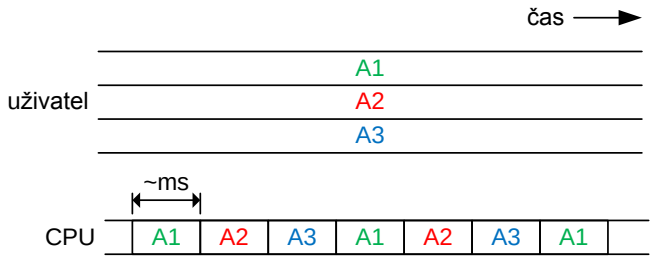
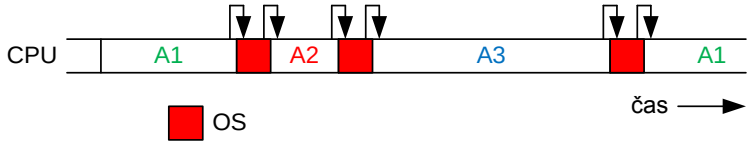
Programy pro OS Windows

Základní pojmy

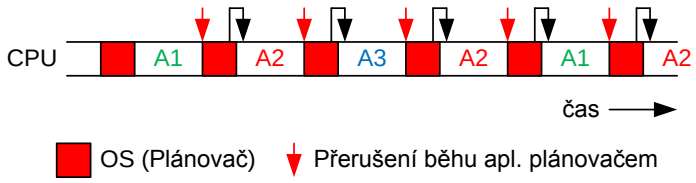
- Operační systém
  - = program, rozhraní mezi člověkem a PC
  - Poskytuje programům přístup k HW (uživatelský vstup, úložiště, RAM, CPU atd.)
- Aplikace (application) = soubor *.exe s kódem programu, proměnnými, a resources (datové zdroje – ikony, obrázky, kurzory, …)
- Multitasking (multitasking)
  - = víceúlohový, běh několika aplikací najednou
  - jedno CPU = simulace, časový multiplex



- kooperativní multitasking (cooperative multitasking)
  - každý program zabere procesorový čas na co nejkratší dobu, a předá jej dalšímu



  - N:
    - 1 program může zablokovat celý OS („zhroucení“ 1 programu = zhroucení OS)
    - Neefektivní řízení (nelze manipulovat s prioritami programů)
- preemptivní multitasking (preemptive multitasking)
  - přidělování času CPU řídí OS (přes tzv. plánovač – scheduler)

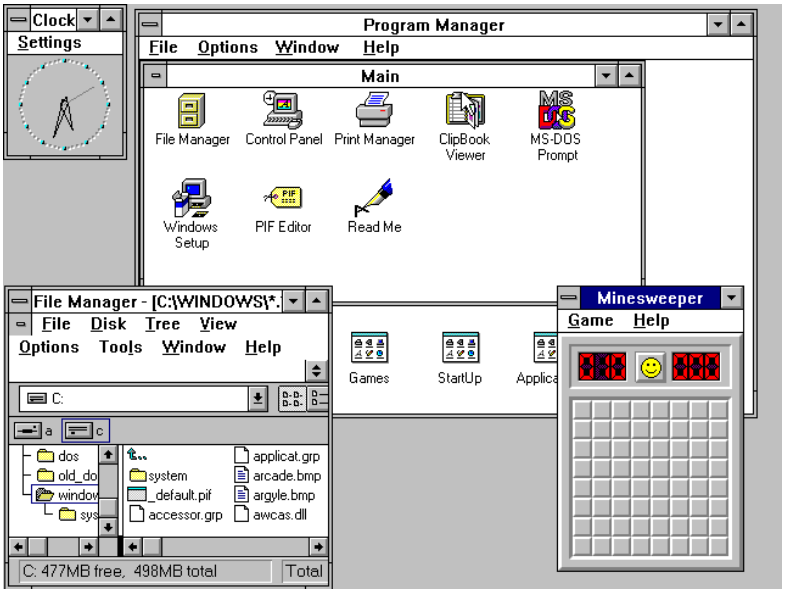


- Proces (Process) = kód aplikace s proměnnými, resources nahraný v paměti PC
  - Při ukončení aplikace OS vše automaticky z paměti smaže
- Vlákno (thread)
  - = posloupnost instrukcí CPU („program“). Každý proces = min. 1 vlákno (main thread)

    - Aplikace může mít více vláken (multithread) – operace na pozadí aplikace
        - Přidělování času vláknům v rámci procesu řídí scheduler (preemptivní princip)
- Ovladač zařízení (device driver)
    - speciální program, který přímo přistupuje na HW a zpřístupňuje jej OS. OS nabízí HW prostřednictvím Hardware abstraction layer (HAL)
    - např. programy nemohou ovládat graf. kartu přímo, pouze přes ovladač
- DLL knihovna (Dynamic Linked Library)
    - = cca *.exe, ale není samostatně spustitelný
    - Princip: program = metody, základ do exe, zbytek do DLL, pokud v programu volána metoda, která je v DLL (není v exe) – OS si ji nahraje z disku
    - Výhody: rychlost, několik procesů může sdílet kód jedné DLL, možnost úprav funkcí programů

Historie Windows

- 1983 – Windows 1.0
    - Spíše přepínač aplikací pro MS-DOS
- 1992 – Windows 3.1
    - Kooperativní multitasking, stále nadstavba MS-DOS, 16-bitový OS
    - Podpora chráněného režimu x86 (až 16MB paměti)
    - Prodej: 50 000 000



- 1993 – Windows NT 3.1
    - NT = pro firemní uživatele
    - Zcela nové jádro, podpora 32b procesorů, preemptivní multitasking, bezpečnost
    - Pokračovatelé: NT 3.51 (1995) – považována za stabilní
- 1995 – Windows 95

    - Spojení MS-DOS 7.0 a Windows, 32b, preemptivní multitasking, podpora Plug'n Play
    - Prodej: 150 000 000
    - Pokračovatelé: Windows 98, Me
- 2000 – Windows 2000 (NT 5.0)
    - spojení 9x (vzhled, multimédia) a NT (jádro, bezpečnost)
- 2001 – Windows XP (NT 5.1)
    - = vylepšené 2000, nový vzhled, rychlejší
    - První systém společný pro firmy i domácnosti
- 2007 – Windows Vista (NT 6.0)
    - nové technologie (WPF, WCF, Aero), masivní příklon k .NET

Architektura OS Windows

- pro W95 a novější
- architektura klient – server
    - obousměrná komunikace
    - klient = běžící procesy
    - server = OS
        - vykonává operace dle přání klientů
        - klientům zasílá zprávy o změnách v OS (uživatel pohnul s myší, …)

Win32 API (WinAPI)

- API = Application Programming Interface
- Zapouzdřuje veškerou funkčnost OS – funkce (metody), datové struktury, datové typy …
    - Tisíce funkcí (= metod), cca 300 zpráv, desítky struktur
- Pohled programátora: OS = API, programování pod Windows = volání WinAPI funkcí
- Ukryto v DLL knihovnách v adresáři Windows/System(32), základ:
    - kernel32.dll – jádro OS (správa paměti, I/O, scheduler)
    - user32.dll – uživatelské rozhraní
    - gdi32.dll – vykreslování obrazu
- SDK (Software Development Kit)
    - = balíček všeho potřebného pro vývoj programů, = přístup k WinAPI
    - pro programování v C/C++ = součást Visual Studia, pro programování v .NET není (bezpodmínečně) nutný

Z pohledu programátora

- Od W 1.0 (W95) stejný programovací model WinAPI
- nové verze W = rozšíření API, jediná změna = .NET
- Pohled do budoucna: příští Windows = .NET, ~~WinAPI~~

Typy aplikací

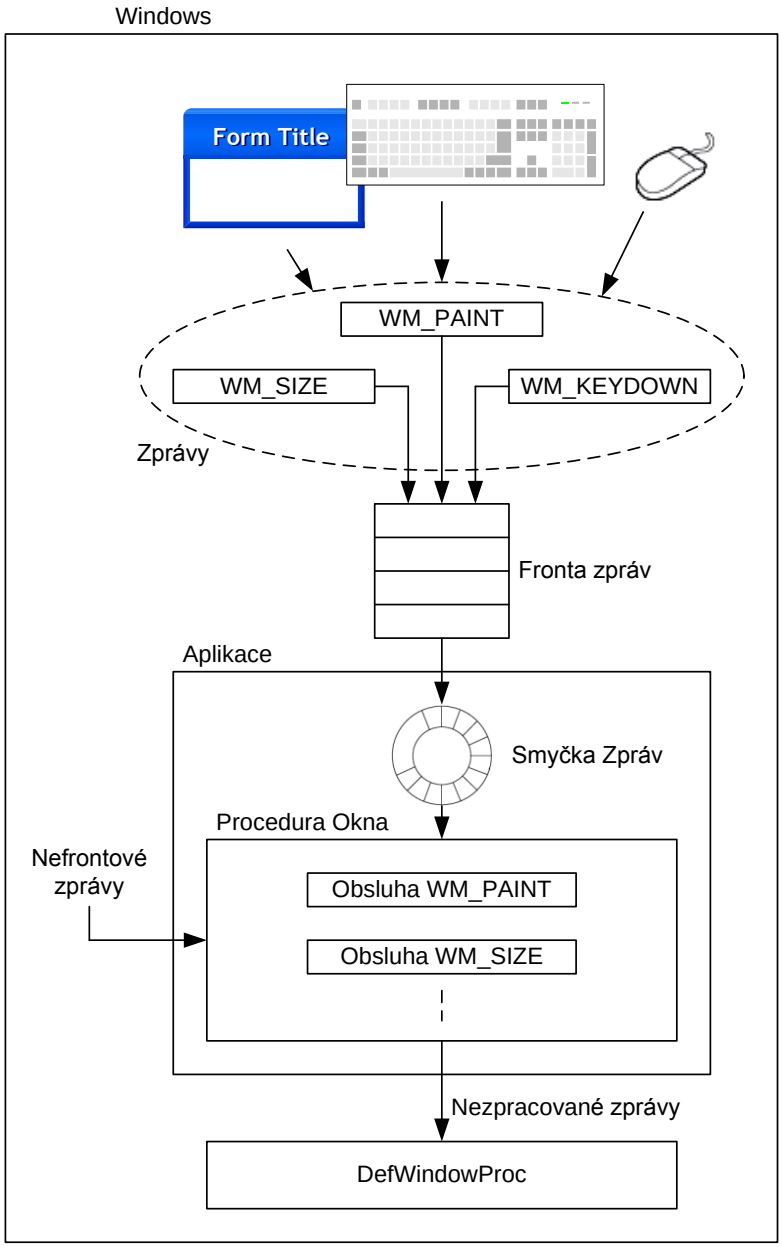
- Konzolové (Console applications) – spojitý vývojový diagram
- S grafickým uživatelským rozhraním (GUI – Graphics User Interface) – nespojitý vv (zprávy, události)

Možnosti vývoje GUI programů

- Přímé volání WinAPI funkcí
  - Nejlépe v C (C++)
  - Výhody: Nejrychlejší, nejmenší programy, lze naprogramovat vše
  - Nevýhody: obtížné, dlouhý vývoj
- Objektové nadstavby nad WinAPI
  - Nejznámější: Microsoft MFC (Foundation Class) – součást placených VS, Borland VCL (Visual Componet Library), QT (zdarma, kód přenositelný na Linux)
  - Zapouzdřují WinAPI do tříd (MFC, QT – C++, VCL – Pascal, C++)
  - Výhody: pohodlnější vývoj, akceptovatelná rychlost, dobrá spolupráce s WinAPI
- RAD nástroje (Rapid Application Development)
  - Možnost konstrukce aplikace vizuální formou, kód píše IDE samo
  - Poprvé Borland Delphi (Pascal) a C++ Builder – RAD nástroje pro VCL
  - Výhody: super rychlý vývoj, snadné
  - Nevýhody: pomalé a velké aplikace, obtížnější spolupráce s WinAPI
- .NET
  - Obsahuje WinForms – zabalení GUI z WinAPI do tříd v .NET
  - Budoucnost vývoje (nejenom) GUI aplikací pod Windows
  - Možnost RAD (Visual Studio)
  - Výhody: bezpečnost (programů – správa paměti, OS – omezenost práv, atd.), unifikace, pohodlný vývoj
  - Nevýhody: Nejpomalejší, obtížná spolupráce s WinAPI (ale lepší než Java – Swing)

Architektura programů pro Windows

- událostmi řízený model (schéma není úplně přesné):



- OS transformuje vše na zprávy (WM_…)
- aplikace reagují na zprávy zasílané op. systémem
- každá aplikace má smyčku zpráv – zprávy si vybírá, třídí a posílá je ke zpracování proceduře okna
  událost = pohyb myši, stisk klávesy, příkaz k překreslení okna …
- aplikací nezpracované zprávy → odeslat zpět OS k výchozímu zpracování

Zpráva (Message)

- frontové
- nefrontové
- = 3 celá čísla (32b)
  - Identifikátor (konstanty WM_…)
  - 2 parametry (`lParam`, `wParam`), význam dle identifikátoru

- Příklady:
  - WM_SIZE – velikost okna aplikace byla změněna
    - Dolních 16b `lParam` = nová šířka okna
    - Horních 16b `lParam` = nová výška okna
    - `wParam`: stav okna (maximalizované, minimalizované, obnovené …)
  - WM_PAINT – systém žádá aplikaci o překreslení obsahu svého okna
    - Žádné parametry
- Možnost definice uživatelských zpráv

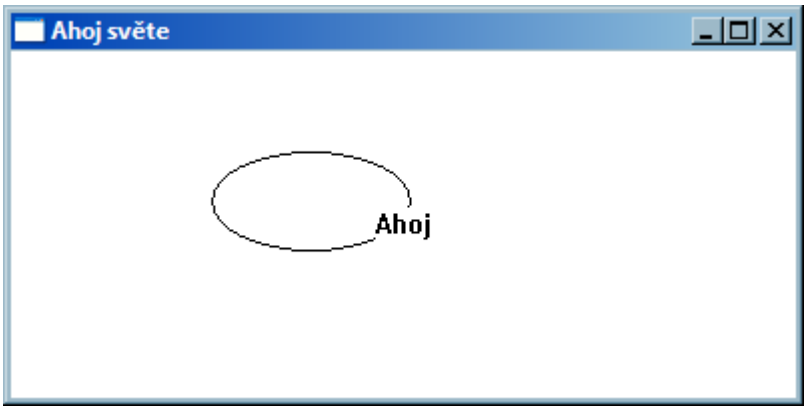
WinAPI aplikace v C

- jen pro ilustraci (WinAPI funkce, konstanty, datové typy)

```
#include <windows.h>
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

// vstupni bod aplikace (ekvivalent Main() z C#)
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
       PSTR szCmdLine, int iCmdShow)
{
       HWND         hwnd;
       MSG          msg;
       WNDCLASS     wndclass;

       // vytvorime tridu okna (jeho budouci podobu)
       wndclass.style         = CS_HREDRAW | CS_VREDRAW;
       wndclass.lpfnWndProc   = WndProc;
       wndclass.cbClsExtra    = 0;
       wndclass.cbWndExtra    = 0;
       wndclass.hInstance     = hInstance ;
       wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION);
       wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW);
       wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
       wndclass.lpszMenuName  = NULL;
       wndclass.lpszClassName = TEXT("MojeTridaOken");

       RegisterClass(&wndclass); // registrace tridy okna

       hwnd = CreateWindow(
              TEXT("MojeTridaOken"),    // window class name
              TEXT("Ahoj světe"),       // Titulek okna
              WS_OVERLAPPEDWINDOW,      // styl okna - ktera tlacitka …
                                        // …v pravem hornim rohu, atd.
              200,                      // initial x position
              200,                      // initial y position
              400,                      // initial x size
              200,                      // initial y size
              NULL,                     // handle nadrazeneho okna (neni)
              NULL,                     // handle pro menu (neni)
              hInstance,                // program instance handle
              NULL) ;                   // creation parameters

       ShowWindow(hwnd, iCmdShow);
       // Windows na miste budouciho okna vymazou obrazovku
       UpdateWindow(hwnd);              // okno aplikace vykreslime
       // smycka zprav - konci prijetim zpravy WM_QUIT (GetMessage() vraci 0)
       while (GetMessage(&msg, NULL, 0, 0))
```

```
        // GetMessage() ceka, az OS da do fronty nejakou zpravu
        {
                TranslateMessage(&msg);
                // predzpracovani zpravy (napr. uprava zprav z klavesnice)
                DispatchMessage(&msg);
                // odesleme procedure okna ke zpracovani
        }
        return 0;    // program konci
}
// procedura okna
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        HDC         hdc;
        PAINTSTRUCT ps;
        RECT        rect;
        // zpracujeme zpravy
        switch (message)
        {
                case WM_PAINT:
                // kdykoli W potrebuji prekreslit obsah okna, zaslou WM_PAINT
                        hdc = BeginPaint(hwnd, &ps);
                        GetClientRect(hwnd, &rect);
                        Ellipse(hdc, 100, 50, 200, 100);
                        DrawText(hdc, TEXT("Ahoj"), -1, &rect,
                                DT_SINGLELINE | DT_CENTER | DT_VCENTER);
                        EndPaint(hwnd, &ps);
                        return 0 ;
                case WM_DESTROY:
                        PostQuitMessage(0);
                        return 0 ;
        }
        return DefWindowProc(hwnd, message, wParam, lParam);
        // nezpracovane odesleme OS k defaultmimu zpracovani
}
```

- Výsledek:

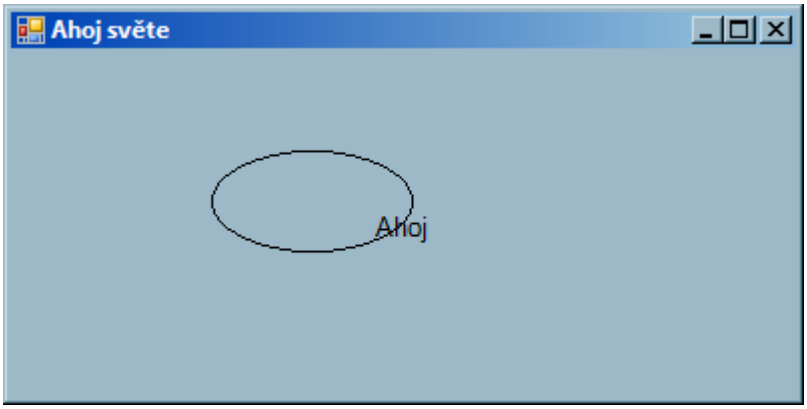


.NET GUI (WinForms) aplikace

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Drawing;

namespace WinForms_Hello
{
```

```
        class Program
        {
                static void Main(string[] args)
                {
                        Form okno = new Form();
                        okno.Text = "Ahoj světe";
                        okno.Size = new Size(400, 200);
                        okno.Paint += new PaintEventHandler(okno_Paint);
                        Application.Run(okno);
                }
                static void okno_Paint(object sender, PaintEventArgs e)
                {
                        Graphics g = e.Graphics;
                        Pen pero = new Pen(Color.Black);
                        Brush stetec = new SolidBrush(Color.Black);
                        Font font = new Font("Arial", 10);
                        g.DrawEllipse(pero, 100, 50, 100, 50);
                        g.DrawString("Ahoj", font, stetec, 180, 80);
                }
        }
}
```

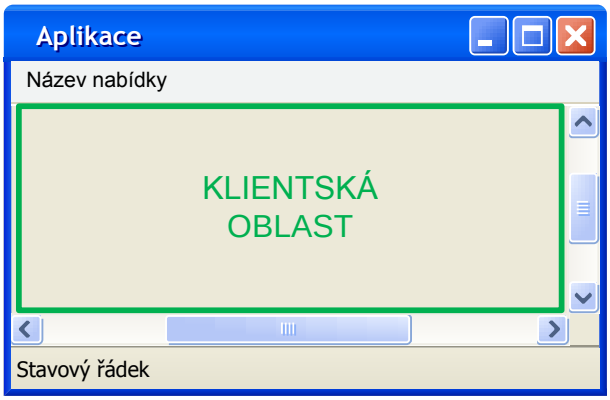
- Ne zcela dokonalá varianta – lepší odvodit vlastní třídu od `Form`
- Výsledek:



UI Windows základní pojmy

Okno

- Základní prvek GUI
- Okno:
  - WinAPI – „vše je okno“ (okno, tlačítka, menu, …)
  - pohled uživatele (i programátora Winforms) – okno = formulář, dialog
- Client area
  - Může obsahovat ovládací prvky (i např. dokument)



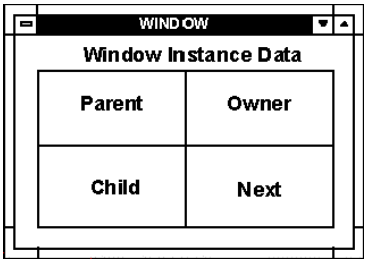
  - Lze do ní kreslit
- Neklientská část
  - Title bar, Menu bar, status bar, H, V scrollbars, …
  - omezeně lze také kreslit
- Stavy okna: Maximalizované, minimalizované, skryté (hidden), normální (zaujímá část obrazovky)

SDI/MDI aplikace

- SDI = Single dokument Interface (poznámkový blok)
- MDI = Multiple …
  - více dokumentů v jedné aplikaci
  - Word, Visual Studio …

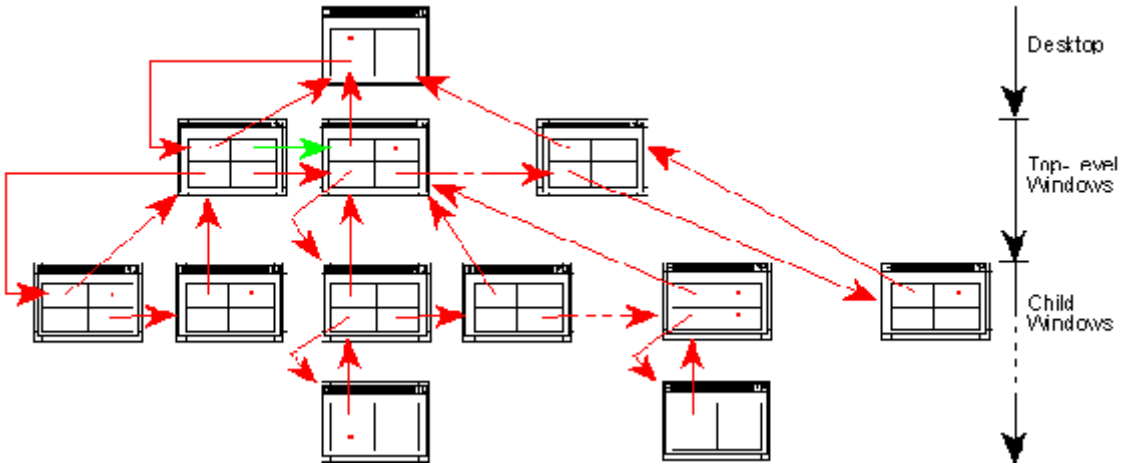
Hierarchie oken

- Každé okno
  - jednoznačné ID v systému – Window handle (32b celé číslo)
  - nese info (handle) o dalších oknech



- Parent Window – nadřazené okno
  - Windows desktop = „nejvyšší“ okno
- Child window – podřízené okno
  - Nemohou samostatně existovat – zobrazovány v klientské oblasti parent w.
  - Změna rodiče ovlivní potomka (existence, viditelnost)
  - Typicky: ovládací prvky („vše je okno“)
- Top-level Windows
  - Nemá vlastníka (resp. vlastník = desktop)
  - Hlavní okno apliakcí
- Owner – vlastník

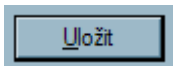
- Okno stejné úrovně (pouze top-level windows)
- aplikace typu „winamp“ (přehrávač + playlist window)



Ovládací prvky

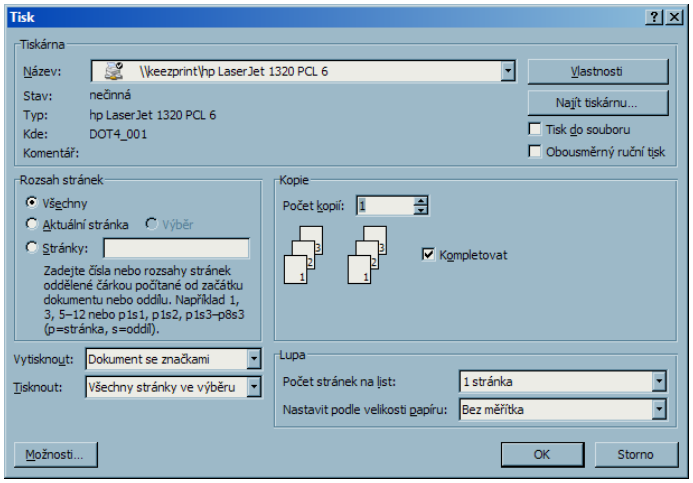
- Tlačítka, checkboxy, edit boxy …
- Focus = aktivní ovládací prvek, přepínání „tab“

- Předvolený prvek (dialogy) – reaguje na enter (OK), esc (Storno) …



Dialogy

- = okno s ovládacími prvky, zobrazení z hlavního okna, zpravidla neměnná velikost



- Určeny pro interakci s uživatelem
- Př.: dialog pro tisk souboru, nastavení aplikace…
- Modální vs. nemodální zobrazení
  - Modální – blokuje ostatní okna aplikace
  - Systémově modální – blokuje všechny aplikace („chyba při čtení z paměti…“)
  - Nemodální – neblokuje (např. výběr barvy v grafických aplikacích)