# X36PJC
# Proseminář #1

# Překlad zdrojových kódů a Makefile

# Úvod

- Rozsah proseminářů
  1) Překlad zdrojových kódů a Makefile
  2) Ladění a profilování aplikace
  3) Unit testing v C/C++
  4) Ladění práce s pamětí
  5) STL + konzultace
  6) STL + konzultace

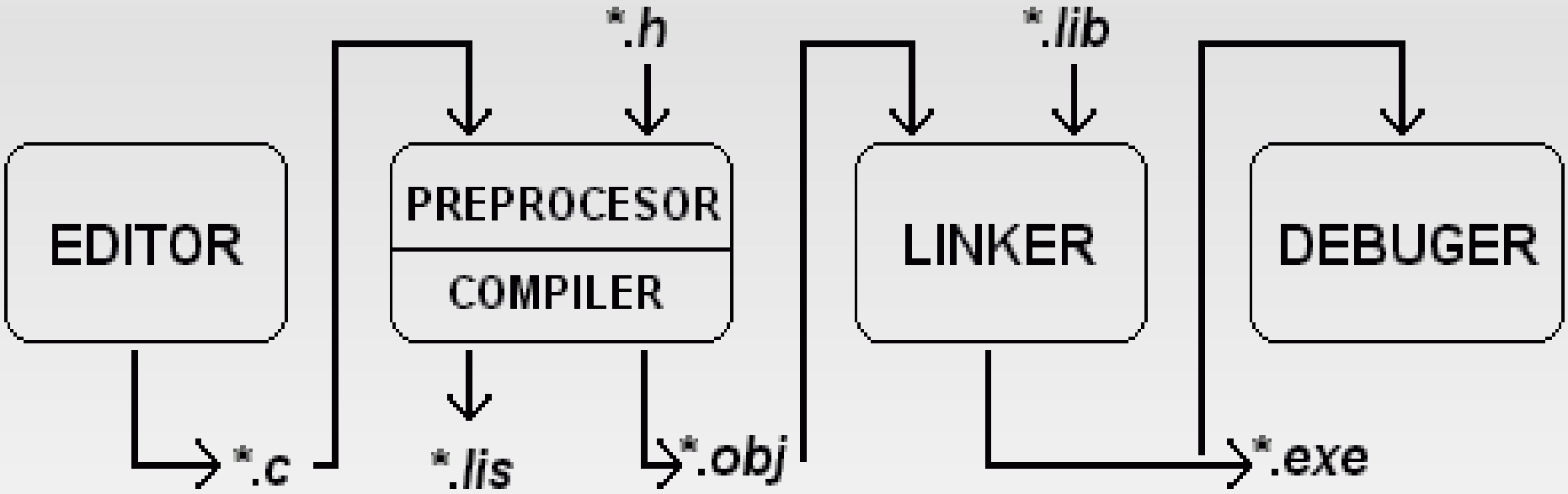
# Osnova cvičení

- Seznámení s editací a překladem na referenční platformě
- Podrobnější seznámení s gcc/g++
- Kompilace z více zdrojových kódů
- Kompilace pomocí make

# Překlad zdrojového kódu I.

- Kompilery používané na X36PJC
  - *gcc* / *g*++ ve verzi 3.4.3 pro Solaris OS
- **Solaris je referenční prostředí!**
- Free alternativy pro Windows
  - *gcc / g++ + cygwin* - www.cygwin.com
  - *Mingw + DevC++* - bloodshed-dev-c.en.softonic.com
  - *Microsoft Visual C++ 2010 Express*
    www.microsoft.com/express/Downloads

# Překlad zdrojového kódu II.



# Překlad zdrojového kódu III.

- Hierarchie souborů při překladu
  - „*.c / *.cc / *.cpp / *.h“ - Soubory zdrojového kódu
  - „*.m“ - Soubor zdrojového kódu zpracovaný preprocesorem
  - „*.s“ - Reprezentace kódu v assembleru
  - „*.o“ - Objektový soubor, binární reprezentace, která je již charakteristická pro danou architekturu
  - „*.dll / *.lib / *.so / *.a“ - Dynamické a statické knihovny
  - „* / *.exe“ - Výsledný spustitelný soubor

# Hello world

```
// Hello.c

#include <stdio.h>

int main(){
    printf("Hello world!\n");
    return 0;
}
```

# Editace a překlad Hello.c

- Editace
  - #>pico Hello.c
  - Ctrl+o – uloží soubor
  - Ctrl+x – ukončí editor
  - Nebo použít Midnight commander
- Překlad
  - #>gcc -o hello Hello.c
  - Nebo pomocí make
- Spuštění
  - #>./hello

# Užitečné parametry pro gcc/g++ I.

- „-o *jméno*“ - jméno výstupní aplikace
- „-c“ - vytvoří objektový soubor
- „-S“ - vyvoří assemblerový kód
- „-Wall“ - (= Warnings all) zajistí, že při překladu budou zobrazena všechna varování
- „-pedantic“ - přísnější kontrola při překladu
- „-o[1,2,3]“ - optimalizace při překladu, čísla reprezentují stupeň optimalizace

# Užitečné parametry pro gcc/g++ II.

- „-g“ - do kompilovaného kódu vloží informace pro debugger
- „-D[název makra [= hodnota]]“ - definuje makro během překladu
- „-l[název dynamické knihovny]“ - přidá k výslednému kódu dynamickou knihovnu
- „-L[cesta k adresáři]“ - přidá překladači další adresář, ve kterém bude hledat dynamické knihovny

# Užitečné parametry pro gcc/g++ III.

- „-I[cesta k adresáři] – (pozor jedná se o velké i!) přidá překladači adresář, ve kterém budou hledány hlavičkové soubory

# Kompilace z více souborů I.

**PrintHello.h:**

```
#include <stdio.h>

void printHello();
```

---

**PrintHello.c:**

```
#include „PrintHello.h“

void printHello(){
    printf(„Hello „);
}
```

# Kompilace z více souborů II.

**PrintWorld.h:**

```
#include <stdio.h>

void printWorld();
```

---

**PrintWorld.c:**

```
#include „PrintWorld.h“

void printWorld(){
    printf(„world!\n„);
}
```

# Kompilace z více souborů III.

**main.c:**

```
#include „PrintHello.h“
#include „PrintWorld.h“

int main(){
    printHello();
    printWorld();

    return 0;
}
```

# Kompilace z více souborů IV.

- #>gcc -o hello main.c PrintHello.c PrintWorld.c
  - Přeloží dané zdrojové soubory „najednou“ do aplikace „hello“

---

- #>gcc -c *.c
  - Vytvoří příslušné objektové soubory main.o, PrintHello.o a PrintWorld.o
- #>gcc -o hello *.o
  - Linkuje objektové soubory do výsledné aplikace

# Překlad pomocí make I.

- Jedná se o mechanizmus pro zjednodušení překladu, zvláště u projektů s více zdrojovými soubory
- Příkaz „make“ je obsažen v systému Solaris
- Informace o způsobu překladu jsou uloženy v souboru „Makefile“
- Při správném použití zrychlí překlad
- Umožňuje jednoduše kompilovat různé konfigurace
- **Makefile bude vyžadováno u každého odevzdaného kódu nebo projektu!**

# Překlad pomocí make II.

- Uvnitř Makefile můžeme:
  - Definovat proměnné

    *NAZEV_PROMENNE = hodnota_promenne*
  - Definovat jednotlivá pravidla

    *cil: cil1 cil2 cil3*

    *akce*
    - *cil* je vlastně název pravidla, ale může označovat i výsledný soubor
    - *cil1, cil2* … jsou cile, na kterých ten aktuální závisí
    - *akce* jsou příkazy daného pravidla

# Překlad pomocí make III.

- Další speciality Makefile
  - Cíl „all“ je volán automaticky při spustění make, je vhodné za ním uvést všechna pravidla, které chceme, aby se provedla defaultně
  - Cíl „.PHONY“ má v závislostech uvedeny názvy všech pravidel, které nemají jako cíl soubor
  - Pro určitý typ cílů lze zavést univerzální pravidlo:

    *%.o: %.c*

    *gcc -c $<*

    % - zástupný znak podobný např. Jako *

    $< - způsobí vložení všech cílů, na kterých je aktuální cíl závislý

# Překlad pomocí make IV.

- Makefile umožňuje provést v těle pravidel prakticky jakoukoliv sadu příkazů. Lze tedy přímo z makefile přeloženou aplikaci rovnou spustit.
- Podle konvencí je vhodné v Makefile vyrobit cíl s názvem clean, který vrátí stav adresáře projektu do stavu před začátkem kompilace

# Použité zdroje a užitečné odkazy I.

- http://www.jazykc.ic.cz
  - Portál věnovaný jazyku C
- http://www.linuxsoft.cz
  - Seriál o C/C++
- http://www.root.cz
  - Seriál o C/C++
- http://www.cplusplus.com
  - Referenční příručka online

# Použité zdroje a užitečné odkazy II.

- http://www.cppreference.com
  - Referenční příručka online
- http://www.gnu.org/software/make/manual/make.html
  - Manuál k make a Makefile