependStudent(l, s);
    DeleteStudentList(l);
    l = NULL;
    return 0;
} /* end of main */
#endif /* MAIN */
/* EOF */

```

Dynamic Data Structures

- Example Session

```
% vi StudentList.c
% make
gcc -Wall -ansi -std=c99 -g -c Student.c -o Student.o
gcc -DMAIN -Wall -ansi -std=c99 -g StudentList.c Student.o
-o StudentList
% valgrind ./StudentList
==5908== Memcheck, a memory error detector
Student ID: 1000
Student Name: New Kid
Student Grade: F
Student ID: 1001
Student Name: Jane Doe
Student Grade: A
Student ID: 1002
Student Name: John Doe
Student Grade: C
==5908== HEAP SUMMARY:
==5908==   in use at exit: 0 bytes in 0 blocks
==5908== total heap usage: 9 allocs, 9 frees, 328 bytes allocated
==5908== All heap blocks were freed -- no leaks are possible
==5908== ERROR SUMMARY: 0 errors from 0 contexts
%
```

EECS22: Advanced C Programming, Lecture 20

(c) 2017 R. Doemer

21

Part 2: Overview

- Dynamic Data Structures
 - Double-linked list
 - Element insertion before another list element
 - Element insertion after another list element
 - Element deletion (left for exercise at home)
 - Example: Student records
 - `StudentList.h`, `StudentList.c`
 - New `InsertStudentBefore`
 - New `InsertStudentAfter`
- Data Display Debugger
 - Graphical data structure display in `ddd`

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

22

Dynamic Data Structures

- Example `studentList.h` (part 1/2)

```

/* StudentList.h: header file for lists of student records */
#ifndef STUDENT_LIST_H
#define STUDENT_LIST_H

#include "Student.h"

typedef struct StudentList SLIST;
typedef struct StudentEntry SENTRY;

struct StudentList
{
    int Length;
    SENTRY *First;
    SENTRY *Last;
};

struct StudentEntry
{
    SLIST *List;
    SENTRY *Next;
    SENTRY *Prev;
    STUDENT *Student;
};

...

```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

23

Dynamic Data Structures

- Example `studentList.h` (part 2/2)

```

...
/* allocate a new student list */
SLIST *NewStudentList(void);

/* delete a student list (and all entries) */
void DeleteStudentList(SLIST *l);

/* prepend/append a student at beginning/end of list */
void PrependStudent(SLIST *l, STUDENT *s);
void AppendStudent(SLIST *l, STUDENT *s);

/* insert a student before/after an existing one */
void InsertStudentBefore(SENTRY *e, STUDENT *s);
void InsertStudentAfter(SENTRY *e, STUDENT *s);

/* remove the first/last student from the list */
STUDENT *RemoveFirstStudent(SLIST *l);
STUDENT *RemoveLastStudent(SLIST *l);

/* print a student list */
void PrintStudentList(SLIST *l);

#endif /* STUDENT_LIST_H */

/* EOF */

```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

24

Dynamic Data Structures

- Example `studentList.c` (part 1/4)

```

/* StudentList.c: maintaining lists of student records */
/* author: Rainer Doemer */
/* modifications: */
/* 11/10/11 RD added InsertStudentBefore, InsertStudentAfter */
/* 11/08/11 RD version for double-linked lists */
/* 11/03/11 RD initial version */

#include "StudentList.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

/* unmodified functions omitted here for brevity */

...

```

Dynamic Data Structures

- Example `studentList.c` (part 2/4)

```

...

/* insert a student before an existing one */
void InsertStudentBefore(SENTRY *e, STUDENT *s)
{
    SENTRY *New;
    New = NewStudentEntry(s);
    New->List = e->List;
    New->Next = e;
    New->Prev = e->Prev;
    e->Prev = New;
    if (New->Prev)
    { New->Prev->Next = New;
    }
    else
    { assert(New->List->First == e);
      New->List->First = New;
    }
    New->List->Length++;
} /* end of InsertStudentBefore */

...

```

Dynamic Data Structures

- Example `studentList.c` (part 3/4)

```

...

/* insert a student after an existing one */
void InsertStudentAfter(SENTRY *e, STUDENT *s)
{
    SENTRY *New;
    New = NewStudentEntry(s);
    New->List = e->List;
    New->Next = e->Next;
    New->Prev = e;
    e->Next = New;
    if (New->Next)
    { New->Next->Prev = New;
    }
    else
    { assert(New->List->Last == e);
      New->List->Last = New;
    }
    New->List->Length++;
} /* end of InsertStudentAfter */

...

```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

27

Dynamic Data Structures

- Example `studentList.c` (part 4/4)

```

#ifdef MAIN
int main(void)
STUDENT *s = NULL;
SLIST *l = NULL;
SENTRY *e = NULL;
l = NewStudentList();
s = NewStudent(1002, "Jim Doe", 'B');
AppendStudent(l, s);
e = l->First;
assert(e->Student == s);
s = NewStudent(1003, "John Doe", 'C');
InsertStudentAfter(e, s);
s = NewStudent(1001, "Jane Doe", 'A');
InsertStudentBefore(e, s);

PrintStudentList(l);

DeleteStudentList(l);
l = NULL;
return 0;
} /* end of main */
#endif /* MAIN */
/* EOF */

```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

28

Dynamic Data Structures

- Example Session

```
% vi StudentList.c
% vi Makefile
% make
gcc -DMAIN -Wall -ansi -std=c99 -g StudentList.c Student.o
-o StudentList
% valgrind ./StudentList
==5908== Memcheck, a memory error detector
Student ID: 1001
Student Name: Jane Doe
Student Grade: A
Student ID: 1002
Student Name: Jim Doe
Student Grade: B
Student ID: 1003
Student Name: John Doe
Student Grade: C
==5908== HEAP SUMMARY:
==5908==    in use at exit: 0 bytes in 0 blocks
==5908== total heap usage: 7 allocs, 7 frees, 264 bytes allocated
==5908== All heap blocks were freed -- no leaks are possible
==5908== ERROR SUMMARY: 0 errors from 0 contexts
%
```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

29

Graphical Data Structure Display

- Data Display Debugger `ddd`

- Graphical frontend for GDB
 - Requires X forwarding client (e.g. *Xming* in addition to *Putty*)
- Displays separate windows
 - Menu bar and command buttons
 - Graphical display area for *pointers* and *data structures!*
 - Source code (and assembly code) browser
 - Command line interface
- Can display data structures built by pointers as graphs
 - Very useful tool to visualize and debug dynamic data structures
- Example: `StudentList.c`

```
% vi StudentList.c
% make StudentList
gcc -Wall -ansi -std=c99 -g -c Student.c -o Student.o
gcc -DMAIN -Wall -ansi -std=c99 -g StudentList.c Student.o
-o StudentList
% ddd ./StudentList
```

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

30

Graphical Data Structure Display

The screenshot shows a debugger window titled "DDD: /users/faculty/doemer/eeecs22/lecture17/StudentSort.c". The main display area shows a graphical representation of a doubly-linked list. At the top, a box indicates the list's state: Length = 3, First = 0x603070, Last = 0x603150. Below this, three nodes are displayed, each with its own List structure (List, Next, Prev, Student) and student details (ID, Name, Grade). The nodes are: ID 1001 (Jane Doe, Grade A), ID 1002 (John Doe, Grade C), and ID 1000 (New Kid, Grade F). Arrows indicate the Next and Prev pointers between nodes. The bottom of the window shows the source code for the list operations:

```

SLIST *l = NULL;
l = NewStudentList();
AppendStudent(l, NewStudent(1001, "Jane Doe", 'A'));
AppendStudent(l, NewStudent(1002, "John Doe", 'C'));
AppendStudent(l, NewStudent(1000, "New Kid", 'F'));
AppendStudent(l, NewStudent(1003, "Jane Doe", 'B'));
  
```

The debugger also shows the command: `(gdb) graph display *(l->Last->Student) dependent on 4` and the current display: `Display 9: *(l->First->Student) (enabled, scope main, address 0x603030)`.

EECS22: Advanced C Programming, Lecture 21

(c) 2017 R. Doemer

31