CS 214 Recitation(Sec. 6)

Zuohui Fu
Ph.D. Department of Computer Science
Office hour: Mon 2pm-3pm
Email: zf87 AT cs dot rutgers dot edu
09/26/2017

Topics

- GDB - GNU Project Debugger
- Dynamic Memory Allocation
- How to use Sizeof
- String Library

GDB - GNU Project Debugger

- The purpose of a debugger such as GDB is to allow you to see what is going on “inside” another program while it executes—or what another program was doing at the moment it crashed.
- For how to use it, please refer to:
  - http://www.gnu.org/software/gdb/documentation/

GDB - GNU Project Debugger

- Useful commands
  - list: l 15, will make 10 lines of code appear, the line 15 will be in the middle:

```
10              time.Sleep(2 * time.Second)
11              c <- i
12          }
13          close(c)
14      }
15
16      func main() {
17          msg := "Starting main"
18          fmt.Println(msg)
19          bus := make(chan int)
```

  - break
    - b 10, will setup the break point in line 10
    - d 10, delete the break point

- Useful commands(continue)
  - backtrace: bt, print the execute of your codes.

```
#0  main.main () at /home/xiemengjun/gdb.go:23
#1  0x000000000040d61e in runtime.main () at /home/xiemengjun/go/src/pkg/runtime/proc.c:244
#2  0x000000000040d6c1 in schedunlock () at /home/xiemengjun/go/src/pkg/runtime/proc.c:267
#3  0x0000000000000000 in ?? ()
```

  - next: n, debug step by step
  - continue: c 2, jump 2 break points to run
  - looking at variables: , know about the parameters
- Key points:
  - When compile, use –g, ex: gcc -g test.c -o test
  - use gdb to start debug, ex: gdb test

Dynamic Memory Allocation

- malloc:
  - void* malloc (size_t size)

  Assign a fixed size of memory, but not initialize it.
- calloc:
  - void* calloc (size_t n, size_t s)

  Assign and initial n successive memory spaces of size s to 0
- free:

  char *ptr = (char*) calloc (10,10);

  free(ptr);

  use free to release the specified block of memory back to the system

How to use Sizeof

- We can know the memory usage of a given type using the sizeof operator. It takes either a variable name or a type name as parameter.
  - sizeof(type), ex, sizeof(int)
  - sizeof(variable_name), can’t use for function
- EX:

```
int n;
printf((sizeof(n));
```

Example

```
struct X
{
        short s;
        int i;
        char c;;
} ;
```

What is the value of sizeof(X)?

Example-Continue

This is because of padding added to satisfy alignment constraints. Data structure alignment impacts both performance and correctness of programs:

```
struct X
{
short s; /* 2 bytes */
        /* 2 padding bytes */
int i; /* 4 bytes */
char c; /* 1 byte */
        /* 3 padding bytes */
};
```

String Library

- Initialize a string:

  char greeting[] = "Hello";

  char *ptr = (char*) malloc (sizeof(char)*10); // the size of pointer, allocating memory dynamically we assigned a memory of **’10 * sizeof(char)' bytes** for address. We used **(char*)** to typecast the pointer returned by malloc to character.

- #include <string.h> contains:
  - strcpy strlen strcat memset memcpy strtok

https://download.mikroe.com/documents/compilers/mikroc/pic/help/ansi_string_library.htm

atoi()

- The C library function **int atoi(const char *str)** converts the string argument **str** to an integer (type int).
- Overflow may appear
- It takes the sign value.

HW2

0.

```
int i = 5;
int *ip = &i;
```

what is ip? What is its value?

1.

Write some code that declares two arrays of size 10 that are string literals.

Make a pointer to one of the arrays, cast it to be an int pointer, and print out its value.

Make a new integer, set it equal to the value of your int pointer, then make a pointer to that integer, cast it to be a char pointer, and print out 8 chars.

What happened? Why?

2.

Write some code that declares two arrays of size 10 that are string literals.

Create a pointer that points to the beginning of the first array, then in a loop, increment the pointer and print out the char it points to, out to index 20.

What happened? Why?