CSCE 155 - C

Lab 11 - Structs

Dr. Chris Bourke

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Review the following free textbook resources:
   - http://en.wikibooks.org/wiki/C_Programming/Complex_types#Structs
   - http://www.cs.cf.ac.uk/Dave/C/node9.html
3. For additional Information on some advanced topics:
   - RSS: http://en.wikipedia.org/wiki/RSS
   - HTTP Protocol: http://en.wikipedia.org/wiki/HTTP
   - C Sockets: http://www.linuxhowtos.org/C_C++/socket.htm

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Be familiar with the concepts of encapsulation & modularity
- Understand how to design, declare, and use C structures (both by reference and by value)
- Have some exposure to advanced topics such as sockets, the HTTP protocol, and XML processing

2 Background

An RSS feed (RDF Site Summary or "Really Simple Syndication") is a format used to publish frequently updated works. RSS enabled clients can subscribe to RSS feeds and update a user as to new or relevant news items. RSS feeds are most commonly formatted using XML (Extensible Markup Language) that use XML tags to indicate what the data represents (the title of the article, a short description, etc.). Clients "read" an RSS feed by making a connection to a server using the HyperText Transfer Protocol (HTTP).

For example, UNL has an RSS news feed available at http://newsroom.unl.edu/releases/?format=xml which serves XML data that looks something like the following:

```
1 <rss xmlns:media="http://search.yahoo.com/mrss/"
  ↪ xmlns:atom="http://www.w3.org/2005/Atom" version="2.0">
2 <channel>
3   <title>UNL News Releases</title>
4   <link>http://newsroom.unl.edu/releases/</link>
5   <description>News from the University of
    ↪ Nebraska-Lincoln</description>
6   <language>en-us</language>
7   <copyright>Copyright 2012 University of
    ↪ Nebraska-Lincoln</copyright>
```

```
8     <image>
9       <title>UNL News Releases</title>
10      <url>http://www.unl.edu/favicon.ico</url>
11      <link>http://www.unl.edu/</link>
12    </image>
13    <item>
14      <title>Guerrilla Girls on Tour perform 'Feminists are Funny'
          ↪ Monday at Sheldon</title>
15
          ↪ <link>http://newsroom.unl.edu/releases/2012/03/09/Guerrilla</link>
16      <description>The Guerrilla Girls on Tour, an internationally
          ↪ acclaimed anonymous theater collective, will perform
          ↪ "Feminists are Funny" at the University of
          ↪ Nebraska-Lincoln's Sheldon Museum of Art, 12th and R
          ↪ streets, at 7 p.m. March 12. The 70-minute play is an...
17      </description>
18      <pubDate>Fri, 09 Mar 2012 02:00:00 -0600</pubDate>
19    </item>
20    ...
21    </items>
22  </channel>
```

Structures in C

An entity may be composed of several different pieces of data. A person for example may have a first and last name (strings), an age (integer), a birthdate (some date/time representation), etc. Its much easier and more natural to group each of these pieces of data into one entity. This is a concept known as *encapsulation*–a mechanism by which data can be grouped together to define an entity.

The C programming language provides a mechanism to achieve encapsulation using structures. Structures are user defined types that have one or more data fields–variables which have a type and a name. To access the member fields of a structure you can use the dot operator; example: `student.firstName`. However, when you have a reference to a structure, you need to use the arrow operator: `student->firstName`.

RSS Client Background

You have been provided with an incomplete RSS client written in C. The client works as follows: based on a URL (which consists of a host and a resource on that host), it opens a socket connection and sends a GET request to the server. The server responds

with a stream of data that the client reads into a buffer. This data stream can, in general, be any type of data, but we're expecting an RSS feed–a stream of plain text XML-formatted data conforming to the RSS standard. Since the data is plain text, it is stored in a character array which is then handed off to another function to parse the XML document. This is done using a library (libxml2, an XML parser and toolkit for Gnome).

3 Activities

Clone the project code for this lab from GitHub by using the following URL: https://github.com/cbourke/CSCE155-C-Lab11.

3.1 A Student Structure

This activity will familiarize you with a completed program in which a C structure has been declared to represent a student. Several functions have been implemented to assist in the construction and printing of this structure. In particular there are two "factory" functions that can be used to help in the construction of an individual structure. The factory function takes the appropriate parameters, allocates memory to hold a new student and sets each field of the student appropriately. In the case of strings, malloc is used to create enough space and the string is then copied. The birthdate is handled specially: it parses the date string and creates a `struct tm` which is a *time* structure defined by the time library (`time.h`). Ultimately, a pointer to the new student structure is returned. There is also a print function that takes a student (by reference) and prints it out to the standard output.

Instructions

1. Examine the syntax and program structure and observe how each factory function works as well as how they are used.
2. Change the values in the `main` function to your name, NUID, and birth date.
3. Compile and run the program. Refer back to this program in the next activity as needed.

3.2 Completing the RSS Client

In this activity, you will complete the RSS Client that connects to a UNL RSS feed, processes the XML data and outputs the results to the standard output. Most of the

client has been completed for you. You just need to complete the design and implementation of a C structure that models the essential parts of an RSS item. Your structure will need to support an RSS item's title, link, description, and publication date.

As a first attempt you can treat the publication date as a string, but ultimately you should use a `tm` structure similar to the `Student` example. This structure has several fields that give you access to the year, month, date, hour, minute, etc. For full documentation, see the following: http://pubs.opengroup.org/onlinepubs/7908799/xsh/time.h.html

Instructions

1. We have provided 3 different RSS feeds to work with in the program. You can switch between them using command line arguments.
2. Design and implement the RSS structure
3. Complete the other functions
   - `createEmptyRss()` - this function should create an "empty" RSS structure
   - `createRss()` - this function should construct an RSS structure based on the input parameters. For the date, use a string representation (RSS and Atom feeds can use a variety of formats and there is no easy way to automatically parse them).
   - `printRss()` - this function should print the given RSS to the standard output in a human readable format (the details are up to you).
4. You will need to use the makefile to compile your program as it contains the flags necessary to import the proper XML libraries. Recall that you can execute the makefile by typing the `make` command.

4 Advanced Activity (Optional)

1. You will notice that internally, we have provided support for both RSS 2.0 feeds and Atom 1.0 feeds (Reddit uses Atom for example). Examine the files where we have defined these feeds. Find an RSS or Atom feed that is of interest to you and integrate it as an option for this program. Be sure to change the main function so that it can be used.
2. Improve the program further by modifying the `parseRSS_XML` function. Currently, this function parses the XML but limits the number of RSS structures to a maximum of 100. Write another function to count the number of items in the XML and use it to instead dynamically allocate an RSS array of a size exactly

equal to the number of item elements in the XML file.