Homework 2

- Questions?
- Complaints?
- Queries?
Debugger

- Text version is gdb
  - Installed on all `nix machines
  - Compile with the \(-g\) option (inserts db info)
- Set break points
- Monitor Variable values.
Graphical debugger.

- Installed on most `nix machines

If remotely accessing from windows box, need to turn on X-server.

X-server allows remote graphics.
#define PI 3.1415

#include <stdio.h>

int main(void) {
    printf("PI = %.2f\n", PI);
    return 0;
}
#include

- Preprocessor command
- Not a variable
- Either literal or macro
- No semicolon
#define preprocessor

Value plugged in

```c
#define HI "Hello world\n"
int main(void) {
    printf(HI);
    return 0;
}
```
Defines to defines

#define ONE 1
#define TWO ONE+ONE
#define THREE ONE+TWO
more #define

#define TRUE 1
#define FALSE 0

int main(void) {
    if (TRUE) { . . . }
}
Function Macros

```c
#define ABS(a) (a)<0 ? -(a) : (a)

int main(void) {
    int j = (3)<0 ? -(a) : (a);
    return 0;
}
```
Why use `#define`

- Changing magic numbers
  - `#define A_SIZE 100`

- Increases Readability

- Macros faster than Functions.
  - Increase code size vs. memory use.
Quick Array review

- Size?
- The Memory runneth over
- Initialization
Strings

- Not Fundamental type
- Not an object
- Just an array of characters
In memory

“abcd” == {a’, b’, c’, d’, \0}

Must have terminating character ‘\0’
Static memory Strings

- `char ch[10] = "Hi there";`
- `char ch[] = "Hello";`
- `char ch[4] = "abcdef";` **WRONG**
Reading and writing strings

- `Scanf and Printf work.`
  - A bit clunky.

- `%s in scanf`
  - Stops at first white space
  - “this is a test”
getchar() <stdio.h>

- Reads characters one at a time
- Starts at newline
- Still has buffering problem
getchar()

- returns a character

  char ch = getchar();

  ch[0] = getchar();
getchar()

- Filling strings
  - Don’t overflow the size

```c
int main(void) {
  char ch[10];
  int k = 0;
  while (k < 10 && (ch[k++] = getchar()) != ‘\n’);
  ch[k-1] = ‘\0’;
  printf(“%s
”, ch);
  return 0;
}
```
getch() & getche()

- work same as getchar()
- No buffering problem
- Read in as soon at key is pressed
- Not in ANSI C

- Not sure if in gcc.
`putchar(char)` `<stdio.h>`

- just writes a character to the screen.

- Easy stuff. No formatting.
puts(char[]) <stdio.h>

- Prints a string starting at the input address
- Writes until terminator
- will auto newline after terminator
gets(char[])

- reads in an entire string.
- Terminates with newline
- auto-adds terminating character
gets

- Giant buffering problem

- Don’t ever use.

- Don’t ever use.
<string.h>

- **strcat**
  - string concat.

- **strcmp**
  - string compare

- **strcpy**
  - string copy

- **strlen**
  - string length