

Modul Japigraf pro kompilátor g95

(rozpracovaný text, stav k 4.12.2007)

Jan Celý, Ústav fyziky kondenzovaných látek, Přírodovědecká fakulta MU

OBSAH

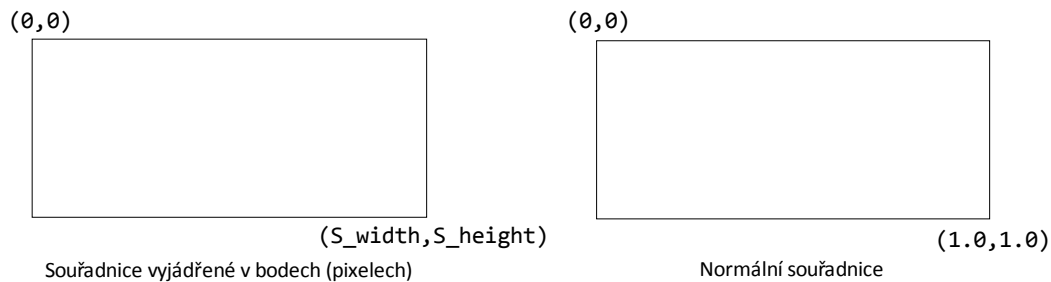
Základní pojmy a používané souřadné soustavy	1
Přirozené a normální souřadnice celé obrazovky	1
Frame, canvas, PlotArea a ViewPort	2
Uživatelské souřadnice	2
Strukturované proměnné pro popis ViewPortu	3
Uživatelské typy doplněné ve verzi 2.4	4
Modul rgbcolor	6
Zahájení práce v prostředí JAPI + Japigraf	6
Procedura StartJapi()	6
Procedura NastavStdGraf	7
Procedura DefViewPort	7
Doplnění a aktivace standardního menu na ViewPort	8
Nezávislé vytvoření <i>frame</i> a <i>canvas</i> , sestavení proměnné <i>infoVP</i>	8

Modul JAPI je soubor procedur napsaných v jazyce Java, které je možné volat z různých jazyků, včetně Fortranu 95. Získat ho můžete na stránkách www.japi.de nebo jako součást balíku *Fortran Tools* od [The Fortran Company](http://www.fortran.com). V dokumentaci k balíku *Fortran Tools* najdete příručku téhož jména, která v kap. 12 uvádí několik příkladů použití modulu JAPI pro tvorbu grafického uživatelského prostředí (GUI); vřele doporučuji tuto kapitolu přečíst a odzkoušet příklady. Modul JAPI sice obsahuje řadu elementárních procedur, které v principu umožňují kresbu grafů, ale neobsahuje žádné procedury "vyšší úrovně", např. zavedení uživatelských jednotek, kresbu os a pod. Napsal jsem proto doplňující modul Japigraf, který obsahuje alespoň základní procedury pro rychlé vytvoření 2D grafů. Neobsahuje však ani zdaleka vše, co by takový modul měl zahrnovat. Chápejte ho proto jako součást programů určených pro výuku, které můžete upravovat a doplňovat podle svých potřeb (chybí zde např. kresba logaritmické a reverzní osy atd.); stejně jako u všech ostatních výukových modulů, které dávám k dispozici, znovu prosím: přihlaste se k provedeným úpravám hned v úvodu modulu, aby případný další uživatel (včetně vás samotných) nebyl později přiváděn k zoufalství, že věci nefungují v souladu s původními komentáři.

Základní pojmy a používané souřadné soustavy

Přirozené a normální souřadnice celé obrazovky

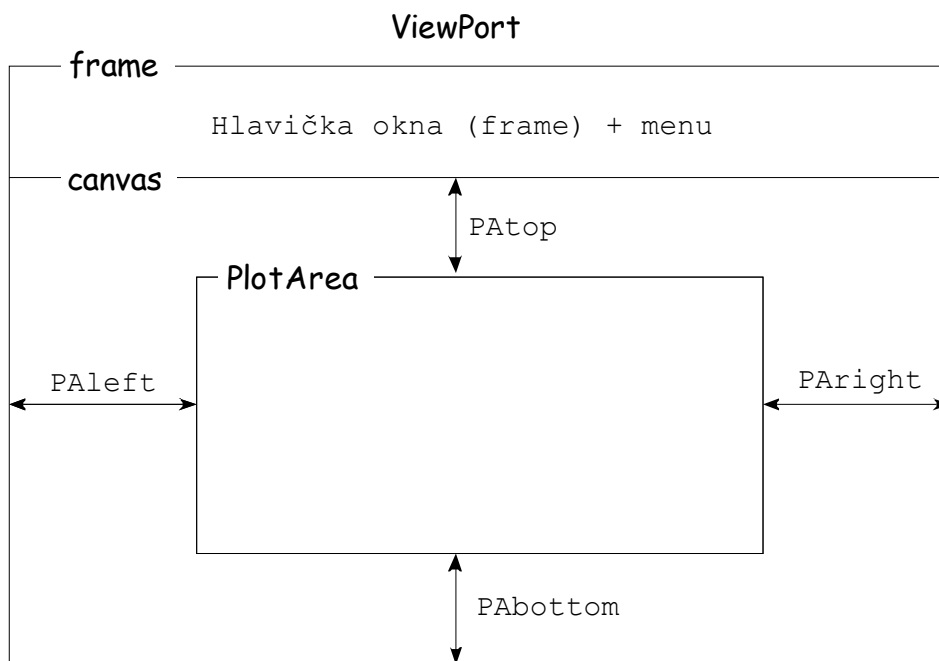
Modul JAPI vychází z rozlišení obrazovky, které je vyjádřeno standarním způsobem – podle nastaveného rozlišení – v bodech (píxelech) a v těchto jednotkách také uvádí rozměry v procedurách. To samozřejmě není praktické, protože na obrazovkách s různým rozlišením bude obraz vypadat různě (např. . se nevejde celý na plochu obrazovky). Zavádím proto *normální souřadnice* typu REAL, tak jak je zřejmé z obr. 1. Hned při inicializaci procedurou *StartJapi()* se zjistí nastavené rozlišení obrazovky, které se uloží do veřejně přístupných (PUBLIC) proměnných *S_width*, *S_height* a nastaví se možnost používat normální souřadnice.



Obr. 1

Frame, canvas, PlotArea a ViewPort

Na obrazovce je možné otevřít okno, které se v JAPI nazývá *frame*. Takových oken je možné současně otevřít více přičemž každé z nich může být skryté nebo viditelné (zobrazené). Do okna se umístí kreslicí plocha – *canvas* – a na ní se ještě může definovat oblast pro nakreslení vlastního grafu, kterou nazývám *PlotArea*. Všem těmto prvkům společně budeme říkat *ViewPort*, znázorněný na obr. 2.

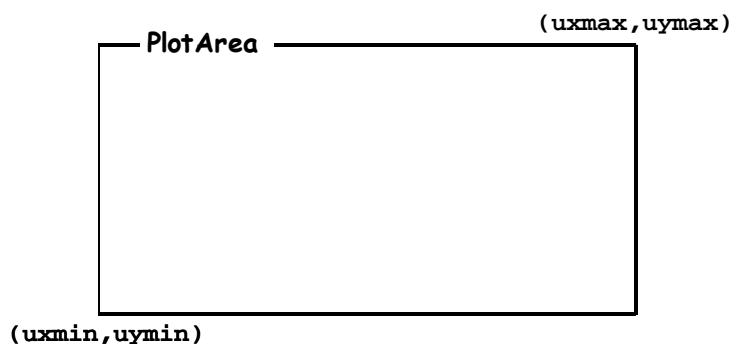


Obr. 2

ViewPort je vymezený oknem (*frame*); v okně je umístěna kreslicí plocha (*canvas*) na ní vybereme oblast (*PlotArea*) pro kresbu grafu. Vše se nastaví voláním procedury *DefViewPort*.

Uživatelské souřadnice

Uživatelské souřadnice jsou typu *REAL* a zavedou se pro *PlotArea* podle obr. 3, použitelné jsou však (rozšířené) na celé *canvas*.



Obr. 3

Pro určení polohy v uživatelských souřadnicích se zavádí *strukturovaná proměnná Ubod*:

```
TYPE,PUBLIC :: Ubod
    REAL(KIND=DP) :: x, y
END TYPE Ubod
```

Poloha bodu v pixelech se zadá analogicky definovanou proměnnou **Pbod**:

```
TYPE,PUBLIC :: Pbod
    INTEGER :: px, py
END TYPE Pbod
```

Pro práci s oběma typy jsou definované *operátory +, -*.

Strukturované proměnné pro popis ViewPortu

Úplný popis všech složek ViewPortu je ve strukturované proměnné typu **ViewPortInfo**, kterou nastaví procedura DefViewPort:

```
TYPE,PUBLIC :: ViewPortInfo
    INTEGER :: frame, canvas
    TYPE(UPinfo) :: canvasUPinfo, PlotAreaUPinfo
    TYPE(rgbcol) :: canvasRGBcolor, PlotAreaRGBcolor
    INTEGER :: pix2ux, pix2uy
END TYPE ViewPortInfo
```

kde jsou použity další strukturované typy

```
TYPE,PUBLIC :: UPinfo
    TYPE(userCO) :: Uinfo
    TYPE(objCO) :: Pinfo
END TYPE UPinfo

TYPE,PUBLIC :: userCO
    REAL(KIND=DP) :: uxmin, uxmax, uymin, uymax
END TYPE userCO

TYPE,PUBLIC :: objCO
    INTEGER :: polx, poly, psirka, pvyska
END TYPE objCO
```

Proměnná typu UPinfo zřejmě obsahuje informaci o uživatelských souřadnicích podle obr. 3 a proměnná typu objCO v pixelech vyjádřenou polohu levého horního rohu oblasti plus jeho šířku a výšku.

Příklad:

deklarujeme proměnnou

```
TYPE(ViewPortInfo) :: infVP
```

kterou použijeme jako výstupní parametr při volání procedury

```
CALL DefViewPort(. . . , infVP).
```

V této proměnné máme poukončení procedury všechny potřebné informace o definovaném ViewPortu, takže např. (jestliže předcházela deklarace: INTEGER :: frame, canvas)

```
frame = infVP%frame,
canvas = infVP%canvas
```

přiřadí do frame (canvas) odkaz (tzv. *handler*) na právě definovanou *frame* (*canvas*); tyto odkazy budeme potřebovat jako parametr (zpravidla první) procedur, které s těmito objekty pracují.

Příklady jiných veličin, které lze získat z infVP:

výraz	veličina
infVP%canvas%canvasUPinfo%Uinfo nebo ¹ canvas%canvasUPinfo%Uinfo	uxmin, uxmax, uymin, uymax PRO canvas!!! TYPE(userCO)
infVP%canvas%PlotAreaUPinfo%Uinfo nebo ¹ canvas%PlotAreaUPinfo%Uinfo	uxmin, uxmax, uymin, uymax PRO PlotArea!!! (zadané v DefViewPort)
canvas%PlotAreaUPinfo%Uinfo%uxmin	uxmin pro PlotArea

Další příklady najdete v doprovodném programu T_navod1.f95.

Uživatelské typy doplněné ve verzi 2.4

Ve verzi 2.4 byly doplněny následující typy, jejichž použití najdete např. v T_navod5.f95:

Pro *informace o objektu v normálních souřadnicích* (frame, canvas, PlotArea); význam stejný jako typ objCO, poloha a rozměr objektu je v normálních souřadnicích místo v pixelech.

```
TYPE, PUBLIC :: NobjCO
REAL(DP) :: Npolx, Npoly, Nsirka, Nvyska
END TYPE NobjCO
```

Pro informace o *frame* typ

```
TYPE, PUBLIC :: FrameInfo
INTEGER :: frame ! handler na frame
CHARACTER(len=255) :: NadpisFR ! nadpis okna
TYPE(objCO) :: PmiryFR ! polx, poly, psirka, pvyska
TYPE(rgbc01) :: BgbarvaFR ! barva pozadí
END TYPE FrameInfo
```

Proměnnou tohoto typu vrátí funkce

```
FUNCTION NastavFrame(FRnobjCO, FRnadjpis, FRbarvaBG) RESULT(infFR)
TYPE(NobjCO), INTENT(IN) :: FRnobjCO ! normalni souradnice
CHARACTER(len=*), INTENT(IN) :: FRnadjpis
TYPE(rgbc01), INTENT(IN) :: FRbarvaBG
TYPE(FrameInfo) :: infFR
```

¹ Předpokládáme existenci výše uvedené proměnné canvas.

Pro informace o *canvas* typ

```

TYPE,PUBLIC :: CanvasInfo
    INTEGER :: canvas
    TYPE(objCO) :: PmiryC,PmiryPA      !
    TYPE(userCO) :: UmezeC,UmezePA    ! uzivatelske souradnice
    TYPE(rgbc1) :: BGbarvaC,BGbarvaPA
    INTEGER :: pix2ux,pix2uy
END TYPE CanvasInfo

```

Pro informace o *PlotArea* (použijí se v následující funkci *NastavCanvas*)

```

TYPE,PUBLIC :: PlotAreaInfo
    INTEGER :: PAleft,PARight,PAbottom,PAtop
END TYPE PlotAreaInfo

```

Proměnnou typu *CanvasInfo* vrací funkce

```

FUNCTION NastavCanvas(frame,CNobjCO,CUmeze,CbarvaBG,CinfPA,Pabarva) &
RESULT(infC)
    ! umožní nastavení více canvas na jednom frame
    ! poloha a velikost v normálních souřadnicích, které se
    ! nyní nevztahují k celé obrazovce, ale k frame !!!
    INTEGER,INTENT(IN) :: frame
        ! handler na frame
    TYPE(NobjCO),INTENT(IN) :: CnobjCO
        ! poloha a velikost canvas v normálních souřadnicích
    TYPE(userCO),INTENT(IN),OPTIONAL :: Cumeze
        ! uživatelské meze pro canvas, DEFAULT:<0.0,100.0>,<0.0,100.0>
    TYPE(rgbc1),INTENT(IN),OPTIONAL :: CbarvaBG, Pabarva
        ! barva pozadí canvas a PlotArea, DEFAULT: bílá
    TYPE(PlotAreaInfo),INTENT(IN),OPTIONAL :: CinfPA
        ! poloha PA na canvas (v pixelech), DEFAULT: PlotArea = canvas
        ! je-li zadané, použijí se uživatelské souřadnice (Cumeze) pro
        ! PlotArea a pro canvas se vypočítají odpovídající nové
    TYPE(CanvasInfo) :: infC

```

Jestliže máme pro zvolené *frame* proměnnou *infFR* a pro všechna canvas na něm *infC1*, *infC2*,... můžeme pro kombinace *FR+Cx* sestavit proměnné *infVPx* pomocí funkce

```

FUNCTION SestavVPinfo(infFR,infC) RESULT(infVP)
    TYPE(FrameInfo),INTENT(IN) :: infFR
    TYPE(CanvasInfo),INTENT(IN) :: infC
    TYPE(ViewPortInfo) :: infVP

```

S takto získanými proměnnými *infVPx* (*x=1,2,...*) pak můžeme volat všechny procedury modulu *Japigraf*, které tento parametr vyžadují.

Modul rgbcolor

K modulu Japigraf existuje ještě doprovodný modul rgbcolor, který zavádí *typ rgbcol* :

```
TYPE,PUBLIC :: rgbcol
    INTEGER :: r,g,b
END TYPE rgbcol
```

a definuje standardní názvy barev doprovázejících v souboru rgb.txt unixovské instalace X windows² (v modulu jsou pouze varianty bez mezer v názvu) takto:

```
TYPE(rgbcol),PARAMETER,PUBLIC :: snow = rgbcol(255,250,250)
TYPE(rgbcol),PARAMETER,PUBLIC :: GhostWhite = rgbcol(248,248,255)
    :                               :                               :
TYPE(rgbcol),PARAMETER,PUBLIC :: DarkMagenta = rgbcol(139,0,139)
TYPE(rgbcol),PARAMETER,PUBLIC :: DarkRed = rgbcol(139,0,0)
```

Zahájení práce v prostředí JAPI + Japigraf

V úvodu programu použijeme: USE JAPI, USE Japigraf, USE rgbcolor.

Při kompilaci do *exe-souboru* nesmíme zapomenout uvést knihovny:

-ljapi -ljapigraf (předpokládáme, že s klíčem -L jsme také uvedli, kde tyto knihovny jsou); kdyby následující procedura StartJapi() hlásila chybu, zkuste doplnit knihovnu -lwsock32.

Procedura StartJapi()

se musí volat jako první; zkontroluje prostředí pro práci JAPI a nastaví PUBLIC proměnné:

AinfVP	proměnná infVP z posledního volání procedury DefViewPort
S_width	šířka obrazovky v pixelech (podle nastaveného rozlišení)
S_height	výška obrazovky v pixelech (rovněž podle rozlišení)
AspRatio	poměr S_width/S_height
PAleft, PARight, PATop, PAbottom	veličiny určující polohu PlotArea podle obr. 2 nastaví na hodnoty (v uvedeném pořadí) : 80, 10, 30,40 pixelů

Pokud není nastavení prostředí v pořádku, vypíše “can’t connect to JAPI server” a ukončí další práci s JAPI.

Proměnné PAleft, PARight, PATop, PAbottom jsou veřejně přístupné (PUBLIC) takže je možné je programově změnit před následujícím voláním procedury DefViewPort.

Po úspěšném provedení procedury StartJapi() můžeme pokračovat několika způsoby:

² Např. v systému cygwin je v adresě usr\X11R6\X11.

Vzorník je např. na <http://www.pitt.edu/~nisdg/cis/web/cgi/rgb.html>

Procedura NastavStdGraf

rychle nastaví prostředí pro kresbu XY-grafu

```

SUBROUTINE NastavStdGraf(infoVP,UodX,UdoX,UodY,UdoY,VPnadpis,&
PlotOx,PlotOy,Velikost,KrokX, KrokY,DilkuX,DilkuY,&
PopisX,PopisY,Nadpis,Grid)
  TYPE(ViewPortInfo),intent(out) :: infoVP
    ! informace o ViewPortu; VYSTUPNI parametr potrebný jako
    ! vstupni parametr pri volani dalsich procedur z Japigrafu
  REAL(KIND=DP),INTENT(IN) :: UodX, UdoX, UodY, UdoY
    ! hranicni hodnoty pro osu X a Y (uzivatelske jednotky)
  CHARACTER(len=*),INTENT(IN) :: VPnadpis
    ! nadpis v zhlavi frame (okna)
  LOGICAL,INTENT(IN),OPTIONAL :: PlotOx, PlotOy
    ! kresba osy X a Y; .true./.false. = ano/ne
  INTEGER,INTENT(IN),OPTIONAL :: Velikost
    ! velikost frame, Velikost z <2,9>, DEFAULT: 5 (1/2 obrazovky)
  REAL(KIND=DP),INTENT(IN),OPTIONAL :: KrokX, KrokY
    ! krok deleni osy X a Y ("velke" dilky) v uziv. jednotkach
  INTEGER,INTENT(IN),OPTIONAL :: DilkuX, DilkuY
    ! pocet "malych" dilku ("malych" carek je DilkuX-1(DilkuY-1)
  CHARACTER(len=*),INTENT(IN),OPTIONAL :: PopisX, PopisY, Nadpis
    ! popis osy X, osy Y a nadpis grafu, DEFAULT: ""
  LOGICAL,INTENT(IN),OPTIONAL :: Grid
    ! kresba mriszky ve "velkych" dilcich

```

Procedura DefViewPort

nastaví *ViewPort* (*frame+canvas+PlotArea*) a vrátí proměnnou *infVP*, která je třeba pro volání většiny ostatních procedur; osy, popisy atd. musíme namalovat sami.

```

SUBROUTINE DefViewPort(NposxFR,NposyFR,nadpisFR,&
NsirkaPA,NvyskaPA,SetAspRatio,&
UxMin,UxMax,UyMin,UyMax,&
canvasCol,PlotAreaCol,&
VPinfo)
  REAL(KIND=DP),INTENT(IN) :: NposxFR,NposyFR,NsirkaPA,NvyskaPA
    ! poloha levého horního rohu frame v normálních souřadnicích
    ! (NposxFR,NposyFR) a šířka a výška PlotArea v normálních
    ! souřadnicích (NsirkaPA,NvyskaPA); canvas se uloží na celou
    ! využitelnou plochu frame, poloha PlotArea na canvas je
    ! určena PUBLIC proměnnými PLeft,PAright,PAbottom,PATop,
    ! které je možné před voláním procedury změnit
  CHARACTER(len=*),INTENT(IN) :: nadpisFR
    ! nadpis v záhlaví okna (frame)
  LOGICAL,INTENT(IN) :: SetAspRatio
    ! pro .true. zkrátí delší osu tak, aby při kresbě „kruh byl
    ! kruh“
  TYPE(rgbcol),INTENT(IN) :: canvasCol,PlotAreaCol
    ! barva canvas a PlotArea
  REAL(KIND=DP),INTENT(IN) :: UxMin, UxMax, UyMin, UyMax
    ! hraniční hodnoty pro PlotArea, rozšíří se na canvas
  TYPE(ViewPortInfo),INTENT(OUT) :: VPinfo

```

S návratovou proměnnou *VPinf* je nyní možné volat ostatní procedury a funkce a kreslit na vytvořené *canvas*.

Doplnění a aktivace standardního menu na *ViewPort*

Menu se na *ViewPort* dodá procedurou

```
SUBROUTINE AddStdMenu(infVP)
      TYPE(ViewPortInfo),INTENT(IN) :: infVP
```

Menu potom obsahuje položky:

- *Soubor / Konec*
(ukončí funkci menu aktivovanou následující procedurou *StartStdMenu*)
- *Tisk / Canvas*
(vytiskne *canvas* na zvolené tiskárně)
- *Graf / Uložit graf jako BMP a Uložit graf jako PPM*
(uloží do souboru *canvas* v zadaném formátu)

Menu je funkční po zavolání procedury

```
SUBROUTINE StartStdMenu(infVP)
      TYPE(ViewPortInfo),INTENT(IN) :: infVP
```

Procedura způsobí, že program v časově neomezené smyčce čeká na stisknutí některé položky menu. Ze smyčky vystoupí až po stisku položky *Soubor/Konec*.

Proceduru lze v programu volat opakovaně; je-li volána jako poslední příkaz programu, položka *Soubor/Konec* ukončí program.

Jestliže máme na jednom *frame* více *canvas* a k nim pomocí funkce *SestavVPinfo* získáme proměnné *infVP1*, *infVP2*, ..., budou položky menu *Tisk*, *Graf* fungovat pro *canvas* spojené s proměnnou *infVPx*, která je uvedena jako argument při volání *StartStdMenu*.

Nezávislé vytvoření *frame* a *canvas*, sestavení proměnné *infoVP*

provedeme pomocí funkcí *NastavFrame* (na na strani 4), *NastavCanvas* (na na strani 5), které umožní uložit na jedno *frame* více *canvas*. Ze získaných informačních proměnných *infFR*, *infC1*, *infC2*,... se pak vytvoří pomocí funkce *SestavVPinfo* (na na strani 5) proměnné *infVP1*, *infVP2*,... potřebné při volání většiny procedur z modulu *Japigraf*. Tento postup podrobně ilustruje doprovodný program *T_navod5.f95*.