**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỆ ĐIỀU HÀNH**

**ĐỀ TÀI: CHƯƠNG TRÌNH NGẮT KẾT NỐI USB AN TOÀN**

*Giảng viên hướng dẫn:* Ts. Phạm Thanh Bình

*Nhóm sinh viên thực hiện:*

1. Phan Anh Đức
2. Phạm Phúc Quang
3. Nguyễn Hoàng Lâm
4. Thái Cao Thiên Đạt
5. Hoàng Thái Duy

**Hà Nội - 2022**

# MỞ ĐẦU

## Giới thiệu:

## Hệ điều hành là một hệ chương trình hoạt động giữa người dùng và phần cứng của máy tính. Từ đó, cung cấp một môi trường để người sử dụng có thể thi hành các chương trình làm cho máy tính dễ sử dụng hơn, thuận tiện và hiệu quả. Các thiết bị USB hiện nay vẫn còn đang phổ biến và nhiều ngưỡi vẫn giữ thói quen lữu trữ dữ liệu quan trọng vào các thiết bị này, nhưng khi copy giữ liệu nhiều ngưỡi vẫn rút thiết bị ra mà không để ý việc ngắt kết nối. Điều này có thể dẫn đến việc mất hoặc hư hỏng dữ liệu. Cho nên chúng em chọn đề tài “Chương trình ngắt kết nối USB an toàn”

## Công cụ sử dụng:

Ngôn ngữ lập trình: C#.

Phần mềm code: Visual studio 2022.

1. **API và các hàm được sử dụng trong bài**

[DllImport("kernel32")]

private static extern IntPtr CreateFile(

string filename,

uint desiredAccess,

uint shareMode,

IntPtr securityAttributes,

int creationDisposition,

int flagsAndAttributes,

IntPtr templateFile);

[DllImport("kernel32")]

private static extern int CloseHandle(IntPtr handle);

[DllImport("kernel32")]

private static extern int DeviceIoControl(

IntPtr deviceHandle,

uint ioControlCode,

IntPtr inBuffer,

int inBufferSize,

IntPtr outBuffer,

int outBufferSize,

ref int bytesReturned,

IntPtr overlapped);

* 1. **API** “**kernel32”**
* Kernel32 quản lý bộ nhớ, hoạt động nhập/xuất và xử lý đóng/ngắt cũng như những hàm ảnh hưởng trực tiếp đến hoạt động của Windows.   
  1. **CreateFile():** Hàm tạo hoặc mở một đối tượng và trả về 1 handle có thể dùng để truy cập vào đối tượng được tạo
* **string filename**: Đường dẫn đến đối tượng hoặc file
* **uint desiredAccess**: Chế độ truy cập (Đọc - Ghi)
* **uint shareMode**: Chế độ chia sẻ
* **IntPtr securityAttributes**: Thuộc tính bảo mật
* **int creationDisposition**: Tạo như thế nào
* **int flagsAndAttributes:** Thuộc tính tệp
* **IntPtr templateFile**: Xử lý tệp với các thuộc tính
  1. **CloseHandle():** Đóng một đối tượng handle đang mở
* Handle: Một Handle dang mở
  1. **DeviceIoControl():** Gửi mã điều khiển trực tiếp đến trình điều khiển thiết bị được chỉ định để thiết bị tương ứng thực hiện hoạt động được chỉ định.
* **IntPtr deviceHandle**: Handle thiết bị sẽ được thao tác
* **Uint ioControlCode:** Chỉ định mã điều khiển cho hoạt động, cụ thể trong bài sẽ dùng mã **IOCTL\_STORAGE\_EJECT\_MEDIA** có giá trị là **0x2D4808** có chức năng ngắt kết nối thiết bị lưu trữ đa phương tiện (USB, CD-ROM,..).
* **IntPtr inBuffer:** Con trỏ đến bộ đệm để cung cấp dữ liệu đầu vào
* **int inBufferSize:** Kích cỡ của bộ đệm dữ liệu đầu vào
* **int outBufferSize:** kích thước của bộ đệm đầu ra
* **ref int bytesReturned:** con trỏ tới biến để nhận số byte đầu ra
* IntPtr overlapped:

**SẢN PHẨM**

1. **Giao diện**

Graphical user interface

Description automatically generated with low confidence

* **Listbox:** Hiển thị list USB đã kết nối
* **Nút Refresh:** Làm mới list box khi có thay đổi
* **Nút Remove:** Để tháo usb

1. **Nguyên lý**

* Tạo 1 handle với đối tượng là ổ đĩa với hàm **CreateFile()**.
* Dùng hàm **DeviceIoControl()** để gửi **ioControlCode** với mã **IOCTL\_STORAGE\_EJECT\_MEDIA** đến handle vừa tạo để ngắt kết nối thiết bị.
* Đóng handle đang mở bằng hàm **CloseHandle()**.

1. **Cách làm**

* Khai báo một số giá trị cần thiết
  + **OPEN\_EXISTING**: Mở file hoặc thiết bị khi tồn tại, có giá trị mặc định là 3
  + **GENERIC\_READ**: Cấp quyền truy cập phù hợp để đọc đối tượng. Có giá trị mặc định là **0x80000000**
  + **GENERIC\_WRITE**: Cấp quyền truy cập phù hợp để cập nhật các thuộc tính trên đối tượng. Có giá trị mặc định **0x40000000**
  + **IOCTL\_STORAGE\_EJECT\_MEDIA**: Dùng để ngắt kết nối thiết bị lưu trữ đa phương tiện. Có giá trị mặc định là **0x2D4808**
* Import thư viện **kernel32.dll** để có thể sử dụng các hàm liên quan
* Khai báo các hàm cần thiết: **CreateFile()**, **DeviceIoControl()**, **CloseHandle()** như đã giải thích ở bên trên
* Tạo 1 function **EjectUSB** với tham số **driveLetterCharacter**
* Khai báo biến **pathfordrive** có giá trị là đường dẫn đến thiết bị với dạng DOS paths device: **“\\.\driveLetterCharacter:”**
* Khai báo 1 biến handle với hàm **CreateFile()** gồm các tham số và giá trị sau:
  + **Filename = pathfordrive** đã khai báo bên trên
  + **desiredAccess = GENERIC\_READ | GENERIC\_WRITE**: quyền phù hợp để đọc và ghi
  + **shareMode = 0**: Ngăn các quy trình khác mở tệp hoặc thiết bị nếu chúng yêu cầu quyền xóa, đọc hoặc ghi.
  + securityAttributes = IntPrt.Zero: Không có thuộc tính hoặc flags nào được gắn
  + **creationDisposition = OPEN\_EXISTING**: Thực hiện một hành động trên một tệp hoặc thiết bị tồn tại hoặc không tồn tại.
  + templateFile = IntPrt.Zero: Không có thuộc tính hoặc flags nào được gắn
* Kiểm tra giá trị trả về của handle:
  + Nếu giá trị trả về của handle < 0 thì có nghĩa là không thể tạo file đã được chỉ định cho nên không thể tháo được USB
  + Nếu giá trị trả về của handle = 0 thì tiếp túc tạo hàm **DeviceIoControl()** để ngắt kết nối USB
* Khai báo hàm DeviceIoControl() với các tham số và giá trị sau:
  + **deviceHandle = hande:** 1 handle của thiết bị mà thao tác sẽ được thực hiện sẽ được thực hiện. Giá trị là biến handle đã khai báo bên trên
  + **ioControlCode = IOCTL\_STORAGE\_EJECT\_MEDIA**: Mã code để thực hiện ngắt kết nối thiết bị đa phương tiện
  + inBuffer = IntPrt.Zero: Không có thuộc tính hoặc flags nào được gắn
  + inBufferSize = 0: Không có thuộc tính hoặc flags nào được gắn
  + outBuffer = IntPrt.Zero: Không có thuộc tính hoặc flags nào được gắn
  + outBufferSize = 0: Không có thuộc tính hoặc flags nào được gắn
  + bytesReturned: Không có thuộc tính hoặc flags nào được gắn
  + overlapped = IntPrt: Không có thuộc tính hoặc flags nào được gắn
* Dùng hàm **CloseHandle()** để đóng handle đang mở và return thông báo đã tháo usb thành công
* Bắt sự kiện vào nút remove và dùng phương thức **DriveInfo.GetDrives()** để truy xuất thông tin ổ đĩa của tất cả các ổ đĩa logic trên máy tính.
* Loại ra thiết bị đang hoạt động và kiểu thiết bị là **Removable**
* Gọi hàm EjectUSB trong class USBManager với tham số truyền vào là tên của thiết bị.

1. **Mã nguồn**
   1. **Class USBManager**
2. public static class USBManager
3. {
5. const int OPEN\_EXISTING = 3;
6. const uint GENERIC\_READ = 0x80000000;
7. const uint GENERIC\_WRITE = 0x40000000;
8. const uint IOCTL\_STORAGE\_EJECT\_MEDIA = 0x2D4808;
10. [DllImport("kernel32")]
11. private static extern IntPtr CreateFile(
12. string filename,
13. uint desiredAccess,
14. uint shareMode,
15. IntPtr securityAttributes,
16. int creationDisposition,
17. int flagsAndAttributes,
18. IntPtr templateFile);
20. [DllImport("kernel32")]
21. private static extern int CloseHandle(IntPtr handle);
23. [DllImport("kernel32")]
24. private static extern int DeviceIoControl(
25. IntPtr deviceHandle,
26. uint ioControlCode,
27. IntPtr inBuffer,
28. int inBufferSize,
29. IntPtr outBuffer,
30. int outBufferSize,
31. ref int bytesReturned,
32. IntPtr overlapped);
34. public static string EjectUSB(char driveLetterCharacter)
35. {
36. string pathfordrive = "\\\\.\\" + driveLetterCharacter + ":";
38. IntPtr handle = CreateFile(pathfordrive, GENERIC\_READ | GENERIC\_WRITE, 0, IntPtr.Zero, OPEN\_EXISTING, 0, IntPtr.Zero);
39. if ((long)handle == -1)
40. {
41. int error = Marshal.GetLastWin32Error();
42. var result = string.Format("Không thể ngắt kết nối USB({0}) - Error code {1}", driveLetterCharacter, error);
43. return result;
44. }
46. int dummy = 0;
47. DeviceIoControl(handle, IOCTL\_STORAGE\_EJECT\_MEDIA, IntPtr.Zero, 0, IntPtr.Zero, 0, ref dummy, IntPtr.Zero);
48. CloseHandle(handle);
49. return "Đã ngắt kết nốt usb thành công";
50. }
51. }
    1. **Refresh Button**
52. private void refreshButton\_Click(object sender, EventArgs e)
53. {
54. var drives = DriveInfo.GetDrives().Where(d => d.IsReady & d.DriveType == DriveType.Removable);
56. if (drives.FirstOrDefault() != null)
57. {
58. double totalSize = (drives.FirstOrDefault().TotalSize) / (1024.0 \* 1024.0 \* 1024.0);
59. double freeSpace = (drives.FirstOrDefault().AvailableFreeSpace) / (1024.0 \* 1024.0 \* 1024.0);
60. listBox1.Items.Add(string.Format("{0}{1} - {2} free of {3}", drives.FirstOrDefault().Name, drives.FirstOrDefault().VolumeLabel, freeSpace.ToString("N1"), totalSize.ToString("N1")));
61. }
62. }
    1. **Remove Button**
63. private void removeButton\_Click(object sender, EventArgs e)
64. {
65. var drives = DriveInfo.GetDrives().Where(d => d.IsReady & d.DriveType == DriveType.Removable);
67. if (drives.FirstOrDefault() != null)
68. {
69. string status = USBManager.EjectUSB(Convert.ToChar(drives.FirstOrDefault().Name.Replace(":\\", "")));
70. DialogResult check = MessageBox.Show(status);
72. if (DialogResult.OK == check && status == "Đã tháo usb thành công")
73. {
74. listBox1.Items.Clear();
75. }
76. }
77. }
78. **Ảnh thực tế**

Graphical user interface, application, chat or text message

Description automatically generated

Graphical user interface, application

Description automatically generated

1. **Github:** [hanngoc1406](https://github.com/hanngoc1406/USB-Manager-Operating-System-Course)

**TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| **1** | [Bài giảng hệ điều hành – ĐH Thủy Lợi](https://dhthuyloi.blogspot.com/2014/02/tai-lieu-he-ieu-hanh.html) |
| **2** | [.NET API browser](https://docs.microsoft.com/en-us/dotnet/api/?view=net-6.0) |
| **3** | [Windows IOCTL Reference](http://www.ioctls.net/) |
| **4** | [aspdotnet-pools](http://www.aspdotnet-pools.com/) |