C Tools

Structs

- Contains an group of item under a single name
- Useful because it
  - Simplifies code
  - Increases code readability
  - Allows for more flexibility

Structs

```
typedef struct process_entry {
    int pid;
    char *command;
} pentry;

pentry *background_queue[100];
background_queue[i]  = calloc(sizeof(pentry));
…
printf(“%s\n”, background_queue[i]->command);
...
free(background_queue[i]);
background_queue[0] = NULL;
```

Function Pointers

- Variables that point to functions
- Useful for
  - Abstracting complexity/naming
  - Passing functions to other functions
  - Defining/changing behavior at runtime

Function Pointers

```
void print_hex(int n) {
    printf(“%x\n”, n);
}
void (*fun_ptr)(int) = &print_hex;

fun_ptr(5);
(*fun_ptr)(20);
```

Object-Oriented Programming in C ???

- Using structs and function pointers, you can provide some object-based features
- This makes programming in procedural languages like C easier to people used to OOP
- Full object-oriented support requires a lot of work and is overkill for small projects
  - You are basically reimplementing C++

Basic Object

```
/*Definitions*/
typedef struct student Student;

Student *student_new();
void student_delete();

void student_init(Student *, char *, int);
void student_print_info(Student *);
```

Basic Object

```
/*Object*/
struct student {
    /*Public*/
    void (*init)(Student*, char*, int);
    void (*print_info)(Student*);

    /*Private*/
    char *name;
    int year;
};
```

Basic Object

```
/*Object Methods*/
Student *student_new() {
    Student *student;
    student = (Student*)calloc(1, sizeof(Student));

    student->init = &student_init;
    student->print_info = &student_print_info;
    student->name = NULL;
    return student;
}
void student_delete(Student *student) {
    if (student->name != NULL)
        free(student->name);
    free(student);
}
```

Basic Object

```
/*Object Methods*/
void student_init(Student *student, char *name, int year) {
    if (student->name != NULL)
        free(student->name);
    student->name = (char*)calloc(strlen(name)+1, sizeof(char));
    strcpy(student->name, name);

    student->year = year;
}
void student_print_info(Student *student) {
    printf("%s has been a student for %d years\n", student->name, student->year);
}
```

Basic Object

```
/*Usage*/
int main() {
    Student *student = student_new();
    student->init(student, "Bob", 4);
    student->print_info(student);
    student_delete(student);
    return 0;
}
```

Some Other Features

- Private members
  - Can't truly hide things in C
  - Potential implementations
    - Comments
    - Nested struct with a separate header
    - Single void* to all data
- Inheritance
  - Can't easily have virtual methods
  - Easiest thing is to just have a copy (composition / aggregation)
- Polymorphism
  - No built in support in C
  - Would need to have a series of function pointers

- http://www.cs.rit.edu/~ats/books/ooc.pdf