Bányász Gábor

Linux Programozás

Fejlesztői eszközök

Fordító (GCC)

- A GNU Compiler Collection (GCC) nem az egyetlen, de a legelterjedtebb a Linuxos világban.
- Több nyelvhez tartalmaz fordító programot:
  - C (gcc)
  - C++ (g++)
  - Objective-C
  - Java (gcj)
  - Fortran
  - Ada (gnat)
  - Go
- A fordító parancssoros számos paraméterrel
  - Többnyire csak néhány paramétert használunk.
  - A pkg-config program egyszerűsítheti a paraméterezést.

gcc (C fordító) paraméterei

Paraméter	Jelentés
`-o fájlnév`	A kimeneti állománynév megadása.
`-c`	Fordítás, linkelés nélkül.
`-Ddefiníció=x`	A *definició* makrót x értékre definiálja.
`-Ikönyvtárnév`	Hozzáadja a könyvtárat ahhoz a listához, amelyben a .h kiterjesztésű (header) állományokat keresi.
`-Lkönyvtárnév`	Hozzáadja a könyvtárat ahhoz a listához, amelyben a library állományokat keresi.
`-llibrary`	A programhoz linkeli a *library* nevű programkönyvtárt.
`-static`	Az alapértelmezett dinamikus linkelés helyett a fordító a statikus programkönyvtárakat linkeli a programhoz.
`-g, -gN` `-ggdb, -ggdbN`	A lefordított állományt ellátja a hibakereséshez szükséges információkkal.
`-O, -ON`	Optimalizálja a programot az N optimalizációs szintre.
`-Wall`	Az összes, általában használt figyelmeztetést bekapcsolja.

Fordítás példa (1)

Kód:

```
#include <stdio.h>
int main()
{
  printf("Hello vilag!\n");
  return 0;
}
```

Fordítás:

```
gcc -o hello hello.c
```

Futtatás:

```
./hello
```

Fordítás példa (2)

- Több forrás állomány esetén:

```
gcc -o prg forras1.c forras2.c
```

- Fordítás majd linkelés:

```
gcc -c hello.c
gcc -o hello hello.o
```

- Több forrás állomány esetén:

```
gcc -c forras1.c
gcc -c forras2.c
gcc -o prg forras1.o forras2.o
```

Fordítás példa (3)

- Saját header könyvtár:

```
gcc -c forras1.c -I/home/myname/headers
gcc -c forras2.c -I/home/myname/headers
gcc -o prg forras1.o forras2.o
```

- Library hozzá linkelése:

...

```
gcc -o prg forras1.o forras2.o -lm
```

- Saját lib hozzálinkelése:

...

```
gcc -o prg forras1.o forras2.o -L/home/myname/libs
-lmy
```

Fordítás példa (4)

- Define a parancssorban:

```
gcc -DDEBUG forras.c -o prg
```

ua.

```
#define DEBUG
```

```
gcc -DXSIZE=16 forras.c -o prg
```

ua.

```
#define XSIZE 16
```

Makefile

- Automatizált fordítást tesz lehetővé.
- Lényegében egy függőségi fát alakítunk ki a műveletek között.
- Ha módosítunk valamit, akkor mindig csak a fának a megfelelő részét járja végig.
- Érzékeny arra, hogy a rendszeridő csak előre haladjon. (A módosítások detektálásához)
- Nem csak fordításra jó, hanem egyéb parancsokat is készíthetünk.
- A „`make`” paranccsal hajthatjuk végre és az alapértelmezett neve: „`Makefile`”

Makefile példa

```
# Makefile

objs = forras1.o forras2.o
libs = -lm

prg: $(objs)
  gcc -o prg $(objs) $(libs)

install: prg
  install -m 644 prg /usr/local/bin

.PHONY: install
```

A Makefile elemei

- Megjegyzések – A # karakterrel kezdődnek és a make figyelmen kívül hagyja őket.
- Explicit szabályok
- Implicit szabályok
- Változódefiníciók
- Direktívák

Explicit szabályok

```
TÁRGY: ELŐKÖVETELMÉNYEK; RECEPT
<TAB>RECEPT
<TAB>...
```

- Ha valamelyik előkövetelmény nincs meg, akkor előállítja.
- Ha valamelyik előkövetelmény fájl módosul, akkor újra elvégzi a fordítást.
- Minden parancssor külön subshellben fut le.
- Ha a make-nek nem adunk meg paramétert, akkor automatikusan az első explicit szabály lesz az alapértelmezett.
- Valójában több tárgyat is megadhatunk. Ilyenkor mindegyikre elvégzi a műveleteket.

Hamis tárgy

- A szabály célja nem egy állomány
- A recept rész tartalmazza a végrehajtandó parancsokat
- Ha mégis létrehozzuk az állományt, akkor nem hajtódnának végre a parancsok
- Megoldás: `.PHONY` kulcsszó

Változódefiníciók

- Értékadás:

```
VÁLTOZÓ = ÉRTÉK
```

- Hivatkozás:

```
$(VÁLTOZÓ) v. ${VÁLTOZÓ}
```

- Nem szekvenciális a kiértékelés, vagyis ez hibás:

```
objs = elso.o
objs = $(objs) masodik.o
```

Speciális értékadás

- Egyszerű kiértékelés:

```
a := egy
b := $(a) ketto
a := harom
```

- Hozzáadás:

```
objs = elso.o
objs += masodik.o
```

- Feltételes értékadás:

```
a = egy
a ?= ketto
```

Speciális hivatkozás

- Helyettesítő hivatkozás:

```
srcs = elso.c masodik.c
objs := $(srcs:.c=.o)
```

- Számított változónevek:

```
a = b
b = c
eredm := $($(a))
```

- Kombinálva:

```
src_1 := elso.c
src_2 := masodik.c
objs := $(src_$(a):.c=.o)
```

Automatikus változók

Változó	Jelentés
`$@`	A célfájl neve.
`$<`	A függőségi lista első elemének neve.
`$?`	Az összes olyan feltételfájl neve (sorban, szóközzel elválasztva), amely frissebb a célállománynál.
`$^`	Az összes feltétel fájl neve szóközzel elválasztva.
`$+`	Jelentése nagyrészt egyezik az előzővel, azonban ha a feltételek közt többször szerepel a fájl, akkor azt ugyanúgy több példányban tartalmazza.
`$*`	A célfájl nevének utótag nélküli része.

Klasszikus ragozási szabályok

- Az egyik kiterjesztésű állományból hogyan állíthatjuk elő a másikat. Vagyis általános szabályok.

```
FsCs:
<TAB>PARANCS

.SUFFIXES: Fs Cs
```

Fs: forrás suffix, Cs: cél suffix

- A parancsok megadásához szükséges az automatikus változók használata.

Statikus minta szabály

- Formátum:

```
TÁRGYAK:TÁRGY-MINTA:ELŐKÖVETELMÉNY-MINTA; RECEPT
<TAB>RECEPT
```

- A TÁRGY-MINTA és az ELŐKÖVETELMÉNY-MINTA megadja hogyan számítható ki a tárgyból az előkövetelmény.
- Példa:

```
objs := elso.o masodik.o
all: $(objs)

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

Minta szabály

- Nincs tárgylista → Általános szabály
- Akkor alkalmazza a rendszer, ha nincs specifikus
- Szintaxis:

```
TÁRGY-MINTA: ELŐKÖVETELMÉNY-MINTA; RECEPT
<TAB>RECEPT
```

- Példa:

```
objs := elso.o masodik.o
all: $(objs)

%.o: %.c
	gcc -c -o $@ $<
```

Implicit szabályok

- Számos minta szabályt beépítetten is tartalmaz a make. Nem szükséges őket külön definiálni.
- Az alapértelmezett szabályok megszokott változókat használnak a paraméterezés kiegészítésére.
- A beépített szabályok kilistázása:

```
make -p
```

- Például:

```
%.o: %.c
	$(CC) $(CFLAGS) $(CPPFLAGS) \
	$(TARGET_ARCH) -c $(OUTPUT_OPTION) $<
```

Speciális célok

- `.PHONY`: Nem nézi az időbélyegeket, mindenképpen végrehajtja a műveletet, ha kell a célhoz.
- `.SUFFIXES`: Kiterjesztések felsorolása
- `.DEFAULT`: Az alapértelmezett cél
- `.SILENT`: Csöndes parancsvégrehajtás
- `.ONESHELL`: A recept egy shellben fut

Direktívák

- Más állományok befűzése:

```
include fájlnév
```

- Feltétel:

```
ifeq ($(CC),gcc)
  libs=$(gcc_libs)
else
  libs=$(normal_libs)
endif
```

- Definiálás:

```
define ket-sor
  echo foo
  echo $(bar)
endef
```

Make alternatívák

- Autotools
- CMake
- qmake
- SCons