David Swanson
CIS 17-A
8 June, 2014

Project 2
Wave Steganographer Part II

For information on wave files and steganography, see the Project 1 write up. This paper will detail the nuts and bolts of the new program.

Project 2 is a working wave file steganographer, plus a set of wave file utilities. Capabilities include:

- Encodes files and strings by replacing LSBs in the data stream of a 'host' file with bits from a 'parasite' file or string.
- Verifies the accuracy of the encoding (no 'skips' in the audio stream).
- Detects and decodes a watermark to file or string, automatically choosing which to do.
- Fixes (canonizes) non-standard wave files.
- Checks for 'true stereo' data, as opposed to 'dual mono'.
- Counts zero crossings in wave data.
- Displays wave file headers in 'friendly' form or raw hex, displays wave data in graph form or raw hex, copies and displays any file.
- Logs events and errors, displaying or writing to file when a class destructor is called.
- I will put an 'Easter Egg' at the last menu choice plus one. I wrote a Towers of Hanoi game to run those damned CIS-7, chapter 5 problems. It allows user play, auto play, plus it contains a teaching mode.

Concepts demonstrated by this project:

- The basics: functions, overloading, arrays and vectors etc.
- Pointers, allocating and destroying
- Structures and classes
- Abstract classes, virtual functions, inheritance and polymorphism
- Template functions
- Error handling and logging
- Files and serialization, binary and text, iteration, random access
- Bit-wise iteration, masking, bib-by-bit byte building (say it fast)
- Conversion between char* and integer. Big- and little-endian format.
- Sorting
- Recursion

Difficulties/learning opportunities:

Nothing is as hard as doing something for the first time. The first difficulty with this project was that there was no way to test the output of certain functions, because that output was being sent to other functions or into a binary file I couldn't easily read. So I had to create functions just to test each step. Those functions are left in the classes for future debugging.

A second difficulty is what any programmer faces: how to deal with mounting size and complexity, until the program becomes like a house of cards. I have dealt with that more successfully with each project completed, creating more logical inheritance structures, using composition with classes and structures, leaving common functions free in a separate file.

Object-oriented programming is 100% for the human writing it, evidenced by the fact that the compiler immediately discards all those nifty curly braces in favor of a stream of procedural code. Classes are conceptual, to limit the size of what a programmer has to understand at any one time. They protect the working code from what is still in development. I can safely forget how something works and never look at it again, as long as I know the ins and outs, and what it does.

The third challenge was the bit-wise manipulation required for the project. It helps to have taken Assembly Language Programming and a basic microprocessor course in the past. Char to int conversions are difficult because if the MSB is 1 (a negative number) the int result will carry that to all additional MSBs, in keeping with 2's complement. For example a char of 0x8A will become FFFFFF8A because the sign bit in the 8 turns into Fs all the way over. This is fine as long as a signed number is the destination. But for an unsigned number, the left-most bits need to be zeroes; otherwise the number becomes very large, making the result undefined.

For conversions from char strings of n length, you need a mask of (2^n*8)-1, a number equal to the unsigned max. The char string is summed to an int, shifting left 8 places with each char added. Further, since C++ creates a temporary int when adding a char to an int sum, it is best to explicitly create a temporary int and then mask it with (2^8)-1 so the sign-stretching does not disrupt the arithmetic.

For signed conversions, you need to preserve the earlier sign-stretching behavior, but only for the byte containing the MSB, since this sets the sign bit for the new number. All other bytes need to be masked with (2^8)-1 as noted earlier.

The chars are of course big-endian, but the strings come from the file in reverse order to what was written, making them little-endian. To switch endians, the conversion loop can be changed from i++ to i--. I wrote functions both ways and checked results with known values to be sure of which to use.

One final challenge, which I chose not to undertake, was a file chooser. I did some research and realized it would be much easier to do in QT. And if I do that I may as well re-write the whole front end. So this program has a hard-coded list with a type-in option, which is fine.

Class and Structure Descriptions:

BFile is the base class, abstract. It is essentially a wrapper for its fstream objects, but with fields to hold file names and sizes, and functions to open and display files. It also holds an object of the ErrLogger class as a repository for all the messages generated during operations.

FileItr is derived from BFile. It runs char-int-char conversions as described above, plus it iterates by sample size, spitting out a properly-converted number with each call. It bit-iterates in similar fashion, breaking bits off the sample number from LSB to MSB, resetting its counter at the finish of each sample. It also has copy and testing functions. It introduces the sampleSize variable, which allows for setting of multi-byte word length.

WaveItr is derived from FileItr, and is specialized to the management of wave files. It introduces the ability to set the sample size automatically based on information in the wave file header. It holds an object of the hData structure, which keeps data from the header locally for display and easy access. It also handles most of the wave file utilities.

WaveSteg is derived from WaveItr, and is specialized to the encoding and decoding of hidden watermarks in the wave data stream.

Helper structure Fact is designed to hold events for ErrLogger, including date-time as string and unsigned int (for sorting), error state, a comment string and a code used to access a global constant array of string messages.

Helper class ErrLogger contains a vector of pointers to Fact structures. It can sort events, output the log to the screen or write it to a file in a subfolder.

Helper structure hData, holds extra data from the header, mainly to keep it conceptually separate from what is needed by WaveItr.

Helper classes Biterator and Reshaper bit-iterate through ints, chars and strings and put them back together. Eight classes are actually included in the inheritance structure, with the above two the abstract base classes. Char, int and string subclasses handle the specifics. These classes are used to encode the parasite's file extension, size, and a magic word used to detect the encoding.

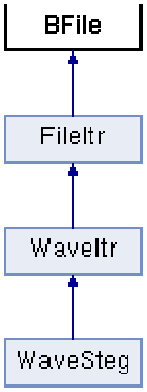
Helper class U_Interface responds to menu choices, combining particular steps necessary to run the primary classes.

Main contains the menu and looping switch.

Commons is a library of general functions, for use by any class.

TemplateFunctions contains whichever common functions are adaptable to template form.

BFile Class Reference abstract



Public Member Functions

void	**openInFile** (string fileName)
void	**openOutFile** (string fileName)
void	**showLogs** (bool show)
void	**saveLogs** (bool save)
bool	**good** ()
unsigned int	**getInSize** ()
unsigned int	**getOutSize** ()
void	**dispFile** (int start, int length, int line)
void	**seekg** (streampos index)
void	**copyBlock** (int start)
void	**copyBlock** (int start, int length)
virtual void	**initGet** (string fileName)=0
virtual void	**initGet** (string fileName, int setSampleSize)=0
virtual void	**initPut** (string fileName)=0
virtual void	**initPut** (string fileName, int setSampleSize)=0
virtual void	**initGetPut** (string inName, string outName)=0
virtual void	**initGetPut** (string inName, string outName, int setSampleSize)=0

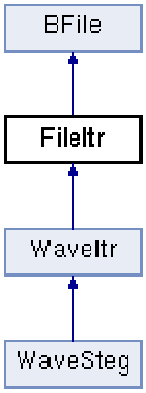
Protected Member Functions

void	**findInSize** ()

Protected Attributes

string	**finName**
string	**foutName**
int	**inSize**
int	**outSize**
bool	**inGood**
bool	**outGood**
ifstream	**fin**
ofstream	**fout**
ErrLogger	**log**
bool	**dispLog**
bool	**saveLog**

FileItr Class Reference



Public Member Functions

void	**initGet** (string filename)
void	**initGet** (string filename, int setsamplesize)
void	**initPut** (string filename)
void	**initPut** (string filename, int setsamplesize)
void	**initGetPut** (string inName, string outName)
void	**initGetPut** (string inName, string outName, int setSampleSize)
void	**copyFile** (string inName, string outName)
void	**copyTest** (string source)
void	**nextSample_test** (int limit)
void	**nextSample_test_unsigned** (int limit)
void	**clearIterator** ()
bool	**nextSample** (unsigned int &n)
bool	**nextSample** (int &n)
bool	**nextSample** (int &n, int interval)
bool	**nextSample** (char &ch)
void	**putSample** (int samp)
void	**nextBit_test** (int limit)

void	**nextBit_test** (string *binNums, int limit)
void	**clearBiterator** ()
bool	**nextBit** (bool &bit)
void	**nextLSB_test** (int limit)
int	**currentSample** ()
bool	**nextLSB** (bool &lsb)
void	**replaceLSB_Test** ()
int	**replaceLSB** (int n, bool bit)
void	**putBit_test** (string *binNums, int size)
void	**putBit** (bool bit)

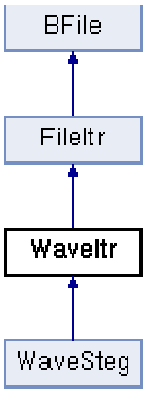
Protected Member Functions

unsigned int	**toUnsignedInt** (char *chunk, int start, int length)
int	**toInt** (char *chunk, int start, int length)
void	**toCharArray** (int n, char *charArray, int start, int length)
void	**toCharArray** (unsigned int n, char *charArray, int start, int length)

Protected Attributes

int	**sampleSize**
int	**samp_int**
int	**unsignedMax**
int	**signedMax**
int	**bitCounter**
char *	**samp_charArray**
int	**currSample**

WaveItr Class Reference



Public Member Functions

void	**initGet** (string filename)
void	**initGet** (string filename, int setsamplesize)
void	**dispHeader** ()
void	**dispHeaderHex** ()
void	**dispDataHex** (int limit)
void	**dispWaveForm** (int limit)
int	**numZeroes** (bool verbose)
void	**seekDataStart** ()
char *	**getHeader** ()
char *	**getHeaderFixed** ()
int	**getHeadSize** ()
int	**getSampleSize** ()
int	**getDataStart** ()
int	**get_iatOneSec** ()
headerData *	**getHeaderData** ()
unsigned int	**getDataStreamLength** ()

bool	**isWave** ()
bool	**isCanon** ()
bool	**trueStereo** ()

Protected Member Functions

bool	**readHeader** ()
unsigned int	**findDataStart** ()
unsigned int	**findStreamLength** ()
char *	**fileToChArray** (streampos start, streampos size)
void	**find_iatOneSec** ()

Protected Attributes

unsigned int	**streamLen**
unsigned int	**iData**
unsigned int	**iatOneSec**
int	**headSize**
char *	**header**
headerData	**hData**

headerData Struct Reference

Public Attributes

unsigned int	**fileLen**
unsigned int	**formatLen**
unsigned int	**formatTag**
unsigned int	**bytes_sec**
unsigned int	**blockAlign**
unsigned int	**numChannels**
unsigned int	**sampleRate**
unsigned int	**bitDepth**

WaveForm Class Reference

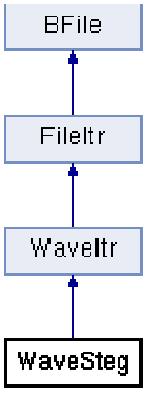
Public Member Functions

	WaveForm (int max)
void	**single** (int val)

Private Attributes

int	**width**
int	**zero**
int	**maxVal**
float	**scale**

WaveSteg Class Reference



Public Member Functions

void	**init_encFToF** (string setHostName, string setParasiteName)
void	**init_encFToF** (string setHostName, string setParasiteName, string setDestName)
void	**encFToF** ()
bool	**verify** (string fName1, string fName2, bool skipLSB, int verbosity)
void	**init_encSToF** (string setHostName, string encodeMe)
void	**init_encSToF** (string setHostName, string encodeMe, string setDestName)
void	**encSToF** ()
void	**initDecode** (string setHostName)
void	**initDecode** (string setHostName, string setDestName)
void	**decode** ()
void	**canonize** (string setHostName, string setDestName)
int	**minHostSize** ()
string	**getEncodeString** ()

Private Member Functions

void	**copyHeader()**

void	**copyDataStream (int iGet, int iPut)**
void	**copyDataStream (int iGet, int iPut, int length)**
void	**encParaStats (int iGet, int iPut)**
void	**encodeFToF_core** ()
void	**encodeSToF_core** ()
void	**decParaStats (int iGet)**
void	**decToFile** ()
void	**decToString** (string setHostName, string setDestName)
bool	**checkSize** ()

Private Attributes

string	**hostName**
string	**paraName**
string	**destName**
string	**paraExt**
string	**encodeString**
int	**paraSize**
int	**encSize**
int	**minSize**
bool	**hangBit**
Waveltr	**host**
Fileltr	**parasite**

Fact Struct Reference

Public Member Functions

	Fact (string setTime, string setFunct, int setMCode, string setMComment, bool setErr)

Public Attributes

int	**mCode**
string	**yr**
string	**time**
string	**funct**
string	**mComment**
unsigned int	**uiTime**
bool	**err**

ErrLogger Class Reference

Public Member Functions

void	**setClient** (string setMe)
void	**addToLog** (string setFunct, int setMCode, string setMComment, bool setErr)
void	**sortByTime** ()
void	**sortByErr** ()
void	**dispLog** ()
void	**logToFile** ()
void	**clrLog** ()

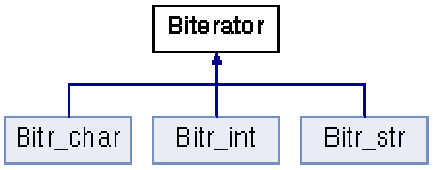
U_Interface Class Reference

Public Member Functions

void	**setPreferences** ()
void	**encodeFileToFile** ()
void	**encodeStringToFile** ()
void	**decode** ()
void	**canonize** ()
void	**testStereo** ()
void	**verify** ()
void	**dispFile** ()
void	**copyFile** ()
void	**dispHeader** ()
void	**dispWaveData** ()
void	**dispZeroes** ()
bool	**encodeFileToFile** (string host, string parasite, string dest)
bool	**encodeStringToFile** (string host, string encodeMe, string dest)
void	**decode** (string host, string dest)
void	**verify** (string fName1, string fName2, bool skipLSB, int verbosity)
void	**canonize** (string fName, bool show, bool save)
void	**canonize** (string fName1, string fName2, bool show, bool save)
void	**testStereo** (string fName)
void	**dispFile** (string fileName, int start, int limit, int line)
void	**dispHeader** (string fName, bool hexed)
void	**dispWaveData** (string fName, int resolution, bool hexed)
void	**dispZeroes** (string fName)

void	**copyFile** (string f1, string f2)

Biterator Class Reference abstract



Public Member Functions

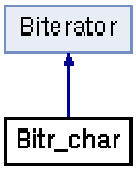
virtual bool	**nextBit** (bool &bit)=0
void	**reset** ()
void	**setMax** (int setMe)
bool	**bitrEnd** ()

Protected Attributes

int	**bitCounter**
int	**max**
bool	**end**

Bitr_char Class Reference

Inheritance diagram for Bitr_char:

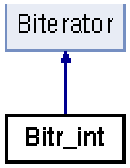


Public Member Functions

	Bitr_char (char setMe)
void	**setSource** (char setMe)
bool	**nextBit** (bool &bit)

Bitr_int Class Reference

Inheritance diagram for Bitr_int:

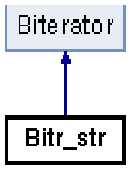


Public Member Functions

	Bitr_int (int setMe)
void	**setSource** (int setMe)
bool	**nextBit** (bool &bit)

Bitr_str Class Reference

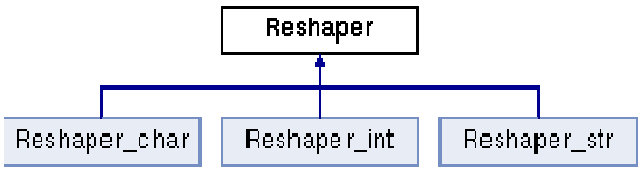
Inheritance diagram for Bitr_str:



Public Member Functions

	Bitr_str (string setMe)
void	**setSource** (string setMe)
bool	**nextBit** (bool &bit)

Reshaper Class Reference abstract



Public Member Functions

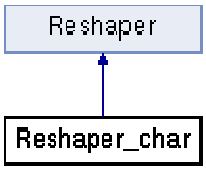
virtual bool	**putBit** (bool bit)=0
virtual void	**clrResult** ()=0
void	**reset** ()
void	**setMax** (int setMe)
bool	**bitrEnd** ()

Protected Attributes

int	**bitCounter**
int	**max**
bool	**end**

Reshaper_char Class Reference

Inheritance diagram for Reshaper_char:

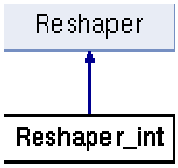


Public Member Functions

char	**getResult** ()
bool	**putBit** (bool bit)
void	**clrResult** ()
void	**reset** ()

Reshaper_int Class Reference

Inheritance diagram for Reshaper_int:

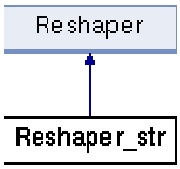


Public Member Functions

	Reshaper_int (int max)
int	**getResult** ()
bool	**putBit** (bool bit)
void	**clrResult** ()

Reshaper_str Class Reference

Inheritance diagram for Reshaper_str:



Public Member Functions

	Reshaper_str (int max)
void	**clrResult** ()
string	**getResult** ()
bool	**putBit** (bool bit)

For additional descriptions of member functions, see Project 1 write up.