EECS 22: Advanced C Programming

Lecture 17

Rainer Dömer

doemer@uci.edu

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

Lecture 17: Overview

- Dynamic Memory Allocation
  - Dynamic Memory Errors
  - Validating Dynamic Memory Usage
  - **`valgrind`**

Dynamic Memory Allocation

- Typical Dynamic Memory Usage Errors
  - Omitting **`malloc()`**: Access to unallocated memory
  - Reading uninitialized memory
  - Omitting **`free()`**: Memory *leak*
  - Freeing memory too early, or multiple times
  - …
- Validating Dynamic Memory Usage
  - **`valgrind`**: A memory error detector (and more)
    - Instruments the program at (right before) run-time
    - Intercepts and checks calls to **`malloc()`** and **`free()`**
    - Intercepts and checks memory accesses
    - Reports any errors to the user (or a log file)
  - ➢ Use **`valgrind`** for testing and debugging!
    - There should be 0 errors and 0 bytes leaked!

Dynamic Memory Allocation

- Example Student Records: **`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 */
```

Dynamic Memory Allocation

- Example Student Records: **`Student.c`** (part 1/3)

```
/* Student.c: maintaining student records */

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

/* allocate a new student record */
STUDENT *NewStudent(int ID, char *Name, char Grade)
{   STUDENT *s;
    s = malloc(sizeof(STUDENT));
    if (! s)
       { perror("Out of memory! Aborting...");
         exit(10);
       } /* fi */
    s->ID = ID;
    strncpy(s->Name, Name, SLEN);
    s->Name[SLEN] = '\0';
    s->Grade = Grade;
    return s;
} /* end of NewStudent */
...
```


Dynamic Memory Allocation

- Example Student Records: **`Student.c`** (part 2/3)

```
...

/* delete a student record */
void DeleteStudent(STUDENT *s)
{
    assert(s);
    free(s);
} /* end of DeleteStudent */

/* print a student record */
void PrintStudent(STUDENT *s)
{
    assert(s);
    printf("Student ID:    %d\n", s->ID);
    printf("Student Name:  %s\n", s->Name);
    printf("Student Grade: %c\n", s->Grade);
} /* end of PrintStudent */

...
```

Dynamic Memory Allocation

- Example Student Records: **`Student.c`** (part 3/3)

```
...
/* test the student record functions */
int main(void)
{   STUDENT *s1 = NULL, *s2 = NULL;
    printf("Creating 2 student records...\n");
    s1 = NewStudent(1001, "Jane Doe", 'A');
    s2 = NewStudent(1002, "John Doe", 'C');

    printf("Printing the student records...\n");
    PrintStudent(s1);
    PrintStudent(s2);

    printf("Deleting the student records...\n");
    DeleteStudent(s1);
    s1 = NULL;
    DeleteStudent(s2);
    s2 = NULL;

    printf("Done.\n");
    return 0;
} /* end of main */

/* EOF */
```


Dynamic Memory Allocation

- Example Student Records: **`Makefile`**

```
# Makefile: Student Records

# macro definitions
CC = gcc
DEBUG = -g
#DEBUG = -O2
CFLAGS = -Wall -ansi -std=c99 $(DEBUG) -c
LFLAGS = -Wall $(DEBUG)

# dummy targets
all: Student

clean:
        rm -f *.o
        rm -f Student

# compilation rules
Student.o: Student.c Student.h
        $(CC) $(CFLAGS) Student.c -o Student.o

Student: Student.o
        $(CC) $(LFLAGS) Student.o -o Student

# EOF
```

Dynamic Memory Allocation

- Example Session

```
% vi Student.h
% vi Student.c
% vi Makefile
% make
gcc -Wall -ansi -std=c99 -g -c Student.c -o Student.o
gcc -Wall -g Student.o -o Student
% ./Student
Creating 2 student records...
Printing the student records...
Student ID:    1001
Student Name:  Jane Doe
Student Grade: A
Student ID:    1002
Student Name:  John Doe
Student Grade: C
Deleting the student records...
Done.
%
```


Dynamic Memory Allocation

- Example Session

```
% valgrind ./Student
==23638== Memcheck, a memory error detector
==23638== […]
==23638== Command: Student
Creating 2 student records...
Printing the student records...
Student ID:    1001
Student Name:  Jane Doe
Student Grade: A
Student ID:    1002
Student Name:  John Doe
Student Grade: C
Deleting the student records...
Done.
==23638== HEAP SUMMARY:
==23638==     in use at exit: 0 bytes in 0 blocks
==23638==   total heap usage: 2 allocs, 2 frees, 96 bytes allocated
==23638==
==23638== All heap blocks were freed -- no leaks are possible
==23638== ERROR SUMMARY: 0 errors from 0 contexts […]
%
```