This is Week 6

Jason Hirschhorn

Fall, 2011

Agenda

- Announcements
- Review
  - Hexadecimal
  - Structs + Structs.c
- Enumerated Types
  - Rps.c
- File I/O
  - Typewriter.c
  - Printer.c

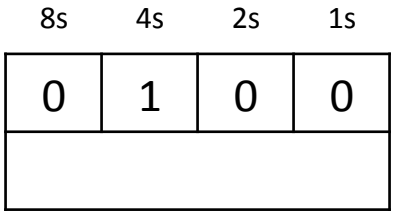
Announcements

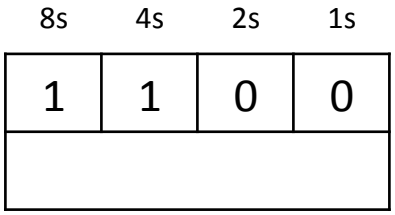
- Problem Set 5 Walkthrough (Sun, 7pm, NW B103) – https://www.cs50.net/psets/
- Office Hours – https://www.cs50.net/ohs/
  - Harvard innovation lab this week
- Lecture videos, slides, source code, scribe notes – https://www.cs50.net/lectures/
- Bulletin Board – http://help.cs50.net
- Quiz 0
  - Glad that's over with…
  - To be returned in section
  - 0 < 5 < 12
- pset5
  - "I Saw You Harvard" competition

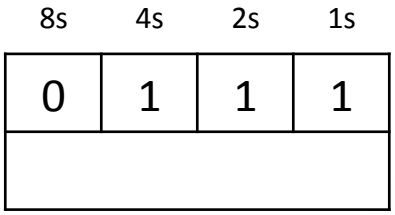
Review

Hexadecimal

- Base 16
  - 0 to 9 then A to F
  - Concise
- Every set of 4 bits can be represented by 1 hex digit
  - “nibble”
- We start hexadecimal numbers with “0x”







Structs

- Bundle together related values
- Field – variable with its own distinct data type
- Access fields with "." or "->"
  - "." used for a normal instance of a struct
  - "->" used for a pointer to a struct

```
struct student
{
        char *name;
        int class_year;
        char *house;
};
```

Structs

Without typedef

```
struct pkmn
{
	char* name;
	char* type;
	int hp;
	bool cute;
};

struct pkmn pikachu;
pikachu.hp = 35;
```

With typedef

```
typedef struct
{
	char* name;
	char* type
	int hp;
	bool cute;
} pkmn;

pkmn charmander;
charmander.hp = 39;
```

Enumerated Types

Enumerated Types

- Create your own type with a finite set of possible values
- Abstraction
  - Easier to read
  - Easier to write
- Examples of possible finite sets

```
enum boolean {false, true};
enum answer {yes, no, maybe};
```

- Of course, we can also use typedef

```
typedef enum {tall, venti, grande} size;
```

File I/O

File I/O

- We are used to reading from and writing to the terminal
  - Read from “stdin”
  - Write to “stdout”
- We may also read from and write to files
  - pset5, I’m looking at you!

File I/O

Step 1: create a reference to the file

```
FILE* fp;
```

Step 2: open the file

```
fp = fopen("file.txt", "r");
```

- 1st argument – path to the file
- 2nd argument – mode
  - "r" – read; "w" – write; "a" – append
  - "w" overwrites the file from the beginning; "a" adds to the end

File I/O

Step 3a: read from the file

- `fgetc` – returns the next character
- `fread` – reads a certain number of objects (you pick the size in bytes) and places them into an array
- `fseek` – moves to a certain position

Step 3b: write to the file

- `fputc` – write a character
- `fprintf` – print a formatted output to a file
- `fwrite` – write an array of objects to a file

File I/O

Step 4: close the file

```
fclose(fp);
```

- Remember
  - Always open a file before reading from or writing to it
  - Always close a file if you open it

That was Week 6

http://www.youtube.com/watch?v=kAG39jKi0lI