Bevezetés a GNU Autoconf rendszer használatába

Bizonyára mindenki ismeri a három parancsot, amellyel egy *.tar.gz fájlból kicsomagolt programot telepíteni lehet: configure, make és make install. E rendszer alapja a GNU Autoconf, amellyel áttekinthetővé tehetjük a fejlesztett programunk forráskódját, és akár egyszerű telepítőkészletet is készíthetünk segítségével.

Bevezetés

Az első, legegyszerűbb programokat parancssorból fordították, a módosítások után kézzel elindítva a fordítót. Később, amikor már több forrás fájlból állítottak össze egyetlen binárist, a fordítási idők lecsökkentéséhez kitalálták a ***Makefile***-okat, amelyekben a programokhoz egy függőségi fa írható le, amely meghatározza, melyik futtatható állományhoz melyik forrásokat kell lefordítani illetve összeszerkeszteni; illetve hogyan, milyen paraméterekkel kell a fordítót indítani. A legnagyobb programoknál persze még ezzel is sok munka van arról nem is beszélve, hogy a programok hordozhatóságára sem ad semmilyen megoldást.
Ma már legtöbbször a ***Makefile***-t sem szokás kézzel írni. Több olyan program is van, amely nagy, több könyvtárból álló forráskódok, összetett programrendszerek kezelését könnyíti meg. Ilyenek a ***Qt*** programok által használt ***qmake***, az ***XFree86 imake*** programja, illetve a ***GNU autoconf***. (Az utóbbi időben az ***X.Org*** is már az ***autoconfot*** használja.)
A ***Linuxot*** használók legtöbbször programok telepítése közben találkoznak az ***autoconf*** rendszerrel. Mindenkinek ismerős a három begépelendő utasítás, a ***configure***, a ***make*** és a ***make install***, amelyekkel az ***Internetről*** forráskódban letöltött programokat installálni tudjuk. Ebben a cikkben azokkal a programokkal foglalkozunk, amelyekkel a ***configure*** parancsfájlt előállíthatjuk.
Az ***autoconf*** rendszernek több haszna is van.

1. lista Helló, világ! GTK+ módra: hello.c

```
#include <gtk/gtk.h>
#include "config.h"

int main(int argc, char *argv[])
{
    GtkWidget *window;

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    g_signal_connect(G_OBJECT(window), "delete-event",
    ➥G_CALLBACK(gtk_main_quit), NULL);
    gtk_window_set_title(GTK_WINDOW(window), PACKAGE_STRING);
    gtk_container_set_border_width(GTK_CONTAINER(window),100);

    gtk_container_add(GTK_CONTAINER(window),
    ➥gtk_label_new("Helló, világ!"));
    gtk_widget_show_all(window);
    gtk_main();
    return 0;
}
```

- Segítségével könnyebben biztosítható a programunk hordozhatósága. Egyszerűen használható makrókat ad a programozó kezébe, amelyekkel meghatározható fordítás közben a cél operációs rendszer, illetve a fordításhoz és futtatáshoz szükséges eszközöket, például fordítóprogramokat, könyvtárakat is automatikusan képes megkeresni.
- Egyszerű telepítőt készíthetünk vele a programunkhoz. Ez a felhasználóknak is könnyebbség, ugyanis minden ***autoconffal***

2. lista A configure.in fájl

```
AC_INIT(Hello, 0.1)
AC_CONFIG_SRCDIR([hello.c])
AM_INIT_AUTOMAKE
AM_CONFIG_HEADER(config.h)

AC_PROG_CC
AC_ISC_POSIX
AC_HEADER_STDC

AM_PROG_INSTALL

AM_PATH_GTK_2_0(2.8.0, :,
➥AC_MSG_ERROR(Test for GTK+
➥failed. See the file
➥'INSTALL' for help.))

AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

készített programot ugyanazon a módon lehet telepíteni.
Ha a rendszer ismerős, még egy tapasztalatlanabb felhasználó is egyből tudja, hogyan használja azt. Nem is beszélve az automatizálhatóságról, gondoljunk csak a *Gentoo Portage* felületére vagy a *BSD portsra*.

Helló, világ! program autoconffal

Úgy döntöttem, hogy a cikkben a rendszert inkább egy kész programon keresztül mutatom be, egy *tutorial*-szerű leíráshoz ugyanis minden fájlból három-négy különböző verziót kellene prezentálni, miközben alig lenne köztünk néhány szó különbség.
A példa megszokott „Helló, világ!" kiíratása, mégpedig annak a GTK+ változata. A forráskód az első listán látható. Gépeljük be, és mentsük el *hello.c* néven, illetve írjuk be a *Makefile.am* és a *configure.in* fájlokat is, elhelyezve őket a forráskóddal egy könyvtárban!
Ha ezt a három fájlt megírtuk, indítsuk el a rendszer egyes programjait, amelyek elkészítenek néhány fájlt:

```
aclocal
autoheader
automake --add-missing
autoconf
```

3. lista A Makefile.am fájl

```
LDADD = @GTK_LIBS@
AM_CFLAGS = @GTK_CFLAGS@
➥-Wall

bin_PROGRAMS = hello

hello_SOURCES = hello.c
```

Amikor először elindítjuk, figyeljük meg: az *automake* program jelez, hogy néhány fájl (*AUTHORS*, *README*, *NEWS*, és *ChangeLog*) hiányzik. A fájlok elvileg részei egy *GNU* kompatibilis szoftver terjesztésnek. Ezekben a program leírása található, illetve különböző információk, amelyek a felhasználók számára érdekesek lehetnek: a szerzők nevei és *e-mail* címei, a program újdonságai, változtatásai az előző verziókhoz képest stb. Néhány hasonló fájlt az *automake* az `--add-missing` argumentum hatására automatikusan létrehozott helyettünk: a *COPYING* fájlba másolta például a *GNU GPL*-t. Készítsük el a szükséges hiányzó szövegfájlokat, vagy legalábbis hozzunk létre egyelőre üres fájlokat helyette! Ha megvagyunk, akkor adjuk ki újra az `automake --add-missing` parancsot, és utána folytassuk az `autoconf` beírásával! A folyamat elég sok fájllal gazdagít minket, legtöbbjükkel szerencsére nem kell foglalkoznunk, ugyanis a rendszer automatikusan kezeli őket.
Ha mindennel elkészültünk, már működik is a megszokott *GNU* módon működő szoftver terjesztésünk! Próbáljuk is ki:

```
./configure
make
make install    # rootként
hello
make uninstall    # rootként
make dist
```

Ha a rendszerhez, ahol dolgozunk, van adminisztrátori jogosultságunk, kipróbálhatjuk a szoftver csomagunk telepítését illetve törlését is (`make install` illetve `make uninstall`). Ez alapértelmezés szerint az */usr/local/bin/hello* fájlt hozza létre.
Ha ezt kihagyjuk, akkor indításnál használjuk inkább a `./hello` parancsot!
Lássuk sorban, mit is jelentenek az egyes fájlokban beírt dolgok!

A configure.in fájl

Ez tulajdonképpen egy *Bourne* héjprogram, amelyet az *m4* makrófeldolgozó kezel. Ezért lehetséges a megszokott `if...then...fi` és hasonló szerkezeteket is használni, de ezekre általában nincsen szükség. (Fontos, hogy a szögletes zárójeleket az *m4* előfeldolgozó kezeli, ezért a *test* program nevét minden esetben ki kell írni, vagyis helytelen a `if [ -f fájlnév ]...` utasítás! A helyes működéshez az `if test -f fájlnév...` szükséges.) Ebből lesz a néha több száz kilobájtosra is megnövő *configure* program, amely a forráskód környezetbe beillesztését és a *Makefile* készítését végzi majd a cél környezetben.
Az egyes bonyolult műveleteket a makrókkat végezhetjük el. Ezeket gyűjti össze az *aclocal* nevű program, ezért kellett azt elsőnek elindítani. A makrók feladatai a következőek:

- `AC_INIT`: Elindítja az *autoconf* rendszert. Mindig ez kell legyen az első makró. Megadhatjuk neki a program nevét, verziószámát, és esetleg további argumentumai is lehetnek, pl. egy e-mail cím, ahova a felhasználók hibajelentéseket küldhetnek.

- `AC_CONFIG_SRCDIR`: Ennek a makrónak adjuk meg a forráskód egy fájlját. Most csak egyetlen egy van, a *hello.c*.

- `AM_INIT_AUTOMAKE`: Elindítja az *automake* rendszert.

- `AM_CONFIG_HEADER`: Megadja, hogy melyik fájlban tároljuk a rendszer által létrehozott definíciókat. Erről bővebben később.

- `AC_PROG_CC`: Megkeresi a rendszer C fordítóját. *Linuxon* ez általában a *gcc*.

- `AC_HEADER_STDC`: Ellenőrzi, hogy megvannak-e a szabványos C

header fájlok. (Ilyen a *hello.c*-ben pont nem szerepel, de elég gyakori a makró.)

- `AC_PROG_INSTALL`: Megkeresi a rendszeren az *install* nevű programot. (Ha nem talál ilyet, akkor az *install.sh* héjprogramot használja majd, amelyet az *automake* hozott létre, ugyanis nem mindegyik operációs rendszerben van ilyen.)

- `AM_PATH_GTK_2_0`: Megkeresi a rendszeren található *GTK+* könyvtár *header* fájljait, könyvtárait, és meghatározza, hogy a C fordítónak illetve szerkesztőnek milyen paramétereket kell átadni *GTK+*-os programok készítéséhez.

- `AC_CONFIG_FILES`: Ennek a makrónak kell megadni, hogy a *configure* héjprogram melyik fájlokat hozza létre.

- `AC_OUTPUT`: Létrehozza az előbb megadott fájlokat.

Általában egy programhoz egy *configure.in* fájlt kell írnunk, annak fő könyvtárában. Alkönyvtárankénti *configure.in* fájlokat csak nagy, több részből szerkesztett programcsomagoknál szokás használni. Ha mindenképpen erre van szükség, érdemes az *autoconf* leírását elolvasni.

Autoheader

A második elindított program az *autoheader* volt. A *configure* szkript sok változót összegyűjt a fordítási, futtatási környezetről, amelyeket a C forráskódban `#ifdef` paranccsal, illetve makróként használhatunk. Részben ezt teszi a forráskódot hordozhatóvá, és könnyen kezelhetővé. A definíciókból viszont elég sok lehet, érdemes inkább őket egy külön beleértett (*include)* fájlban tárolni. Korlátozott amúgy a parancssorok hosszúsága is, és könnyen túlléphetnénk azt a C fordítónak átadott sok `-D` paraméterrel.
Az ilyen beleértett fájl készítését támogatja az *autoheader*, illetve *autoconf* `AC_CONFIG_HEADER` makrója. A fájl szokásos neve *config.h*, ezt a forráskódba a `#include "config.h"` sorral illeszthetjük be. Elvileg ehhez is készítenünk kellene egy *config.h.in* bemeneti szövegfájlt, azonban ezt az *autoheader* elvégzi helyettünk. (Az *autoconf* a definíciókat az */usr/share/autoconf/acconfig.h* fájlból másolja. Ha valamilyen makró eltérő bejegyzést igényel, akkor egy *acconfig.h* nevű fájlt kell készítenünk, egy könyvtárban a *configure.in*-nel, és az *autoheader* figyelembe fogja venni.)
Érdemes egyébként fordítás után megvizsgálni a *config.h* fájlt. Láthatjuk például, hogy a program neve, verziószáma és még sok minden más makróként használható.
A *hello.c* programban az ablak nevét egy ilyen alapján állítjuk be a `gtk_window_set_title(...)` függvényhívással. Ezeket a C előfeldolgozó (*preprocessor*) helyettesíti be.

Automake

Az *autoconf* rendszer egyik célja, hogy minden programot ugyanolyan módon kelljen telepíteni. Közismertek a `make` és a `make install` parancsok, ezeken kívül sok kevésbé ismert is van: `make distclean`, `make dist`... Ezek minden *GNU* kompatibilis szoftver terjesztésnek részei.
Az *automake* egy adott *Makefile.am*-ből készíti el a *Makefile.in*-t, amelyből a futtatási környezetben elkészül később az a *Makefile*, amely támogat minden megszokott funkciót. Ez a fájl tulajdonképpen csak néhány változót tartalmaz, amelyeket az *automake* beilleszt egy előre elkészített *Makefileba.* A szintaxis ezért a megszokott: megyjegyzéseket a # karakter után írhatunk, a többsoros listákat pedig a \ karakterrel választhatjuk el.

Nézzük ezt is soronként!

- `LDADD`: ebbe a változóba tehetjük be, hogy a programok szerkesztéséhez (*linking*) milyen paraméterek szükségesek. Például a matematikai könyvtár (*libm*) beépítéséhez: `-lm`. Néhány változót a *configure.in* fájlban megadott makrók is beállítanak, azokat ide beírhatjuk: a *GTK+* használatához pl. `@GTK_LIBS@`.

- `AM_CFLAGS`: a C fordítónak átadandó paraméterek. Ezek hivatkozhatnak például könyvtárakra, ahol az *include* fájlokat keresi; a *GTK+* használatához ez `@GTK_CFLAGS@`, de emögé érdemes írnunk a `-Wall` paramétert is, amely a fordítás figyelmeztetéseit kapcsolja be.

- `bin_PROGRAMS`: azoknak a programoknak a neve, amelyet installálás után az */usr/local/bin*-ben (vagy */usr/bin*-ben) szeretnénk látni.

- `hello_SOURCES`: a hello nevű program előállításához használt forráskódok listája.

Egyéb gyakran használt változók:

- `sbin_PROGRAMS`: a programok, amelyek az sbin-be kerülnek.

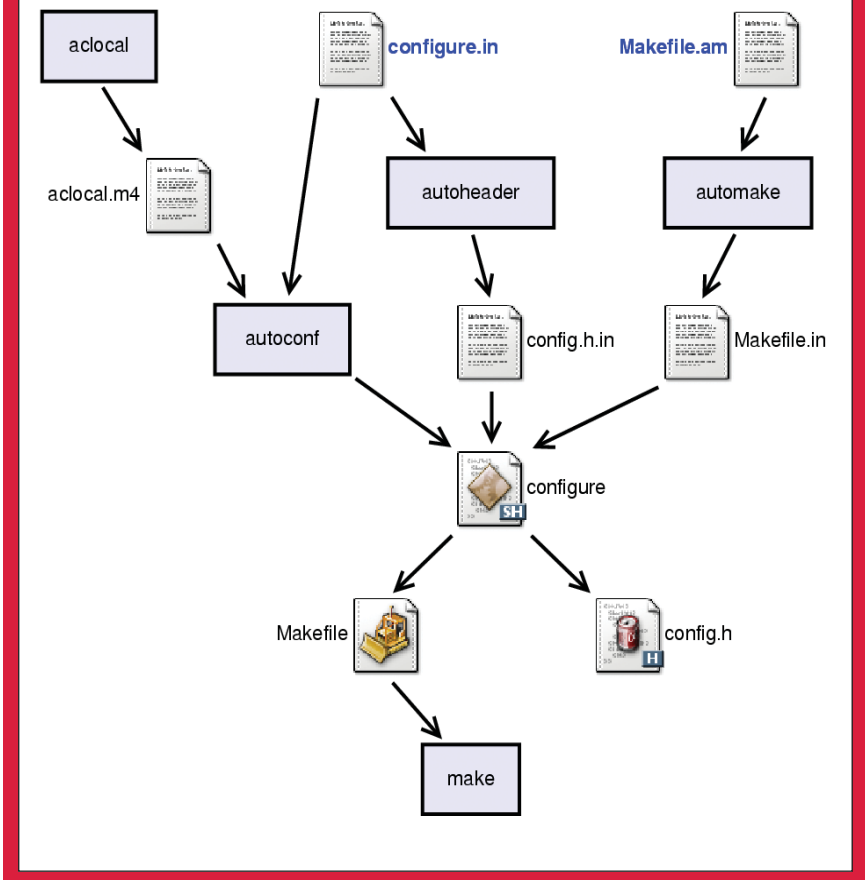
- `man_MANS`: kézikönyv (*manual*) oldalak.

- `noinst_HEADERS`: olyan *header* fájlok, amelyeket csak a fordítás közben kell használni.

- `EXTRA_DIST`: egyéb fájlok, amelyek a terjesztéshez tartoznak, például képek vagy más adatfájlok

Az autoconf rendszer programjai

A bemutatott rendszer elég összetett, érdemes egy ábrán áttekinteni, hogy melyik program mit csinál, és milyen fájlokat használ.
A használathoz tulajdonképpen két fájlt kell megírnunk. Az egyik a bemutatott *Makefile.am*, amely leírja, hogy milyen programokat kell a fordítónak létrehoznia, és hogy az egyes programokat melyik források alapján kell összeszerkesztenie. A fordításhoz szükségesek különböző paraméterek, amelyeket egyszerűbb, néhány soros *Makefile* vagy parancssori fordítás esetén a *pkg-config* programtól kérünk el; ezeket itt változóként kell megadni. A *Makefile.am* írja le azokat az egyéb fájlokat, amelyek a programhoz tartoznak, például képek, kézikönyv oldalak stb. Ebből a fájlból hozza létre az *automake* a *Makefile.in*-t, amely a szoftvercsomag része, és a *configure* héjprogram egyik bemeneti fájlja.



1. ábra Az autoconf rendszer programjai

A másik a *configure.in*. Ebben a szoftverünk által használt könyvtárakat, eszközöket soroljuk fel, illetve ez használható a fordítás folyamatának konfigurálására. Az *autoheader* program ennek alapján hozza létre a *config.h.in*-t, amelyből később egy *C* (vagy más) nyelvű *include* fájl lesz, amely a forráskódunk kezelhetőségét javítja. Az *autoconf* pedig ez alapján hozza létre a *configure* programot, amely már a cél környezetben fog futni.

Eddig tartott a programozó feladata. Ha egy felhasználó letölti, kicsomagolja a programot, akkor a telepítéshez először elindítja a *configure* programot. Ez a meglévő előfeldolgozott fájlok segítségével megvizsgálja a cél környezetet. (Ez akár teljesen más is lehet, mint ahol a programot eredetileg fejlesztették, *FreeBSD*, *MinGW*...) Létrehozza az abban a környezetben érvényes *config.h* fájlt és a *Makefile*-t. Utána indulhat a fordítás és a telepítés.

Az ábra alapján könnyen eldönthető az is, hogy az egyes bemeneti fájlok módosítása esetén melyik programokat kell újra futtatnunk. Szerencsére, ha már van egy elkészített *Makefileunk*, abba az *autoconf* elhelyez olyan szabályokat, amelyek szükség esetén az *automake* vagy *autoconf* programot automatikusan elindítják, így legtöbbször elég csak egy sima *make* parancs.

Terjesztések létrehozása

Van még néhány *Makefile* cél, amelyet eddig nem részleteztem. Az egyik a `make dist`, amely *.tar.gz* fájlba tömöríti a programcsomagunkat, amelyet egyből akár fel is tölthetünk a netre, hogy mások használhassák. A fenti példában a rendszer a *hello-0.1.tar.gz* fájlt hozza létre; a név és a verziószám természetesen megfelel a *configure.in* első sorában megadottnak. Ha ezt kicsomagoljuk, a jól ismert dolgot fogjuk látni, létrejön egy *hello-0.1*/ nevű könyvtár. Innentől pedig már álmunkból felkeltve is tudjuk a teendőt:

```
./configure && make && make
➥install
```

Terjesztés létrehozásakor érdemes egyébként inkább a `make distcheck` parancsot használni, ez ugyanis ki is próbálja a létrejövő fájlt, konkrétan konfigurálja, telepíti egy ideiglenes könyvtárba, aztán letörli. Így rögtön kiderül, ha valamelyik fájl kimaradt volna – főként az adatfájloknál (EXTRA_DIST) szokott ez előfordulni. Működnek egyébként a jól ismert `make clean` és `make distclean` parancsok is, ezekkel a fordítás, illetve a konfigurálás közben keletkezett fájlokat tudjuk törölni.

Autoconf kompatibilis programok

A legtöbb integrált fejlesztői környezet, mint például az *Anjuta*, eleve *autoconfra* épülő projektet hoznak létre. Használható ilyen célra az *autoproject* nevű program is. Ha pedig van egy meglévő forráskódunk, amely még nem használja ezt az eszközt, érdemes kipróbálni az *autoscan* nevű programot, amely egy kezdetleges *configure.in* fájl létrehozásában segít, megvizsgálva a forrást, hogy milyen szokásos nehezen hordozható részek vannak benne.

Czirkos Zoltán

Jelenleg diplomatervező a Budapesti Műszaki Egyetem Elektronikus Eszközök Tanszékén. Kutatási területe az operációs rendszerek betörésvédelme és a P2P kommunikáció. 2005-ben a Tudományos Diákköri Konferencián II. helyezést ért el. Kedvencei a boszorkányos és a rózsaszín párducos filmek.

KAPCSOLÓDÓ CÍMEK

Autoconf leírás:
➲ http://www.gnu.org/manual/autoconf/

Autoconf bevezető:
➲ http://www.seul.org/docs/autotut/

Autoproject:
➲ http://directory.fsf.org/autoproject.html

Anjuta:
➲ http://anjuta.sourceforge.net/