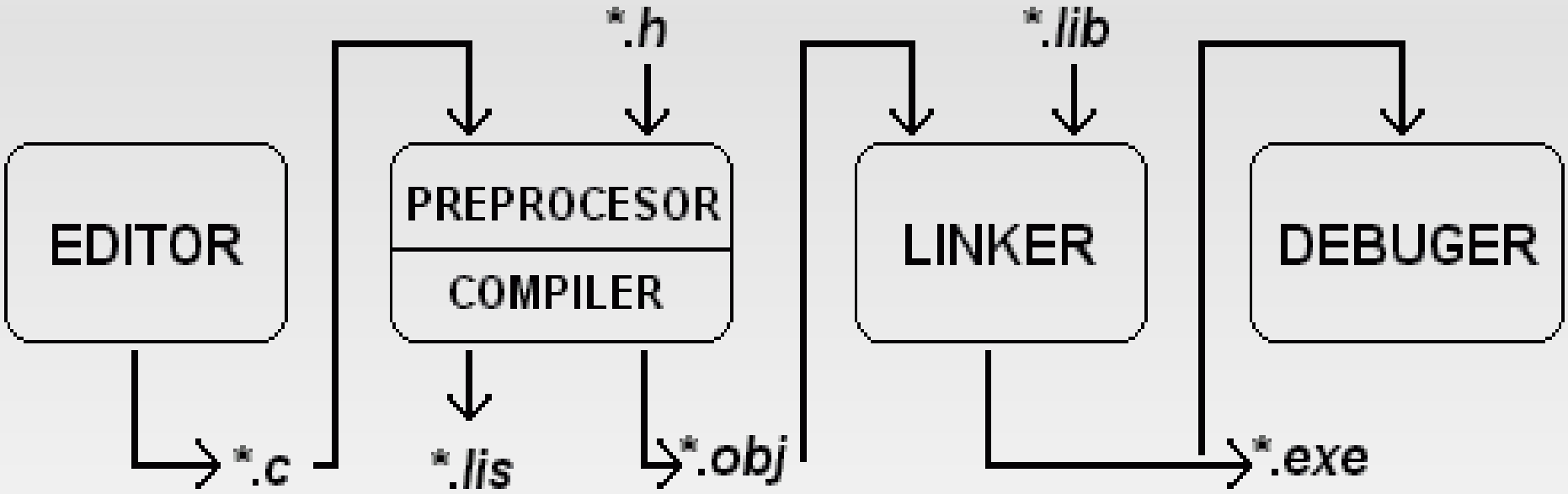
PJC
Cvičení #1

Překlad zdrojových kódů a Makefile

Překlad zdrojového kódu I.

- Kompilery používané na A7B36PJC
  - *gcc / g++*
- Free alternativy pro Windows
  - *gcc / g++ + cygwin* - www.cygwin.com
  - *Mingw - www.mingw.org*
  - *Microsoft Visual C++ 2010 Express*
  - *Borland – edn.embarcadero.com*
  - *Intel - software.intel.com*

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
- Jednotlivé výstupy je možno shlédnout v přiložené ukázce HelloWorld-simple

Váš první program v C++

- Nebo zkusme raději něco snazšího...

Hello world

```
// Hello.cpp

#include <iostream>

int main(){
   std::cout << "Hello world!" << std::endl;
   return 0;
}
```

Editace a překlad Hello.cpp

- Editace v příkazové řádce
  - #>pico Hello.cpp
  - Ctrl+o – uloží soubor
  - Ctrl+x – ukončí editor
  - Nebo použít Midnight commander
- Překlad v příkazové řádce
  - #>g++ -o hello Hello.cpp
  - Nebo pomocí make
- Spuštění v příkazové řádce
  - #>./hello

Užitečné parametry pro gcc/g++ I.

- „-o *jméno*“ - jméno výstupní aplikace
- „-c“ - vytvoří objektový soubor
- „-S“ - vytvoří 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

- Větší projekty se bez toho neobejdou

Kompilace z více souborů I.

PrintHello.h:

```
#include <iostream>

void printHello();
```

PrintHello.cpp:

```
#include „PrintHello.h“

void printHello(){
   std::cout << „Hello „ ;
}
```

Kompilace z více souborů II.

PrintWorld.h:

```
#include <iostream>

void printWorld();
```

PrintWorld.cpp:

```
#include „PrintWorld.h“

void printWorld(){
   std::cout << „world!„ << std::endl;
}
```

Kompilace z více souborů III.

main.cpp:

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

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

  return 0;
}
```

Kompilace z více souborů IV.

- #>g++ -o hello main.cpp PrintHello.cpp PrintWorld.cpp
  - Přeloží dané zdrojové soubory „najednou“ do aplikace „hello“

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

Proces kompilace je náročný

- Je třeba jej trochu zjednodušit...

Překlad pomocí make I.

- Jedná se o mechanizmus pro zjednodušení překladu, zvláště u projektů s více zdrojovými soubory
- Příkaz „make“ je obsažen v každém unixovém systému.
- Informace o způsobu překladu jsou uloženy v souboru „Makefile“
- Při správném použití zrychlí překlad
- Umožňuje jednoduše kompilovat různé konfigurace
- **Makefile bude vyžadováno v odevzdaných semestrálních pracích!**

Překlad pomocí make II.

```
all: hello

.PHONY: clean

hello: main.o PrintHello.o PrintWorld.o
   g++ main.o PrintHello.o PrintWorld.o -o hello

main.o: main.cpp
   g++ -c main.cpp

PrintHello.o: PrintHello.cpp
   g++ -c PrintHello.cpp

PrintWorld.o: PrintWorld.cpp
   g++ -c PrintWorld.cpp

clean:
   rm -rf *.o hello
```

Překlad pomocí make III.

- 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 IV.

- Pozor! Před akcí za uvedenými cíli musí být jeden tabulátor!
- 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

Překlad pomocí make V.

- Makefile umožňuje provést v těle pravidel prakticky jakoukoliv sadu příkazů. Lze tedy přímo z makefile přeloženou aplikaci rovnou spustit nebo testovat nějakým vstupem.
- Podle konvencí je vhodné v Makefile vyrobit cíl s názvem clean, který vrátí stav adresáře projektu do stavu před začátkem kompilace
- V přiložené ukázce HelloWorld-multiple je názorná demonstrace funkčnosti

Vývojová prostředí

- Code::Blocks
  - www.codeblocks.org
- Eclipse + CDT
  - www.eclipse.org
- NetBeans
  - netbeans.org
- DevCpp
  - www.bloodshed.net
- Microsoft Visual C++
  - www.microsoft.com

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