CS221
C and Systems Programming

Wouldn't it be nice if…

…structs let us add methods and access control?

People do primitive Object Oriented Programming in C

- Methods take pointer-to-struct as first argument

`initBoard(&myBoard, …)` instead of

`myBoard.initBoard(…)`

E.g. for a game of chess or checkers

Preventing memory errors

```
char buf[256];
strncpy(buf, 256, "Really long text...")
strncat(buf, 256, "More really long text");

scanf("%255s", buf);

snprintf(buf, 256, "Here is a really long string: %s", bigStr)

char* buf2;
asprintf(&buf2, "My formats: %d %f", 10, 3.14);
```


Force a max length of data copied—do not overwrite past end of string.
USE THESE!

System Functions

Systems capabilities - timekeeping

What time is it? What time zone are we in? How long does some job take?

System capabilities - timekeeping

```
time_t time(0);
struct tm* localtime(const time_t*);
char* ctime(const time_t*);


char* tzname[2];
long timezone;
int daylight;
```

```
struct tm {
  int tm_sec;
  int tm_min;
  int tm_hour
  int tm_mday;
  int tm_month;
  int tm_year;
  int tm_wday;
  int tm_yday;
  int tm_isdst;
}
```

6

"seconds since the epoch"

Try this out for different values of time_t
Try printing out current time, with timezone, reflecting daylight savings time

Measuring time intervals

```
clock()


CLOCKS_PER_SEC
```


CPU time expended, not wall clock time expended

Measuring time intervals ACCURATELY

```
clock_gettime()


CLOCK_REALTIME
CLOCK_MONOTONIC
CLOCK_MONOTONIC_RAW
```


NTP and time tracking

I have had some spectacular bugs in the past…

SEE timedemo.c

File Input/Output

How to read and write files?

Two different "libraries" – low-level and high-level

- Do need both at different times
- Low level reads and writes binary data
- High level reads/writes binary data or formatted strings

Low level I/O using File Descriptors

```
int fd = open(char* path, int flags);
read()
write()
close();

O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_APPEND, O_TRUNC
```


Do a sample program!

High level I/O using Streams

```
FILE* fptr = fopen(char* path, char* mode);
fread()
fwrite()
fclose();

"r", "w", "a"
"b" flag:  binary
```


Do a sample program!

High level I/O using Streams

`fprintf()`
`fscanf()`
`fgetc()`
`feof()`

`stdin, stdout, stderr`

Windows vs Linux line endings…

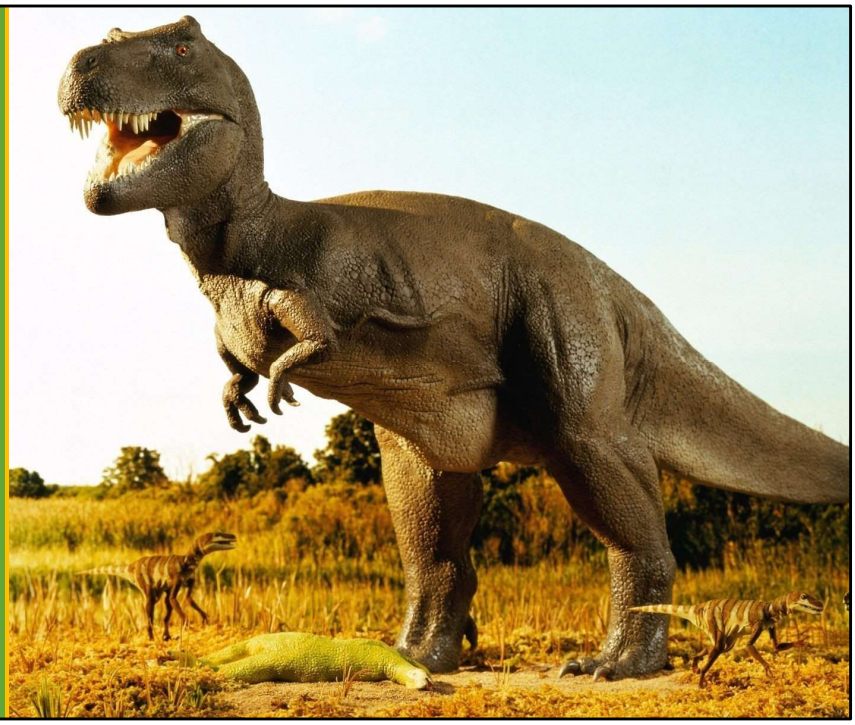
No built-in `readLine()` method!

Do a sample program!

Write ("test\n") on Windows and look at the actual chars. Get a \r

IGNORE \r when you see it, when text processing.

Why are we learning this terrible primitive language?



Why learn C?

Historical interest
- Parent of so many other languages, which are similar

Very lightweight compiler and runtime
- Available on more hardware than any other language – embedded devices

Best support for Operating System function calls – "system programming"

We want the dangerous low-level tools sometimes
- Especially for very high-performance computing

Java, C++, C#, Objective-C, Go, Ruby, etc