

– Lab1 –

Introduktion

Förberedelse för planetlabben genom att kapsla in (skapa wrappers) systemanrop.

1 Syfte

Få en känsla av hur Win32API fungerar, dvs programmerarens interface gentemot Windows. Känsla för tråd och processbegreppet, kritiska sektioner, samt kommunikation via mailslots.

En förberedelse inför planet laborationen genom att kapsla in systemanrop (göra wrappers)

2 Innehåll

- Win32API
- Skapa trådar
- Kritiska sektioner
- Mailslothantering

3 Förberedelse & Tips

- Se hur man sätter upp projekt i Visual Studio.
- Bekanta er hur man använder hjälpen för att hitta rätt systemanrop, den lär ni behöva!
- Se `wrapper.h` för att se vilka systemanrop ni behöver kapsla in! Denna fil får ni ej ändra, dvs interfacet är spikat!
- `wrapper.c` innehåller alla skelett för dessa.
- Skapa filen `intro.c` och gör uppgifterna i denna fil.
- Fyll i wrappers direkt (i `wrapper.c`) och använd dessa i era uppgifter istället för att fixa dem på slutet.

4 Redovisning

Du behöver bara redovisa uppgift 4 b)! Tänk dock på att `wrapper.c` kommer att användas i senare labbar.

5 Uppgifter

1. Du skall köra en process som skriver "Hello world!" 10 gånger på skärmen med en fördröjning på 1 sekund mellan varje. Processen ska terminera efter detta.
2. Skapa nu ytterligare en tråd som skriver "Hello moon" på skärmen i all evighet. Fördröjning mellan varje utskrift skall vara 0.2 sekunder. Skapa den nya tråden *innan* programmet skriver ut "Hello world!" tio gånger. Varför avslutas måntråden?
Prova att skicka en parameter till den nya tråden. Om du inte gjort det fyll i wrappern `threadCreate` i `wrapper.c`.
3. a) Ändra programmet från uppgift 2, så att 10 "Hello world!" skrivs ut, sedan 10 "Hello moon", och sedan så repeteras detta beteende i evighet.
b) Kan du garantera att ditt program i alla lägen uppför sig enligt ovan? Om inte, modifiera programmet så att det alltid gör det. *Tips:* Kolla på kritiska sektioner.
4. a) Skapa en process som innehåller *exakt* två trådar. Användaren matar in text i den ena tråden, som skall koppla upp sig mot en brevlåda skapad av den andra tråden. Denna andra tråd skall läsa ur brevlådan och skriva ut mottagen text på skärmen. Systemet

skall snurra tills strängen "slut" skrivs av användaren.



- b) Gör nu samma sak men istället för att skicka strängar skickar du över poster (**struct**) av valfri typ.
- c) Om du inte gjort det, fyll i wrappers för samtliga systemanrop som du använt i denna lab i `wrapper.c`.

A Lathund för hur man skapar projekt i MS Visual Studio

OBS! Denna lathund är skriven för en äldre version av Visual Studio (version 6), men ni kan säkert få lite tips från den ändå.

A.1 Sammanfattning

I denna kurs kommer vi att arbeta helt i Visual studio. För er som tidigare arbetat i Borland C så är det möjligt att genomföra laborationerna i Borland. Dock rekommenderar vi er att använda Visual Studio istället eftersom det är en av de vanligaste kompilatorerna på marknaden idag. Ni som tidigare arbetat i Borland kommer att märka att det mesta är sig likt, men att Visual Studio ibland lägger till saker som man inte riktigt har kontroll över. Det är dock möjligt att göra allt arbete själv från grunden. Det är på det sättet vi kommer att jobba i den här kursen.

A.2 Några ord om miljön

Visual studio är en komplett utvecklingsmiljö som inte bara innehåller en editor, en kompilator och en länkare för C. Den innehåller dessutom en miljö för Visual Basic, ett dokumenthanteringssystem, Visual Sourcesafe, en drös med analysverktyg, debuggers och mycket annat. Till Visual Studio finns också ett hjälpsystem som heter Microsoft developers network, MSDN. De delar ni kommer att använda är Visual C++ miljön som innehåller ungefär samma saker som Borland C++ samt MSDN för att hitta hjälp.

Visual C++ (VC++) hittar du på Startmenyn under fliken Microsoft visual studio. MSDN når du, om det är installerat på datorn, genom att välja hjälpen i VC++, eller genom att markera ett ord i koden och trycka på F1. Skulle MSDN inte vara installerat på datorn så hittar du samma information på <http://www.msdn.microsoft.com>. Det är lite krångligare att använda MSDN på det sättet eftersom det inte är direkt kopplat till VC++.

A.3 Workspace och projekt

När du startar VC++ möts du av något som i princip liknar Borland C++. Du har en tom editor, en navigationspanel till vänster och ett statusfönster längst ner (se figur 1).

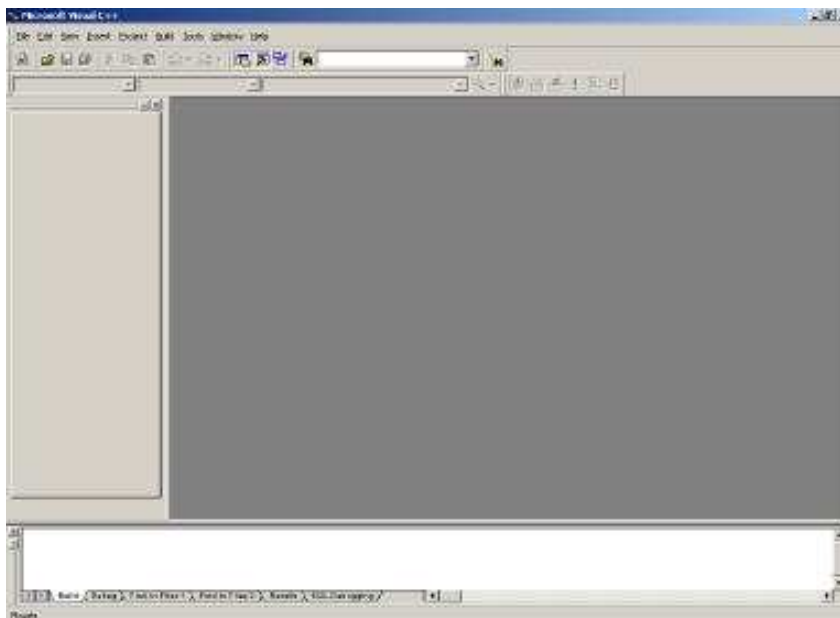


Figure 1: VC++ som det ser ut när du startar det.

Nu skall du öppna ett workspace/solution¹. Ett workspace är inte ett projekt utan kan ses mer som ett överprojekt. I en workspace kan du samla flera projekt som hör ihop. Detta kommer ni att ha stor nytta av när ni gör planetlabben senare. Ett workspace öppnar du genom att välja *File->New*. Välj fliken workspace, välj lämpligt ställe att spara workspacet och hitta på något klurigt namn, t.ex. *os*. Som ni märker händer inte så mycket förutom att en liten markering i navigationspanelen visar att *os* nu finns.

När detta är klart är det dags att lägga till det första projektet. Detta kan göras på två sätt, antingen väljer ni *File->New* igen, eller också högerklickar ni på *os* i navigationspanelen. Hur ni än väljer att göra möts ni av följande fönster:

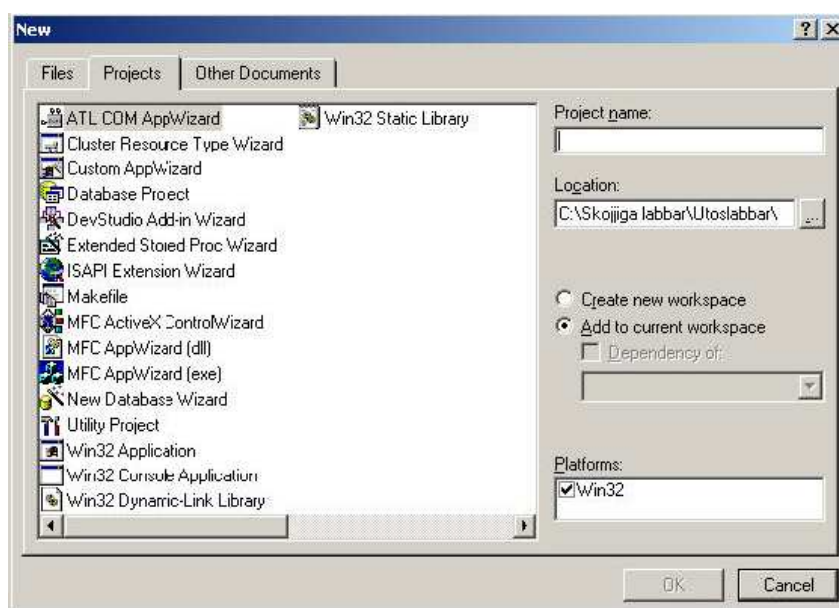


Figure 2: Fönster i VC++ för att skapa ett nytt projekt.

Studerar man listan över alla tänkbara projekttyper så märker man att man, relativt enkelt kan skapa de mest intressanta applikationer. Till exempel ATL COM AppWizard där man kan göra COM objekt som kan nås från alla program i din dator, t ex Excel, Visual Basic eller något eget skapat program. Vidare finns Win32 Dynamic-Link Library där ni kan göra en DLL fil som kan distribueras med er applikation. De två projekttyper ni behöver för denna kurs är:

- **Win32 Console Application** som är en enkel applikation som normalt kommunicerar via Text. Denna typ kan jämföras med Borlands Easywin. Den har alltid en Main funktion och fungerar som ni är vana vid.
- **Win32 Application** är en windows applikation. Den är tänkt för applikationer som kommunicerar via windowsfönster, med knappar och grafik. Den är med andra ord expert på att rita punkter i alla möjliga färger (läs planeter hint hint). I denna applikation har man ingen Main funktion utan en Winmain, som inte egentligen innehåller särskilt mycket som ni kommer att märka.

Eftersom detta är ert första projekt utgår vi ifrån att ni vill skapa en *Win32 Console Application*. Välj följaktligen detta alternativ i listan och hitta på ett fifflurigt namn på projektet, t ex Kommandotolk. Innan ni klickar på OK-knappen se till att ni har valet *Add to current workspace*

¹Namnet skiljer mellan olika versioner av Visual Studio

ifyllt.


Nu kommer ett fönster upp där ni har möjlighet att välja ett uppstartat projekt eller ett tomt. Det kan vara bra att välja ett uppstartat projekt om man vill veta hur strukturen skall vara på en viss projekttyp. Dock skall ni i denna kurs välja *Empty project*. Då slipper ni dessutom alla märkliga filer som VC++ annars skapar.

När ni slutfört skapandet av projektet har ett antal saker hänt. Ni har fått upp projektet i navigationspanelen och ni kan se att H-filer och C-filer skall ligga i separata mappar.




A.4 Att programmera i VC++

För att lägga till filer i projektet högerklickar ni bara på *Source files* respektive *Header files* i navigationspanelen och väljer Add files to folder. Vill ni skapa en ny fil skriver ni in ett filnamn, t ex Kommandotolk.c och klickar OK, efter en fråga (som ni svarar ja på), skapas filen.

Öppna filen och koda som vanligt. För att kompilera koden trycker du på knappen  eller trycker på F7. All kompilerings och länkningsoutput visas i statusfönstret längst ner. Vill du


köra programmet (och kompilera vid behov) trycker du på knappen  alternativt trycker på CTRL-F5.

Värt att notera som bra features är till exempel:

- En liten hjälp ramlar upp när man har skrivit in ett funktionsanrop och just ska till och skriva in första argumentet. Hjälpen visar vilka argument och vilka returvärden som funktionen har.
- Samtliga funktioner i koden kan återfinnas i fönstret ovanför editorfönstret. Man kan välja vilken funktion man vill titta på och klicka på  så hoppar man dit. Perfekt när antalet funktioner växer. En liknande översikt kan man också få om man klickar på fliken  i navigationspanelen.
- Knappen  och tillhörande textruta kan användas för att leta efter förekomster av en viss sträng i alla filer i projektet. En `#define` som ligger i någon bortglömd fil kanske

A.5 Debuggern i VC++

En bra debugger finns naturligtvis i Visual C++ också. Det enklaste sättet att starta debuggern är att infoga en breakpoint någonstans i programmet. Detta görs enklast med att man markerar en rad i editorn och högerklickar på den. Välj sedan Insert/Remove Breakpoint och du får en

röd cirkel i marginalen. Programmet kör du sedan genom att klicka på knappen .

Det som händer då är att du byter miljö i VC++ till Debug miljö. Du får bland annat upp en liten kontrollpanel där du kan styra exekveringen av koden på ett liknande sätt som i Borland C++.

Det finns naturligtvis massor av intressanta funktioner, här är några:

- Om du vill veta det aktuella värdet på någon variabel i koden räcker det att du håller muspekaren på den så hoppar det upp ett lite fönster som talar om det.
- Rutan längst ner till vänster på skärmen visar de variabler som påverkas av raden som just exekverats. Vill man istället se alla variabler kan man välja att klicka på fliken Locals.

- Naturligtvis finns det variabler som man vill hålla ett extra öga på, då är det lämpligt att använda rutan längst ner till höger istället. Den fungerar som Borlands Watch window. Med andra ord kan du själv välja vilka variabler du vill titta på genom att till exempel högerklicka på en variabel i koden och välja *Quickwatch*. I fönstret som då dyker upp klickar du på *Add Watch*, och den hoppar ner till rutan. Du kan givetvis också manuellt tvinga in nya värden på variabler under körning. Kan vara bra ibland.

A.6 Konfigurationer och filstrukturer i VC++

När du skapar ett projekt i VC++ träder det automatiskt in i en konfiguration som kallas Debug. Detta är inte att förväxla med debuggern även om det finns kopplingar där emellan. Debug-konfigurationen talar om för VC++ att projektet är i utvecklingsfasen. I och med detta läggs debuginformation in i den kompillerade exe-filen. Den filen, tillsammans med intermediära filer, hamnar i en underkatalog till projektet som heter Debug. När ni är klara med programmet kan ni, om ni vill, välja *Win32 Release* i fönstret *Set active Configuration* som återfinns i menyn *Build*. Då kommer den färdiga programfilen att hamna i en ny underkatalog som heter *Release*.

A.7 Att arbeta med flera projekt i samma workspace

Det är fullt möjligt, som tidigare nämnts, att arbeta med flera projekt i samma workspace. Vad man gör är helt enkelt samma procedur som när du skapade första projektet. Det finns dock några saker man bör tänka på när man arbetar på detta sätt. När man vill köra sitt program från VC++ så måste VC++ veta vilket projekts programfil som skall köras. Detta görs genom att välja vilket projekt som skall vara aktivt. Detta görs genom att i navigationspanelen högerklicka på det projekt man vill ha aktivt och välja *Set as Active Project*.

Ibland vill man använda kodfiler från andra projekt i sitt projekt. Pondera att ni utvecklat en kanonfunktion för att vända en sträng i projekt A och nu behöver ni just denna i projekt B. Det enklaste sättet att göra detta är att lägga till H-filen och C-filen från projekt A i projekt B. Observera att det räcker att inkludera filerna i VC++. Du behöver inte fysiskt flytta filerna från ett projekt till ett annat. Ändrar du sedan något i någon av filerna ändras den ju i alla projekt eftersom den bara finns på ett ställe fysiskt.