Metody připojování periferií

BI-MPP

Přednáška 10

Ing. Miroslav Skrbek, Ph.D.

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

ZS2021/22

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Agenda

- Ovladače ve Windows (pokračování předchozí přednášky)

Literatura

- Gook, M.: Hardwarová rozhraní – Průvodce programátora. Computer Press, Brno 2006. ISBN 80-251-1019-2
- WDK Documentation, Windows Drivers Kits, Microsoft Corporation, 2009.

Skutečná práce s objekty

WdfObjectCreate - vytvoření objektu
alokace základní datové struktury instance třídy objekt
WdfObjectDelete – zrušení (dealokace) objektu

Každý objekt má počítadlo odkazů (referencí), které se při vytvoření objektu nastaví na jedničku. Vytváříme-li odkazy na objekt v jiných objektech, je nutno programově zvyšovat počítadlo odkazů funkcí **`WdfObjectReference`**, případně snižovat funkcí **`WdfObjectDereference`**, pokud není odkaz dále používán.

Události

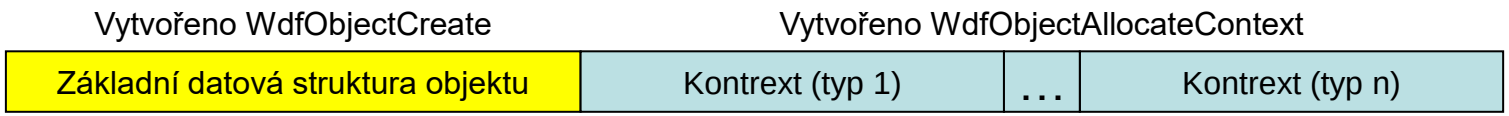
`EvtCleanupCallback` – musí uvolnit veškeré odkazy na daný objekt pomocí funkce **`WdfObjectDereference`** tak, aby počet odkazů dosáhl nuly. Je volán při požadavku na smazání objektu funkcí **`WdfObjectDelete.`**

`EvtDestroyCallback` – volán pokud počítadlo odkazů dosáhlo nuly, tj. těsně před dealokací objektu. Funkce by měla uvolnit veškeré alokované paměťové bloky vázané na daný objekt.

Vytvoření instančních proměnných objektu

`WdfObjectAllocateContext` – alokuje paměťový blok o velikosti struktury, která obsahuje instanční proměnné. Velikost se předává v parametru funkce. Je-li takových struktur více, pak je možné tuto funkci volat vícekrát pro jeden objekt.

`WdfObjectGetTypedContext` – vrací ukazatel na požadovaný kontext.



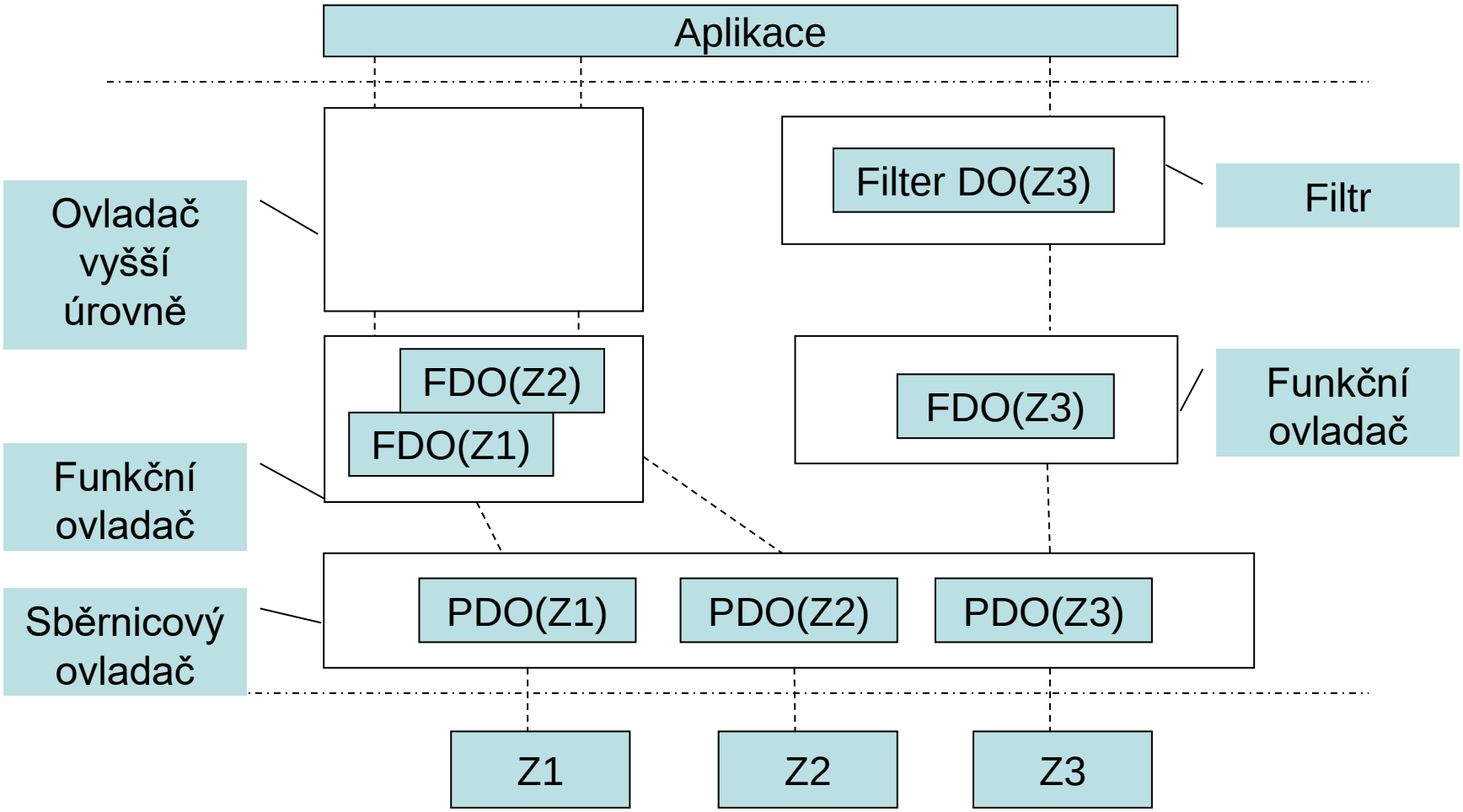
Typy objektů pro zařízení

- PDO (Physical Driver Object) – reprezentuje zařízení na sběrnici u sběrnicových ovladačů
- FDO (Funcional Driver Object) – reprezentuje zařízení u funkčních ovladačů
- Filter DO (Filter Driver Object) – reprezentuje zařízení u filtrů

Objekt ovladače (Driver Object)

- Reprezentuje ovladač (strojový kód ovladače) nahraný do paměti nahraný do paměti).
- Tento objekt vytváří IO manager při nahrání ovladače a předává se jako parametr do vstupního bodu ovladače (funkce DriverEntry).
- Aplikací funkce **`WdfObjectAllocateContext`** se alokuje paměťový prostor pro proměnné, které jsou společné pro všechna zařízení obsluhovaná tímto ovladačem

Objekty v hierarchii ovladačů



Vstupní bod ovladače (EntryPoint)

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT  DriverObject,
                     IN PUNICODE_STRING  RegistryPath)
{
    WDF_DRIVER_CONFIG  config;
    NTSTATUS  status;
    WDF_OBJECT_ATTRIBUTES  attributes;

    WDF_OBJECT_ATTRIBUTES_INIT(&attributes);
    WDF_DRIVER_CONFIG_INIT(&config, EvtDriverDeviceAdd);


    status = WdfDriverCreate(DriverObject, RegistryPath,
                 &attributes, &config, WDF_NO_HANDLE);
        return status;
}
```

Callback volaný při detekci nového zařízení daného typu

Inicializace objektu ovladače. Funkce poskytnutá frameworkem, nutno volat v každém ovladači

EvtDriverDeviceAdd callback

- Volán kdykoliv je přidáno nové zařízení daného typu
- Musí vytvořit FDO pro dané zařízení voláním WdfDeviceCreate
- Vytvořit interfacy ovladače
- Zviditelnit ovladač aplikacím (pokud je požadováno)
- Vytvoření V/V front

Zviditelnění ovladače pro aplikace (přes symbolický link)

```
PCWSTR deviceFilenameStr = L"\\DosDevices\\mpp";

RtlInitUnicodeString(&deviceFilename,
                     deviceFilenameStr);

status = WdfDeviceCreateSymbolicLink(
                     device,
                     &deviceFilename);

if (!NT_SUCCESS(status)) {
        return status;
}
```

V aplikaci

```
h = CreateFile("\\\\.\\mpp", …);
```

Zviditelnění ovladače pro aplikace (přes device interface)

- V ovladači zavolat funkci `WdfDeviceCreateDeviceInterface` a vytvořit (přiřadit) typ rozhraní (myšleno rozhraní aplikace-ovladač) k právě připojenému zařízení
- V aplikaci funkcemi SetupDi…nalézt požadované zařízení dle typu rozhraní a získat souborovou cestu k tomuto zařízení (DevicePath)
- Funkcí CreateFile otevřít ovladač, získat maipulační číslo (handle) a souborovými funkcemi zařízení ovládat

Vytvoření rozhraní (Device Interface)

```
status = WdfDeviceCreateDeviceInterface(device,
              &GUID_DEVINTERFACE_FIT, NULL);
if (!NT_SUCCESS (status)) {
       return status;
}
```

V ovladači se při připojení nového zařízení (konkrétně v callbacku `EvtDriverDeviceAdd`) funkcí `WdfDeviceCreateDeviceInterface` vytvoří rozhraní požadovaného typu (v našem případě reprezentovaného `GUID_DEVINTERFACE_FIT`).

Fakticky se jedná o přiřazení GUID typu rozhraní a jeho záznam do systémových registrů (Registry). Můžeme použít standardní GUID např. `GUID_DEVINTERFACE_DISK`, pak ale musí ovladač implementovat funkce CreateFile, ReadFile, WriteFile, DeviceIoControl, CloseHandle tak, jak standardně požadováno pro zařízení typu DISK. Můžeme si také vytvořit vlastní GUID (jako naše `GUID_DEVINTERFACE_FIT`) a funkcionalitu definovat sami

Detekce zařízení pro požadovaný typ rozhraní

```
// Vytvoření seznamu všech připojených zařízení
// počet zařízení lze specificky omezit vhodnými parametry funkce)
HDEVINFO hdi = SetupDiGetClassDevs(NULL, NULL, NULL,
    DIGCF_ALLCLASSES | DIGCF_PRESENT | DIGCF_DEVICEINTERFACE);

// Vyhledání všech zařízení s rozhraním daného typu

for(i = 0; TRUE; i++) {

          int r = SetupDiEnumDeviceInterfaces(hdi, NULL,
                        &GUID_DEVINTERFACE_FIT, i, &interf);
          if (!r) {
                    int err = GetLastError();
                    if (err == ERROR_NO_MORE_ITEMS) {
                              break;
                    }
                    // report an error !
                return;

            }
          // funkcí SetupDiGetDeviceInterfaceDetail zjisti
          // detailnější informace včetně DevicePath
}
// Použij DevicePath ve funkci CreateFile k získání
// manipulačního čísla (handle)
```

Zpracování V/V požadavků (vytvoření fronty)

```
WDF_IO_QUEUE_CONFIG  ioQueueConfig;
WDFQUEUE  hQueue;

NTSTATUS EvtDriverDeviceAdd (
    IN WDFDRIVER  Driver,
    IN PWDFDEVICE_INIT  DeviceInit
          ) {
    …
          WDF_IO_QUEUE_CONFIG_INIT_DEFAULT_QUEUE(
                    &ioQueueConfig, WdfIoQueueDispatchSequential);

          ioQueueConfig.EvtIoDefault = EvtIoDefault;
          ioQueueConfig.EvtIoRead = EvtIoRead;
          ioQueueConfig.EvtIoWrite = EvtIoWrite;

          status = WdfIoQueueCreate(device,
                    &ioQueueConfig, WDF_NO_OBJECT_ATTRIBUTES,
                              &hQueue);
          if (!NT_SUCCESS (status)) {
                    return status;
          }
    …
    return status;
}
```

Požadavky budou zpracovávány sekvenčně

Callback pro operaci ReadFile

Callback pro operaci WriteFile

Callback pro zpracování požadavku EvtIoRead a EvtIoWrite

```
void EvtIoRead(IN WDFQUEUE  Queue, IN WDFREQUEST  Request,
      IN size_t  Length) {
    //
    // … zpracování požadavku bude zde …
    //
    // signalizace, že požadavek byl zpracován
    WdfRequestComplete(Request, STATUS_SUCCESS);
}

void EvtIoWrite(IN WDFQUEUE  Queue, IN WDFREQUEST  Request,
     IN size_t  Length)
{
    //
    // … zpracování požadavku bude zde …
    //
    // signalizace, že požadavek byl zpracován
    WdfRequestComplete(Request, STATUS_SUCCESS);
}
```