

User-Lever I/O Overview

主講人：虞台文

1

Content

- Introduction
- Synchronous vs. Asynchronous I/O
- User-Level APIs
- Demonstrations
- Exercises

2

User-Lever I/O Overview

Introduction

3

Introduction

- The main purpose of this talk is:
 - to introduce the basic mechanism to invoke kernel-mode methods from user-level applications or user-mode drivers.
 - to write user-level program to test your driver.

4

User-Lever I/O Overview

Synchronous vs. Asynchronous I/O

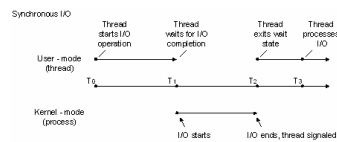
5

Synchronous vs. Asynchronous I/O

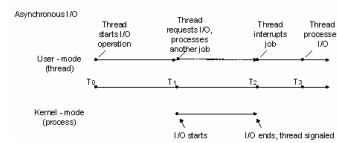
- Synchronous I/O
 - A thread starts an I/O operation and immediately enters a wait state until the I/O request has completed.
- Asynchronous I/O
 - A thread sends an I/O request to the kernel.
 - If accepted, it continues processing another job until the kernel signals completion.
 - It then interrupts its current job and processes the data from the I/O operation as necessary.

6

Synchronous vs. Asynchronous I/O



If many fast I/O operations need to be made, Synchronous I/O would be better (less overhead).



Good when I/O request is expected to take a large amount of time, such as a refresh or backup of a large database.

7

User-Lever I/O Overview

User-Level APIs

8

Basic-Level APIs

- ✗ CreateFile
- ✗ ReadFile/ReadFileEx
- ✗ WriteFile/WriteFileEx
- ✗ DeviceIoControl
- ✗ CloseHandle
- ✗ Event Objects

9

If zero, the object cannot be shared and cannot be opened again until the handle is closed.

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

13

CreateFile

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

10

If NULL, the object gets a default security descriptor.

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

14

Testing your driver, specify the device name in this field.

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
```

| Example string | Meaning |
|--------------------|---------------------------------|
| \\.\PHYSICALDRIVE0 | Opens the first physical drive. |
| \\.\PHYSICALDRIVE2 | Opens the third physical drive. |
| \\.\COM1 | Opens the first COM port. |

11

Action to take on files that exist, and which action to take when files do not exist, e.g., OPEN_EXISTING

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

15

Access mode (Read/Write) e.g., GENERIC_READ|GENERIC_WRITE

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

12

Has little meaning for most devices drivers. Most of flag values only applied to file-based system. The only interesting value for us is FILE_FLAG_OVERLAPPED, which indicates that the user wants to use asynchronous I/O.

```
HANDLE CreateFile(
    [in] LPCTSTR lpFileName,
    [in] DWORD dwDesiredAccess,
    [in] DWORD dwShareMode,
    [in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    [in] DWORD dwCreationDisposition,
    [in] DWORD dwFlagsAndAttributes,
    [in] HANDLE hTemplateFile
);
```

16

Handle to a template file, with the GENERIC_READ access right. The template file supplies file attributes and extended attributes for the file being created. Specify NULL for devices drivers.

```
[in] LPCTSTR lpFileName,
[in] DWORD dwDesiredAccess,
[in] DWORD dwFlagsAndAttributes,
[in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
[in] DWORD dwCreationDisposition,
[in] DWORD dwFlagsAndAttributes,
[in] HANDLE hTemplateFile
);
```

17

ReadFile

Address of OVERLAPPED structure for overlapped I/O.

```
BOOL ReadFile(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToRead,
[out] LPDWORD lpNumberOfBytesRead,
[in] LPOVERLAPPED lpOverlapped
);
```

```
BOOL ReadFileEx(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToRead,
[out] LPDWORD lpNumberOfBytesRead,
[in] LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

Address of completion routine.

21

CreateFile



```
HANDLE CreateFile(
[in] LPCTSTR lpFileName,
[in] DWORD dwDesiredAccess,
[in] DWORD dwShareMode,
[in] LPSECURITY_ATTRIBUTES lpSecurityAttributes,
[in] DWORD dwCreationDisposition,
[in] DWORD dwFlagsAndAttributes,
[in] HANDLE hTemplateFile
);
```

18

OVERLAPPED

```
typedef struct _OVERLAPPED{
    ULONG_PTR Internal; // system use
    ULONG_PTR InternalHigh; // system use
    DWORD Offset; // file position (low)
    DWORD OffsetHigh; // file position (high)
    HANDLE hEvent; // for completionsignaling
} OVERLAPPED;
```

Call CreateEvent to get the event handle.

22

CloseHandle

```
BOOL CloseHandle( HANDLE hObject );
```



19

FileIOCompletionRoutine

```
VOID CALLBACK FileIOCompletionRoutine(
    DWORD dwErrorCode,
    DWORD dwNumberOfBytesTransferred,
    LPOVERLAPPED lpOverlapped
);
```

23

ReadFile/Ex

```
BOOL ReadFile(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToRead,
[out] LPDWORD lpNumberOfBytesRead,
[in] LPOVERLAPPED lpOverlapped
);
```

```
BOOL ReadFileEx(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToRead,
[out] LPDWORD lpNumberOfBytesRead,
[in] LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

20

FileIOCompletionRoutine

```
VOID CALLBACK FileIOCompletionRoutine(
    DWORD dwErrorCode,
    DWORD dwNumberOfBytesTransferred,
    LPOVERLAPPED lpOverlapped
);
```

The system does not use the hEvent member of the structure; the completion routine can deallocate the memory used by the overlapped structure.

24

WriteFile/Ex

```

BOOL WriteFile(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToWrite,
[out] LPDWORD lpNumberOfBytesWritten,
[in] LPOVERLAPPED lpOverlapped
);

BOOL WriteFileEx(
[in] HANDLE hFile,
[out] LPVOID lpBuffer,
[in] DWORD nNumberOfBytesToWrite,
[out] LPDWORD lpNumberOfBytesWritten,
[in] LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
    
```

25

DeviceIoControl

- Provides a device input and output control (IOCTL) interface through which an application can communicate directly with a device driver.
- A general-purpose interface that can send control codes to a variety of devices.
- Each control code represents an operation for the driver to perform.
 - E.g., a control code can ask a device driver to return information about the corresponding device, or direct the driver to carry out an action on the device, such as formatting a disk.

26

DeviceIoControl

```

BOOL DeviceIoControl(
[in] HANDLE hDevice,
[in] DWORD dwIoControlCode,
[in] LPVOID lpInBuffer,
[in] DWORD nInBufferSize,
[out] LPVOID lpOutBuffer,
[in] DWORD nOutBufferSize,
[out] LPDWORD lpBytesReturned,
[in] LPOVERLAPPED lpOverlapped
);
    
```

27

De

[in] Call the *CreateFile* function to obtain a device handle.

```

BOOL DeviceIoControl(
[in] HANDLE hDevice,
[in] DWORD dwIoControlCode,
[in] LPVOID lpInBuffer,
[in] DWORD nInBufferSize,
[out] LPVOID lpOutBuffer,
[in] DWORD nOutBufferSize,
[out] LPDWORD lpBytesReturned,
[in] LPOVERLAPPED lpOverlapped
);
    
```

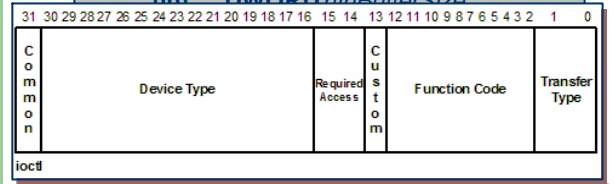
28

DeviceIoControl

[in] Control code for the operation.

```

BOOL DeviceIoControl(
[in] HANDLE hDevice,
[in] DWORD dwIoControlCode,
[in] LPVOID lpInBuffer,
[in] DWORD nInBufferSize,
[out] LPVOID lpOutBuffer,
[in] DWORD nOutBufferSize,
[out] LPDWORD lpBytesReturned,
[in] LPOVERLAPPED lpOverlapped
);
    
```



29

DeviceIoControl

Input and output buffers

```

BOOL DeviceIoControl(
[in] HANDLE hDevice,
[in] DWORD dwIoControlCode,
[in] LPVOID lpInBuffer,
[in] DWORD nInBufferSize,
[out] LPVOID lpOutBuffer,
[in] DWORD nOutBufferSize,
[out] LPDWORD lpBytesReturned,
[in] LPOVERLAPPED lpOverlapped
);
    
```

30

DeviceIoControl

```

BOOL DeviceIoControl(
[in] HANDLE hDevice,
[in] DWORD dwIoControlCode,
[in] LPVOID lpInBuffer,
[in] DWORD nInBufferSize,
[out] LPVOID lpOutBuffer,
[in] DWORD nOutBufferSize,
[out] LPDWORD lpBytesReturned,
[in] LPOVERLAPPED lpOverlapped
);
    
```

[in] Pointer to an OVERLAPPED structure.

31

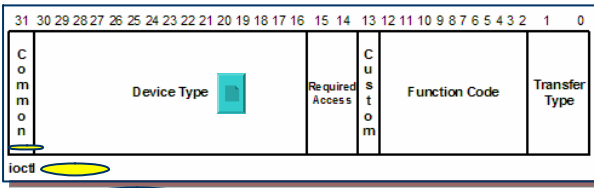
Example

- An application can use the DeviceIoControl function to perform direct input and output operations on, or retrieve information about, a floppy disk drive, hard disk drive, tape drive, or CD-ROM drive.
- The example demonstrates how to retrieve information about the first physical drive in the system.
- It Uses IOCTL_DISK_GET_DRIVE_GEOMETRY control code to fill a DISK_GEOMETRY structure with information about the drive.

Source Code Demo

32

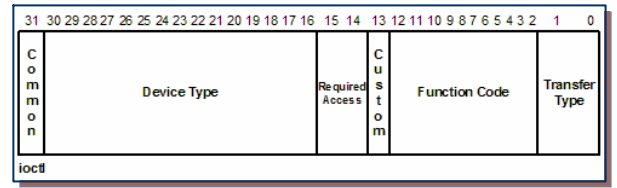
More On I/O Control Code



Set to 1 for customer defined device-type, i.e., 32768..65535

33

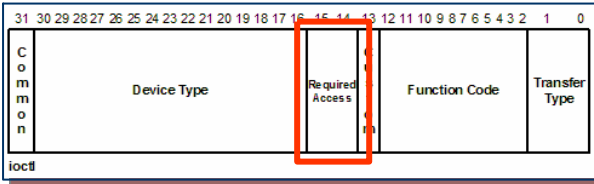
CTL_CODE Marco



```
CTL_CODE( DeviceType, Function, Method, Access );
```

37

More On I/O Control Code



| Flag | Description |
|-------------------|------------------------------|
| FILE_ANY_ACCESS | Request all access. |
| FILE_READ_ACCESS | Request read access. |
| FILE_WRITE_ACCESS | Request write access. |

34

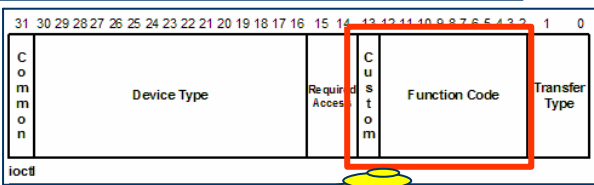
System Support Control Codes

- ⚡ Communications Control Codes
- ⚡ Device Management Control Codes
- ⚡ Directory Management Control Codes
- ⚡ Disk Management Control Codes
- ⚡ File Management Control Codes
- ⚡ File Systems Control Codes
- ⚡ Power Management Control Codes
- ⚡ Volume Management Control Codes

See Platform SDK for the detail

38

More On I/O Control Code



Defines an action within the device category. Function codes 0-2047 are reserved for Microsoft Corporation, and 2048-4095 are reserved for OEMs/IHV's.

35

Example

I/O Control Code Decoder

```
#define IOCTL_DISK_GET_DRIVE_GEOMETRY \
    CTL_CODE( IOCTL_DISK_BASE, \
              0x0000, \
              METHOD_BUFFERED, \
              FILE_ANY_ACCESS)
```

IOCTL_DISK_GET_DRIVE_GEOMETRY=0x00070000

IOCTL_DISK_GET_DRIVE_GEOMETRY Detail

39

More On I/O Control Code

| Flag | Description | Transfer Type |
|-------------------|--|---------------|
| METHOD_BUFFERED | Suitable for transfers small amounts of data for the request. | Transfer Type |
| METHOD_IN_DIRECT | Read a large amount of data for the request using DMA or PIO and must transfer the data quickly. | |
| METHOD_OUT_DIRECT | Write a large amount of data to the device for the request using DMA or PIO and must transfer the data quickly. | |
| METHOD_NEITHER | Workable only if running in the context of the thread that originates the I/O control request . | |

36

User-Level I/O Overview

Demonstrations

40

Read Disk Geometry

```

hDevice = CreateFile(
    "\\.\PhysicalDrive0", // drive to open
    0, // no access to the drive
    FILE_SHARE_READ |
    FILE_SHARE_WRITE, // share mode
    NULL, // default security attributes
    OPEN_EXISTING, // disposition
    0, // file attributes
    NULL); // do not copy file attributes

```



IOExplorer.exe

41

Read Disk Geometry

```

bResult = DeviceIoControl(
    hDevice, // device to be queried
    IOCTL_DISK_GET_DRIVE_GEOMETRY, // I/O operation
    NULL, 0, // no input buffer
    pdg, sizeof(*pdg), // output buffer
    &junk, // # bytes returned
    (LPOVERLAPPED) NULL); // synchronous I/O

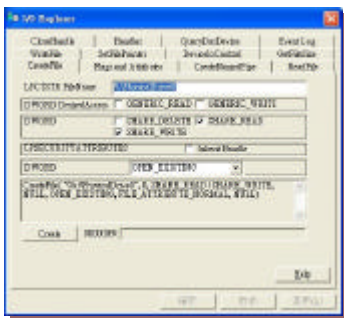
```



IOExplorer.exe

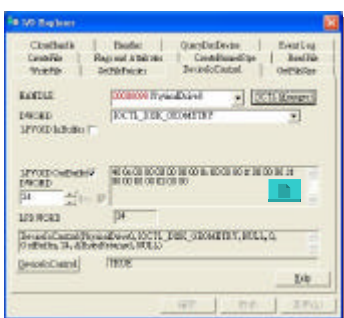
42

Read Disk Geometry



43

Read Disk Geometry



44

User-Lever I/O Overview

Exercise

45

Programming Task

- ≈ Write a Win32 Application to perform I/O controls (DeviceIoControl) on
 - Device Management Control Codes
 - Disk Management Control Codes
- ≈ Be sure to have friendly user interface.

46