ISA 563: Fundamentals of Systems Programming

Complex Data Types

Jan. 29, 2013

Outline

- Recall primitive data types
- Explore complex data types
  - struct
  - enum
  - union
  - bit fields

Primitive Types

- char, int, float, double
- Fine for basic primitive types
- Building blocks for more complex types that we will discuss

Complex Types

- What about a type that
  - has multiple dimensions or properties
  - are aggregates of primitive types
- Storage model doesn't work very well
  - how do you store a complex type in 1 32-bit memory cell
    - you don't (usually)

Example: represent an MP3

- A simple int does not cut it
- Multiple properties about 1 single logical entity

```
/* some properties for an MP3 */
char file_name[256] = {0};
char audio_name[256] = {0}
long length = 0;
int bit_rate = 144; // kbps
```

Repeating Properties

- Will “run out” of variable names
- Parallel maintenance of data
- Need a template for this logical collection of data

A Structure (`struct` keyword)

- Collection of logically-related data

```
struct mp3_audio
{
  char file_name[256];
  char audio_name[256];
  long length;
  int  bit_rate;
  char data[1000000];
}
```

Enumerations

- A way to declare a set of constants
- Has scope

```
enum days {MON, TUES, WED, THUR, FRI, SAT, SUN};

enum months {Jan = 1, Feb, Mar, Apr, May, Jun,
             Jul, Aug, Sept, Oct, Nov, Dec};

enum {FALSE=0, TRUE, false=0, true=1};
```

Unions

- Like a struct, but has multiple personalities depending on context:

```
union pet
{
   char cat;
   int  bird;
   float turtle;
}
```

Struct vs. Union

- Struct contains all things at once
  - distinct cells allocated for all members
- Union
  - memory allocated for the largest member
  - union instance is treated as only 1 member at a "time"
  - programmer must keep track
  - size depends on the largest member

Demo

pi.c

typedef

```
typedef union _packet_flags
{
  int tcp_opts;
  short udp_opts;
  char open_opts;
} PacketFlags;

typedef struct _packet
{
  Header header;
  PacketFlags flags;
  Payload payload;
  struct _packet* _next;
} Packet;
```