

CSSE 490-04

Dynamic Storage Reclamation
Course Introduction

Roll call and introductions

- Name (nickname)
- Hometown
- Local residence
- Major(s)
- Something exciting you did over the summer break

Administrivia!

- Background
- Syllabus
- Schedule
- Index page
- First assignment due next Friday

Course logistics

- Goals
 - Explore GC
 - Perform research
- Discussion and presentations
 - Read and present papers
- Individual or group project
 - Build a collector
- Documentation
 - Important part of project

Key concepts in managing memory

- Key challenges and key ideas
 - **Explicit vs Automated memory management**
 - In which languages is each done?
 - Why?
 - **Memory allocation**
 - Contiguous allocation
 - Free-list allocation
 - **Memory reclamation**
 - Tracing
 - Reference counting

What is memory management?

- Programs contain
 - Objects
 - Data
 - Occupy memory
- Runtime system must allocate and reclaim memory for program in an efficient manner
 - Why is this important?
 - Why is this hard?
 - Why is this interesting?

Allocation and Reclamation

- Allocation
 - Objects dynamically allocated on HEAP
 - `malloc()`, `new()`
- Reclamation
 - Manual/Explicit
 - `free()`
 - `delete()`
 - Automated
 - Garbage collection (GC)

Explicit memory management challenges

- Consumes software development time
 - new → allocate storage for new object
 - delete → reclaim storage
- Prone to software faults (reclaim too soon)

```
Foo* p = new Foo();  
Foo* q = p;  
delete p;  
p->DoSomething();  
p = NULL;  
q->ProcessFoo();
```

- **Statically undecidable**
- **Problem for developers**

Explicit memory management challenges

- Memory leak (never reclaim)

```
#include <stdlib.h>
void f(void){
    void* s;
    s = malloc(50);
    return;
}

int main(void){
    while (1) f();
    return 0;
}
```

Explicit memory management pluses

- Efficiency can be very high
- Puts the programmer in control

Automated memory management

- Runtime system automatically
 - Detects dead objects (garbage detection)
 - Reclaims dead objects (garbage reclamation)
 - Garbage collection
- Preserves software development time
 - Relieves programmer burden
 - Less prone to errors
- Utilized by most modern OOP and scripting languages
 - Python, Java, C#, php

Garbage collection challenges

- Occurs an unpredictable times
- Duration is unbounded
- Performance efficiency issues

```
public void f(){
    startLaser();
    Obj o = new Obj();
    stopLaser();
}

public static void main(...){
    while (true) f();
}
```



Time

GC, Bad
for Real-
Time

Major concerns

- Explicit memory management
 - Reclaiming objects at the right time
- Garbage collection
 - Discriminating **live** objects from garbage
- Both
 - Fast allocation
 - Fast reclamation
 - Low fragmentation