Error Detection with CRC-32 table-based algorithm

Information Theory Lab 12

Objective

Understand the table-based implementation of the CRC-32 algorithm for error detection, by implementing it as a C application.

Theoretical notions

Basic CRC algorithm

Consider an (N+1)-bit generator polynomial in binary format as g. (i.e. CRC-16, CRC-32).

Given a block of binary data, the algorithm for CRC computation is as follows:

1. Locate the first 1 in the sequence
2. Starting from this location, perform XOR of the next data bits with **g**.
3. Repeat until all original data has been zeroed out.

The **CRC value** is the value remaining at the end of the data.

Table-based implementation

Instead of doing the XOR operations at every bit 1 in the sequence, with shifted versions of **g**, the individual XOR operations can be grouped into a single XOR operation for one byte of data, and then chained together, as follows:

1. Set CRC = 0
2. While there are bytes in the input data

1. Read the next byte b
2. XOR b with the first byte of the existing CRC
3. Read the new CRC for b from a pre-computed table
4. XOR the old CRC (from the second byte onwards) over the new CRC

The algorithm requires a precomputed table with the CRC value for all the 256 byte values.

This is the usual algorithm described in the literature (e.g. search Wikipedia for CRC32).

Exercises

1. Write a C program that computes the CRC-32 value of a data file and appends it to the file.

   The program shall be called in two possible ways:

   a. with two arguments. In this case the program takes the first as input and produces the encoded file as output.

   ```
   CRC16.exe original.dat encoded.dat
   ```

   b. with one argument. In this case the program takes the encoded file as input and checks if the CRC is OK or not.

   ```
   CRC16.exe encoded.dat
   ```

   - The arguments are:
     - `original.dat`: the input file (original / encoded)
     - `encoded.dat` [optional]: the encoded file, with the CRC value appended
   - The program should consist of the following steps:
     - declare one large vector of `unsigned char` for input bits
     - open the input file and read everything into the input vector
     - implement the table-based algorithm
     - there will be 32 bits remaining at the end of the original input vector (the CRC-32 value)
     - then:
       a. If the program is called with two arguments:
          * write the vector to the output data file, including the CRC-32 value at the end
       b. If the program is called with one argument:
          * if the CRC-32 value is 0, display "File OK\n", otherwise display "Data corrupted\n"
2. Run the program (with two arguments) on a sample data file. Take the output encoded file run the program again on it (with a single argument). Observe the

result.

3. Use the website `hexed.it` to introduce a few errors anywhere in the encoded file. Save the modified file and run the program on it (with a single argument). Observe the result.

Program design

1. The pre-computed table of CRC-32 values for all the possible byte values is given below:

```
static const unsigned int table[256] = {
0x00000000U,0x04C11DB7U,0x09823B6EU,0x0D4326D9U,
0x130476DCU,0x17C56B6BU,0x1A864DB2U,0x1E475005U,
0x2608EDB8U,0x22C9F00FU,0x2F8AD6D6U,0x2B4BCB61U,
0x350C9B64U,0x31CD86D3U,0x3C8EA00AU,0x384FBDBDU,
0x4C11DB70U,0x48D0C6C7U,0x4593E01EU,0x4152FDA9U,
0x5F15ADACU,0x5BD4B01BU,0x569796C2U,0x52568B75U,
0x6A1936C8U,0x6ED82B7FU,0x639B0DA6U,0x675A1011U,
0x791D4014U,0x7DDC5DA3U,0x709F7B7AU,0x745E66CDU,
0x9823B6E0U,0x9CE2AB57U,0x91A18D8EU,0x95609039U,
0x8B27C03CU,0x8FE6DD8BU,0x82A5FB52U,0x8664E6E5U,
0xBE2B5B58U,0xBAEA46EFU,0xB7A96036U,0xB3687D81U,
0xAD2F2D84U,0xA9EE3033U,0xA4AD16EAU,0xA06C0B5DU,
0xD4326D90U,0xD0F37027U,0xDDB056FEU,0xD9714B49U,
0xC7361B4CU,0xC3F706FBU,0xCEB42022U,0xCA753D95U,
0xF23A8028U,0xF6FB9D9FU,0xFBB8BB46U,0xFF79A6F1U,
0xE13EF6F4U,0xE5FFEB43U,0xE8BCCD9AU,0xEC7DD02DU,
0x34867077U,0x30476DC0U,0x3D044B19U,0x39C556AEU,
0x278206ABU,0x23431B1CU,0x2E003DC5U,0x2AC12072U,
0x128E9DCFU,0x164F8078U,0x1B0CA6A1U,0x1FCDBB16U,
0x018AEB13U,0x054BF6A4U,0x0808D07DU,0x0CC9CDCAU,
0x7897AB07U,0x7C56B6B0U,0x71159069U,0x75D48DDEU,
0x6B93DDDBU,0x6F52C06CU,0x6211E6B5U,0x66D0FB02U,
0x5E9F46BFU,0x5A5E5B08U,0x571D7DD1U,0x53DC6066U,
0x4D9B3063U,0x495A2DD4U,0x44190B0DU,0x40D816BAU,
0xACA5C697U,0xA864DB20U,0xA527FDF9U,0xA1E6E04EU,
0xBFA1B04BU,0xBB60ADFCU,0xB6238B25U,0xB2E29692U,
0x8AAD2B2FU,0x8E6C3698U,0x832F1041U,0x87EE0DF6U,
0x99A95DF3U,0x9D684044U,0x902B669DU,0x94EA7B2AU,
0xE0B41DE7U,0xE4750050U,0xE9362689U,0xEDF73B3EU,
0xF3B06B3BU,0xF771768CU,0xFA325055U,0xFEF34DE2U,
0xC6BCF05FU,0xC27DEDE8U,0xCF3ECB31U,0xCBFFD686U,
0xD5B88683U,0xD1799B34U,0xDC3ABDEDU,0xD8FBA05AU,
```

```
0x690CE0EEU,0x6DCDFD59U,0x608EDB80U,0x644FC637U,
0x7A089632U,0x7EC98B85U,0x738AAD5CU,0x774BB0EBU,
0x4F040D56U,0x4BC510E1U,0x46863638U,0x42472B8FU,
0x5C007B8AU,0x58C1663DU,0x558240E4U,0x51435D53U,
0x251D3B9EU,0x21DC2629U,0x2C9F00F0U,0x285E1D47U,
0x36194D42U,0x32D850F5U,0x3F9B762CU,0x3B5A6B9BU,
0x0315D626U,0x07D4CB91U,0x0A97ED48U,0x0E56F0FFU,
0x1011A0FAU,0x14D0BD4DU,0x19939B94U,0x1D528623U,
0xF12F560EU,0xF5EE4BB9U,0xF8AD6D60U,0xFC6C70D7U,
0xE22B20D2U,0xE6EA3D65U,0xEBA91BBCU,0xEF68060BU,
0xD727BBB6U,0xD3E6A601U,0xDEA580D8U,0xDA649D6FU,
0xC423CD6AU,0xC0E2D0DDU,0xCDA1F604U,0xC960EBB3U,
0xBD3E8D7EU,0xB9FF90C9U,0xB4BCB610U,0xB07DABA7U,
0xAE3AFBA2U,0xAAFBE615U,0xA7B8C0CCU,0xA379DD7BU,
0x9B3660C6U,0x9FF77D71U,0x92B45BA8U,0x9675461FU,
0x8832161AU,0x8CF30BADU,0x81B02D74U,0x857130C3U,
0x5D8A9099U,0x594B8D2EU,0x5408ABF7U,0x50C9B640U,
0x4E8EE645U,0x4A4FFBF2U,0x470CDD2BU,0x43CDC09CU,
0x7B827D21U,0x7F436096U,0x7200464FU,0x76C15BF8U,
0x68860BFDU,0x6C47164AU,0x61043093U,0x65C52D24U,
0x119B4BE9U,0x155A565EU,0x18197087U,0x1CD86D30U,
0x029F3D35U,0x065E2082U,0x0B1D065BU,0x0FDC1BECU,
0x3793A651U,0x3352BBE6U,0x3E119D3FU,0x3AD08088U,
0x2497D08DU,0x2056CD3AU,0x2D15EBE3U,0x29D4F654U,
0xC5A92679U,0xC1683BCEU,0xCC2B1D17U,0xC8EA00A0U,
0xD6AD50A5U,0xD26C4D12U,0xDF2F6BCBU,0xDBEE767CU,
0xE3A1CBC1U,0xE760D676U,0xEA23F0AFU,0xEEE2ED18U,
0xF0A5BD1DU,0xF464A0AAU,0xF9278673U,0xFDE69BC4U,
0x89B8FD09U,0x8D79E0BEU,0x803AC667U,0x84FBDBD0U,
0x9ABC8BD5U,0x9E7D9662U,0x933EB0BBU,0x97FFAD0CU,
0xAFB010B1U,0xAB710D06U,0xA6322BDFU,0xA2F33668U,
0xBCB4666DU,0xB8757BDAU,0xB5365D03U,0xB1F740B4U,
};
```

2. You can check a sample implementation *here*

Final questions

1. What is the size of the output file, compared to the input file?