STATISK OCH DYNAMISK LÄNKNING

I C, även om det oftast inte syns som olika steg i LW, måste källkoden för ett program kompileras och *sedan* länkas till olika bibliotek för att man ska få en exekverbar fil (.exe). De olika biblioteken kan till exempel vara standardbibliotek <bibl.h> eller egna bibliotek ”bibl.h”. De kan innehålla förkompilerade funktioner (*precompiled functions*) som kan anropas från programmet för att utföra olika uppgifter, exempelvis spara information på en hårddisk eller allokera minne.

När dessa funktioner länkas till en applikation (d.v.s. ett program, eller i LW snarare ett *projekt*) kopieras de in och blir en permanent del av applikationens exekverbara fil. Samtliga anrop till funktioner i något av de länkade biblioteken *översätts* eller *utförs* bara en gång, nämligen under själva länkningen. Detta kallas **statisk länkning** och följaktligen **statiska bibliotek**. Detta är praktiskt om man vill låta andra använda funktionerna utan att vilja/behöva avslöja koden.

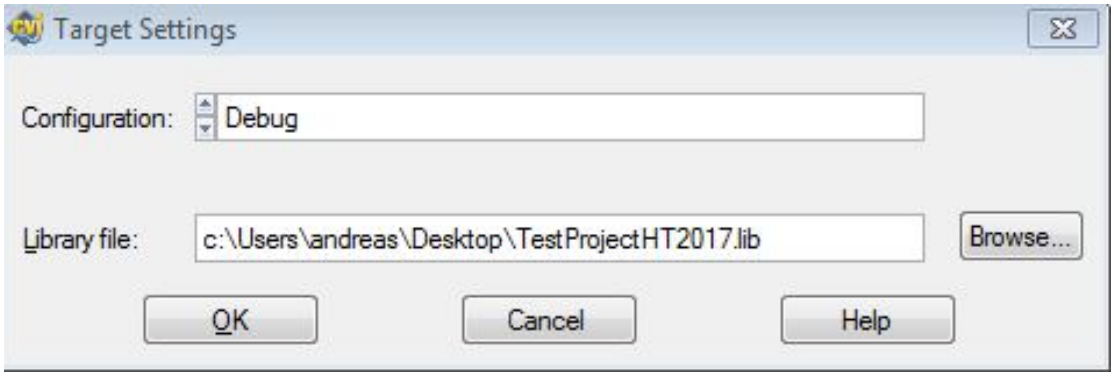
De flesta Windows-applikationer använder en annan typ av länkning, **dynamisk länkning** (*dynamic linking*), som gör det möjligt att länka en applikation till ett eller flera bibliotek under själva programkörningen. Till skillnad från statiska så kopieras **dynamiska bibliotek** aldrig in i en applikations exekverbara fil. Ett dynamiskt bibliotek kallas vanligen ett DLL (Dynamic Link Library) – en .DLL-fil. Vid programkörning laddar operativsystemet in DLL-filen i minnet och länkar alla referenser till funktioner I DLL-filen, så de kan anropas av applikationen. Sedan exekverar de delarna av applikationen som använder DLL-funktionerna. Slutligen frigörs minnet när DLLfunktionerna ej längre behövs.

Nedan följer praktiska ***steg-för-steg guider*** för länkade bibliotek i LW.

ATT SKAPA STATISKA BIBLIOTEK (.LIB-FILER) MED LW/CVI

1. Ta fram ett projekt med minst en fil i.
2. Välj *Build -> Target Type*.
3. Välj *Static Library*.
4. Välj *Build -> Create Static Library*.

(Hur en .lib-fil är uppbyggd beror på vilket utvecklingsmiljö man använder. Därför finns det, under *Build -> Target Settings...*, möjlighet att välja konfigurationen.



ATT SKAPA OCH ANVÄNDA DYNAMISKA BIBLIOTEK (.DLL-filer) MED LW/CVI

Innan du börjar – lite bakgrund:
i) källkoden i en DLL innehåller en *DllMain*-funktion (OBS *ej* DLLMain) som anropas när en exekverande applikation laddar. Detta skapar en möjlighet för programmeraren att utföra olika typer av *initialiseringar* när DLL-funktionerna laddas i minnet och att *frigöra* vissa minnesobjekt när DLL-funktionerna ej behövs längre.
ii) en .h fil bör innehålla prototyper för alla de funktioner man vill ska kunna anropas I DLL-filen (dessa funktioner kallas *exporterade funktioner*). Kompilatorn måste få veta exakt vilka funktioner som ska finna tillgängliga för anropande funktioner. Detta då DLL-filer kan innehålla funktioner som används internt och *inte* kan anropas utifrån.

// Att skapa ett .dll

1. Skapa ett nytt LW-projekt *firstdll.prj*.

2. Välj *Build -> Target Type -> Dynamic Link Library*.

3. Skapa en ny källkodsfil och skriv in eller kopiera DLL.c enligt nedan:

```
#include <cvirte.h>
#include <userint.h>
int _stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID
lpvReserved)  {
                if(fdwReason==DLL_PROCESS_ATTACH){
                        MessagePopup("Inne i DllMain", "Jag har laddats...");
                }
                else if (fdwReason==DLL_PROCESS_DETACH){
                MessagePopup("Inne i DllMain", "Jag har urladdats...");
                }
                return 1;
}
void FunkInternal(void) {
                        MessagePopup("", "Inne i interna DLL-funktionen");
}
void FunkDLLTest(void) {
                        MessagePopup("Inne i FunkDLLTest", "Exporterad DLL-
                        funktion");
                        FunkInternal();
}
```

// *__stdcall* är anropskonventionen som rekommenderas för exporterade funktioner.
// de andra funktioner vi använder är standardfunktioner.

4. Spara filen som *firstdll.c* och lägg till ditt projekt.

5. Skapa en ny .h-fil och skriv in följande:

void FunkDLLTest(void); //extern function

6. Spara filen som *firstdll.h* och lägg den till ditt projekt.

7. Välj *Build -> Configuration -> Release*.

8. Välj *Build -> Target Settings ->* välj *Change* i rutan *Exports*. I fönstret *DLL Export Options* välj *firstdll.h* och *OK*.
// LW kommer att använda denna header-fil för att bestämma vilka funktioner
// som skall exporteras. Den enda funktionsprototypen i denna header-fil är
// *FunkDLLTest* och detta är alltså den enda funktion som kommer att
// exporteras.

9. Välj *OK*.

10. Välj *Build -> Create Release Dynamic Link Library*.
// Nu skapas *firstdll.dll* och flera olika kopior av *.lib*.
// En del kopior placeras i en egen mapp med kompilatorns namn.
// .lib är importeringsbibliotek som LW använder, .dll inkluderas aldrig i ”build”

11. Stäng alla fönster utom projektfönstret. Spara alla ändringar (*Save All*).

// Att använda din .dll

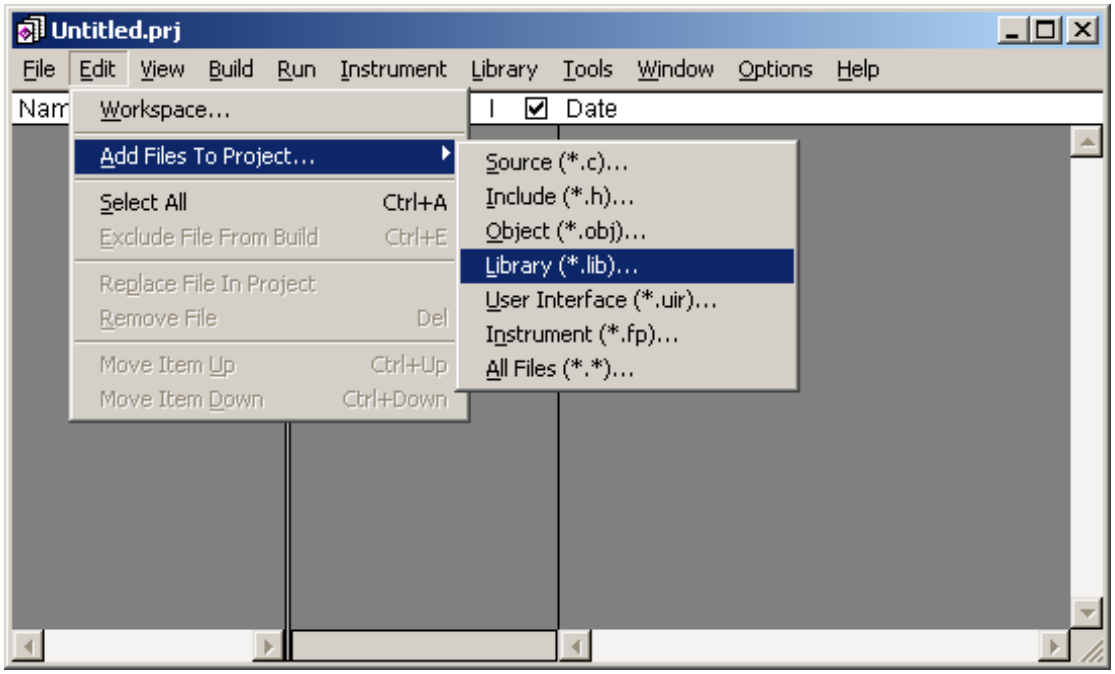
12. Skapa ett nytt projekt: *usedll.prj*.

13. Skapa en ny källkodsfil och skriv in eller kopiera *usedll.c* enligt nedan:

```
#include <stdio.h>
#include "firstdll.h"
int main (void) {
FunkDLLTest(); //Här anropas den exporterade funktionen
return 0;
}
```

14. Lägg till filen till ditt projekt (*usedll.c*).

15. Välj Edit -> *Add Files to Project -> .lib*
// Välj ditt .lib från steg 13 ovan. OBS *ej* .dll
// I detta importeringsbibliotek finns ju referensen till funktionen...



16. Kör projektet:
- Fyra meddelanden, som du klickar bort med *"OK"*:
    1. laddning av DLL
    2. anrop av *FunkDLLTest*
    3. internt anrop från *FunkDLLTest* till *FunkInternal*
    4. urladdning av DLL.

Det kan vara en mycket bra idé att titta på flera exempel i *...\samples\dll*.