C Structures in Practice

CS 2060

Prof. Jonathan Ventura

Use of C Structures in Unix/Linux

- To further illustrate C structures, we will review some uses of `struct` in system calls.
- Here is a function from BSD to get the current time (found in `sys/time.h`):

```
struct timeval {
    time_t      tv_sec;     /* seconds since Jan. 1, 19
    suseconds_t tv_usec;    /* and microseconds  */
};

int gettimeofday( struct timeval * tp, void * tzp );
```

- `time_t` and `suseconds_t` are integer types.
- `tp` will contain the time after the function returns.
- `tzp` is for getting the timezone (can pass NULL to ignore it).

Example of gettimeofday

```
#include <sys/time.h>

double timeInSeconds()
{




}
```

Example of gettimeofday

```c
#include <sys/time.h>

double timeInSeconds()
{
    struct timeval tp;

    gettimeofday( &tp, NULL );

    return tp.tv_sec + tp.tv_usec*1e-6;
}
```

Example of gettimeofday

```c
#include <sys/time.h>

double timeInSeconds()
{
    struct timeval *tp;

    // what about this?
    gettimeofday( tp, NULL );

    return tp->tv_sec + tp->tv_usec*1e-6;
}
```

Example of gettimeofday

```c
#include <sys/time.h>

double timeInSeconds()
{
    struct timeval *tp;

    // tp is an uninitialized pointer
    // the struct is never allocated in memory
    // this will probably crash
    gettimeofday( tp, NULL );

    return tp->tv_sec + tp->tv_usec*1e-6;
}
```

Socket example

- Unix uses "sockets" to connect across a network.
- You "connect" a socket to some network address and then begin reading/writing data from it.

```
#include <sys/types.h>
#include <sys/socket.h>

struct sockaddr {
    unsigned short    sa_family;    // address family, AF_xxx
    char              sa_data[14];  // 14 bytes of protocol address
};

int connect( int socket, const struct sockaddr *address,
             socklen_t address_len );
```

- `socket` is the socket number.
- `address` contains the address information.
- `address_len` is the address length in bytes (?)

Socket example

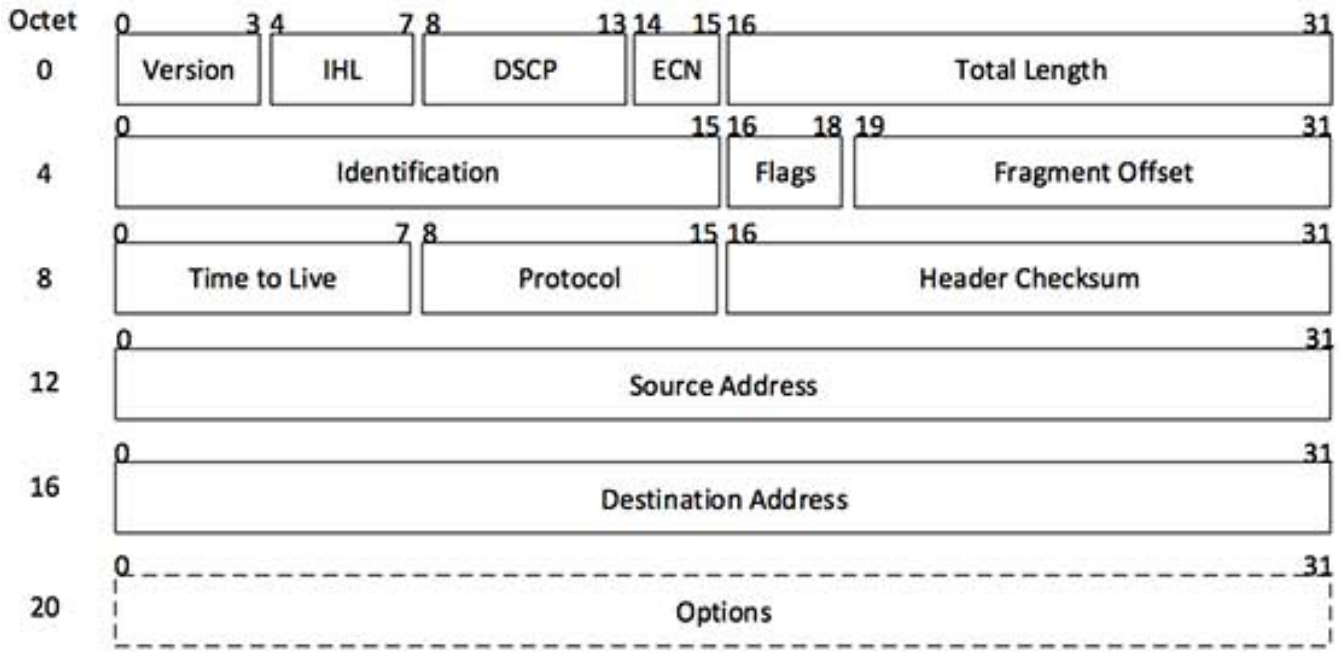
- This connect call pre-dates the Internet (introduced in 1983).
- To connect to an IP address, we use a different structure:

```
#include <sys/types.h>
#include <sys/socket.h>

struct sockaddr_in {
    short           sin_family;   // e.g. AF_INET, AF_INET6
    unsigned short  sin_port;     // e.g. htons(3490)
    struct in_addr  sin_addr;     // see struct in_addr, below
    char            sin_zero[8];  // zero this if you want to
};

struct in_addr {
    unsigned long s_addr;         // load with inet_pton()
};
```

- `AF_INET` is for IPv4, `AF_INET6` is for IPv6.
- `sin_port` is the port number; e.g. 80 is for HTTP (web).
- `sin_addr` is the IP address in integer form.

Structures for data formats

- Structures can be used to encode a data format.
- For example, here is the format of an Internet Protocol (IP) packet:



[Image: IP Header]

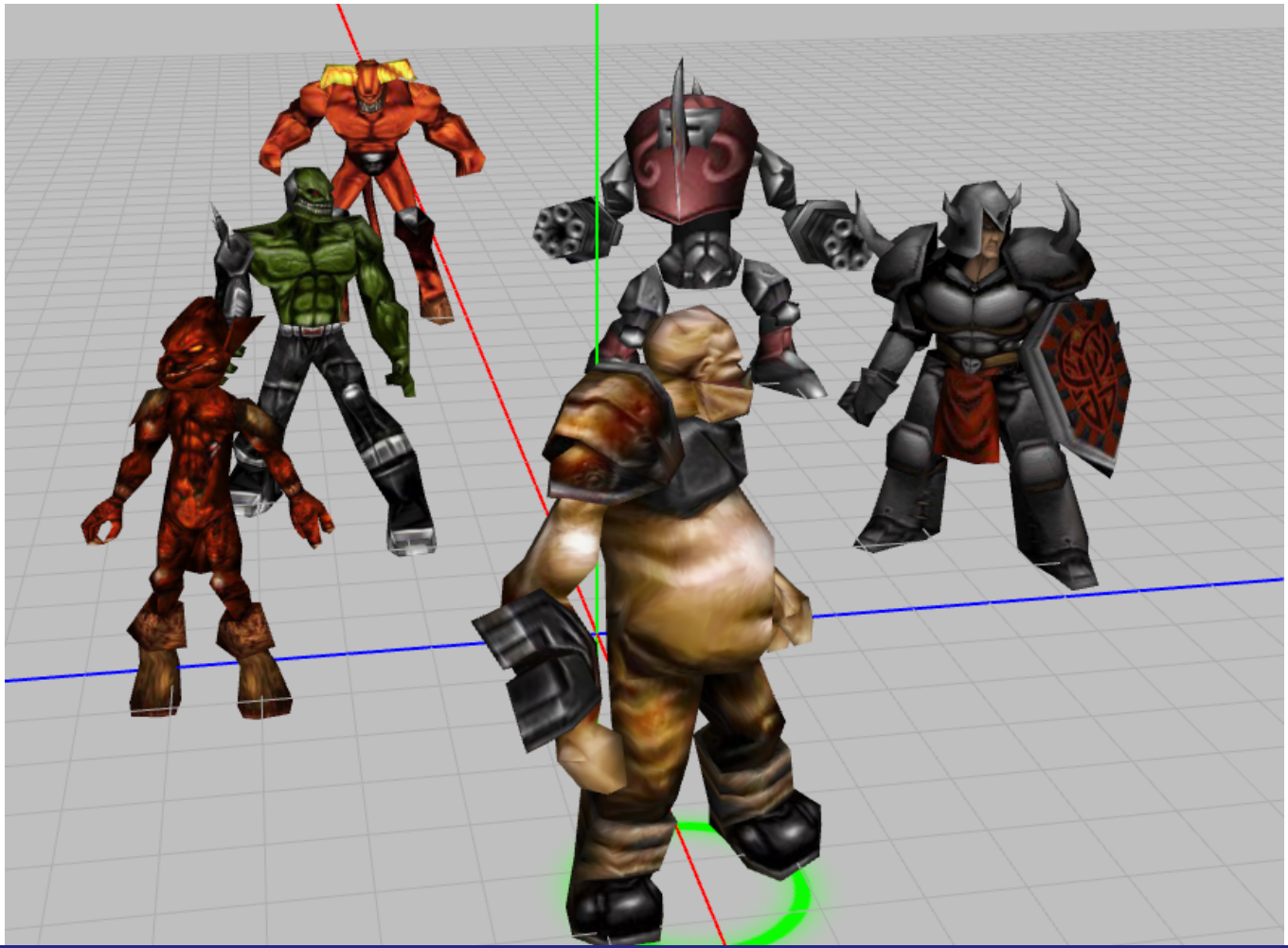
Structures for file formats

- Files typically have a fixed header structure with metadata about the file.
- A `struct` can be used to represent this header and to read it from or write it to a file.

- For example, JPEG, PNG, or any other image file format will have some type of pre-defined header.

MD2 Model Format

- MD2 is a 3D model format used by the Quake II engine.



MD2 Model Format

- The MD2 header has the following format:

Offset	Data type	Name	Description
0	int	ident	Magic number. (IDP2)
4	int	version	MD2 version. (8)
8	int	skinwidth	Width of the texture
12	int	skinheight	Height of the texture
16	int	framesize	Size of one frame in bytes
20	int	num_skins	Number of textures
24	int	num_xyz	Number of vertices
28	int	num_st	Number of texture coordinates
32	int	num_tris	Number of triangles
...	...	...	...

MD2 Model Format Header in Struct

- We can represent this using a `struct`:

```
struct MD2Header {
    int ident;      // Magic number
    int version;    // MD2 version
    int skinwidth;  // Width of the texture
    int skinheight; // Height of the texture
    int framesize;  // Size of one frame in bytes
    int num_skins;  // Number of textures
    int num_xyz;    // Number of vertices
    int num_st;     // Number of texture coordinates
    int num_tris;   // Number of triangles
};
```

MD2 Model Format Header in Struct

- Now we can easily read and write the header from a file:

```
struct MD2Header header;

// open file
f = fopen("model.md2","r");

// read header from file
fread( &header, sizeof(MD2Header), 1, f );

// ...
```