

PB071 – Úvod do jazyka C

Překladové systémy

Jiri Slaby

Fakulta informatiky
Masarykova univerzita

2. 5. 2016

- 1 Úvod a motivace
- 2 Vývojový nástroj make
- 3 Autotools, CMake
- 4 Závěr

Sekce 1

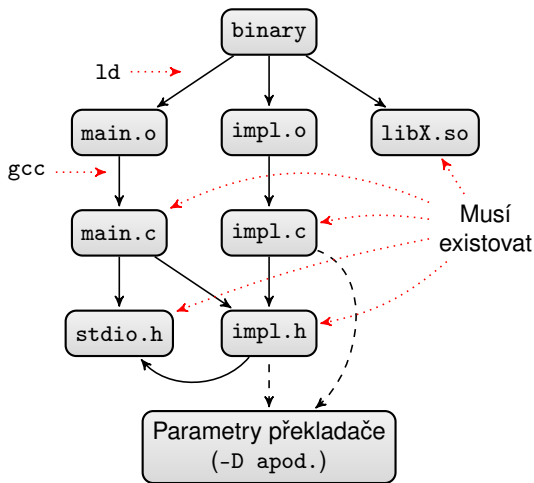
Úvod a motivace

- Máme zdrojové soubory
- Chceme binárku
- Použijeme překladač
 - `gcc *.c`
- **Demo:** jádro
- Chceme raději nějaký nástroj
 - Popis, jak sestavit projekt
 - Ale jaký?

Spousta řešení

- Svoje vlastní
 - Skripty v bashi, perlu, pythonu
 - *Vždy* peklo
 - *Nikdy* nevynalézejte kolo
- Cizí
 - Udržované nástroje
 - make, cmake, qmake, ...

Popis závislostí + popis výroby



Sekce 2

Vývojový nástroj make

- Pravidla pro překlad projektu
 - V textové formě
- `make` pravidla načte, vyhodnotí a spustí
- Mezi pravidly jsou typicky závislosti
 - Viz předchozí slajd
- `make` hledá `Makefile`, popř. `makefile` v lokálním adresáři
 - `-f` mění název souboru
 - `-C` mění adresář
- <https://www.gnu.org/software/make/manual/make.html>

- Pravidla: `cil:` závislosti
 - Implicitně se začne vyhodnocením prvního v souboru
 - Ale lze specifikovat na příkazové řádce: `make cil`
- Pravidla mohou mít akce, jak `cil` vyrobit ze závislosti
 - Na dalším řádku za pravidlem, *odsazené tabulátorem*
 - Volá se shell
 - Jinak se použije implicitní akce

Příklad

```
all : main
```

```
main: main.o impl.o  
    gcc -o main main.o impl.o
```

Hello world Makefile

- 1 Napište jednoduchý Makefile
- 2 V něm budou 3 pravidla
 - p1, p2, p3
- 3 Každé vypíše své jméno
- 4 Každé z nich vyvolejte
 - Z příkazové řádky
- 5 Nyní mezi nimi vytvořte závislosti
 - Ať se vypíše po řadě p1, p2, p3
- 6 Nyní spusťte jen samotný `make`

- Nastavení proměnných
 - Na příkazové řádce
 - V souboru: `CC=gcc`
- Reference pomocí `$(...)`

Příklad

```
CFLAGS=-Wall
```

```
all : main
```

```
main: main.o impl.o
```

```
$(CC) -o main main.o impl.o
```

Proměnné v Makefile

- 1 Vytvořte nové pravidlo `m.o`
- 2 Bude záviset na `main.c`
- 3 Bude překládat `main.c` na `m.o`
- 4 Bude brát v úvahu proměnné `CC` a `CFLAGS`
- 5 Upravte jedno z předchozích `p*` pravidel, aby záviselo na `m.o`
- 6 Vyzkoušejte
 - Zkuste předat různé `CFLAGS`
 - Nezapomeňte pozměnit alespoň čas souboru (`touch main.c`)

- % je zástupný symbol
- Lze psát pravidla: %.o: %.c
- Jak ale zjistit jména souborů?
 - \$@ – jméno cíle
 - \$< – jméno první závislosti
 - \$^ – jména všech závislostí

Příklad

```
%.o: %.c  
$(CC) -c -o$@ $<
```

Obecná pravidla v Makefile

- 1 Vytvořte nové pravidlo k překladu z `.c` do `.o`
- 2 Bude brát v úvahu proměnné `CC` a `CFLAGS`
- 3 Kromě `CFLAGS` bude překládat s `-W`
- 4 Upravte jedno z předchozích `p*` pravidel, aby záviselo na `main.o`
- 5 Vyzkoušejte

Makefile

Vnitřní příkazy

- Make má vnitřní příkazy
 - `$(prikaz parametry)`
- `shell command`
 - Spustí shell a vykoná v něm příkaz `command`
- `wildcard pattern`
 - Expanduje `pattern` (* ne %) a vrací seznam
- `patsubst pattern, replacement, text`
 - Nahradí v `text` všechny výskyty `pattern` za `replacement`
- ...

Příklad

```
PATH=$(shell pwd)
SOURCES=$(wildcard *.c)
OBJS=$(patsubst %.c,%o,$(SOURCES))
```

Vnitřní příkazy v Makefile

- 1 Přeložte všechny `.c` (wildcard) v aktuálním adresáři do `.o` (patsubst)
- 2 Slinkujte všechny `.o` do jedné binárky
- 3 Vytvořte si 2 zdrojové soubory
 - Jeden s funkcí `main`
 - Druhý s funkcí, která je volána z `main`
- 4 Vyzkoušejte

- Nutno popsat všechny závislosti
- `make clean` ručně
- Rychlost
 - Není lineární vzhledem k počtu souborů
- Složitý zápis komplikovanějších pravidel
 - Např. neexistuje explicitní podpora pro podadresáře
- Méně přenositelný
- Může být příliš upovídaný

Sekce 3

Autotools, CMake

- Nadstavby nad `make` (a ostatními)
 - Často generují `Makefile` pro `make`
- Podpora externích závislostí
 - Je v dosahu překladač C?
 - Umí překladač C99?
 - Je v systému knihovna X?
 - Máme hlavičku `Y.h`?
- Podpora podadresářů
- Konfigurace uživatelem
 - Ne/chci ve výsledné binárce podporu pro X
 - Ne/optimalizuj při překladač (ne/chci ladit)
- ...

Tři nástroje

- 1 Autoconf
 - Vyhledání externích závislostí
 - Konfigurace uživatelem
 - Z popisu v `configure.ac` generuje skript `configure`
 - <http://www.gnu.org/software/autoconf/manual/>
- 2 Automake
 - Popis překladač (včetně rozdělení do podadresářů)
 - Z popisu v `Makefile.am` generuje `Makefile.in`
 - <http://www.gnu.org/software/automake/manual/>
- 3 Libtool
 - Podpora pro tvorbu knihoven na různých POSIX systémech
 - <https://www.gnu.org/software/libtool/manual/>

Výsledek: jeden `configure` a mnoho `Makefile.in`

Uživatel spustí `./configure` a dostane mnoho `Makefile`

- Popis konfigurace a překladač v jednom
 - Soubory CMakeLists.txt
 - Znáte z cvičení (kontr_lessons)
- <https://cmake.org/cmake/help/latest/>

Příklad

```
cmake_minimum_required(VERSION 2.8.11)
project(my_project)

set(CMAKE_C_FLAGS "-std=c99 -pedantic -Wall -Wextra")

add_subdirectory(my_lib)
add_executable(main main.c impl.c)
target_link_libraries (main LINK_PUBLIC my_lib)
```

Sekce 4

Závěr

Nepište překladové systémy na vlastní pěst

Děkuji za pozornost!

Dotazy?