15-213 / 18-243
Recitation Week-10

Miray Kas

07/18/2011

Optimizing your program

- You have to get everything working first
- Premature optimization will give you headaches.
- Gprof: Profiler in the hints part.

What is Gprof?

- *Gprof* is a profiling program which collects and arranges statistics on your programs.

- Using it for malloc lab???

Step-by-Step

- bambooshark.ics.cs.cmu.edu
- Change the make file to include –pg flag
  - CFLAGS = -Wall -Wextra -Werror -O2 -pg –DDRIVER
- Make
- Run your code so that gmon.out is created:
  - ./mdriver -V -f traces/malloc.rep
- Run the profiles
  - gprof mdriver > rep_stats

Your gprof results: rep_stats (The file name you gave to gprof)

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
63.19	0.12	0.12	14481156	0.00	0.00	get_counter
31.60	0.18	0.06	10	6.00	18.01	start_comp_counter
5.27	0.19	0.01	1	10.01	10.01	mhz
0.00	0.19	0.00	120	0.00	0.00	mem_sbrk
0.00	0.19	0.00	120	0.00	0.00	mm_malloc
0.00	0.19	0.00	20	0.00	0.00	mem_heap_hi
0.00	0.19	0.00	20	0.00	0.00	mem_heap_lo
0.00	0.19	0.00	16	0.00	0.00	rio_readlineb
0.00	0.19	0.00	12	0.00	0.00	mem_reset_brk
0.00	0.19	0.00	12	0.00	0.00	mm_init
0.00	0.19	0.00	12	0.00	0.00	reinit_trace

What is each field?

- **% time**: Percentage of the total execution time your program spent in this function. These should all add up to 100%.
- **Cumulative seconds:** Cumulative total number of seconds the computer spent executing this functions, plus the time spent in all the functions above this one in this table.
- **self seconds:** Number of seconds for this function alone.
- **Calls:** Number of times the function was called. Blank if never called.
- **self ms/call:** Average number of milliseconds spent in this function per call.
- **total ms/call:** Average number of milliseconds spent in this function and its descendants per call.
- **name :** Name of the function.

Valgrind

- Valgrind is a multipurpose code profiling and memory debugging tool for Linux
- Login to a shark machine, go to your folder
- Run valgrind:
  - valgrind --tool=memcheck --leak-check=yes ./mdriver
  - valgrind --tool=memcheck --leak-check=yes ./mdriver -V -f traces/malloc.rep
- To redirect to an output file, instead of screen:
  - valgrind --tool=memcheck --leak-check=yes ./mdriver >& valgrind_outputs

Valgrind Example

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char *p;
     // Allocation #1 of 19 bytes
    p = (char *) malloc(19);
    // Allocation #2 of 12 bytes
    p = (char *) malloc(12);
    free(p);
    // Allocation #3 of 16 bytes
    p = (char *) malloc(16);
    return 0;
}
```

Compile:

gcc -o test -g test.c

Call Valgrind:

valgrind --tool=memcheck --leak-check=yes ./test

What is the problem?

Valgrind Example

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char *p;
     // Allocation #1 of 19 bytes
    p = (char *) malloc(19);
    // Allocation #2 of 12 bytes
    p = (char *) malloc(12);
    free(p);
    // Allocation #3 of 16 bytes
    p = (char *) malloc(16);
    return 0;
}
```

Allocation #1 (19 byte leak) is lost because p is pointed elsewhere before it is free'd.

Valgrind says it is (test.c:9)

Similar for Allocation #3

Static Keyword

- Applied to global variable
- Applied to a function

```
static int x;
static int add(int a, int b)) { ... }
```

The scope of the variables/functions decreases from the entire project to the current source file.

- Applied to local variable

```
void accumulate(int x) {
    static int total;
    total += x;
}
```

The value of total will remain in the memory and will be accumulated every time accumulate function is called.

mm_init

- mm_init function forbade the calling of mem_init. How to create the heap if we can't do this?
  - Get the tablehead and heapend.
    - For heapend, mem_heap_hi is useful
    - For tablehead, mem_sbrk is useful
    - mem_sbrk extends the heap pointer by incr bytes and returns the start address of the new area.
    - Basically, you need to extend heap by table_size + hdft_size
  - Create and initialize the freelist table at the beginning of the heap.
  - Append the prologue and epilogue blocks after the table.