Tworzenie bibliotek dynamicznych w kompilatorach GNU Fortran

Dwa kompilatory GNU Fortran pozwalają na stworzenie bibliotek dynamicznych (DLL – Dynamic Link Library). Zarówno kompilator g95 jak i gfortran tworzą bibliotekę dynamiczną przy użyciu tych samych instrukcji. Dla g95:

```
g95 -c biblioteka.f90
g95 -shared -mrtd -o biblioteka.dll biblioteka.o
```

Dla gfortran:

```
gfortran -c biblioteka.f90
gfortran -shared -mrtd -o biblioteka.dll biblioteka.o
```

Przy czym:

-c	Jedynie kompiluje i tworzy plik obiektowy biblioteka.o.
-o	Nazwa pliku wyjściowego biblioteka.dll.
-shared	Tworzy plik dll.
-mrtd	Ustawia konwencję zgodną z stdcall.

Jednakże nazwa pliku wyjściowego *.o musi być zgodna z nazwą pliku źródła. Natomiast nazwa biblioteki nie musi. Plik obiektowy jest niezbędny przy kompilacji do pliku biblioteki. Tutaj przykładowy kod źródłowy zawarty jest pliku biblioteka.f90.

Kod biblioteki ma bardzo prostą formę:

```
subroutine procedura(r)
real*8 r,a
r=r+1
a=3.7
r=atan2(r,a)
return
end
```

Jak widać, jest to po prostu procedura zwracająca wartość ośmiobajtowej zmiennej typu **`real`**. Nie ma tu potrzeby tworzenia interfejsu, jak w bibliotekach statycznych.

Po skompilowaniu biblioteki, wystarczy skompilować przykładowy program:

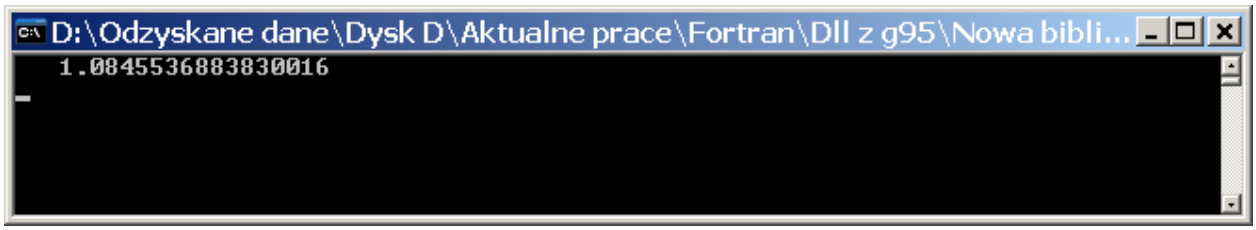
```
program aplikacja
real*8 :: r
r = 6
call procedura(r)
print *, r
read(*, *)
end
```

Do tego celu potrzebna będzie dla g95 taka instrukcja kompilatora:

```
g95 -o aplikacja aplikacja.f90 -L. biblioteka.dll
```

- gdzie –L linkuje bibliotekę biblioteka.dll do pliku wynikowego programu aplikacja.exe. Dla gfortran wystarczy zmienić nazwę kompilatora, jak w przypadku kompilowania biblioteki.

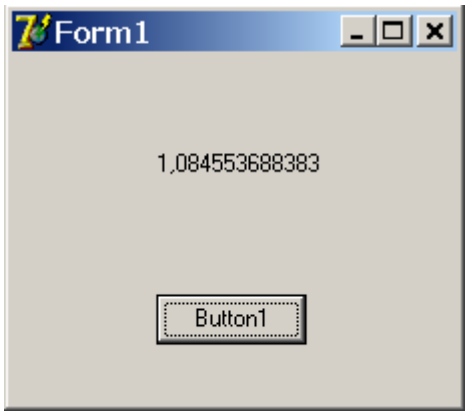
I po wszystkim. Po uruchomieniu aplikacji, skorzysta ona z biblioteki biblioteka.dll i wyświetli na konsoli wynik procedury `procedura()`. Dzieje się tak nawet po (podwójnym) kliknięciu na program, które otworzy okienko konsoli z wynikiem programu. Zapewnia to wywołanie **`read`**`(*, *)`, zatrzymujące konsolę na ekranie.



Taka biblioteka ma jeszcze jedną zaletę, gdyż można ją podłączyć do innych programów, tworzonych na innych kompilatorach. Przykładowo w środowisku Delphi będzie to wyglądało tak:

```
unit Biblioteka;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
  private
  public
  end;

var
  Form1: TForm1;

const
  B_DLL='biblioteka.dll';

implementation

procedure procedura_(var i:Real);stdcall; external B_DLL;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var r:Real;
begin
   r:=6;
   procedura_(r);
   Label1.Caption:=FloatToStr(r);
end;

end.
```

Zatem pod zdarzeniem OnClick przycisku Button1 wywołana będzie procedura biblioteki biblioteka.dll i wynik, zwrócony poprzez zmienną `r` typu `Real`, zostanie wyświetlony w kontrolce tekstowej Label1.



Kruczkiem tego wykorzystania procedury bibliotecznej GNU Fortrana jest to, że trzeba, wczytując ją, nadać jej nazwę zakończoną podkreśleniem _, czyli gdy w oryginale ma nazwę `procedura`, to w Delphi musi być `procedura_`. Inaczej aplikacja, ładując bibliotekę biblioteka.dll, nie znajdzie punktu wejścia do tej procedury. Delphi skraca wynik aż o cztery miejsca znaczące od końca, ale to w niczym nie zmienia faktu, że taka biblioteka współdziała z innymi programami niż tylko napisanymi w GNU Fortranie.

Oczywiście przykłady Fortrana wymagają ustawienia zmiennej środowiskowej, kierującej instrukcje do katalogu, w którym jest dany kompilator i jego biblioteki. Opisany przykład jest bardzo prosty, ale ilustruje ogólne zasady postępowania przy tworzeniu i podłączaniu bibliotek dołączanych dynamicznie napisanych w GNU Fortranie.

Andrzej Marek Hendzel