UNIVERSITY OF
BIRMINGHAM

# Year 3 C++ Project
# E-Publication Document Creator

School of Physics and Astronomy
University of Birmingham

**Josh Wainwright**
UID:1079596

Project Supervisor: Dr Steve Hillier
Date: December 2012

## Contents

**1 Motivation** **2**

**2 Details** **2**

**3 Project** **3**

3.1 Graphical User Interface . . . . . . . . . . 3

3.2 Internal Structure . . . . . . . . . . 4

**4 Critique** **5**

**5 Notes** **5**

**Appendix** **6**

## 1 Motivation

This project is to build an application that aids in the creation of a simple ebook. Following the defined file structure, the user will have a number of options to customise the book, including title, author and date etc; and choose the html files that make up the book and any images that are associated with it.

## 2 Details

The e-publication document format, or epub for short, is a free and open e-book standard created by the International Digital Publishing Forum (IDPF). It is designed to contain reflowable content which is content that can adapt its presentation depending on the device used to view it. This means that it is ideally suited for use by publishers on devices like e-book readers that have different sizes and means that some customisation by the user, like screen rotation and text size, is possible.

The epub document consists of the html, or similar, files that make up the text of the document and are referenced from a number of files required by the ebook-reader. These files are then wrapped in a ZIP archive along with the directory structure. The file structure is shown in figure 1.

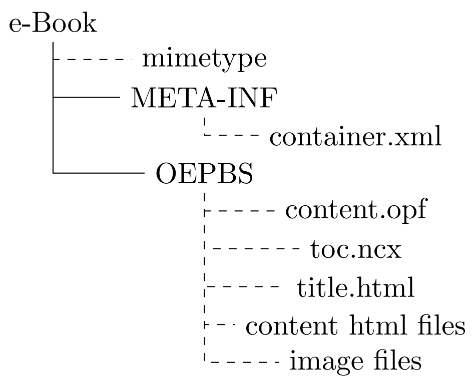


Figure 1: The required directory structure for an epub ebook. Files are shown with dotted lines.

Each of the files included is required to exist by the reader, as long as that reader conforms to the standards of the format, but they contain information that will vary from book to book.

## 3 Project

### 3.1 Graphical User Interface

My implementation of an epub creator uses a single main window that is opened when the user starts the program, shown in figure 2. From here the main details of the book can be entered.

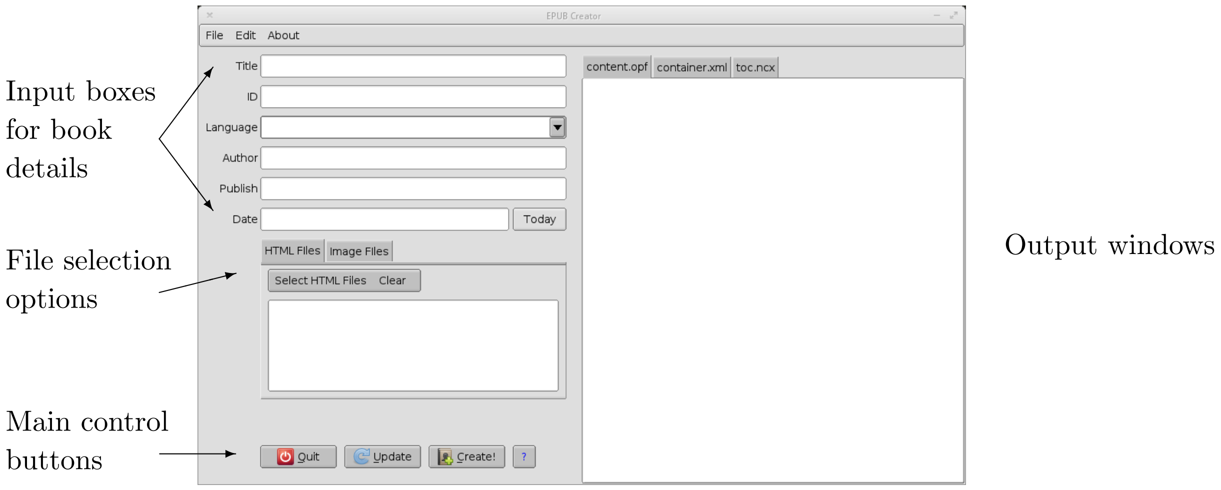
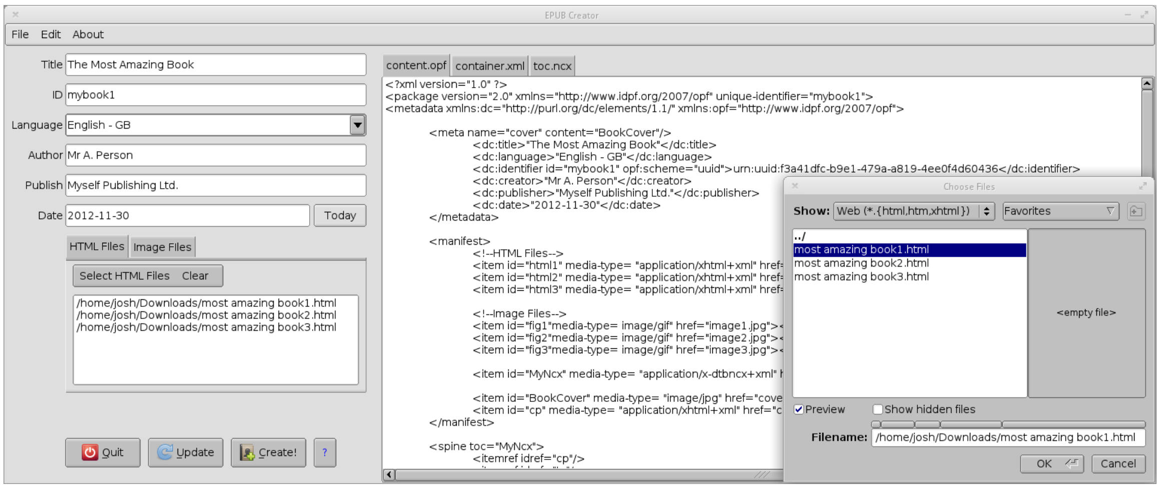


Figure 2: The main window of the application that is shown when the program is started.



Title: The Most Amazing Book
ID: mybook1
Language: English - GB
Author: Mr A. Person
Publish: Myself Publishing Ltd.
Date: 2012-11-30

```
/home/josh/Downloads/most amazing book1.html
/home/josh/Downloads/most amazing book2.html
/home/josh/Downloads/most amazing book3.html
```

```
<?xml version="1.0" ?>
<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-identifier="mybook1">
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org/2007/opf">

        <meta name="cover" content="BookCover"/>
                <dc:title>"The Most Amazing Book"</dc:title>
                <dc:language>"English - GB"</dc:language>
                <dc:identifier id="mybook1" opf:scheme="uuid">urn:uuid:f3a41dfc-b9e1-479a-a819-4ee0f4d60436</dc:identifier>
                <dc:creator>"Mr A. Person"</dc:creator>
                <dc:publisher>"Myself Publishing Ltd."</dc:publisher>
                <dc:date>"2012-11-30"</dc:date>
        </metadata>

        <manifest>
                <!--HTML Files-->
                <item id="html1" media-type= "application/xhtml+xml" href=
                <item id="html2" media-type= "application/xhtml+xml" href=
                <item id="html3" media-type= "application/xhtml+xml" href=

                <!--Image Files-->
                <item id="fig1"media-type= image/gif" href="image1.jpg"><
                <item id="fig2"media-type= image/gif" href="image2.jpg"><
                <item id="fig3"media-type= image/gif" href="image3.jpg"><

                <item id="MyNcx" media-type= "application/x-dtbncx+xml" h

                <item id="BookCover" media-type= "image/jpg" href="cove
                <item id="cp" media-type= "application/xhtml+xml" href="c
        </manifest>

        <spine toc="MyNcx">
                <itemref idref="cp"/>
```

Show: Web (*.{html,htm,xhtml})

```
../
most amazing book1.html
most amazing book2.html
most amazing book3.html
```

<empty file>

[✓ Preview] Show hidden files

Filename: /home/josh/Downloads/most amazing book1.html

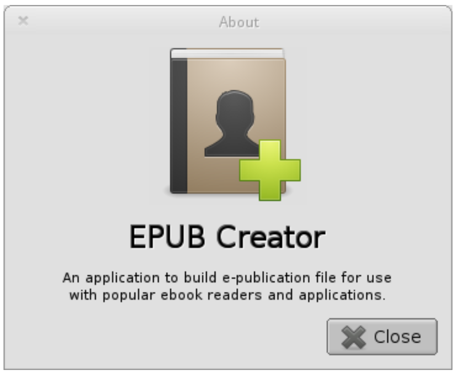
Figure 3: An example with the dialogue for choosing the html files to be included as well as an example of when data has been added to the relevant fields and the update button has been pressed to get a preview of the files.

Also, from here, the html files can be chosen, as can the images used in the book. Tabs are used to keep the interface clean and less cluttered. The file chooser that is opened when the button to select the files is clicked is an FLTK built in file chooser.

Below these entry boxes are the main controls for the program. "Exit" closes the window and ends the application, "Refresh" updates the in application information about the book to reflect and changes made by the user, and "Create!" takes the entered information and writes it to the relevant files. It also copies the selected files from the location chosen to the temporary folder used by the application. Once this has finished, it ZIPs the resulting folder as specified in the document specification and names it according the user's choice of name for the book.

Next to the input boxes, on the right side of the window, are a number of tabs that inform the user what will be written when they click the create button to make their book. These tabs provide information for those interested in the structure of epub ebooks and provide a means of checking that the book will be created as expected. Each separate tab contains what will be included in each of several separate files (see figure 1).

At the top of the window, there is a menu bar with a few options and operations. These include buttons that perform the same functions as the buttons previously mentioned, exit, update and create as well as the "About" dialogue, shown in figure 4, and an option to set the current operating system (see below for details).



An application to build e-publication file for use with popular ebook readers and applications.

Figure 4: The "About" window provides information about the application and could be extended to include licensing information, bug reporting information and developer details etc.

### 3.2 Internal Structure

The code for this project is shown at the end of this document in appendices 5 through 5. The project started as a single long file with all of the various functions distributed throughout. I took the decision to split the code through several files that are interdependent, but which contain a smaller subset of the code, grouped according to the function that it achieves. Each of the files is used by one or more other files, the include tree is shown in figure 5.

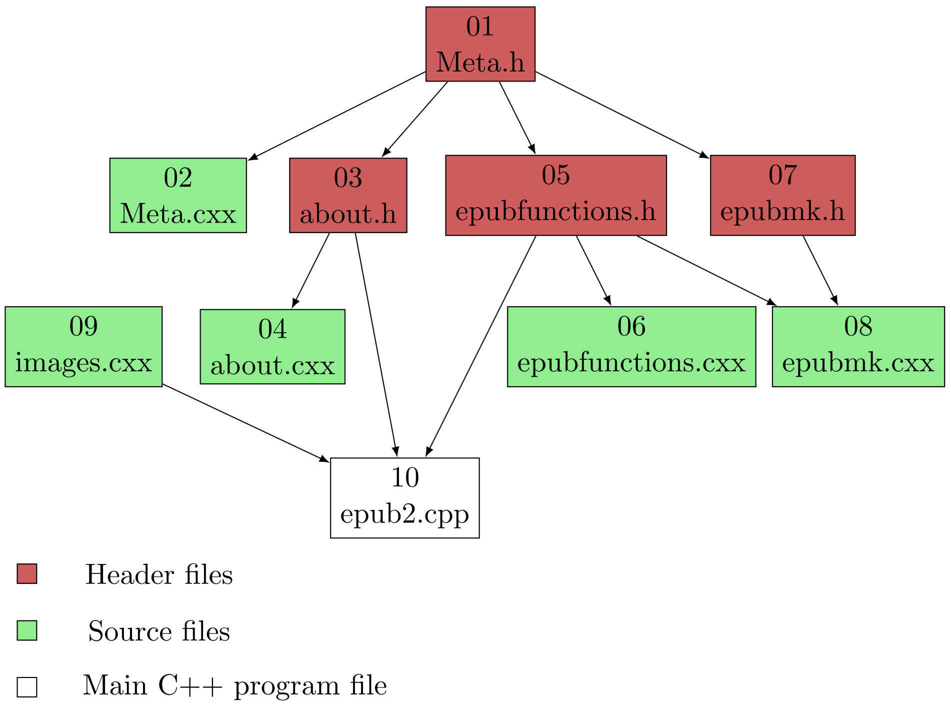


Figure 5: A diagram showing the structure of the epub project files and their interdependencies.

The most important aspect of the code is the class called "Meta". This contains all of the variables and the associated functions that make up and allow interaction with a book. Meta stands for meta-data which includes information like the title, author and language as well as more extensive information, like the contents of the static files for inclusion in the final book. The class contains these data, as well as the means to access and modify them.

## 4 Critique

The aim of this project was to build an application to aid in the building of ebook according to the specification written by the IDPF. This, the application achieves. You are able to select the pre-written files, include them in the document and specify the details about the document that you wish, or are required, to be specified. It also has some flexibility so that the number and type of included text files does not matter, and man or no images can be selected.

However, there are several areas where this application would require more work and possibly redesign were it to be used on a larger scale. Some of these that I have identified are listed below.

- There is no way to rearrange the files that are included so that they are added to the book in the right order (the only way currently is to name the files sequentially as the file chooser adds the files to the array in alphanumerical order).
- Some of the elements of the GUI do not behave as expected, for example on resize some of the buttons change location and/or size instead of remaining static and the space around them changing.
- To extend the functionality of the application, there are several features that could be added, such as:
  - Front cover chooser,
  - Text editor for main text files.
- The code could be further compartmentalised into functions to simplify and reduce the repetition of elements.

## 5 Notes

- This application is written for the Linux operating system. Though it shall run under other operating systems, like Windows, the core functionality of the program require Linux. The program “zip”, which is by default installed on almost all systems, is a requirement for this program to function properly.
- Included in the files for the program is a makefile that can be used to build the program under the same Linux environment. This requires all of the files for the program to be in the same directory along with the makefile. The program can then be built using the command `make` from the terminal (alternatively use `make && ./epub` to build and then run the program). This will create several directories as well as the program executable, “epub”. Running `make clean` will remove the auxillary files that are created during the build process.
- Also included in the program files is a folder called “Example” which includes a set of html and image files that can be used to test the program. These are merely a web page that has been saved locally and been stripped of the JavaScript, CSS elements and some header information. To use this example, simply run the program as usual and choose the relevant files from the

## Meta.h

```
//**************************************************************************
// Author: Josh Wainwright                                                 *
// Date: 30/11/2012                                                        *
// File Number: 01                                                         *
//                                                                         *
// Description: Header file for the "Meta" class. Here all variable types are  *
//              declared along with functions.                             *
//**************************************************************************

#include <string>
#include <vector>

#include <FL/Fl.H>
#include <FL/Fl_Input.H>

using namespace std;

class Meta{
    private:
        string title;
        string title_under;
        string id;
        string lang;
        string author;
        string publish;
        string date;
        string uuid;
        bool LINUX=true;
        vector<string> files_list;
        vector<string> images_list;

        string content_contents;
        string container_contents;
        string toc_contents;
    public:
        string get(string get);
        bool get_os();
        int get_files_size();
        int get_images_size();
        string get_file(int i);
        string get_image(int i);

        void clear(string list);

        void set_title(string title_set);
        void set_title_under(string title_under);
        void set_id(string id_set);
        void set_lang(string lang_set);
        void set_author(string author_set);
        void set_publish(string publish_set);
        void set_date(string date_set);
        void set_files(string files);
        void set_images(string images);
        void set_uuid(string uuid_set);
        void set_os(bool os);

        void set_content(string content);
        void set_container(string container);
        void set_toc(string toc);

        static void openimage_cb(Fl_Widget*, void*);
        static void openhtml_cb(Fl_Widget*, void*);
        static void clearhtml_cb(Fl_Widget*, void*);
        static void clearimages_cb(Fl_Widget*, void*);
        static void clear_all_cb(Fl_Widget*);

        static void inlang_cb(Fl_Widget*, void*);
        static void intitle_cb(Fl_Input* o);
        static void inid_cb(Fl_Input* o);
        static void inauthor_cb(Fl_Input* o);
        static void inpublish_cb(Fl_Input* o);
        static void indate_cb(Fl_Input* o);
        static void today_cb(Fl_Input* o);
        static void create_cb(Fl_Input* o);
        static void print_cb(Fl_Input* o);
};
```

## Meta.cxx

```
//***************************************************************************
// Author: Josh Wainwright                                                  *
// Date: 30/11/2012                                                         *
// File Number: 02                                                          *
//                                                                          *
// Description: Function prototypes for the "Meta" class                    *
//***************************************************************************

#include <stdlib.h>
#include <iostream>

#include "Meta.h"

//Allows the retrieval of any of the listed variables from the Meta class. The
// name of the variable must be used as the argument within quote marks/
string Meta::get(string get){
    if(get == "title"){return title;}
    else if(get == "title_under"){return title_under;}
    else if(get == "id"){return id;}
    else if(get == "lang"){return lang;}
    else if(get == "author"){return author;}
    else if(get == "publish"){return publish;}
    else if(get == "date"){return date;}
    else if(get == "content_contents"){return content_contents;}
    else if(get == "container_contents"){return container_contents;}
    else if(get == "toc_contents"){return toc_contents;}
    else if(get == "uuid"){return uuid;}
    else{return 0;}
}

//Allows the retrieval of some of the more varied variables. These all simply
// return the same data type as the included variable.
int Meta::get_files_size(){return files_list.size();}
int Meta::get_images_size(){return images_list.size();}
bool Meta::get_os(){return LINUX;}
string Meta::get_file(int i){return files_list[i];}
string Meta::get_image(int i){return images_list[i];}

//Allows the large strings to be set to blank if the user wants to clear
// everything that they have entered. There is no clear function for the other
// variables as the relevant input can be set to zero individually.
void Meta::clear(string list){
    if(list == "files_list"){files_list.clear();}
    else if(list == "images_list"){images_list.clear();}
    else if(list == "content"){content_contents = "";}
    else if(list == "container"){container_contents = "";}
    else if(list == "toc"){toc_contents = "";}
}

//Functions to set the value of each of the different variables. Most simply
// replace the existing data with the new, but some concatenate the new onto the
// existing, hence the clear functions for these cases.
void Meta::set_title(string title_set){title = title_set;}
void Meta::set_title_under(string title_under_set){title_under=title_under_set;}
void Meta::set_id(string id_set){id = id_set;}
void Meta::set_lang(string lang_set){lang = lang_set;}
void Meta::set_author(string author_set){author = author_set;}
void Meta::set_publish(string publish_set){publish = publish_set;}
void Meta::set_date(string date_set){date = date_set;}
void Meta::set_files(string file){files_list.push_back(file);}
void Meta::set_images(string image){images_list.push_back(image);}
void Meta::set_uuid(string uuid_set){uuid = uuid_set;}
void Meta::set_os(bool os){LINUX = os;}

void Meta::set_content(string content){content_contents += content;}
void Meta::set_container(string container){container_contents += container;}
void Meta::set_toc(string toc){toc_contents += toc;}
```

## about.h

```
//*****************************************************************************
// Author: Josh Wainwright                                                    *
// Date: 30/11/2012                                                           *
// File Number:03                                                            *
//                                                                            *
// Description: Header file for the "About" dialogue.                         *
//*****************************************************************************

#ifndef about_h
#define about_h
#include <FL/Fl.H>
#include <FL/Fl_Double_Window.H>
extern Fl_Double_Window *about_window;
#include <FL/Fl_Box.H>
#include <FL/Fl_Button.H>
extern Fl_Button *close_about;
void about();
#endif
```

## about.cxx

```
//*****************************************************************************
// Author: Josh Wainwright                                                    *
// Date: 30/11/2012                                                           *
// File Number: 04                                                            *
//                                                                            *
// Description: "About" dialogue, Here the window is created and drawn. All   *
//              FLTK elements are defined and used here. This file should     *
//              remain able to be run standalone.                             *
//*****************************************************************************

#include "about.h"
#include "images.cxx"

Fl_Double_Window *about_window=(Fl_Double_Window *)0;
Fl_Button *close_about=(Fl_Button *)0;

static void cb_close_about(Fl_Button* o, void*) {
  about_window->hide();
}

void about() {
    about_window = new Fl_Double_Window(360, 270, "About");
    {
        about_window->set_modal();
            about_window->color(FL_LIGHT2);
        Fl_Box* picture = new Fl_Box(10, 10, 340, 128);
            picture->image(image_big);

        Fl_Box* title = new Fl_Box(10, 148, 340, 35, "EPUB Creator");
            title->labeltype(FL_ENGRAVED_LABEL);
            title->labelsize(24);

        Fl_Box* text1 = new Fl_Box(10, 188, 340, 20,
                    "An application to build e-publication file for use");
            text1->labelsize(11);

        Fl_Box* text2 = new Fl_Box(10, 202, 340, 20,
                    "with popular ebook readers and applications.");
            text2->labelsize(11);

        close_about = new Fl_Button(260, 230, 90, 30, " Close");
            close_about->image(image_window);
            close_about->callback((Fl_Callback*)cb_close_about);
            close_about->align(Fl_Align(256));

        about_window->size_range(360, 270, 360, 270);
        about_window->end();
    }
    about_window->show();
}
```

## epubfunctions.h

```
//*****************************************************************************
// Author: Josh Wainwright                                                    *
// Date: 30/11/2012                                                           *
// File Number: 05                                                            *
//                                                                            *
// Description: Header file for the "epubfunctions.cxx" with function        *
//              prototypes                                                    *
//*****************************************************************************

#include <string>

using namespace std;

string space_underscore(string text);

string space_backslash(string text);

string removenewline(string text);

string catfile(string filename);

void copy_file(string fromfile, string tofile);
```

## epubfunctions.cxx

```
//*****************************************************************************
// Author: Josh Wainwright                                                    *
// Date: 30/11/2012                                                           *
// File Number: 06                                                            *
//                                                                            *
// Description: various functions used by the program to manipulate strings   *
//              etc.                                                          *
//*****************************************************************************

#include <stdlib.h>

#include "epubfunctions.h"

//Converts all of the spaces in a string to underscores so that the string can
// be used in functions.
string space_underscore(string text){
    for(int unsigned i=0;i<text.length();i++){
        if(text[i] == ' ') text[i] = '_';
    }
    return text;
}

//Adds a backslash before all spaces to escape the space, allows the string to
// be used as a file location.
string space_backslash(string text){
    string newtext;
    for(int unsigned i=0;i<text.length();i++){
        if(text[i]==' '){
        //  text[i] = '\\';
            newtext += "\\ ";
        } else {
            newtext += text[i];
        }
    }
    return newtext;
}

//Removes all instances of a newline character from a string so that a different
// deliminator can be used to separate the string when read.
string removenewline(string text){
    for(int unsigned i=0;i<text.length();i++){
        if(text[i]=='\n') text[i] = ' ';
    }
    return text;
}

//Removes all of the string apart from the last section that contains a filename
// so that that filename can be used independantly.
```

```
string catfile(string filename){
    size_t found;
    found = filename.find_last_of("/\\");
    filename = filename.substr(found+1);
    return filename;
}

//Use system commands to copy a file from one location to
void copy_file(string fromfile, string tofile){
//  fromfile = space_backslash(fromfile);
    fromfile = removenewline(fromfile);
//  tofile = space_backslash(tofile);
    string cp = "cp \"" + fromfile + "\" \"" + tofile + "\"";
    system(cp.c_str());
}
```

## epubmk.h

```
//****************************************************************************
// Author: Josh Wainwright                                                   *
// Date: 30/11/2012                                                          *
// File Number:07                                                           *
//                                                                           *
// Description: Header file for the "epubmk.cxx" with function prototypes    *
//****************************************************************************

#include <string>

#include "Meta.h"

using namespace std;

Meta mkcontent(Meta metadata);

Meta mkcontainer(Meta metadata);

Meta mktoc(Meta metadata);

void mkstructure(Meta metadata);
```

## epubmk.cxx

```
//****************************************************************************
// Author: Josh Wainwright                                                   *
// Date: 30/11/2012                                                          *
// File Number: 08                                                           *
//                                                                           *
// Description: Various functions that create the final epub document. These *
//              are all static and are generally used only once.            *
//****************************************************************************

#include <iostream>
#include <sstream>
#include <fstream>

#include "epubmk.h"
#include "epubfunctions.h"

Meta mkcontent(Meta metadata){
    // mkcontent ----------------------------------------------------------------
    metadata.clear("content");

    //Switch for setting the Unique User Identifier depending on if the user is
    // on a Linux system or not
    if(metadata.get_os()){
        //uuidgen is a
        system("uuidgen > temp");
        ifstream temp("temp");
        string uuid;

        getline (temp,uuid);
        metadata.set_uuid(uuid);
        system("rm temp");
    }else{
        metadata.set_uuid("00000000-0000-0000-0000-000000000000");
    }

    metadata.set_content(
        "<?xml version=\"1.0\" ?>\n"
        "<package version=\"2.0\" xmlns=\"http://www.idpf.org/2007/opf\" unique-identifier=\""
             + metadata.get("id") + "\">\n"
        "<metadata xmlns:dc=\"http://purl.org/dc/elements/1.1/\" xmlns:opf=\"http://www.idpf.
            org/2007/opf\">\n\n"
        "\t<meta name=\"cover\" content=\"BookCover\"/>\n"
        "\t\t<dc:title>\"" + metadata.get("title") + "\"</dc:title>\n"
        "\t\t<dc:language>\"" + metadata.get("lang") + "\"</dc:language>\n"
        "\t\t<dc:identifier id=\"" + metadata.get("id") + "\" opf:scheme=\"uuid\">urn:uuid:" +
             metadata.get("uuid") + "</dc:identifier>\n"
        "\t\t<dc:creator>\"" + metadata.get("author") + "\"</dc:creator>\n"
        "\t\t<dc:publisher>\"" + metadata.get("publish") + "\"</dc:publisher>\n"
        //could also add subject, discription, editor. All optional.
        "\t\t<dc:date>\"" + metadata.get("date") + "\"</dc:date>\n"
        "\t</metadata>\n\n"
        "\t<manifest>\n"
        "\t\t<!--HTML Files-->\n"
    );

    stringstream ss;//create a stringstream
    for (int i = 0; i < metadata.get_files_size(); i++){
        ss.str("");
        ss << i+1;//add number to the stream

        string filename;
        filename = catfile(metadata.get_file(i));
        metadata.set_content(
            "\t\t<item id=\"html" + ss.str() + "\" media-type= \"application/xhtml+xml\" href
                =\"" + filename + "\"></item>\n"
        );
    }

    metadata.set_content("\n\t\t<!--Image Files-->\n");

    for (int i = 0; i < metadata.get_images_size(); i++){
        ss.str("");
        ss << i+1;//add number to the stream
        ss.str();

        string imagename;
        imagename = catfile(metadata.get_image(i));
        //FOR SOME REASON, WHEN THIS IS REMOVED, THE IMAGES WORK
        //metadata.set_content(
        //    "\t\t<item id=\"fig" + ss.str() + "\"media-type= image/gif\" href=\"" +
            imagename + "\"></item>\n"
        //);
    }

    metadata.set_content(
        "\n\t\t<item id=\"MyNcx\" media-type= \"application/x-dtbncx+xml\" href=\"toc.ncx\"></
            item>\n"
    );

    //Chooser for BOOK COVER
    metadata.set_content(
        "\n\t\t<item id=\"BookCover\" media-type= \"image/jpg\" href=\"cover.jpg\"></item>\n"
    );

    //<!-- Cover page [optional] -->
    metadata.set_content(
        "\t\t<item id=\"cp\" media-type= \"application/xhtml+xml\" href=\"coverpg.html\"></
            item>\n"
        "\t</manifest>\n\n"
        "\t<spine toc=\"MyNcx\">\n"
        // "\t\t<itemref idref=\"cp\"/>\n"
        // "\t\t<itemref idref=\"tc\"/>\n"
    );

    for(int i=0;i<metadata.get_files_size();i++){
        ss.str("");
        ss << i+1;
        string numb = ss.str();
        metadata.set_content("\t\t<itemref idref=\"html" + numb + "\"/>\n");
    }

    metadata.set_content(
        "\t</spine>\n\n"
        "\t<guide>\n"
        "\t\t<reference type=\"cover\" title=\"Cover\" href=\"coverpg.html\"></reference>\n"
        "\t\t<reference type=\"toc\" title=\"Table of Contents\" href=\"toc.html\"></reference
            >\n"
        "\t\t<reference type=\"text\" title=\"Beginning\" href=\"intro.html\"></reference>\n"
        "\t</guide>\n\n"
        "</package>"
    );
    return metadata;
}

Meta mkcontainer(Meta metadata){
// mkcontainer ------------------------------------------------------------
    metadata.clear("container");

    metadata.set_container(
        "<?xml version=\"1.0\"?>\n"
        "<container version=\"1.0\" \n"
        "xmlns=\"urn:oasis:names:tc:opendocument:xmlns:container\">\n"
        "\t<rootfiles>\n"
        "\t\t<rootfile full-path=\"OEPBS/content.opf\" media-type=\"application/oebps-package+
            xml\"/>\n"
        "\t</rootfiles>\n"
        "</container>"
    );

    return metadata;
}

Meta mktoc(Meta metadata){
    //mktoc----------------------------------------------------------------
    metadata.clear("toc");

    metadata.set_toc(
        "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
        "<!DOCTYPE ncx PUBLIC \"-//NISO//DTD ncx 2005-1//EN\" \"http://www.daisy.org/z3986
            /2005/ncx-2005-1.dtd\">\n"
        "<ncx xmlns=\"http://www.daisy.org/z3986/2005/ncx/\" version=\"2005-1\" xml:lang=\"en
            \">\n"
        "\t<head>\n"
        //<!-- The content of dtb:uid must be exactly the same as the uuid
        // specified in the OPF file. -->
        "\t\t<meta name=\"dtb:uid\" content=\"urn:uuid:" + metadata.get("uuid") + "\"/>\n"
        //<!-- The NCX specification allows for a hierarchal table of contents.
        // It is recommended that the depth be set at 1 since that is all that
        // the Kindle and Nook support. -->
        "\t\t<meta name=\"dtb:depth\" content=\"1\"/>\n"
        //<!-- The total page count and the max page number are not needed.
        // Leave them set to zero. -->
        "\t\t<meta name=\"dtb:totalPageCount\" content=\"0\"/>\n"
        "\t\t<meta name=\"dtb:maxPageNumber\" content=\"0\"/>\n"
        "\t</head>\n"
        "\t<docTitle><text>" + metadata.get("title") + "</text></docTitle>\n"
        "\t<docAuthor><text>" + metadata.get("author") + "</text></docAuthor>\n"
        //<!-- The navMap section is the heart of the NCX file. For EPUB eBooks
        // this section specifies the entries in the navigational table of
        // contents. For the older keyboard Kindles this section defines a set
        // of predefined bookmarks. You can jump between the preset bookmarks by
        // pressing the right or left arrows of the 5-way controller. It appears
        // that the Kindle Touch and the Kindle Fire do not use the NCX file.
        // The order of the entries is specified by the playOrder attribute. The
        // location is specified by the content src attribute. This consists of
        // an HTML file name such as "filename.html" or an HTML filename with a
        // named anchor added such as "filename.html#mark". The preset bookmarks
        // for the Kindle will appear as little dots on the progress bar at the
        // bottom of the screen.-->
        "\t<navMap>\n"
    );

    stringstream ss;//create a stringstream
    for(int i=0;i<metadata.get_files_size();i++){
        ss.str("");
        ss << i+1;
        string numb = ss.str();
        metadata.set_toc(
            "\t\t<navPoint id=\"navpoint-" + numb + "\" playOrder=\"" + numb + "\">\n"
            "\t\t\t<navLabel><text>file" + numb + "</text></navLabel>\n"
            "\t\t\t<content src=\"" + catfile(metadata.get_file(i)) + "\"/>\n"
            "\t\t</navPoint>\n"
        );
    }

    metadata.set_toc(
        "\t</navMap>\n"
        "</ncx>"
    );
    return metadata;
}

void mkstructure(Meta metadata){
    struct stat sb;

    string folder = metadata.get("title_under") + "/";
    if(!(stat(folder.c_str(), &sb) == 0 && S_ISDIR(sb.st_mode))){
        string mkdir = "mkdir " + metadata.get("title_under") + "/";
        system(mkdir.c_str());
    }
    string mimetype = metadata.get("title_under") + "/mimetype";
    ofstream mimetype2(mimetype.c_str());
    mimetype2 << "application/epub+zip";
    mimetype2.close();

    string meta = metadata.get("title_under") + "/META-INF";
    if(!(stat(meta.c_str(), &sb) == 0 && S_ISDIR(sb.st_mode))){
        string meta2 = "mkdir " + metadata.get("title_under") + "/META-INF";
        system(meta2.c_str());
    }

    string oepbs = metadata.get("title_under") + "/OEPBS";
    if(!(stat(oepbs.c_str(), &sb) == 0 && S_ISDIR(sb.st_mode))){
        string oepbs2 = "mkdir " + metadata.get("title_under") + "/OEPBS";
        system(oepbs2.c_str());
    }

    string toc_ncx = metadata.get("title_under") + "/OEPBS";
    string toc_ncx2 = toc_ncx + "/toc.ncx";
    ofstream toc_ncx3(toc_ncx2.c_str());
    toc_ncx3 << metadata.get("toc_contents");
    toc_ncx3.close();

    string content_opf = metadata.get("title_under") + "/OEPBS";
    string content_opf2 = content_opf + "/content.opf";
    ofstream content_opf3(content_opf2.c_str());
    content_opf3 << metadata.get("content_contents");
    content_opf3.close();

    string dest = content_opf + "/";
    for(int i=0;i<metadata.get_files_size();i++){
        copy_file(metadata.get_file(i), dest);
    }
    for(int i=0;i<metadata.get_images_size();i++){
        copy_file(metadata.get_image(i), dest);
    }

    string container_xml = metadata.get("title_under") + "/META-INF";
    string container_xml2 = container_xml + "/container.xml";
    ofstream container_xml3(container_xml2.c_str());
    container_xml3 << metadata.get("container_contents");
    container_xml3.close();

    string zip_mimetype = "zip " +
                          metadata.get("title_under") + ".epub -DX0 " +
                          metadata.get("title_under") + "/mimetype";
    string zip_else = "zip " +
                      metadata.get("title_under") + ".epub -rDX9 " +
                      metadata.get("title_under") + "/META-INF " +
                      metadata.get("title_under") + "/OEPBS";
//  cout << zip_mimetype << " and \n" << zip_else << endl;
    system(zip_mimetype.c_str());
    system(zip_else.c_str());
}
```

## epub2.cpp

```
//****************************************************************************
// Author: Josh Wainwright                                                   *
// Date: 30/11/2012                                                          *
// File Number: 09                                                           *
//                                                                           *
// Description: Main file containing "main" and the declaration and position of*
//              all FLTK elements and their respective callback functions and *
//              attributes.                                                  *
//****************************************************************************

#include <iostream>

#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Value_Output.H>
#include <FL/Fl_Multiline_Output.H>
#include <FL/Fl_Text_Display.H>
#include <FL/Fl_Tabs.H>
#include <FL/Fl_Group.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Menu_Bar.H>
#include <FL/Fl_Menu.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Text_Editor.H>
#include <FL/Fl_Input_Choice.H>

#include "epubfunctions.h"
#include "epubmk.h"
#include "about.h"
#include "images.cxx"

using namespace std;

Meta metadata;

Fl_File_Chooser* chooser;

Fl_Double_Window* mywindow;
Fl_Menu_Bar* mainmenu;
Fl_Tile* tile;
Fl_Group* lefthalf;
    Fl_Input* intitle;
        Fl_Button* info;
    Fl_Input* inid;
    Fl_Input_Choice* inlang;
    Fl_Input* inauthor;
    Fl_Input* inpublish;
    Fl_Input* indate;
        Fl_Button* today;

    Fl_Tabs* tabs;
        Fl_Group* htmls;
            Fl_Menu_Bar* htmlmenu;
            Fl_Text_Buffer* htmlbuff;
                Fl_Text_Display* htmltext;

        Fl_Group* images;
            Fl_Menu_Bar* imagemenu;
            Fl_Text_Buffer* imagebuff;
                Fl_Text_Display* imagetext;

    Fl_Box* hidden;
    Fl_Button* update;
    Fl_Button* create;
    Fl_Button* close;

Fl_Box* righthalf;
    Fl_Tabs* outputtabs;
        Fl_Group* outputwindow1;
            Fl_Text_Buffer* code1buff;
                Fl_Text_Editor* code1text;
        Fl_Group* outputwindow2;
            Fl_Text_Buffer* code2buff;
                Fl_Text_Editor* code2text;
        Fl_Group* outputwindow3;
            Fl_Text_Buffer* code3buff;
```

```
                Fl_Text_Editor* code3text;


//FLTK--------------------------------------------------------------------------FLTK

void Meta::openimage_cb(Fl_Widget*, void*){
    string images = "";
    Fl_File_Chooser chooser("~",
                            "Image Files (*.{bmp,gif,jpg,jpeg,png})",
                            Fl_File_Chooser::MULTI,
                            "Choose Image Files"
                            );
    chooser.show();

    while(chooser.shown()){
        Fl::wait();
    }

    if ( chooser.value() == NULL ){
        return;
    }
    for ( int t=1; t<=chooser.count(); t++ ) {
        metadata.set_images(chooser.value(t));
        images = images + chooser.value(t) + "\n";
    }
    imagebuff->text(images.c_str());
}

void Meta::openhtml_cb(Fl_Widget*, void*){
    string files = "";
    Fl_File_Chooser chooser("~",
                            "Web (*.{html,htm,xhtml})\tText (*.txt)",
                            Fl_File_Chooser::MULTI,
                            "Choose HTML Files"
                            );
    chooser.show();

    while(chooser.shown()){
        Fl::wait();
    }

    if ( chooser.value() == NULL ){
        return;
    }
    for ( int t=1; t<=chooser.count(); t++ ) {
        metadata.set_files(chooser.value(t));
        files = files + chooser.value(t) + "\n";
    }
    htmlbuff->text(files.c_str());
}

void Meta::clearhtml_cb(Fl_Widget*, void*){
    metadata.clear("files_list");
    htmlbuff->text("");
}

void Meta::clearimages_cb(Fl_Widget*, void*){
    metadata.clear("images_list");
    imagebuff->text("");
}

void Meta::inlang_cb(Fl_Widget *w, void*){
    /*if(inlang->value()=="English"){metadata.set_lang("en");}
    else if(inlang->value()=="English - GB"){metadata.set_lang("en-gb");}
    else if(inlang->value()=="English - US"){metadata.set_lang("en-us");}
    else if(inlang->value()=="French"){metadata.set_lang("fr");}
    else if(inlang->value()=="French - Canada"){metadata.set_lang("fr-ca");}
    else if(inlang->value()=="Spanish"){metadata.set_lang("es");}
    else{metadata.set_lang("none");}*/
    metadata.set_lang(inlang->value());
}
void Meta::intitle_cb(Fl_Input* o){
    metadata.set_title(intitle->value());
    metadata.set_title_under(space_underscore(intitle->value()));
}
void Meta::inid_cb(Fl_Input* o){
    metadata.set_id(inid->value());
}
```

```
void Meta::inauthor_cb(Fl_Input* o){
    metadata.set_author(inauthor->value());
}
void Meta::inpublish_cb(Fl_Input* o){
    metadata.set_publish(inpublish->value());
}
void Meta::indate_cb(Fl_Input* o){
    metadata.set_date(indate->value());
}

void Meta::today_cb(Fl_Input* o){
    time_t rawtime;
    struct tm* timeinfo;
    char buffer[80];
    time(&rawtime);
    timeinfo = localtime(&rawtime);

    strftime(buffer, 80, "%Y-%m-%d", timeinfo);
    metadata.set_date(buffer);
    indate->value(buffer);
}

static void info_cb(Fl_Input* o){
    const char *message =
            "Title - The title of the publication\n"
            "ID - A unique identifier for the book\n"
            "Language - The language of the publication\n"
            "Author - The author of the book\n"
            "Publisher - Name of publisher or your name if self published\n"
            "Date - Date the book was published\n";
    fl_message(message);
}

void Meta::create_cb(Fl_Input* o){
    if(metadata.get("title").empty()){
        const char *message = "Please enter a Title.";
        fl_message(message);
    }else if(metadata.get("id").empty()){
        const char *message = "Please enter an ID.";
        fl_message(message);
    }else if(metadata.get("lang").empty()){
        const char *message = "Please select a language.";
        fl_message(message);
    }else if(metadata.get("author").empty()){
        const char *message = "Please enter an author.";
        fl_message(message);
    }else if(metadata.get("date").empty()){
        const char *message = "Please enter a date.";
        fl_message(message);
    }else{
        mkstructure(metadata);
        const char *message = "Your file was created successfully.\n Enjoy.";
        fl_message(message);
    }
}

//PRINT
void Meta::print_cb(Fl_Input* o){
    metadata = mkcontent(metadata);
    metadata = mkcontainer(metadata);
    metadata = mktoc(metadata);
    code1buff->text(metadata.get("content_contents").c_str());
    code2buff->text(metadata.get("container_contents").c_str());
    code3buff->text(metadata.get("toc_contents").c_str());
    mywindow->redraw();
}

void Meta::clear_all_cb(Fl_Widget* o){
    if(fl_ask("Are you sure?") == 1){
        metadata.clear("files_list");
        metadata.clear("images_list");
        metadata.clear("content");
        metadata.clear("container");
        metadata.clear("toc");
        code1buff->text("");
        code2buff->text("");
        htmlbuff->text("");
        imagebuff->text("");
        intitle->value("");
        inid->value("");
        inlang->value("");
        inauthor->value("");
        inpublish->value("");
        indate->value("");
        mywindow->redraw();
    }
}

void close_cb( Fl_Widget* o, void*) {
   exit(0);
}

void os_linux_cb( Fl_Widget* o, void*){
    metadata.set_os(true);
    cout << "Linux" << endl;
}

void os_windows_cb( Fl_Widget* o, void*){
    metadata.set_os(false);
    cout << "Windows" << endl;
}

void about_cb(Fl_Widget* o, void*){
    about();
}

void drag_cb( Fl_Widget* o, void*){
    mywindow->redraw();
}

//MAIN-----------------------------------------------------------------------MAIN

int main(){
    int w=1000; // Width
    int h=600;  // Height
    int tx=30;  // Text height, for single line
    mywindow = new Fl_Double_Window(w,h,"EPUB Creator");
    mywindow->color(Fl_Color(fl_rgb_color(222, 222, 222)));
    {
        mainmenu = new Fl_Menu_Bar(0, 0, w, tx);
            mainmenu->color(Fl_Color(fl_rgb_color(222, 222, 222)));
            Fl_Menu_Item menu_menubar[] = {
                {"File", 0,  0, 0, 64, FL_NORMAL_LABEL, 0, 14, 0},
                {"OS",  0,   0, 0, FL_SUBMENU},
                    {"Linux", 0, (Fl_Callback*)os_linux_cb, 0, FL_MENU_RADIO|FL_MENU_VALUE},
                    {"Windows", 0, (Fl_Callback*)os_windows_cb, 0, FL_MENU_RADIO},
                {0},
                {"Exit", 0,  (Fl_Callback*)close_cb, 0, 0, FL_NORMAL_LABEL, 0, 14, 0},
                {0,0,0,0,0,0,0,0,0},
                {"Edit", 0,  0, 0, 64, FL_NORMAL_LABEL, 0, 14, 0},
                {"Update",0, (Fl_Callback*)Meta::print_cb,0,0, FL_NORMAL_LABEL,0,14,0},
                {"Clear",0, (Fl_Callback*)Meta::clear_all_cb,0,0, FL_NORMAL_LABEL,0,14,0},
                {0,0,0,0,0,0,0,0,0},
                {"About",0, (Fl_Callback*)about_cb,0,0, FL_NORMAL_LABEL,0, 14,0},
                {0,0,0,0,0,0,0,0,0}
            };
            mainmenu->menu(menu_menubar);

        tile = new Fl_Tile(0,0,w,h);
            tile->callback(drag_cb,0);
        {
            lefthalf = new Fl_Group(0, 0, w/2, h);
            {
                intitle = new Fl_Input(80, 40, w/2-100, tx, "Title");
                    intitle->callback((Fl_Callback*)Meta::intitle_cb);
                    intitle->tooltip("Please enter a title");

                inid = new Fl_Input(80, 80, w/2-100, tx, "ID");
                    inid->callback((Fl_Callback*)Meta::inid_cb);

                inlang = new Fl_Input_Choice(80, 120, w/2-100, tx, "Language");
                    inlang->add("English");
                    inlang->add("English - GB");
                    inlang->add("English - US");
                    inlang->add("French");
                    inlang->add("French - Canada");
                    inlang->add("Spanish");
                    inlang->add("German");
                    inlang->callback((Fl_Callback*)Meta::inlang_cb);

                inauthor = new Fl_Input(80, 160, w/2-100, tx, "Author");
                    inauthor->callback((Fl_Callback*)Meta::inauthor_cb);

                inpublish = new Fl_Input(80, 200, w/2-100, tx, "Publish");
                    inpublish->callback((Fl_Callback*)Meta::inpublish_cb);

                indate = new Fl_Input(80, 240, w/2-175, tx, "Date");
                    indate->callback((Fl_Callback*)Meta::indate_cb);
                    today = new Fl_Button(410, 240, 70, tx, "Today");
                        today->callback((Fl_Callback*)Meta::today_cb);
                        today->color(Fl_Color(fl_rgb_color(222, 222, 222)));

                tabs = new Fl_Tabs(80, 280, w/2-100, 210);
                {
                    // HTML tab
                    htmls = new Fl_Group(80, 310, w/2-100, 180, "HTML FIles");
                    htmls->color(Fl_Color(fl_rgb_color(222, 222, 222)));
                    {
                        htmlmenu = new Fl_Menu_Bar(90, 320, 200, tx);
                            htmlmenu->add("Select HTML Files", 0, Meta::openhtml_cb);
                            htmlmenu->add("Clear", 1, Meta::clearhtml_cb);
                        htmltext = new Fl_Text_Display(90, 360, w/2-120, 120);
                        htmlbuff = new Fl_Text_Buffer();
                            htmltext->buffer(htmlbuff);
                    }
                    htmls->end();

                    // Images tab
                    images = new Fl_Group(80, 310, w/2-100, 180, "Image FIles");
                    images->color(Fl_Color(fl_rgb_color(222, 222, 222)));
                    {
                        imagemenu = new Fl_Menu_Bar(90, 320, 200, tx);
                            imagemenu->add("Select Image Files", 0, Meta::openimage_cb);
                            imagemenu->add("Clear", 1, Meta::clearimages_cb);
                        imagetext = new Fl_Text_Display(90, 360, w/2-120, 120);
                        imagebuff = new Fl_Text_Buffer();
                            imagetext->buffer(imagebuff);
                            imagetext->resizable();
                    }
                    images->end();
                }
                tabs->end();

                close = new Fl_Button(80, h-50, 100, tx, " &Quit");
                    close->callback(close_cb);
                    close->image(image_exit);
                    close->align(Fl_Align(256));
                update = new Fl_Button(190, h-50, 100, tx, " &Update");
                    update->callback((Fl_Callback*)Meta::print_cb);
                    update->image(image_update);
                    update->align(Fl_Align(256));
                create = new Fl_Button(300, h-50, 100, tx, " &Create!");
                    create->callback((Fl_Callback*)Meta::create_cb);
                    create->image(image_create);
                    create->align(Fl_Align(256));
                info = new Fl_Button(410, h-50, 30, tx, "?");
                    info->callback((Fl_Callback*)info_cb);
                    info->labelcolor(FL_BLUE);
                hidden = new Fl_Box(440, h-50, 30, tx);
                    hidden->hide();
            }
            lefthalf->end();
            lefthalf->resizable(hidden);

            righthalf = new Fl_Box(500, 0, w/2, h);
            {
                outputtabs = new Fl_Tabs(500, 40, w/2, h-40);
                {
                    outputwindow1 = new Fl_Group(500, 70, w/2, h-70, "content.opf");
                    {
                        code1text = new Fl_Text_Editor(500, 70, w/2, h-70);
                        code1buff = new Fl_Text_Buffer();
                            code1text->buffer(code1buff);
                    }
```

```
                    outputwindow1->end();
                    outputwindow2 = new Fl_Group(500, 70, w/2, h-70, "container.xml");
                    {
                        code2text = new Fl_Text_Editor(500, 70, w/2, h-70);
                        code2buff = new Fl_Text_Buffer();
                            code2text->buffer(code2buff);
                    }
                    outputwindow2->end();
                    outputwindow3 = new Fl_Group(500, 70, w/2, h-70, "toc.ncx");
                    {
                        code3text = new Fl_Text_Editor(500, 70, w/2, h-70);
                        code3buff = new Fl_Text_Buffer();
                            code3text->buffer(code3buff);
                    }
                    outputwindow3->end();
                }
                tabs->end();
            }
        }
        tile->end();
    }
    mywindow->resizable(tile);
    mywindow->end();
    Fl::scheme("gtk+");
    mywindow->show();

    Fl::run();
}
```