CSc 352
Bit Manipulation

Benjamin Dicken

What will this print?

```
#define PI 3.1415926535897

double radians(double degrees) {
  return degrees * (PI /180);
}

int main() {
  double r = 100.0;
  double tp = 100.0;
  double a = r * (sin(radians(tp)) * cos(radians(tp)));
  double b = r * sin(radians(tp)) * cos(radians(tp));
  if (a == b) { printf("equal\n"); }
  else { printf("unequal\n"); }
  return 0;
}
```

Announcements

- Student Course Survey
  - 1 PA grade dropped if response percentage gets to 80% or more
  - 76.84% (before class)
- PA 10
- Final exam, May 6, 1-3pm, this room
  - Study guide for Final will be posted Friday
- First 80 bytes of binary file

Data Representation

Each row represents:
studentID, exam 1, exam 2, final exam

How many bytes would it take to represent this with a CSV ASCII file?

How many bytes would it take to represent this in binary? How compact could we get it?

grade_info.csv

```
19311233,80,90,100
91246834,75,85,82
21245122,43,76,87
18673124,90,75,90
```

Implement Conversion

Write the code to:

- Open this text file
- Re-write the same data to **`binary_grade_info.bin`**
- Close the file

grade_info.csv

```
19311233,080,090,100
91246834,075,085,082
21245122,043,076,087
18673124,090,075,090
```

Bit Operations

C supports a number of operations to manipulate the ones and zeros in memory

Shifting:	`>>` `<<`
Masking:	`&` `|` `^`
Flipping:	`~`

```c
uint8_t x, y;
x = 1;      // 00000001
y = x<<2;   // 00000100
y = y>>2;   // 00000001

for (int i = 0; i < 8; i++) {
  y = y<<1;
  printf("%u\n", y);
}
```

Viewing bits on stdout

- Implement the function

```
void print_bits(uint8_t data);
```

- Should print out the 1s and 0s stored in **data** to standard output
- For example:

```
uint8_t x = 4;
print_bits(x); // Should print 00000100
```

Viewing bits on stdout

- Implement the function

```
void print_bits(uint8_t * data, int size);
```

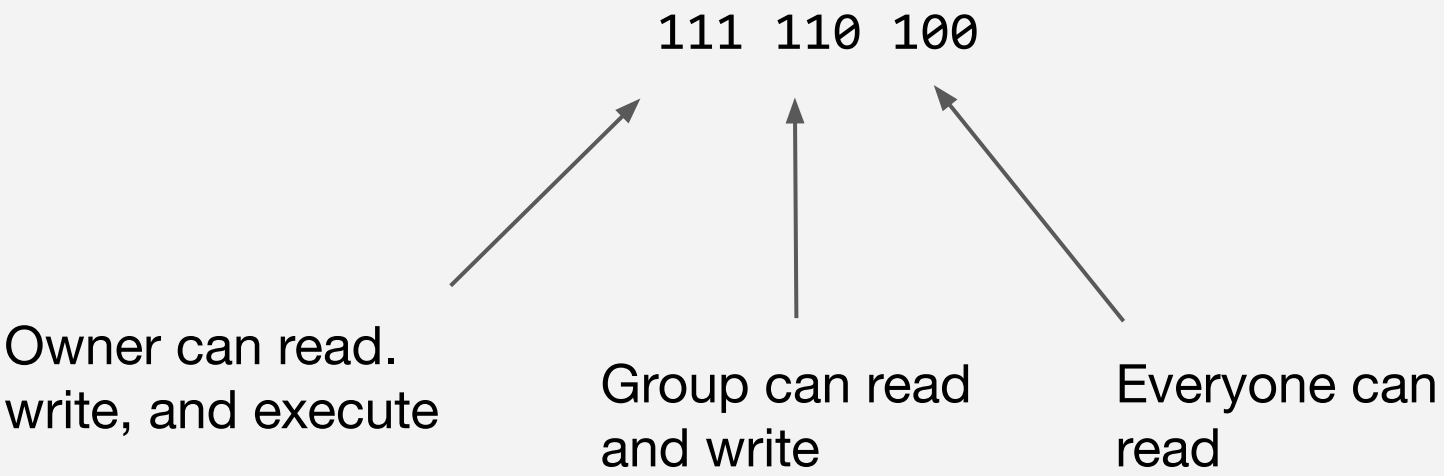
- Should print out the 1s and 0s stored in the array of length **size** that **data** points to
- For example:

```
uint16_t x = 4;
print_bits(x, 2); // Should print 00100000 00000000
```

```c
void print_bits(uint8_t * data, int size){
  uint8_t* copy = malloc(size);
  memcpy(copy, data, size);
  for(int i = 0; i < size; i++){
    for(int j = 0; j < 8; j++){
      uint8_t temp = copy[i];
      temp = temp<<(7-j);
      temp = temp>>7;
      printf("%u", temp != 0 ? 1 : 0);
    }
    printf(" ");
  }
  printf("\n");
  free(copy);
}
```

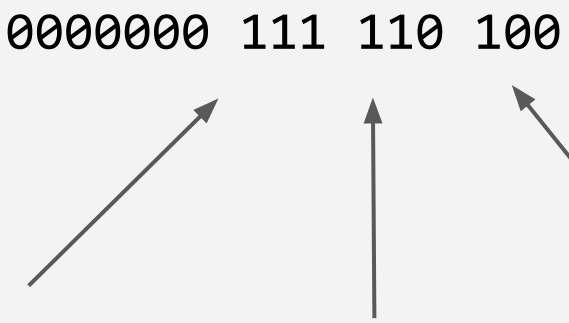
Permissions

Recall that permissions for files can be represented as a binary sequence:



Permissions

Could represent this with a uint16_t



Owner can read. write, and execute

Group can read and write

Everyone can read

Permissions

- Implement the function

```
uint16_t owner_permissions(uint16_t * permissions);
```

- Should take the Owner permissions and set those same permissions as the group and every permissions too, return the number
- For example:

```
uint16_t x = 372; // 000000000 101 110 100
owner_permissions(x); // Should return 000000000 101 101 101
```