KIV/CPP – Programování v jazyce C++

A. Práce s knihovnami (extra)

Martin Úbl

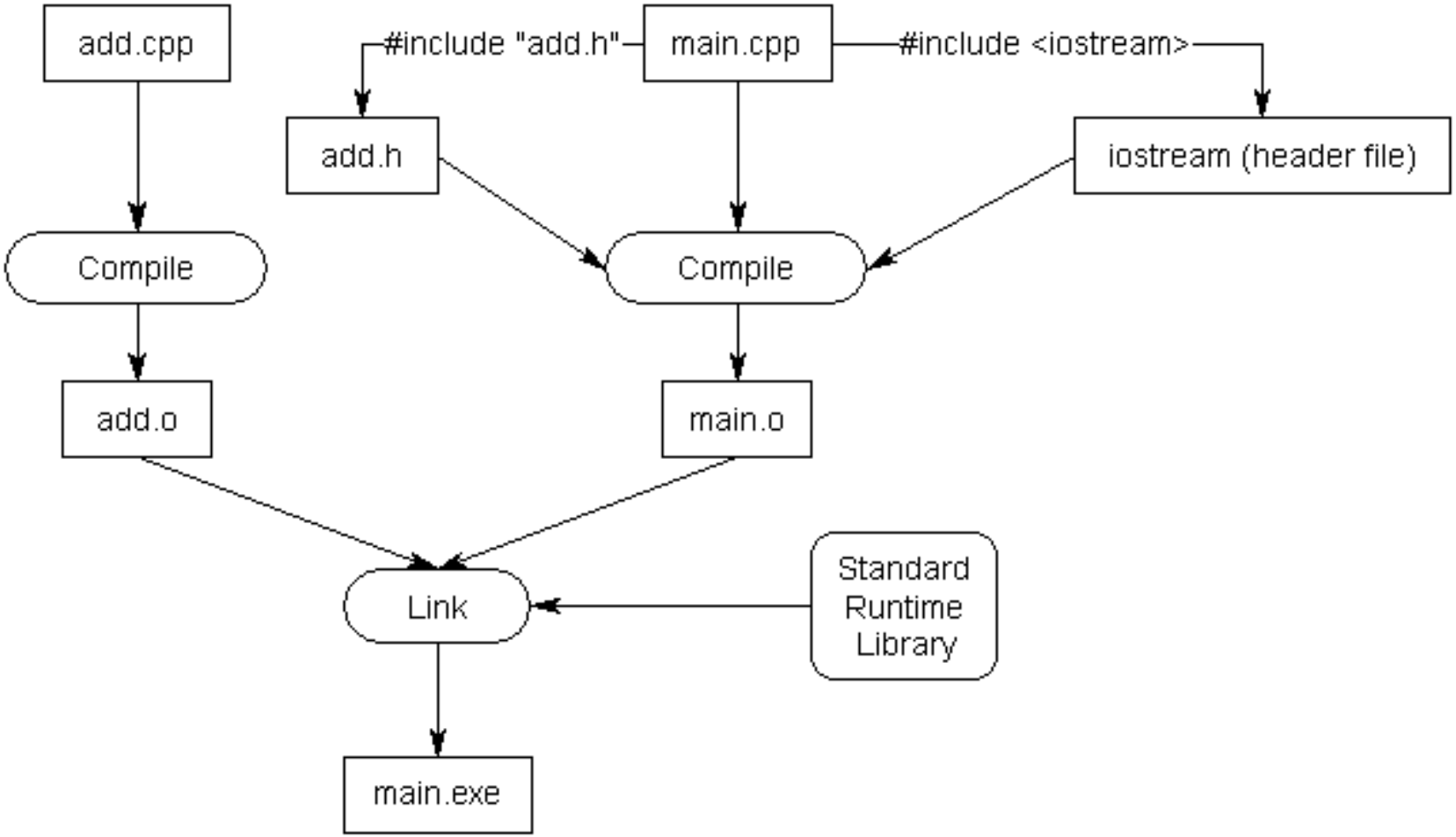
KIV ZČU

2022/2023

Práce s knihovnami

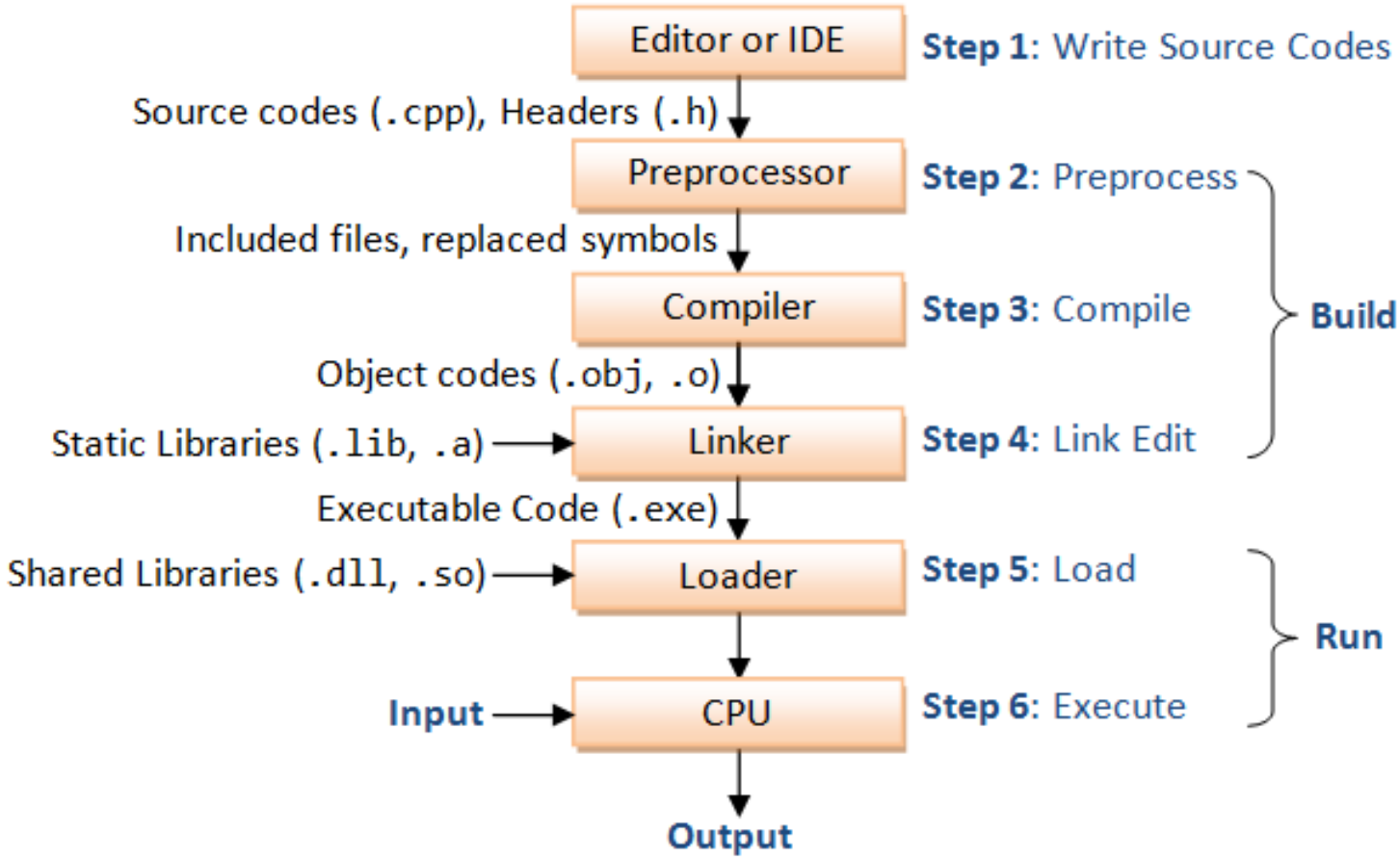
- knihovny
  - zakompilované
    - s kompilovanými moduly
    - „hlavičkové“ - pouze v .h / .hpp souboru
  - statické (staticky linkované)
    - .lib (MS Windows)
    - .a (GNU/Linux, macOS)
  - dynamické (dynamicky linkované)
    - .dll (MS Windows)
    - .so (GNU/Linux)
    - .dylib (macOS)

Práce s knihovnami



Obrázek: Znázornění procesu kompilace a (statického) linkování

Práce s knihovnami



Obrázek: Znázornění procesu kompilace a linkování

Práce s knihovnami

- statické knihovny
- musí být přítomny v čase kompilace
- již jsou zkompilované
- obsahují zatím jen symbolicky adresovaný kód (ne konkrétní adresy)
- linker je v čase kompilace přikompiluje k sestavované aplikaci
- musí být kompatibilní s daným kompilátorem
- dodavatel typicky poskytuje zkompilovanou .lib/.a knihovnu a hlavičkové soubory

Práce s knihovnami

- dynamické knihovny
- musí být přítomny v čase běhu (spouštění)
- načítání
  - automatické (dynamic sekce spustitelného souboru)
  - ruční (dlopen, LoadLibrary, ...)
- knihovny musí symboly tzv. exportovat
  - na MS Windows explicitně
  - na GNU/Linux / macOS jsou exportované všechny

Práce s knihovnami

- dynamické knihovny
- C++ - name mangling
- „serializace“ jména (symbolu)
- často přidá podtržítka, čísla, další znaky
- unikátnost

Práce s knihovnami

```
martin@rattmann:~/test2$ nm a.out
0000000000004058 B __bss_start
0000000000004170 b completed.7389
                 U __cxa_atexit@@GLIBC_2.2.5
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000004040 D __data_start
0000000000004040 W data_start
00000000000010c0 t deregister_tm_clones
0000000000001130 t __do_global_dtors_aux
00000000000003db0 t __do_global_dtors_aux_fini_array_entry
0000000000004048 D __dso_handle
0000000000004050 V DW.ref.__gxx_personality_v0
                                                ...
0000000000004178 B _end
0000000000001354 T _fini
0000000000001175 T main
00000000000010f0 t register_tm_clones
0000000000001090 T _start
0000000000004058 D __TMC_END__
                 U _Unwind_Resume@@GCC_3.0
                 U _ZNSolsEi@@GLIBCXX_3.4
                 U _ZNSolsEPFRSoS_E@@GLIBCXX_3.4
                 U _ZNSt8ios_base4InitC1Ev@@GLIBCXX_3.4
                 U _ZNSt8ios_base4InitD1Ev@@GLIBCXX_3.4
0000000000001264 W _ZN11NejakaTridaC1Ev
0000000000001264 W _ZN11NejakaTridaC2Ev
0000000000001270 W _ZN11NejakaTridaD1Ev
0000000000001270 W _ZN11NejakaTridaD2Ev
000000000000128a W _ZN11NejakaTridapLERKi
000000000000127c W _ZN11NejakaTrida12NejakaMetodaEi
0000000000002004 r _ZStL19piecewise_construct
0000000000004171 b _ZStL8__ioinit
0000000000004060 B _ZSt4cout@@GLIBCXX_3.4
                 U _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@@GLIBCXX_3.4
0000000000001205 t _Z41__static_initialization_and_destruction_0ii
000000000000129c W _Z9MojePrintIiEvRKT_
```

Obrázek: Name mangling C++ (výpis nástroje nm)

Práce s knihovnami

- zamezení name manglingu
- `extern "C"`

```
extern "C" void perform_magic(int param1);

extern "C" unsigned long long SharedCounter;
```

Práce s knihovnami

- přes rozhraní dynamických knihoven by měly jít pouze POD typy
- dynamické knihovny by měly být nezávislé na jazyce
- nemáme jistotu, že hostitelská aplikace např. psaná v Pascalu bude mít binárně kompatibilní implementaci `std::vector`
- problém na jiné platformě
- problém i s jinou verzí téhož kompilátoru (standardní knihovny)

Práce s knihovnami

- export symbolů
- na GNU/Linux a macOS jsou exportované všechny symboly
- na Windows je nutné explicitně exportovat
  - `_declspec(dllexport)`
  - .def soubor (specifikuje se kompilátoru)
  - a další.. (export přepínač, pragma)

Práce s knihovnami

- `_declspec(dllexport)`

```
extern "C" _declspec(dllexport)
       void perform_magic(int param1);
```

- pokud linkujeme automaticky, může být v definici (hlavičkový soubor) pro hostitelskou aplikaci uveden `_declspec(dllimport)`
- jde o optimalizaci pro linkování

Práce s knihovnami

- .def soubor (specifikuje se kompilátoru)

```
LIBRARY magic.dll

EXPORTS
    perform_magic
    SharedCounter
```

Práce s knihovnami

- import symbolů
  - automaticky (generovaná statická knihovna se „spojkami“ + hlavičkový soubor)
  - ručně

Práce s knihovnami

- import symbolů automaticky
- kompilátor při překladu vytvoří malou statickou knihovnu
- tam se nachází resolvery funkcí z knihovny
- hostitelská aplikace přilinkuje statickou knihovnu
- příslušný modul includuje hlavičkový soubor
- dynamická knihovna je načtena při spuštění programu
- programátor se nestará o import symbolu, „prostě volá“ funkce co byly definované v hlavičkovém souboru
  - v malé statické knihovně jsou funkce s těmito jmény
  - nemají ale požadovanou funkci, pouze „návod“, jak najít a zavolat funkci z dynamické knihovny
  - viz GOT/PLT v KIV/OS

Práce s knihovnami

- import symbolů ručně
- je vytvořena jen dynamická knihovna
- sada funkcí pro hostitelskou aplikaci
  - Windows / GNU/Linux+macOS
  - `LoadLibrary` / `dlopen` - otevření knihovny
  - `GetProcAddress` / `dlsym` - načtení symbolu
  - `FreeLibrary` / `dlclose` - uzavření knihovny

Práce s knihovnami

- GetProcAddress / dlsym - načtení symbolu
- vyzvedne pouze adresu
- musíme znát signaturu funkce (např. nějaký hlavičkový soubor knihovny)
- adresa je validní, dokud nezavoláme `FreeLibrary / dlclose`

Práce s knihovnami

```
using MagicFunc = void(*)(int);

auto lib = LoadLibrary("magic.dll");

MagicFunc perform_magic =
    reinterpret_cast<MagicFunc>(
        GetProcAddress(lib, "perform_magic")
    );

perform_magic(42);

FreeLibrary(lib);
```

Práce s knihovnami

- Volací konvence
- „dohoda“ toho, jak se volá funkce/metoda
  - jak se předávají parametry
  - jak se předává návratová hodnota
  - kdo „uklidí“ zásobník
- Například:
  - `cdecl` (x86) - parametry přes zásobník zprava doleva, návratová hodnota v (E)AX/ST0 registru, ...
  - `stdcall` (x86) - podobná, jen volaná funkce uklízí zásobník
  - x64 (x86_64) - parametry částečně registry, částečně zásobníkem, „shadow“ místo v zásobníku před rámcem volání, ... (MS/*nix řeší jinak)
- `https://www.agner.org/optimize/calling_conventions.pdf`
- volací konvenci možno uvést v signatuře funkce

Práce s knihovnami

- objektové knihovní konvence
- např. COM model
- objekty vždy dědí od `IUnknown` rozhraní
  - první tři položky v tabulce virtuálních metod jsou `AddRef`, `Release` a `QueryInterface`
- každá COM třída může podporovat další rozhraní
- rozhraní identifikovány pomocí GUID
- lze zavolat `QueryInterface` nad objektem s daným GUID
  - vrací `S_OK` a přetypovaný pointer
  - nebo vrací `E_NOINTERFACE` pokud objekt rozhraní nepodporuje