

به نام خداوند

عرفان آهنگر سريزدي

۱۴۰۲۴۱۲۲

تمرین دوم شبکه های کامپیوتر

دکتر پهلواني

برای حل این تمرین در محیط ویندوز نیاز به نصب ۵ نرم افزار داریم:

(۱) نصب MinGW

(۲) نصب CMake

(۳) نصب git

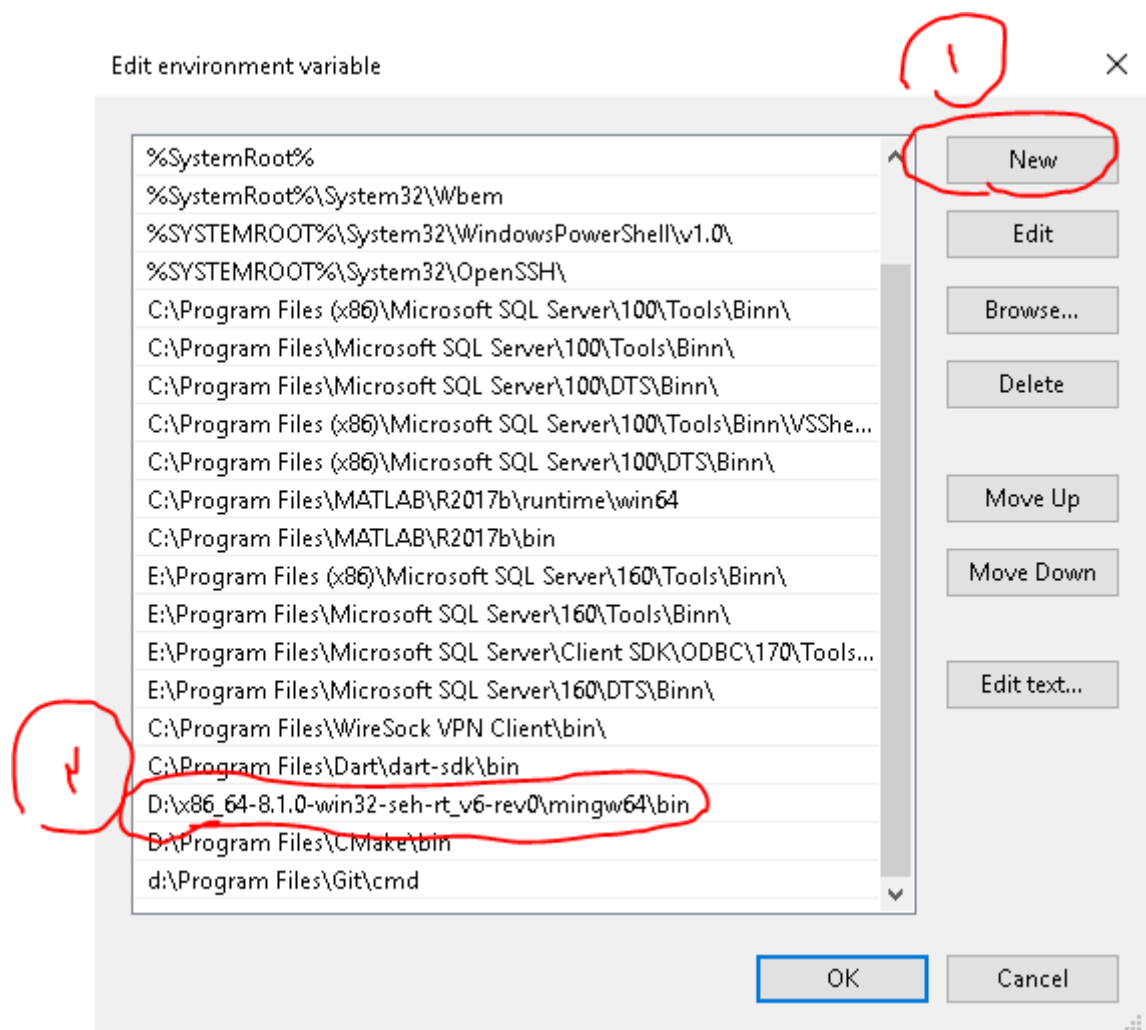
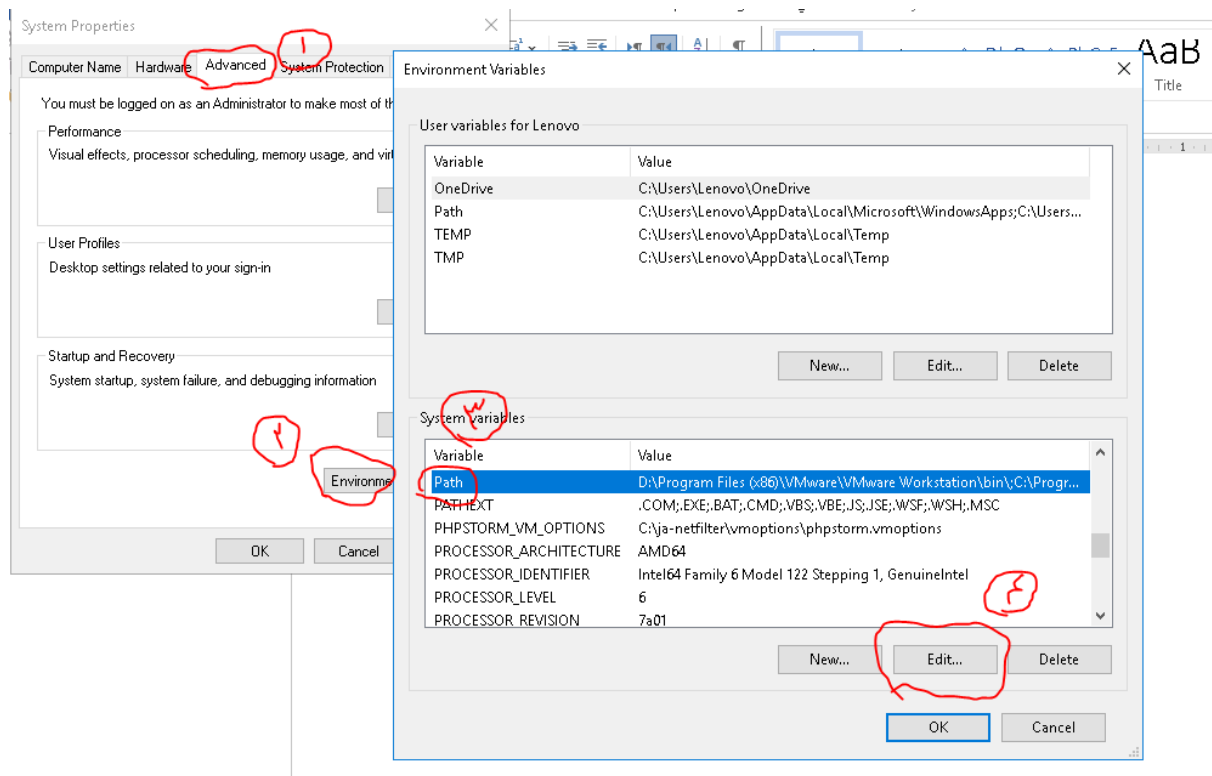
(۴) نصب کلاینت FTP (FileZilla)

(۵) نصب ویرایشگر کد مثل VS Code

خب در مرحله اول برای نصب MinGW ما نیاز داریم که فایل را دانلود کنیم و سپس از حالت فشرده خارج کنیم و حالا حال پوشه ساخته شده را در محل ثابتی (خیلی مهمه که تغییر نکنه در آینده!) کپی کنیم و سپس وارد پوشه Bin میشویم به طور پیش فرض من در این محل ریختم:

D:\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw64\bin

بعد در منوی سرچ ویندوز ، عبارت environment variables رو جست و جو میکنیم و حالا طبق عکس زیر پیش میریم:



باقی صفحات باز رو هم با زدن ok میبندیم و حالا محیط cmd را باز میکنم و دو دستور زیر را وارد میکنم

g++

gcc

و اجرا که کردم با پیغام زیر مواجه میشم یعنی نصب با موفقیت انجام شده:

fatal error: no input files
compilation terminated.

خب نصب مرحله اول با موفقیت انجام شده 😊

در مرحله دوم برای نصب CMake از سایت CMake دانلود میکنم و سپس آن را نصب کردم و گزینه اضافه شدن به PATH را در حین نصب انتخاب کردم

در مرحله سوم برای نصب git از سایت زیر برنامه را دانلود کردم و نصب کردم:

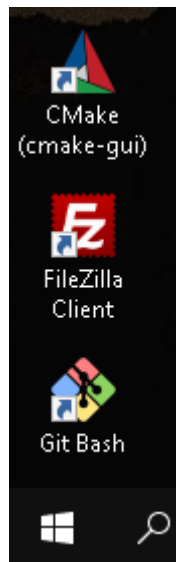
<https://git-scm.com/>

مرحله چهارم هم برای نصب کلاینت از سایت زیر برنامه را دانلود کردم و نصب کردم

<https://filezilla-project.org/>

مرحله پنجم هم vs code از قبل در سیستم داشتم و نیازی به نصب نبود 😊

پس کل برنامه های لازم برای انجام این تمرین نصب شده است:



خب نصب برنامه های مورد نظر تمام شد و وارد بخش بعدی میشویم

بخش نوشتن کد سرور FTP است که یه پوشه در محیط دیسکتاپ ایجاد کردم به اسم FTPServer



درون پوشه ساختار زیر را ایجاد کردم:

```
/FTPServer
CMakeLists.txt —┘
/src —┘
main.cpp —┘ |
ftp_server.cpp —┘ |
ftp_server.h —┘ |
/build —┘ L
```

یعنی دو تا فایل به اسم build و src و یه فایل txt به اسم CMakeLists

build	*V/*A/*F=*۳ ظ,ق,۱۱:۲۳	File folder	
src	*V/*A/*F=*۳ ب *V:۱۸...	File folder	
CMakeLists.txt	*V/*A/*F=*۳ *V:۲۳ ...	Text Document	1 KB

فایل CMakeLists.txt نحوه ساخت پروژه با استفاده از CMake را تعریف می کند.

```
CMakeLists.txt - Notepad
File Edit Format View Help
cmake_minimum_required(VERSION 3.10)
project(FTPServer)

set(CMAKE_CXX_STANDARD 17)

add_executable(ftp_server src/main.cpp src/ftp_server.cpp)
```

حالا وارد فایل SRC میشویم:

FTPServer > src				
	Name	Date modified	Type	Size
	ftp_server.cpp	*V/*A/*F=*۳ *V:۲۷ ...	C++ Source File	4 KB
	ftp_server.h	*V/*A/*F=*۳ *V:۲۶ ...	C Header File	1 KB
	main.cpp	*V/*A/*F=*۳ *V:۲۹ ...	C++ Source File	1 KB

در فایل ftp_server.h ما توابعی برای آپلود، دانلود، حذف، جستجو و لیست کردن فایل ها تعریف میکنیم

در فایل ftp_server.cpp توابع را با استفاده از برنامه‌نویسی سوکت و کتابخانه winsock2 (مخصوص ویندوز) پیاده‌سازی میکنیم

محتوای داخل فایل ftp_server.h:

```
ftp_server.h - Notepad
File Edit Format View Help
#ifndef FTP_SERVER_H
#define FTP_SERVER_H

#include <string>

// برای برنامه‌نویسی سوکت در ویندوز Winsock کتابخانه
#include <winsock2.h>

// تعریف کلاس FTPServer
class FTPServer {
public:
    FTPServer(int port); // سازنده سرور
    void start(); // شروع به کار سرور
    void handleClient(int clientSocket); // مدیریت اتصال یک کلاینت

    // توابع مختلف برای عملیات FTP
    void uploadFile(int clientSocket, const std::string& filename);
    void downloadFile(int clientSocket, const std::string& filename);
    void deleteFile(int clientSocket, const std::string& filename);
    void searchFile(int clientSocket, const std::string& filename);
    void listFiles(int clientSocket);

private:
    int serverSocket; // سوکت سرور
    int port; // پورت سرور
    void sendMessage(int clientSocket, const std::string& message); // ارسال پیام به کلاینت
    std::string receiveMessage(int clientSocket); // دریافت پیام از کلاینت
};

#endif
```

در ftp_server.cpp این توابع را پیاده‌سازی میکنیم:

ftp_server.cpp - Notepad

File Edit Format View Help

```
#include "ftp_server.h"
#include <iostream>
#include <fstream>
#include <filesystem>
namespace fs = std::filesystem;

#pragma comment(lib, "ws2_32.lib") // لینک کردن کتابخانه‌های winsock

FTPServer::FTPServer(int port) : port(port) {
    // راه اندازی Winsock
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);

    // ایجاد سوکت سرور
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);

    // تنظیم آدرس سرور
    sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(port);

    // اتصال سوکت به آدرس
    bind(serverSocket, (sockaddr*)&serverAddr, sizeof(serverAddr));

    // گوش دادن به اتصالات
    listen(serverSocket, SOMAXCONN);

    std::cout << "Server started on port " << port << std::endl;
}

void FTPServer::start() {
    while (true) {
        sockaddr_in clientAddr;
        int clientSize = sizeof(clientAddr);
        int clientSocket = accept(serverSocket, (sockaddr*)&clientAddr, &clientSize);

        // مدیریت هر کلاینت در یک رشته‌ای جدید (thread)
        handleClient(clientSocket);
    }
}

void FTPServer::handleClient(int clientSocket) {
    std::string command = receiveMessage(clientSocket);
    if (command == "UPLOAD") {
        std::string filename = receiveMessage(clientSocket);
        uploadFile(clientSocket, filename);
    }
}
```



```
] else if (command == "DOWNLOAD") {
    std::string filename = receiveMessage(clientSocket);
    downloadFile(clientSocket, filename);
} else if (command == "DELETE") {
    std::string filename = receiveMessage(clientSocket);
    deleteFile(clientSocket, filename);
} else if (command == "SEARCH") {
    std::string filename = receiveMessage(clientSocket);
    searchFile(clientSocket, filename);
} else if (command == "LIST") {
    listFiles(clientSocket);
}
closesocket(clientSocket);
}

void FTPServer::uploadFile(int clientSocket, const std::string& filename) {
    std::ofstream outFile(filename, std::ios::binary);
    char buffer[1024];
    int bytesRead;
    while ((bytesRead = recv(clientSocket, buffer, sizeof(buffer), 0)) > 0) {
        outFile.write(buffer, bytesRead);
    }
    outFile.close();
    sendMessage(clientSocket, "File uploaded successfully.");
}

void FTPServer::downloadFile(int clientSocket, const std::string& filename) {
    std::ifstream inFile(filename, std::ios::binary);
    if (!inFile) {
        sendMessage(clientSocket, "File not found.");
        return;
    }

    char buffer[1024];
    while (inFile.read(buffer, sizeof(buffer))) {
        send(clientSocket, buffer, sizeof(buffer), 0);
    }
    inFile.close();
    sendMessage(clientSocket, "File downloaded successfully.");
}

void FTPServer::deleteFile(int clientSocket, const std::string& filename) {
    if (fs::remove(filename)) {
        sendMessage(clientSocket, "File deleted successfully.");
    } else {
        sendMessage(clientSocket, "File not found.");
    }
}
```

```

    }
}

void FTPServer::listFiles(int clientSocket) {
    std::string fileList;
    for (const auto& entry : fs::directory_iterator(".")) {
        fileList += entry.path().string() + "\n";
    }
    sendMessage(clientSocket, fileList);
}

void FTPServer::sendMessage(int clientSocket, const std::string& message) {
    send(clientSocket, message.c_str(), message.size(), 0);
}

std::string FTPServer::receiveMessage(int clientSocket) {
    char buffer[1024];
    int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);
    return std::string(buffer, bytesReceived);
}

```

در main.cpp، سرور را ایجاد میکنیم و آن را اجرا میکنیم:

 main.cpp - Notepad

File Edit Format View Help

```

1 #include "ftp_server.h"
#include <iostream>

```

```

int main() {
    // پورت سرور را مشخص میکنیم (مثلاً ۵۴۰۰۰)
    int port = 54000;

    // FTPServer ایجاد یک نمونه از
    FTPServer server(port);

    // اجرای سرور
    server.start();

    return 0;
}

```

😊 خب این مرحله هم تمام شد و وارد مرحله بعد میشویم

خب حالا برای کامپایل در ویندوز و اجرای CMake ما نیاز به نصب برنامه داریم مثل Visual Studio یا برنامه ای کم حجم تر مثل MSYS2 که ما از برنامه دومی یعنی MSYS2 استفاده کردیم برای کامپایل در ویندوز

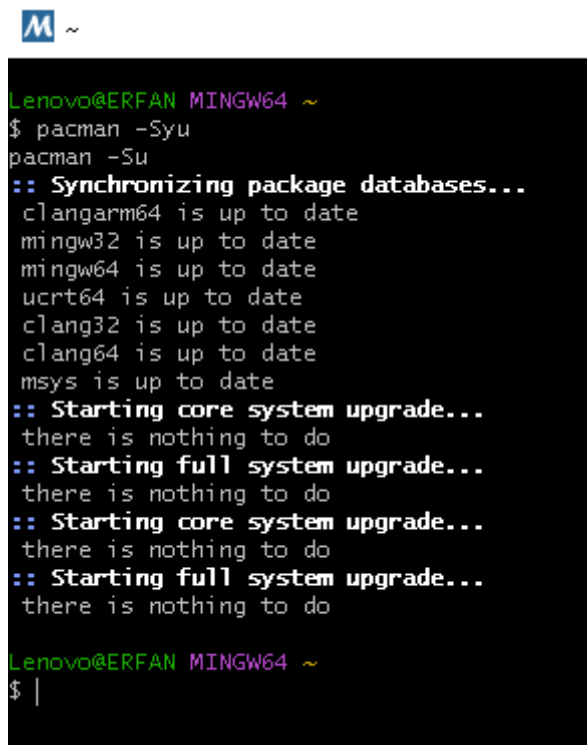
ابتدا برنامه مورد نیاز را از لینک زیر دانلود کردیم:

<https://www.msys2.org/>

سپس به طور کامل نصب کردیم. برای شروع نیاز به بروزرسانی MSYS2 است که برای بروزرسانی پایگاه داده بسته ها در محیط MSYS2 این کد را میزنیم تا بروزرسانی شود:

pacman -Syu

pacman -Su



```
Lenovo@ERFAN MINGW64 ~  
$ pacman -Syu  
pacman -Su  
:: Synchronizing package databases...  
clangarm64 is up to date  
mingw32 is up to date  
mingw64 is up to date  
ucrt64 is up to date  
clang32 is up to date  
clang64 is up to date  
msys is up to date  
:: Starting core system upgrade...  
there is nothing to do  
:: Starting full system upgrade...  
there is nothing to do  
:: Starting core system upgrade...  
there is nothing to do  
:: Starting full system upgrade...  
there is nothing to do  
Lenovo@ERFAN MINGW64 ~  
$ |
```

سپس مرحله نصب کامپایلر GCC و CMake در MSYS2 است که با دستورات زیر نصب میکنیم:

pacman -S mingw-w64-x86_64-gcc

pacman -S mingw-w64-x86_64-cmake

```

Lenovo@ERFAN MINGW64 ~
$ pacman -S mingw-w64-x86_64-gcc
warning: mingw-w64-x86_64-gcc-14.2.0-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) mingw-w64-x86_64-gcc-14.2.0-1

Total Installed Size: 224.83 MiB
Net Upgrade Size:      0.00 MiB

:: Proceed with installation? [Y/n] |

```

```

Lenovo@ERFAN MINGW64 ~
$ pacman -S mingw-w64-x86_64-cmake
warning: mingw-w64-x86_64-cmake-3.30.5-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) mingw-w64-x86_64-cmake-3.30.5-1

Total Installed Size: 45.38 MiB
Net Upgrade Size:      0.00 MiB

:: Proceed with installation? [Y/n] |

```

با نوشتن Y نصب شروع میشود که من قبلا نصب کردم نیازی به نصب مجدد نیست

پس از نصب نوبت به کامپایل پروژه با استفاده از CMake است

با دستور زیر وارد پوشه ftp server میشوم

```
cd /c/Users/YourUsername/Desktop/FTPServer
```

```

Lenovo@ERFAN MINGW64 ~
$ cd /c/Users/Lenovo/Desktop/FTPServer

Lenovo@ERFAN MINGW64 /c/Users/Lenovo/Desktop/FTPServer
$ |

```

سپس cd build میزنم که وارد پوشه build شوم

```

Lenovo@ERFAN MINGW64 /c/Users/Lenovo/Desktop/FTPServer/build
$ |

```

سپس دستور زیر را برای پیکربندی پروژه اجرا میکنم:

```
cmake -G "MSYS Makefiles" ..
```

این دستور CMake را برای ایجاد فایل‌های ساخت با استفاده از make تنظیم میکند

و پیکربندی با موفقیت انجام میشود و فایل‌های زیر در پوشه ایجاد میشود:

FTPServer > build

Name	Date modified	Type	Size
CMakeFiles	۱۰:۴۴ ق.ظ ۸/۸/۱۴۰۳	File folder	
cmake_install.cmake	۱۰:۴۴ ق.ظ ۸/۸/۱۴۰۳	CMAKE File	2 KB
CMakeCache.txt	۱۰:۳۸ ق.ظ ۸/۸/۱۴۰۳	Text Document	3 KB
Makefile	۱۰:۴۴ ق.ظ ۸/۸/۱۴۰۳	File	7 KB

حالا نوبت به ساخت پروژه میرسد که بعد از پیکربندی، پروژه را با دستور زیر کامپایل میکنیم:

make

و در مرحله آخر اجرای پروژه است که پس از اتمام کامپایل، فایل اجرایی پروژه باید در پوشه build موجود باشد. برای اجرای برنامه:

ftp_server/.

میزنیم و برنامه اجرا میشود...

این مرحله تمام شد 😊

مرحله بعدی تست سرور است. برای تست سرور FTP خود، می‌توانیم از یک کلاینت FTP مانند FileZilla یا WinSCP استفاده کنیم که من از filezilla استفاده کردم تا عملیات‌های مختلف (آپلود، دانلود، حذف و ...) را بررسی کنم.

اجرای سرور

ابتدا مطمئن میشویم که سرور FTP ما در حال اجراست:

وارد پوشه‌ی پروژه در محیط MSYS2 MinGW میشم و از طریق اجرای فایل اجرایی سرور که ساختم، سرور را اجرا میکنم. نام فایل من ftp_server است، دستور زیر را وارد میکنم:

ftp_server/.

حالا سرور باید در این حالت آماده پذیرش اتصالات از کلاینت‌ها باشد که خوشبختانه هست 😊

حالا از FileZilla برای تست اتصال به سرور استفاده میکنم

FileZilla Client را دانلود و نصب کردم و حالا نوبت به پیکربندی اتصال است که FileZilla را باز میکنیم و اطلاعات اتصال به سرور را به شکل زیر وارد میکنیم:

Host: آدرس IP سرور از localhost یا 127.0.0.1 استفاده کردم

Username: نام کاربری FTP که در سرور تنظیم کرده‌ایم

Password: در صورت نیاز، رمز عبور وارد میکنم

Port: پورتی که سرور من روی آن گوش می‌دهد مثلاً 21

اتصال به سرور: روی دکمه‌ی Quickconnect کلیک میکنیم تا اتصال برقرار شود.

Quickconnect	:	Port	:	Password	:	Username	:	localhost	:	Host
Resolving address of localhost										Status:
...Connecting to [::1]:21										Status:
...Connection established, waiting for welcome message										Status:
...Initializing TLS										Status:
.TLS connection established										Status:
USER anonymous										Command:
331 Please, specify the password.										Response:
PASS *****										Command:
530 Login incorrect.										Response:

حالا نوبت به تست عملیات مختلف در filezilla است 😊

آپلود فایل:

- یک فایل را از سیستم انتخاب میکنیم و آن را به پنجره سرور (سمت راست) در FileZilla میکشیم و رها میکنیم
- FileZilla باید فایل را به سرور آپلود کند. مطمئن میشویم که فایل در سرور ذخیره شده است.

دانلود فایل:

- روی یک فایل موجود در سرور کلیک راست میکنیم و **Download** را انتخاب میکنیم
- FileZilla باید فایل را از سرور دانلود کند که حتما اینکارا میکند

حذف فایل:

- روی فایلی در سرور کلیک راست میکنیم و گزینه **Delete** را انتخاب میکنیم.
- فایل باید از سرور حذف شود.

جستجو و مشاهده فایل‌ها:

- لیست فایل‌ها و پوشه‌ها را در پنجره‌ی سرور مشاهده میکنیم تا مطمئن شویم که سرور توانایی فهرست کردن فایل‌ها را دارد.

حالا نوبت به مرحله آخر میرسید یعنی یک فایل PDF ایجاد میکنیم و لینک به مخزن GitHub که پروژه در آن قرار دارد اضافه میکنیم این مراحل به ما کمک می‌کند تا سرور خود را تست و مستندات لازم را تهیه کنیم. که این مرحله هم براتون توضیح میدم:

به وبسایت GitHub میرویم

وارد حساب کاربری میشویم.

روی گزینه‌ی **New** در گوشه‌ی سمت راست بالای صفحه کلیک میکنیم یا از قسمت **Repositories** گزینه **New** را انتخاب میکنیم

در صفحه‌ی ساخت مخزن:

- **Repository name:** یک نام برای مخزن انتخاب میکنیم
- **Description:** توضیحات دلخواه درباره پروژه وارد میکنیم (اختیاری)
- **Public** یا **Private:** انتخاب میکنیم که مخزن عمومی یا خصوصی باشد.
- گزینه‌ی **Add a README file** را انتخاب میکنیم (اختیاری).

روی دکمه Create repository کلیک میکنیم تا مخزن جدید ایجاد شود.

حالا باید پروژه را از سیستم به مخزن GitHub منتقل کنیم

مرحله ۱: تنظیم Git در سیستم

- پس از نصب، از طریق **Command Prompt** یا **Git Bash** دستورات زیر را برای تنظیم Git وارد میکنیم:

```
git config --global user.name "YourName"
```

```
git config --global user.email "YourEmail@example.com"
```

ایجاد یک مخزن محلی و اتصال آن به GitHub

۱. به پوشه‌ی پروژه‌ی خودم میریم

۲. در **Command Prompt** یا **Git Bash** ، دستورات زیر را وارد میکنیم:

```
git init
```

```
git add .
```

```
git commit -m "اولین ارسال پروژه"
```

git init مخزن محلی Git را در پوشه‌ی پروژه ایجاد می‌کند.

git add . تمام فایل‌ها را برای ارسال به GitHub آماده می‌کند.

git commit -m "اولین ارسال پروژه**"** تغییرات را با یک پیام توضیحی ثبت می‌کند.

به صفحه‌ی مخزن جدیدی که در GitHub ساختیم میریم. در آنجا دستورات لازم برای اتصال مخزن محلی به مخزن GitHub را مشاهده می‌کنیم. معمولاً دستوراتی به این شکل هستند:

```
git remote add origin https://github.com/YourUsername/FTPServer.git
```

```
git push -u origin master
```


git remote add origin آدرس مخزن GitHub را به مخزن محلی اضافه می کند.
git push -u origin master تمام فایل ها را به شاخه اصلی (main) در GitHub ارسال می کند.

پس از اجرای این دستورات، پروژه من در مخزن GitHub بارگذاری شده و می توانم آن را در GitHub مشاهده کنیم

برای افزودن لینک مخزن به فایل PDF

۱. لینک مخزن GitHub را از نوار آدرس مرورگر کپی میکنیم
۲. در فایل PDF ، در قسمت مناسب، این لینک را اضافه میکنیم

و اینگونه شد که داکيومنت ما تموم شد 😊

ممنون از شما

خسته نباشین