Prevajanje C programa

Prevajalnik

> Prevajalnik je program, ki zapis v programskem jeziku prevede v strojni jezik računalnika ali v vmesno kodo. Program, ki je preveden v strojni jezik, lahko neposredno izvajamo na računalniku. Program, ki je preveden v vmesno kodo, lahko izvajamo s tolmačem za vmesno kodo.
> -- MaFiRa [0]

[0] http://wiki.fmf.uni-lj.si/wiki/Prevajalnik

Prevajalnik

> Prevajalnik je program, ki zapis v programskem jeziku prevede v strojni jezik računalnika ali v vmesno kodo. Program, ki je preveden v strojni jezik, lahko neposredno izvajamo na računalniku. Program, ki je preveden v vmesno kodo, lahko izvajamo s tolmačem za vmesno kodo.
> -- MaFiRa [0]

Najpogosteje uporabljeni prevajalniki:

+ GNU Compiler Collection (GCC) [1]
+ Clang/LLVM [2]
+ Microsoft Visual C++ (MSVC) [3]

[0] http://wiki.fmf.uni-lj.si/wiki/Prevajalnik
[1] https://en.wikipedia.org/wiki/GNU Compiler Collection
[2] https://en.wikipedia.org/wiki/Clang
[3] https://en.wikipedia.org/wiki/Microsoft Visual C%2B%2B

Štiri faze prevajanja:

Štiri faze prevajanja:

1. Predprocesiranje (preprocessing)

Štiri faze prevajanja:

1. Predprocesiranje (preprocessing)

2. Prevajanje v ožjem smislu (compilation)

Štiri faze prevajanja:

1. Predprocesiranje (preprocessing)
2. Prevajanje v ožjem smislu (compilation)
3. Zbiranje (assembly)

Štiri faze prevajanja:

1. Predprocesiranje (preprocessing)
2. Prevajanje v ožjem smislu (compilation)
3. Zbiranje (assembly)
4. Povezovanje (linking)

Prevajanje C programa

Štiri faze prevajanja:

1. Predprocesiranje (preprocessing)
2. Prevajanje v ožjem smislu (compilation)
3. Zbiranje (assembly)
4. Povezovanje (linking)

Vzorčni program: program.c [0]

[0] https://git.io/vdDoI

1. Predprocesiranje

- dani C program prepiše v pravi C program
- nadomesti kontrolne vrstice (vrstice, ki se začenjajo z višajem #), # include direktive itd., z vsebinami standardnih in ostalih glav, makro definicij itd.
- združi vrstice, ki se končajo z nagibnico \
- odstrani komentarje

1. Predprocesiranje

- dani C program prepiše v pravi C program
- nadomesti kontrolne vrstice (vrstice, ki se začenjajo z višajem #), # include direktive itd., z vsebinami standardnih in ostalih glav, makro definicij itd.
- združi vrstice, ki se končajo z nagibnico \
- odstrani komentarje

```
gcc -E program.c > program.i
```

1. Predprocesiranje

- dani C program prepiše v pravi C program
- nadomesti kontrolne vrstice (vrstice, ki se začenjajo z višajem #), # include direktive itd., z vsebinami standardnih in ostalih glav, makro definicij itd.
- združi vrstice, ki se končajo z nagibnico \
- odstrani komentarje

```
gcc -E program.c > program.i
```

gcc tukaj kliče cpp (C predprocesor):

```
cpp program.c > program.i
```

2. Prevajanje v ožjem smislu (compilation)

- predprocesiran program prevajalnik prevede v zbirni jezik, assembler
- pri prevajanju nastane datoteka s končnico `.s`
- običajno se tu vključi še optimizator, ki skuša program skrajšati in pospešiti. Rezultat je spet program v zbirniku, upajmo da krajši in boljši

2. Prevajanje v ožjem smislu (compilation)

- predprocesiran program prevajalnik prevede v zbirni jezik, assembler
- pri prevajanju nastane datoteka s končnico .s
- običajno se tu vključi še optimizator, ki skuša program skrajšati in pospešiti. Rezultat je spet program v zbirniku, upajmo da krajši in boljši

```
gcc -S program.i
```

3. Zbiranje (assembly)

Program napisan v zbirniku končno prevzame zbirnik, assembler, ki predela `.s` datoteko v `.o` datoteko. To je objektna ali binarna, za ljudi nečitljiva datoteka. Zbirnik je v celi verigi prvi program, ki mora vedeti kako je datoteka `.o` zgrajena.

3. Zbiranje (assembly)

Program napisan v zbirniku končno prevzame zbirnik, assembler, ki predela `.s` datoteko v `.o` datoteko. To je objektna ali binarna, za ljudi nečitljiva datoteka. Zbirnik je v celi verigi prvi program, ki mora vedeti kako je datoteka `.o` zgrajena.

```
gcc -c program.s
```

3. Zbiranje (assembly)

Program napisan v zbirniku končno prevzame zbirnik, assembler, ki predela `.s` datoteko v `.o` datoteko. To je objektna ali binarna, za ljudi nečitljiva datoteka. Zbirnik je v celi verigi prvi program, ki mora vedeti kako je datoteka `.o` zgrajena.

```
gcc -c program.s
```

gcc tukaj kliče as:

```
as -o program.o program.s
```

3. Zbiranje (assembly)

Program napisan v zbirniku končno prevzame zbirnik, assembler, ki predela `.s` datoteko v `.o` datoteko. To je objektna ali binarna, za ljudi nečitljiva datoteka. Zbirnik je v celi verigi prvi program, ki mora vedeti kako je datoteka `.o` zgrajena.

```
gcc -c program.s
```

gcc tukaj kliče as:

```
as -o program.o program.s
```

Poglejmo si vsebino datoteke program.o

```
hexdump -C program.o | head -n 20
```

4. Povezovanje (linking)

Zadnja faza je povezovalnik, program ld. Ta pobere vse navedene, za nastanek izvršljive datoteke potrebne, `.o` in `.a` objektne datoteke in jih združi v izvedljiv program `a.out`, če s stikalom `-o` ne ukažemo drugače.

4. Povezovanje (linking)

Zadnja faza je povezovalnik, program ld. Ta pobere vse navedene, za nastanek izvršljive datoteke potrebne, `.o` in `.a` objektne datoteke in jih združi v izvedljiv program `a.out`, če s stikalom `-o` ne ukažemo drugače.

Ukaz povezovanja je dokaj zapleten. Na mojem sistemu zgleda nekako takole:

4. Povezovanje (linking)

Zadnja faza je povezovalnik, program ld. Ta pobere vse navedene, za nastanek izvršljive datoteke potrebne, `.o` in `.a` objektne datoteke in jih združi v izvedljiv program `a.out`, če s stikalom `-o` ne ukažemo drugače.

Ukaz povezovanja je dokaj zapleten. Na mojem sistemu zgleda nekako takole:

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 \
/usr/lib64/crt1.o /usr/lib64/crti.o \
/usr/lib64/crtn.o program.o \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtbegin.o -L \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0 -lgcc -lgcc_eh \
-lc -lgcc -lgcc_eh \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtend.o \
-o program
```

4. Povezovanje (linking)

Zadnja faza je povezovalnik, program ld. Ta pobere vse navedene, za nastanek izvršljive datoteke potrebne, .o in .a objektne datoteke in jih združi v izvedljiv program a.out, če s stikalom -o ne ukažemo drugače.

Ukaz povezovanja je dokaj zapleten. Na mojem sistemu zgleda nekako takole:

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 \
/usr/lib64/crt1.o /usr/lib64/crti.o \
/usr/lib64/crtn.o program.o \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtbegin.o -L \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0 -lgcc -lgcc_eh \
-lc -lgcc -lgcc_eh \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtend.o \
-o program
```

Zato izvedimo raje:

4. Povezovanje (linking)

Zadnja faza je povezovalnik, program ld. Ta pobere vse navedene, za nastanek izvršljive datoteke potrebne, .o in .a objektne datoteke in jih združi v izvedljiv program a.out, če s stikalom -o ne ukažemo drugače.

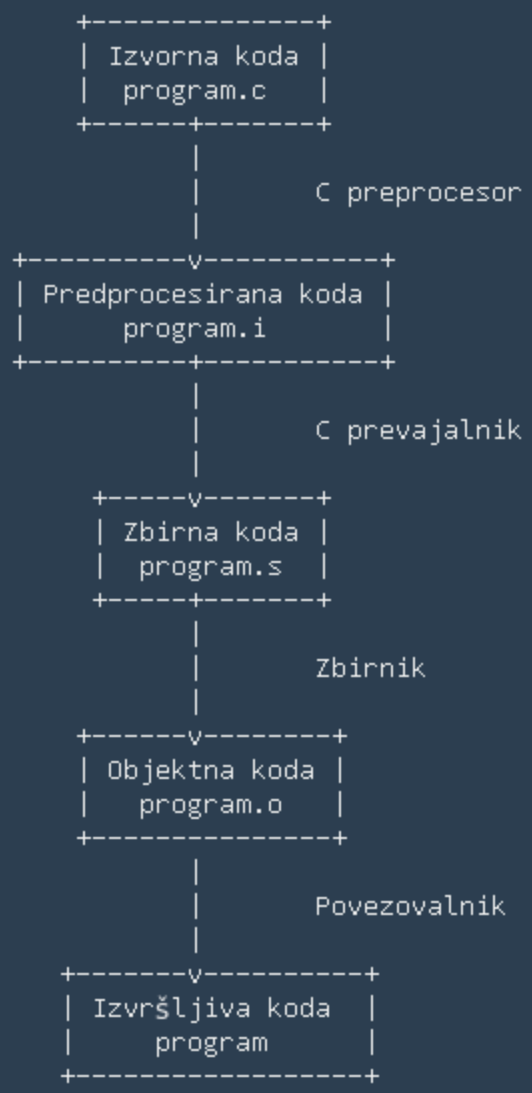
Ukaz povezovanja je dokaj zapleten. Na mojem sistemu zgleda nekako takole:

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 \
/usr/lib64/crt1.o /usr/lib64/crti.o \
/usr/lib64/crtn.o program.o \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtbegin.o -L \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0 -lgcc -lgcc_eh \
-lc -lgcc -lgcc_eh \
/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/crtend.o \
-o program
```

Zato izvedimo raje:

```
gcc -o program program.c
```

Povzetek


Izvorna koda
program.c
C preprocesor
Predprocesirana koda
program.i
C prevajalnik
Zbirna koda
program.s
Zbirnik
Objektna koda
program.o
Povezovalnik
Izvršljiva koda
program

Prevajalnik zagotovi, da je skladnja (sintaksa) programa pravilna, ne mora pa zagotoviti, da je program tudi logično pravilen.

[Dodatek 1:]

Nekaj opcij prevajalnika

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2:
  (Sedem optimizacijskih nivojev)

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2: (Sedem optimizacijskih nivojev)

```
gcc -Wall -O2 -o program program.c
```

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2: (Sedem optimizacijskih nivojev)

```
gcc -Wall -O2 -o program program.c
```

- Odstrani nepotrebne simbole:

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2: (Sedem optimizacijskih nivojev)

```
gcc -Wall -O2 -o program program.c
```

- Odstrani nepotrebne simbole:

```
gcc -Wall -s -o program program.c
```

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2: (Sedem optimizacijskih nivojev)

```
gcc -Wall -O2 -o program program.c
```

- Odstrani nepotrebne simbole:

```
gcc -Wall -s -o program program.c
```

- Ustvari razhroščevalne informacije, s katerimi lahko dela GDB:

[Dodatek 1:]

Nekaj opcij prevajalnika

- Omogoči vsa opozorila prevajalnika, opozorila pretvori v napake in preveri, če program ustreza ANSI standardu:

```
gcc -Wall -pedantic -Werror -Wextra -o program program.c
```

- Shrani trenutne datoteke, ki nastanejo med prevajanjem:

```
gcc -Wall -save-temps -o program program.c
```

- Optimiziraj program z zastavico -O2: (Sedem optimizacijskih nivojev)

```
gcc -Wall -O2 -o program program.c
```

- Odstrani nepotrebne simbole:

```
gcc -Wall -s -o program program.c
```

- Ustvari razhroščevalne informacije, s katerimi lahko dela GDB:

```
gcc -Wall -g -o proram program.c
```

[Dodatek 2:]

Sedem optimizacijskih nivojev

Nivo	Pomen
-O -O1	Zmanjšanje velikosti programa in hitrosti izvajanja brez velikega vpliva na čas prevajanja.
-O2	Se večji nivo zmanjšanja velikosti programa in hitrosti izvajanja.
-O3	Največji nivo zmanjšanja velikosti programa in hitrosti izvajanja.
-O0	Brez optimizacije, skrajša čas prevajanja in olajša razhroščevanje (privzeto).
-Os	Optimizacija velikosti programa.
-Ofast	Optimizacija hitrosti izvajanja.
-Og	Vklop vseh optimizacij, ki ne vplivajo na težavnost razhroščevanja

[Dodatek 3:]

Informacije o izvršljivi datoteki

[Dodatek 3:]

Informacije o izvršljivi datoteki

▶ Prikaži informacije o objektni datoteki:

```
objdump -S -l -C -F -t -w program
```

[Dodatek 3:]

Informacije o izvršljivi datoteki

▶ Prikaži informacije o objektni datoteki:

```
objdump -S -l -C -F -t -w program
```

▶ Prikaži informacije o ELF datoteki:

```
readelf -a program
```

[Dodatek 3:]

Informacije o izvršljivi datoteki

- Prikaži informacije o objektni datoteki:

```
objdump -S -l -C -F -t -w program
```

- Prikaži informacije o ELF datoteki:

```
readelf -a program
```

- Razhroščevalniki in razbirniki:

```
gdb program
```

```
r2 program
```

...

[Dodatek 3:]

Informacije o izvršljivi datoteki

- Prikaži informacije o objektni datoteki:

```
objdump -S -l -C -F -t -w program
```

- Prikaži informacije o ELF datoteki:

```
readelf -a program
```

- Razhroščevalniki in razbirniki:

```
gdb program
```

```
r2 program
```

...

Windows: `IDA Pro`, `OllyDBG`, `x64dbg`, `WinDBG`, ...

[Dodatek 3:]

Informacije o izvršljivi datoteki

- Prikaži informacije o objektni datoteki:

```
objdump -S -l -C -F -t -w program
```

- Prikaži informacije o ELF datoteki:

```
readelf -a program
```

- Razhroščevalniki in razbirniki:

```
gdb program
```

```
r2 program
```

...

Windows: `IDA Pro`, `OllyDBG`, `x64dbg`, `WinDBG`, ...

- Šestnajtiški urejevalnik (hexeditor):

```
dhex program
```

Viri:

- ZAKRAJŠEK, E. ANSI C. Ljubljana:
  Fakulteta za matematiko in fiziko, 1998

- program.c [0]

[0] https://git.io/vdDoI

Ta predstavitev je narejena z mdp [0]

[0] https://github.com/visit1985/mdp