# CSc 553

Principles of Compilation

13 : Garbage Collection — Uncooperative Languages

Department of Computer Science
University of Arizona

collberg@gmail.com

## Introduction

## Uncooperative Languages

There is some information which is necessary in order to perform automatic memory management:

1. We need to find the roots of the object graph, i.e. the pointers from the stack, registers, or global variables which point to objects on the heap.
2. We need to know the size, the beginning, and end of each object.
3. For each object we need to find which of its fields are pointers.

- Unfortunately, some languages have been designed so that it is impossible to determine this information.
- C and C++ are the two most popular such languages.

## Uncooperative Languages...

- C and C++ don't separate safe and unsafe features (such as address and bit manipulation) which are sometimes needed in systems programming.
- Modula-3 has similar unsafe features as C and C++ but they can be encapsulated into unsafe modules, which don't mess up the safety of the main (safe) part of the program.

## Uncooperative Languages...

- Most GC algorithms assume that there is always a pointer to the beginning of every object. Depending on the code generator, that may or may not be true.

  ```
  f(g,s) char (*g)(); char * s;
  {  int i; int l = strlen(s);
     for (i = 0; i < l; i++)
        s[i] = (*g)(s[i]); }
  ```

  There may be no pointer to `s[0]`.

## Uncooperative Languages...

We need to know

1. the roots of the object graph.
2. the size, the beginning, and end of each object.
3. which object fields are pointers.

**Finding Roots:**

```
Foo* f = new foo; // f = 0x53f36
f = NULL;         // f* is garbage
int i = 0x53f36;  // points to f...
```

## Uncooperative Languages...

**Finding the beginning:**

```
char* str = new char[26];
strcpy(str, "This is a string");
str += 10;     // Only ptr to str...
```

**Finding pointers:**

```
union Unsure {char* str; int i} x;
```

## Conservative GC

- Works OK for uncooperative languages (C, C++) where we can't distinguish between pointers and integers. Sometimes fails to reclaim all garbage.

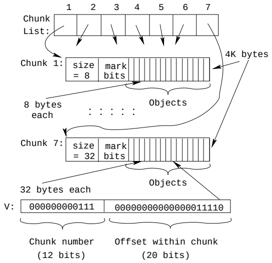
**Main Ideas:**

- Allocate memory in chunks. Each chunk holds a collection of objects of a certain size (i.e. it's easy to find the start of objects).
- Chunks are numbered. A pointer consists of 12 bits of chunk number ($C$) + 20 bits of offset within the chunk ($O$).

## Conservative GC. . .

- To check whether a value $V = (C, O)$ is a pointer to some object we check that
  1. Heap-bottom $\leq V \leq$ Heap-top,
  2. FirstChunk# $\leq C \leq$ LastChunk#
  3. the offset $O$ is a multiple of the object size in chunk $C$.

## Conservative GC. . .



## Readings and References

- Read Scott, pp. 389.