

# KV Betriebssysteme

## Windowsprogrammierung

KV Betriebssysteme  
Windowsprogrammierung  
Arten von Ressourcen

**Michael Sonntag**  
Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: sonntag@fim.uni-linz.ac.at

Michael Sonntag KV Betriebssysteme 1

Bitmaps

Bitmaps sollten als externe Datei gespeichert werden und nur in die RC-Datei incluiert werden:  
**nameID BITMAP filename**

**Verwendung im Programm:**

- **HANDLE LoadImage(HINSTANCE hInst, LPCTSTR lpszName, IMAGE\_BITMAP, int cx, int cy, UINT fuLoad)**
  - hInst: Programminstanz
  - lpszName: Name der Bitmap, Bitmap-Identifizier oder Dateiname
  - cx, cy: Breite/Höhe der Bitmap (auf 0 setzen für phys. Größe)
  - fuLoad: Lade-Parameter (LR\_DEFAULTSIZE, LR\_LOADFROMFILE, ...)

Michael Sonntag KV Betriebssysteme 2

Bitmaps - Beispiel (1)

```
case WM_PAINT: {
    hDC=BeginPaint(hWndMain, &ps);
    GetClientRect(hWndMain, &r);
    memDC=CreateCompatibleDC(hDC);
    oldBmp=SelectObject(memDC, bkgnd);
    BitBlt(hDC, 0, 0, r.right, r.bottom, memDC, 0, 0, SRCCOPY);
    SelectObject(memDC, oldBmp);
    DeleteDC(memDC);
    EndPaint(hWndMain, &ps);
}

int InitInstance(HINSTANCE hInst, int nCmdShow) {
    BITMAP bmp;
    bkgnd=LoadImage(hInstCurr, MAKEINTRESOURCE(IDB_BGR),
        IMAGE_BITMAP, 0, 0, LR_DEFAULTCOLOR);
    if(!bkgnd) return FALSE;
}
```

Michael Sonntag KV Betriebssysteme 3

Bitmaps - Beispiel (2)

```
if(!GetObject(bkgnd, sizeof(bmp), &bmp)) {
    DeleteObject(bkgnd); return FALSE; }
hWndMain = CreateWindow(szWndClassName,
    "Fenster mit Bitmap-Hintergrund",
    /* Nicht größenveränderbar! */
    WS_DLGFRAME | WS_MINIMIZEBOX | WS_SYSMENU,
    CW_USEDEFAULT, CW_USEDEFAULT,
    /* Horizontal: Zwei mal Rahmenbreite addieren */
    bmp.bmWidth+2*GetSystemMetrics(SM_CXFIXEDFRAME),
    /* Vertikal: Zwei mal Rahmenbreite und Titelzeilenhöhe addieren */
    bmp.bmHeight+2*GetSystemMetrics(SM_CYFIXEDFRAME)+
    GetSystemMetrics(SM_CYCAPTION),
    0, 0, hInstCurr, NULL);
```

Michael Sonntag KV Betriebssysteme 4

KV Betriebssysteme  
Windowsprogrammierung  
Timer

**Michael Sonntag**  
Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: sonntag@fim.uni-linz.ac.at

Michael Sonntag KV Betriebssysteme 5

Timer

Mit Timern können periodisch Benachrichtigungen ausgelöst werden:

- **WM\_TIMER**
- **Callback-Prozedur**

Die Funktion wird regelmäßig ausgelöst, bis der Timer gelöscht wird.

Michael Sonntag KV Betriebssysteme 6

# KV Betriebssysteme

## Windowsprogrammierung

### Timer - API-Funktionen

- ```
UINT SetTimer( HWND hWnd, UINT nIdEvent,
              UINT uElapse,
              TIMERPROC lpFunc)
```
- **uElapse:** Timeout in Millisekunden
  - **lpTimerFunc** angegeben: Prozedur wird aufgerufen
  - **lpTimerFunc==NULL:** an das Fenster hWnd wird die Nachricht WM\_TIMER mit dem Parameter nIdEvent (als wParam) geschickt
  - **Rückgabewert:** ID des Timers (für KillTimer!)

Michael Sonntag

KV Betriebssysteme

7

### Timer - API-Funktionen

- ```
BOOL KillTimer(HWND hWnd, UINT uIdEvent)
```
- **hWnd:** Muß gleich dem Aufruf von SetTimer sein.
  - **uIdEvent:** Wenn hWnd!=NULL, muß dies der Parameter nIdEvent von SetTimer sein, ansonsten der Rückgabewert dieser Prozedur.
  - **Der Rückgabewert** ist ungleich 0 bei Erfolg.

Michael Sonntag

KV Betriebssysteme

8

### Timer - Beispiel

```
Jede Sekunde ein Pieps-Ton:
#define ID_TIMER 123
....
In WndProc:
    case WM_TIMER: MessageBeep(MB_OK); break;
in WinMain:
    SetTimer(hWndMain, ID_TIMER, 1000, NULL);
    while(GetMessage(&msg, NULL, 0, 0)>0) {}
    KillTimer(hWndMain, ID_TIMER);
```

Michael Sonntag

KV Betriebssysteme

9

### KV Betriebssysteme Windowsprogrammierung Dateien

Michael Sonntag

Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: sonntag@fim.uni-linz.ac.at

Michael Sonntag

KV Betriebssysteme

10

### Dateioperationen

- Datei:**
- Erzeugen
  - Öffnen
  - Lesen
  - Schreiben
  - Schließen
- Temporäre Dateien**  
**Kopieren/Verschieben/Löschen von Dateien**  
**Locking/Unlocking**  
**Verzeichnisse**

Michael Sonntag

KV Betriebssysteme

11

### Dateien öffnen/erzeugen

```
HANDLE CreateFile(LPCTSTR lpFileName,
                  DWORD dwDesiredAccess, DWORD shareMode,
                  LPSECURITY_ATTRIBUTES lpSecAttrib,
                  DWORD dwCreationDisposition,
                  DWORD dwFlagsAndAttributes,
                  HANDLE hTemplateFile);
```

- Wird verwendet für:** Erzeugen und Öffnen  
**Für Objekte vom Typ:** Datei, Pipe, Mailslot,  
Netzwerkressourcen, Laufwerken (nur NT),  
Konsolen, Verzeichnissen (nur öffnen)

Michael Sonntag

KV Betriebssysteme

12

# KV Betriebssysteme

## Windowsprogrammierung

### Parameter von CreateFile (1)

- **lpFileName:** Null-terminierter String des Dateinamens
- **dwDesiredAccess:** Wie zugegriffen werden soll
  - 0 (nur Laufwerks-Attribute lesen), GENERIC\_READ, GENERIC\_WRITE
- **shareMode: 0 (Kein anderer Zugriff)**
  - FILE\_SHARE\_READ: Lesezugriff erlaubt
  - FILE\_SHARE\_WRITE: Schreibzugriff erlaubt
  - FILE\_SHARE\_DELETE: Löschezugriff erlaubt
- **lpSecAttrib:** Ob Child-Prozesse das Handle verwenden können (NULL: nur dieser Prozess)

Michael Sonntag

KV Betriebssysteme

13

### Parameter von CreateFile (2)

- **dwCreationDisposition:** Modus für existierende/nicht existierende Dateien
  - CREATE\_NEW, CREATE\_ALWAYS, OPEN\_EXISTING, OPEN\_ALWAYS, TRUNCATE\_EXISTING
- **dwFlagsAndAttributes:** Archiv, Versteckt, System, ...
  - Zusätzlich: Buffering, Delete on close, Asynchroner Zugriff, ...
- **hTemplateFile:** Handle auf eine Template-Datei, von der Attribute (normale und erweiterte) übernommen werden
- **Rückgabewert:** Handle auf die Datei (kann auch 0 sein!)
  - Fehler: INVALID\_HANDLE\_VALUE

Michael Sonntag

KV Betriebssysteme

14

### Dateien lesen (1)

**BOOL** ReadFile(**HANDLE** hFile, **LPVOID** lpBuffer, **DWORD** nBytesToRead, **LPDWORD** lpBytesRead, **LPOVERLAPPED** lpOverlapped)

- hFile:** Handle auf die Datei  
**lpBuffer:** Zeiger auf den Buffer, in den gelesen wird  
**nBytesToRead:** Wieviele Bytes gelesen werden sollen  
**lpBytesRead:** Anzahl der gelesenen Bytes  
**lpOverlapped:** Für asynchrones lesen  
**Rückgabewert:** Ungleich 0 bei Erfolg

Michael Sonntag

KV Betriebssysteme

15

### Datei lesen (2)

- Lesen beginnt am Dateizeiger, der um die tatsächlich gelesenen Bytes aktualisiert wird.
- Asynchrones Lesen: KEINE Aktualisierung des Dateizeigers!
- Der Buffer darf während des Lesens nicht verändert werden (insbesondere bei asynchronem Zugriff!)
- Dateiende:**
- Synchroner Zugriff: Rückgabewert TRUE, lpBytesRead ist 0

Michael Sonntag

KV Betriebssysteme

16

### Dateien schreiben (1)

**BOOL** WriteFile(**HANDLE** hFile, **LPCVOID** lpBuffer, **DWORD** nBytesToWrite, **LPDWORD** lpBytesWritten, **LPOVERLAPPED** lpOverlapped)

- hFile:** Handle auf die Datei  
**lpBuffer:** Zeiger auf Buffer, aus dem geschrieben wird  
**nBytesToWrite:** Anzahl der zu schreibenden Bytes  
**lpBytesWritten:** Zeiger auf tatsächlich geschriebene Bytes  
**lpOverlapped:** Für asynchrones schreiben  
**Rückgabewert:** Ungleich 0 bei Erfolg

Michael Sonntag

KV Betriebssysteme

17

### Dateien schreiben (2)

- Schreiben beginnt am Dateizeiger, der um die tatsächlich geschriebenen Bytes aktualisiert wird.
- Asynchrones Schreiben: KEINE Aktualisierung des Dateizeigers!
- Der Buffer darf während des Schreibens nicht verändert werden (insbesondere bei asynchronem Zugriff!)
- Null Bytes zu schreiben wird einfach ignoriert (NOP); dies bedeutet KEIN Truncate oder EOF
- Zum Beenden der Datei:**
- BOOL** SetEndOfFile(**HANDLE** hFile)

Michael Sonntag

KV Betriebssysteme

18

# KV Betriebssysteme

## Windowsprogrammierung

### Dateizeiger setzen

**DWORD SetFilePointer(HANDLE hFile, LONG lDist, PLONG lpDistHigh, DWORD dwMoveMethod)**  
**hFile:** Datei-Handle  
**lDist:** Abstand: +  $\mathcal{D}$  nach hinten, -  $\mathcal{D}$  nach vorn  
**lpDistHigh:** NULL oder höherwertige 32 Bit für Abstand  
**dwMoveMethod:** Anfangspunkt für Abstand  
• FILE\_BEGIN, FILE\_CURRENT oder FILE\_END  
**Rückgabewert:** Neuer Dateizeiger  
• Bei Fehlern: Kompliziert (0xfffffff bei lpDistHigh==NULL)!

Michael Sonntag

KV Betriebssysteme

19

### Dateien schließen

**BOOL CloseHandle(HANDLE hObject)**  
• **hObject:** Handle auf ein offenes Objekt  
• **Rückgabewert:** Ungleich 0 bei Erfolg  
  
• **Handles dürfen nur einmal geschlossen werden!**

Michael Sonntag

KV Betriebssysteme

20

### Temporäre Dateien

**UINT GetTempFileName(LPCTSTR lpPathName, LPCTSTR lpPrefix, UINT uUnique, LPCTSTR lpTempFileName)**  
**lpPathName:** Pfad für die temporäre Datei (Meist: "." oder GetTempPath(...))  
**lpPrefix:** Ersten drei Zeichen des Namens  
**uUnique:** 0...Eindeutiger String wird erzeugt  
!=0...Diese Nummer wird verwendet  
**lpTempFileName:** Buffer für den Namen der Datei  
• Buffer-Länge: MAX\_PATH

Michael Sonntag

KV Betriebssysteme

21

### Temporäre Dateien

Wird keine Nummer angegeben (uUnique=0), so wird die Datei erzeugt (und wieder geschlossen!).  
**Rückgabewert ist uUnique bzw. die verwendete Zahl**  
**Die Extension ist immer ".TMP"**  
**Temporäre Dateien werden NICHT automatisch gelöscht.**  
**Dateiname:**  
• path\preuuuu.TMP  
• path Pfad aus lpPathName  
• pre Die ersten drei Zeichen von lpPrefix  
• uuuu Hexadezimalwert von uUnique (bzw. aus Systemzeit erstellt)

Michael Sonntag

KV Betriebssysteme

22

### Kopieren

**BOOL CopyFile(LPCTSTR source, LPCTSTR dest, BOOL bFailIfExists)**  
**source, dest:** Nullterminierte Dateinamen  
**bFailIfExists:** Fehler wenn das Ziel schon existiert  
**Rückgabewert:** Ungleich Null bei Erfolg  
**Die Datei muß geschlossen oder für Nur-Lese-Zugriff geöffnet sein!**  
**Sicherheitsattribute werden nicht kopiert, normale Datei-Attribute schon.**  
**Siehe auch: CopyFileEx(...)!**

Michael Sonntag

KV Betriebssysteme

23

### Verschieben

**BOOL MoveFile(LPCTSTR source, LPCTSTR dest)**  
**source, dest:** Nullterminierte Dateinamen  
**Rückgabewert:** Ungleich Null bei Erfolg  
  
**Die Datei muß geschlossen sein!**  
**Siehe auch: MoveFileEx(...)!**  
  
**Auch für Verzeichnisse, aber NUR auf dem selben Laufwerk.**

Michael Sonntag

KV Betriebssysteme

24

# KV Betriebssysteme

## Windowsprogrammierung

### Löschen

**BOOL DeleteFile(LPCTSTR lpFilename)**

**lpFilename:** Nullterminierter Dateiname

**Rückgabewert:** Ungleich Null bei Erfolg

**Die Datei muß geschlossen sein!**

### Locking

Eine Datei kann von mehreren Programmen gleichzeitig geschrieben werden. Um Probleme zu verhindern kann man (=USER!) Locking verwenden.

Gesperrte Bereiche können von anderen Prozessen weder gelesen noch geschrieben werden.

Man verwendet dazu:

**LockFile, LockFileEx**

**UnlockFile, UnlockFileEx**

Gesperrte Bereiche müssen 1:1 freigegeben werden.

### Verzeichnisse

**CreateDirectory:** Verzeichnis erzeugen

**RemoveDirectory:** Verzeichnis löschen (muß leer sein!)

**GetCurrentDirectory:** Verzeichnis abfragen

**SetCurrentDirectory:** Verzeichnis wechseln

**Diese Funktionen arbeiten alle auf Dateinamen, nicht auf Handles!**

### Verzeichnis-Inhalt

**HANDLE FindFirstFile(LPCTSTR lpFileName, LPWIN32\_FIND\_DATA lpFindData)**

- Sucht nur nach Wildcards; lpFindData ist das erste Ergebnis.

**BOOL FindNextFile(HANDLE hFindFile, LPWIN32\_FIND\_DATA lpFindData)**

- Das nächste Ergebnis (Rückgabe 0 wenn keine weiteren Dateien)

**BOOL FindClose(HANDLE hFindFile)**

- Beenden der Suche

**FindFirstFileEx erlaubt komplexere Suchen!**