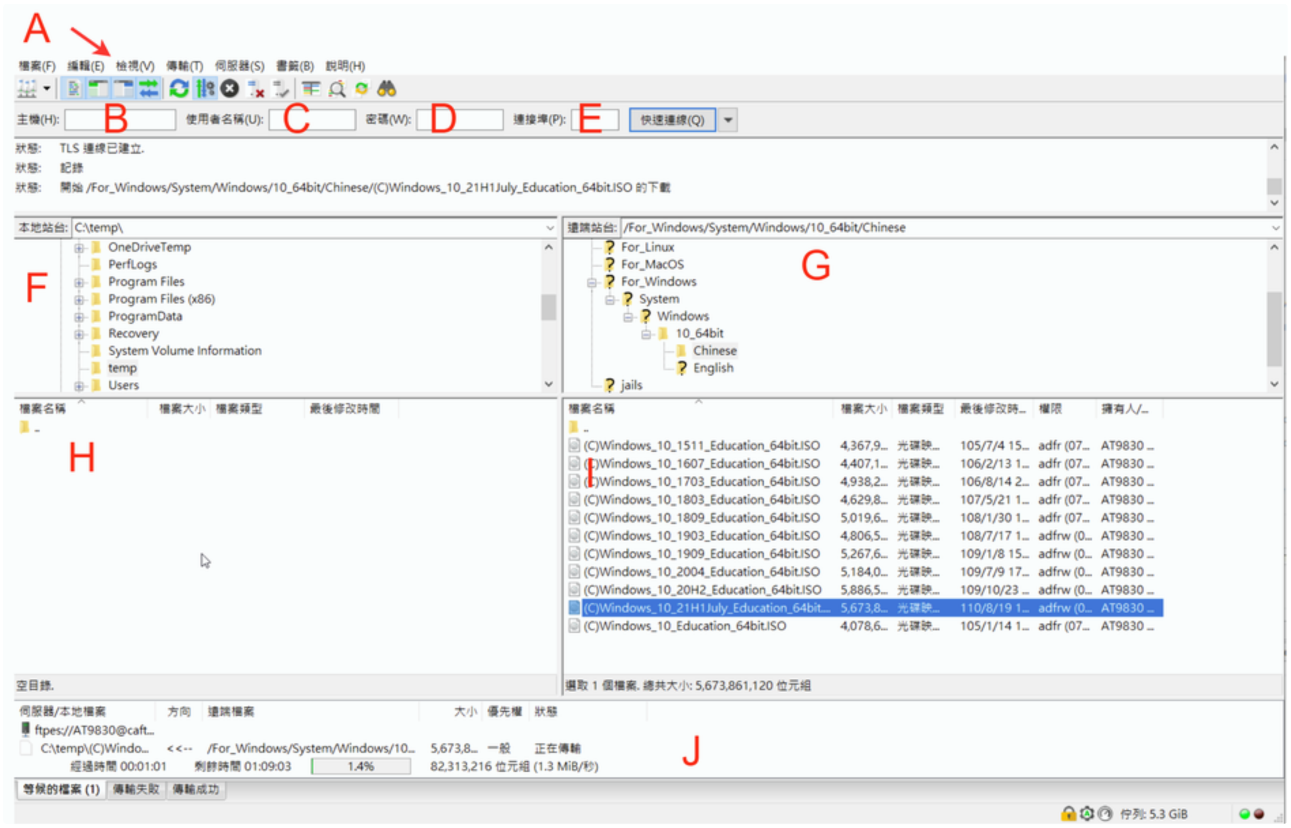
Client GUI Design (2022 OS)

<Table of Contents>

1. Overview

In this part of the project, we will implement the user interface of the FTP client as shown in the following.



```
狀態:	TLS 連線已建立.
狀態:	記錄
狀態:	開始 /For_Windows/System/Windows/10_64bit/Chinese/(C)Windows_10_21H1July_Education_64bit.ISO 的下載
```

本地站台: C:\temp\

OneDriveTemp
PerfLogs
Program Files
Program Files (x86)
ProgramData
Recovery
System Volume Information
temp
Users

檔案名稱	檔案大小	檔案類型	最後修改時間
..			

空目錄.

遠端站台: /For_Windows/System/Windows/10_64bit/Chinese

For_Linux
For_MacOS
For_Windows
System
Windows
10_64bit
Chinese
English
jails

檔案名稱	檔案大小	檔案類型	最後修改時...	權限	擁有人/...
..					
(C)Windows_10_1511_Education_64bit.ISO	4,367,9...	光碟映...	105/7/4 15...	adfr (07...	AT9830 ...
(C)Windows_10_1607_Education_64bit.ISO	4,407,1...	光碟映...	106/2/13 1...	adfr (07...	AT9830 ...
(C)Windows_10_1703_Education_64bit.ISO	4,938,2...	光碟映...	106/8/14 2...	adfr (07...	AT9830 ...
(C)Windows_10_1803_Education_64bit.ISO	4,629,8...	光碟映...	107/5/21 1...	adfr (07...	AT9830 ...
(C)Windows_10_1809_Education_64bit.ISO	5,019,6...	光碟映...	108/1/30 1...	adfr (07...	AT9830 ...
(C)Windows_10_1903_Education_64bit.ISO	4,806,5...	光碟映...	108/7/17 1...	adfrw (0...	AT9830 ...
(C)Windows_10_1909_Education_64bit.ISO	5,267,6...	光碟映...	109/1/8 15...	adfrw (0...	AT9830 ...
(C)Windows_10_2004_Education_64bit.ISO	5,184,0...	光碟映...	109/7/9 17...	adfrw (0...	AT9830 ...
(C)Windows_10_20H2_Education_64bit.ISO	5,886,5...	光碟映...	109/10/23 ...	adfrw (0...	AT9830 ...
(C)Windows_10_21H1July_Education_64bit...	5,673,8...	光碟映...	110/8/19 1...	adfrw (0...	AT9830 ...
(C)Windows_10_Education_64bit.ISO	4,078,6...	光碟映...	105/1/14 1...	adfr (07...	AT9830 ...

選取 1 個檔案. 總共大小: 5,673,861,120 位元組

伺服器/本地檔案	方向	遠端檔案	大小	優先權	狀態
ftpes://AT9830@caft...					
C:\temp\(C)Windo...	<<--	/For_Windows/System/Windows/10...	5,673,8...	一般	正在傳輸
經過時間 00:01:01	剩餘時間 01:09:03	1.4%	82,313,216 位元組 (1.3 MiB/秒)		

佇列: 5.3 GiB

FTP Client GUI (from FileZilla)

Using FileZilla as our example, the client UI should have the following elements:

A. Drop-down menu bar

B. Server hostname/IP address field

C. Login ID field

D. Password field

E. Server port number field

F. Local directory view

G. Remote directory view

H. Local file list view

I. Remote file list view

J. Job queue and status window

We will use Qt for building the graphical user interface. The target platform for the client program is Ubuntu with Unity Desktop.

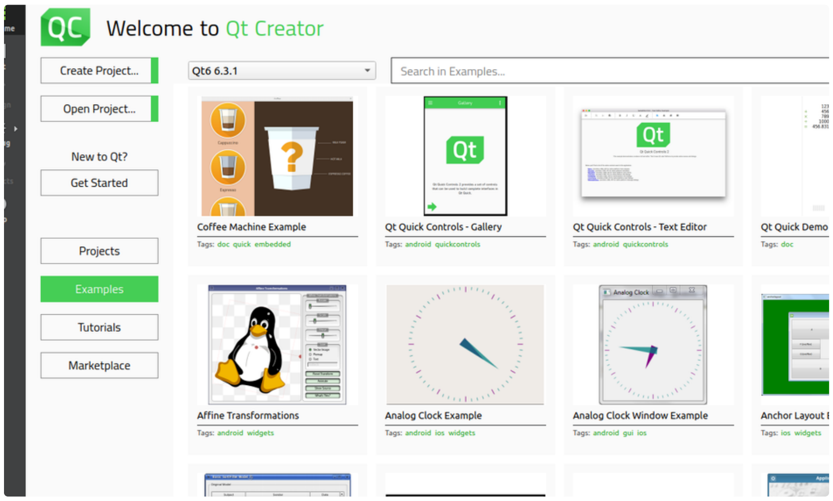
2. Qt Programming

Here we will walk you through the setup of Qt. The section will also cover how to create a GUI window with Qt along with adding the widgets (e.g., menus, buttons, combo box) to a window. We will also talk about how to set up the signals and slots for handling GUI events (e.g., mouse click, etc).

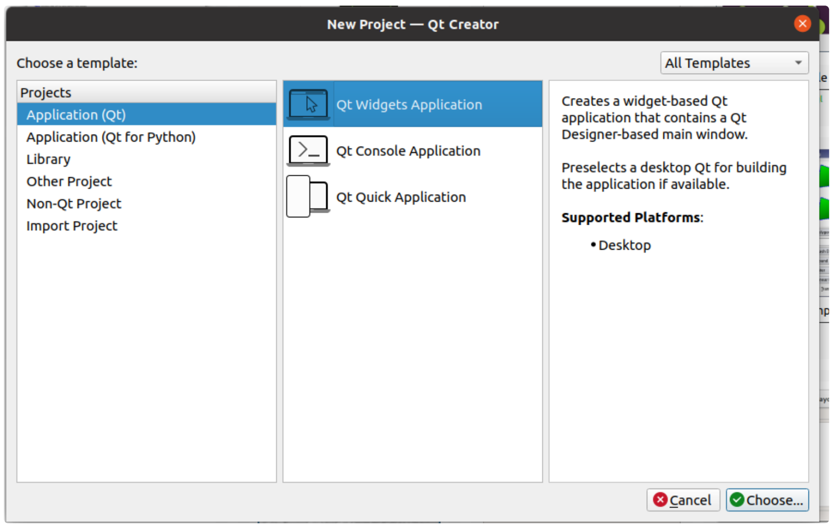
For a more complete introduction to Qt, please refer to the tutorial and documentation (Introduction to Qt | Qt 6.4)

2.1. Installation

1. Install prerequisites
   `sudo apt-get update && sudo apt-get upgrade`
   `sudo apt-get -y install build-essential openssl libssl-dev libssl1.0 libgl1-mesa-dev libqt5x11extras5`
2. Download QT installer
   Download Qt: Get Qt Online Installer
3. Run the Qt installer
   Remeber to select “qt 6.3 desktop” to install



Create Project

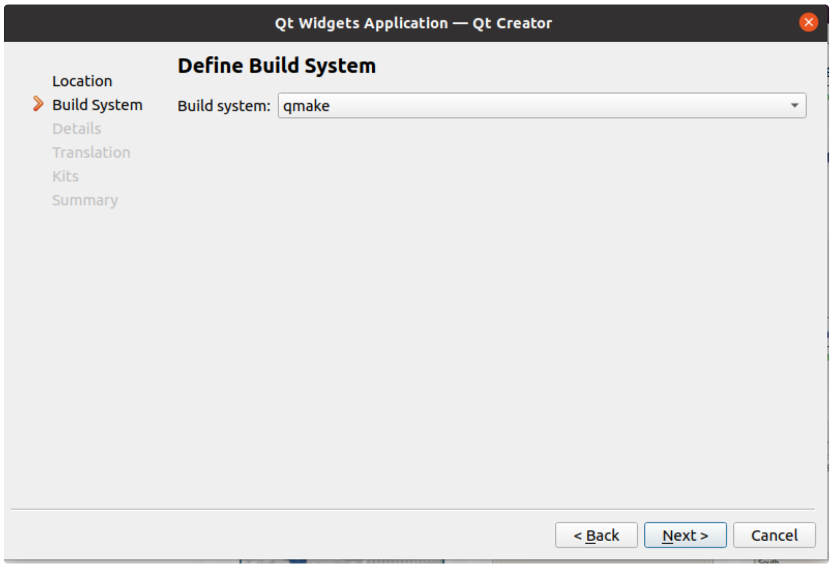


Creates a widget-based Qt application that contains a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms:

- Desktop



Define Build System

Build system: qmake

***.pro**

Define makefile for qmake.

The .pro file should be edited according to the qmake corresponding to the library you included.

Please refer to the .pro file provided by us for the homework.

Compiled command

```
qmake
make
```

2.2. Overview of Qt UI Components

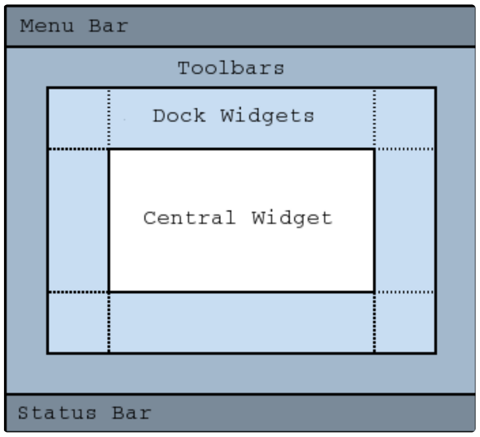
2.2.1. Header file MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
};
#endif
```

The `MainWindow` class extends `QMainWindow` from the Qt framework. We have a `ui` member variable in the private fields. The type is a pointer of `Ui::MainWindow`, which is defined in the `ui_MainWindow.h` file generated by Qt. It's the C++ transcription of the UI design file `MainWindow.ui`. The `ui` member variable will allow you to interact with your UI components (`menubar`, `statusbar`, and so on) from C++.

`Q_OBJECT` is macro allows the class to define its own signals/slots and more globally Qt's meta-object system.

Please note that the drop-down menu must be written on the menubar which is a member of ui(i.e ui->menubar)



the component under ui

2.2.2. A Drop-Down Menu

`QMenu` is a widget that provides a list of options. If you want a **drop-down menu**, you must add the `QMenu` type widget to the `QMenuBar` type widget.

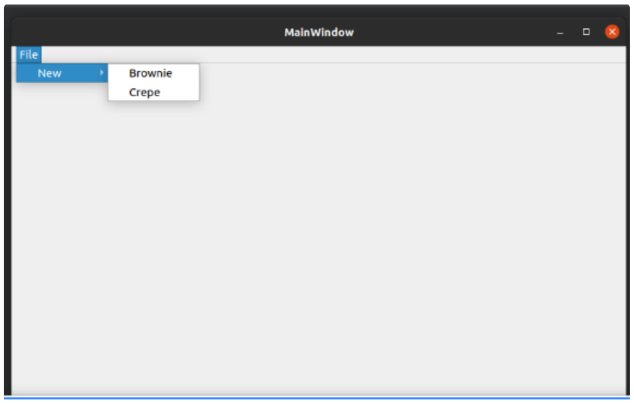
In the most basic GUI framework, there is a member `menubar` as type `QMenuBar` in the private field `ui`. We can directly add a custom `QMenu` or `QAction` to the menubar. There are two cases:

First, if the option you want to add to the main menu is **another menu**, then what you want to add to the main menu is a `QMenu*` type variable. This can be done simply with the `QMenu* addMenu(QMenu*)` function.

Second, if you want to add **a command (user action)**, then you should add a `QAction*` type variable to the main menu, you must use `QAction* addAction(String option)` function to complete it.

For example,

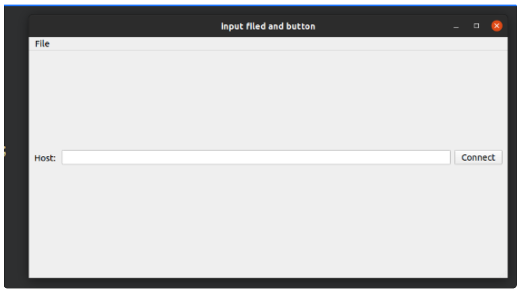
```
auto menu = new QMenu("File");
auto submenu = menu->addMenu("New"); //add a submenu
submenu -> addAction("Brownie");// add a user action(command)
submenu -> addAction("Crepe");
menu -> addMenu(submenu);
ui->menubar->addMenu(menu);//add to the ui->menubar
```



2.2.3. Input Field And Button

The components are mostly extend `QWidget` which is a UI component. It can be a label, a textbox, a button, and so on, in the following example is **label, lineEdit and button**. If you define a parent-child relationship between your window, layout, and other UI widgets, memory management of your application will be easier.For example, in below we add a `QHBoxLayout* horizon` as child to `widget` first , then we add widget as child to ui which is implement by `setCentralWidget(widget);`.For example,

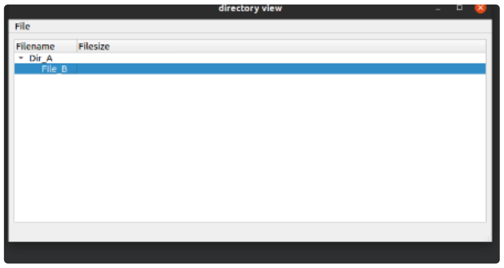
```
//input field
auto widget = new QWidget();
auto horizon = new QHBoxLayout();
std::vector<Qwidget*> userinput_filed;
userinput_field.push_back(new QLabel("Host: "));
userinput_field.push_back(new QLineEdit());
userinput_field.push_back(new QPushButton("Connect"));
for(auto uf : userinput_field)
  horizon -> addWidget(uf);
widget -> setLayout(horizon);
//set the centrolwidget in ui to your widget
setCentralWidget(widget);
```



2.2.4. Directory View

We can use `QTreeWidget` to implement **Directory View**, here is introduction for the functions and objects that will be used.
The first is `setHeaderLabel(const QStringList)`, which can be used to determine **what headers are, such as file name, size, and so on.**
The second is `QTreeWidgetItem`, which refers to an object that can be managed and displayed by `QTreeWidget`.
The third is `addTopLevelItem(QTreeWidgetItem*)`, which can be used to **determine the top level** of the file structure, and the last is `addChild(QTreeWidgetItem*)`, which is used to **increase the directory** of a `QTreeWidgetItem`.For example,

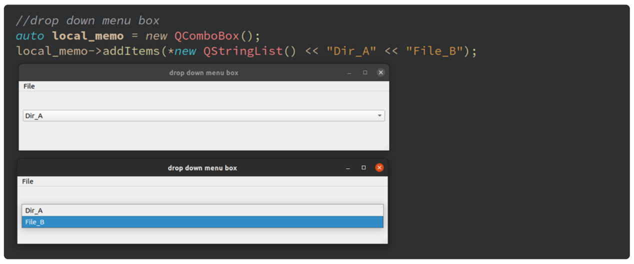
```
//directory view
auto local_site = new QTreeWidget();
local_site -> setHeaderLabels(*new QStringList() << "Filename" << "Filesize");
auto first_layer = new QTreeWidgetItem(*new QStringList()<< "Dir_A")
local_site->addTopLevelItem(first_layer);
first_layer->addChild(new QTreeWidgetItem(*new QStringList()<< "File_B"));
auto widget = new QWidget();
auto horizon = new QHBoxLayout();
horizon->addWidget(local_site);
wiget->setLayout(horizon);
setCentralWidget(widget);
```



2.2.5. ComboBox

To create a drop-down lists that can display the option to user, we can use `QCombox` to implement it.

Use `addItems(const QStringList)` to add options to the box.



```
//drop down menu box
auto local_memo = new QComboBox();
local_memo->addItems(*new QStringList() << "Dir_A" << "File_B");
```

2.2.6. Signal And Slot

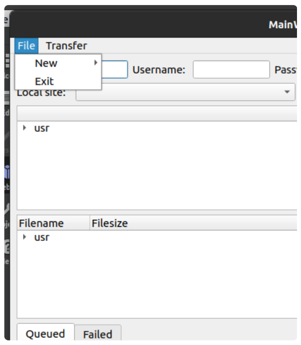
In GUI programming, when we change one widget, we often want another widget to be notified. More generally, we want objects of any kind to be able to communicate with one another. For example, if a user clicks a Close button, we probably want the window's `close()` function to be called.

Usage:

```
connect(the_object_emit_signal, signal, the_object_call_the_slot_function, slot);
```

Example:

There is a “Exit” option in “File” menu:



When the “Exit” is clicked ,the window should be closed, then we have

```
auto act = menu -> addAction("Exit");
connect(act, SIGNAL(triggered()), this, SLOT(close()));
```

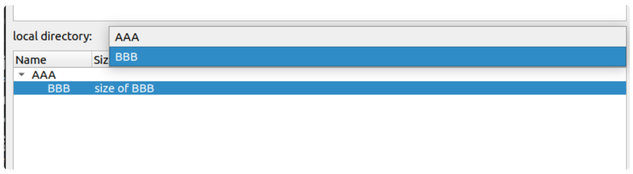
in MainWindow.cpp.

learn more signal and slot

2.2.7. The Signal And Slot between Directory View And ComboBox

Mimicking the UI design of FileZilla, we would like the file list view (H) to show the files in the directory selected in the directory view (F). So, when we change to a new directoy in F, the file list content in H needs to be updated accordingly.

To make this happen, we need to define a slot function in `QTreeWidget` to handle the signal of mouse clicking. In the slot function, we will update the list of files in the combobox for the file list view, like:



You can go to Qt documentation to check the signal of QTreewidget.

1. Declare the slot function in the header and implement the function to be invoked when the item in the treewidget gets clicked.

   In the column 8, 9 of header file:

```
 1  class MainWindow : public QMainWindow
 2  {
 3      Q_OBJECT
 4
 5  public:
 6      ...
 7  //Declare the slot function
 8  public slots:
 9      void clickitem(QTreeWidgetItem *item);
10  private:
11      Ui::MainWindow *ui;
12      ...
13  };
```

   In the MainWindow.cpp:

```
void MainWindow::clickitem(QTreeWidgetItem *item)
{
    combobox->addItem(item->text(0));
}
```

   Look at the **slot function's parameter** `QTreeWidgetItem *item`, this is refer to the item is clicked **in the caller**(`treeWidget`) **of signal function** `itemClicked(QTreeWidgetItem*, int)`

2. Associate the slot function with the signal via connect(…).

   In the MainWindow.cpp:

```
auto treeWidget = new QTreeWidget();
auto combobox = new QComboBox();
//some design for treeWidget and combobox
...
//connect signal and slot
connect(treeWidget,SIGNAL(itemClicked(QTreeWidgetItem*, int)),this, SLOT(clickitem(QTreeWidgetItem*)));
```

- Purpose two: When the item in the combobox is changed, the treeWidget will also display the changed item.

1. Declare slot function:

```
public slots:
    ...
    void dir_combo_changed(const QString&);
```

2. Define slot function:
   We can use `QTreeWidgetItemIterator` to iterate the item in the `treeWidget`.

```
void MainWindow::dir_combo_changed(const QString &text)
{
    QTreeWidgetItemIterator it(treeWidget);
    while (*it)
    {
          if ((*it)->text(0) == text)
              treeWidget->setCurrentItem((*it));
          ++it;
    }
}
```

3. Connect signal and slot

```
connect(combobox, SIGNAL(currentTextChanged(QString)),this,SLOT(dir_combo_changed(QString)));
```

2.2.8. TableWidget

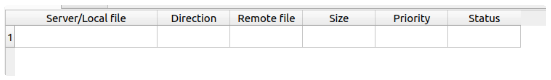
You can create a table view by `QTableWidget`

```
auto queue = new QTableWidget(1, 6);
```

and set its header labels

```
queue->setHorizontalHeaderLabels(*new QStringList()<< "Server/Local file" << "Direction"
<< "Remote file" <<"Size"<< "Priority" << "Status");
```

It would look like

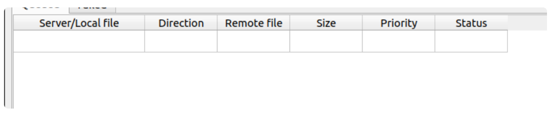


	Server/Local file	Direction	Remote file	Size	Priority	Status
1						

You can use `QHeaderView` to customize the header(the `verticalHeader()` return the type `QHeaderView*`), for example,

```
queue->verticalHeader()->hide();
```

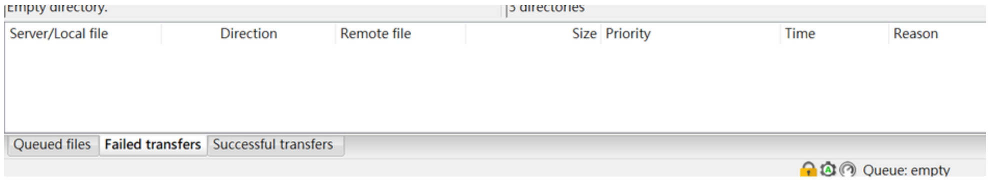
then the vertical header would be hidden.



Server/Local file	Direction	Remote file	Size	Priority	Status

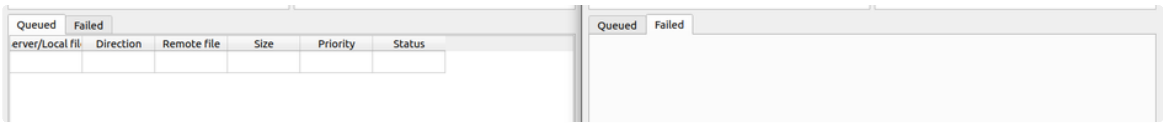
2.2.9. TabWidget

Remeber that in Filezilla there is a status view component with tab to change the queued cases , the success cases , and failed cases like,



We can do it by `QTabWidget`, and add the table we create in the previous subtitle **TableWidget**.

```
auto transfer_status = new QTabWidget();
transfer_status ->addTab(queue,"Queued");//add the QTableWidget queue with tab's name "Queued"
transfer_status ->addTab(new QWidget(),"Failed");//add a nuul widget
```



If you want to adjust the position of tabs like the tab view in Filezilla, you can write,

```
transfer_status->setTabPosition(QTabWidget::South);
```

2.2.9. Layout

QLayout is useful for arrangging the widget, it allows the components in the layout to adapt to the size of the window.

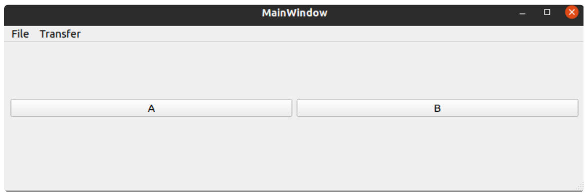
The most we use are:

`QHBoxLayout`: it can arrange widgets horizontally.

`QVBoxLayout`: arrange widgets vertically.

You can add some widgets to layout:

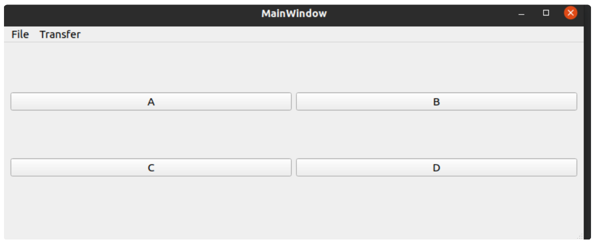
```
auto widget = new QWidget();
auto horizon = new QHBoxLayout();
horizon->addWidget(new QPushButton("A"));
horizon->addWidget(new QPushButton("B"));
widget->setLayout(horizon);
setCentralWidget(widget);
```



Also you can add layouts to another layout:

```
auto horizon = new QHBoxLayout();
horizon->addWidget(new QPushButton("A"));
horizon->addWidget(new QPushButton("B"));

auto horizon2 = new QHBoxLayout();
horizon2->addWidget(new QPushButton("C"));
horizon2->addWidget(new QPushButton("D"));

auto vertical = new QVBoxLayout();
vertical->addLayout(horizon);
```

```
vertical->addLayout(horizon2);

widget->setLayout(vertical);
setCentralWidget(widget);
```



<Submission Checklist>

An unfinished implementation of the FTP client's GUI is provided to help you finish the project. Unzip the zip file and follow the following steps to build the unfinished FTP client program.

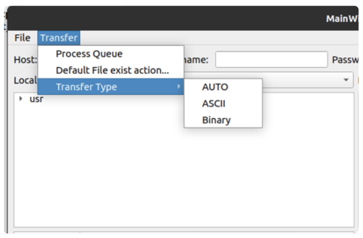
```
sudo apt-get install make
sudo apt-get install g++
sudo apt-get install qt5-default
#If you use Ubuntu 22.04, try 'sudo apt-get install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools' instead
qmake
make
```

You may now launch the FTP client program by running the executable ftp_gui generated by the build process.

It is recommended that you read the code thoroughly and understand the usage of the Qt APIs.

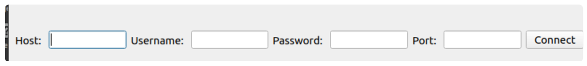
Now you need to add back the following missing UI elements:

1. MenuBar
   You need to add back the "Transfer" menu in `mainmenu()` function. Under the "Transfer" menu, there should be the following menu items
   a. Process Queue
   b. Default File exist action...
   c. Transfer Type
      i. AUTO
      ii. ASCII
      iii. Binary
      It should look like,



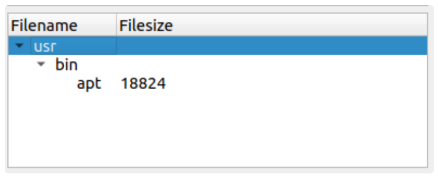
2. Add the input fields for the connection parameters and the "Connect" button in the `input_field()` function:

   a. You should implement the label(`QLabel`) "Host:", "Username:", "Password:" and "Port:", and each label follows a line edit(`QLineEdit`) component.
   b. There is a “Connect” button(`QButton`) to the right of the last label.
   c. Use the `QHBoxLayout` for typesetting.
   It should look like the figure below,

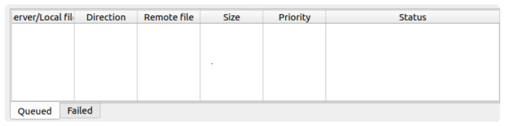


The fully implemented input fields and connect button

3. Local file view(in `local_view()` function)
   a. add an item to represent file “apt” under bin/ with file size 18824
   b. add the header label “Filename” and “Filesize”
   like this,



4. Status_view(in `status()` function)
   a. set the header of QTableWidget in tab “Queued” to "Server/Local file", "Direction" , "Remote file", "Size", "Priority" and "Status".
   b. add a null tab page with the tab name “Failed”



5. Signal and Slot(in `mainmenu` function)
   a. Add the menu item “Exit” to the “File” menu; Finish the implementation of the signal and slot so that the window will get closed when “Exit” is clicked.

Please complete the missing parts and submit the following items

1. A zip file of your FTP client code.
   a. With a Makefile so the TA can build your code with the make command
      i. main.cpp
      ii. mainwindow.h
      iii. mainwindow.cpp
      iv. mainwindow.ui
      v. *.pro
   b. Make sure that the code work will work on Ubuntu 20.04.4 LTS (or Ubuntu 22.04 LTS)
2. A PDF report describing the status of your FTP client implementation (i.e., which of the missing elements have been completed). It would be good if you can attach some screenshots so the TA can check if they match with the output from running your code (Item 1).

If you have any questions, please post it on the E3 forum for the course project. If needed, you can reach the TA for the project via Lyrae <lyra20545@gmail.com>.