

## Slackware csomagok készítése

2004. április 25.

<http://www.linuxpackages.net/howto.php?page=package&title=Package+Howto>

Szerző: TG [tg.nospam@nospam.linuxpackages.net](mailto:tg.nospam@nospam.linuxpackages.net)

Fordította: Békési József [jbekesi@axelero.hu](mailto:jbekesi@axelero.hu)

Pdf változat: Tuza László ([tuza\\_laszlo@ujcolor.hu](mailto:tuza_laszlo@ujcolor.hu))

Ennek a HOGYAN-nak az a célja, hogy segítse a felhasználókat a Slackware csomagok elkészítésében. Ennek megfelelően csak hozzávetőleges útmutatónak tekinthető, nem részletes HOGYAN-nak. Először is fel kell állítanunk néhány szabályt a csomagjaink számára. Az első az, hogy mivel másoknak készítjük a csomagokat, különös figyelmet kell fordítanunk a folyamatra. Ez a HOGYAN a gyors és piszkos módszer. A *build szkriptek* használat mindig jobb. A *build szkriptek* megtekinthetők a Slackware ftp site-ján, vagy tükrein, a forrás csomagok között, általában *programnév.build* vagy *Slackbuild* néven. Emellett érdemes megnézni, milyen eszközöket vehetünk igénybe a tökéletes csomagok elkészítéséhez. Ezt illetően lásd a HOGYAN részleget <http://www.linuxpackages.net/howto.php>

### Különös odafigyelést érdemelnek:

- Jogosultságok
- Függőségek
- Telepítési útvonal

A későbbiekben mindegyikre részletesen kitérünk, most lássuk, hogyan készül el egy csomag.

Először is hozzunk létre egy tiszta munkaterületet. Ezen a munkaterületen van egy hely, ahol a fordításokat végzem, és van egy könyvtár, amibe az installálásokat irányítom. Példának okáért legyen egy *work* könyvtár, azon belül pedig két másik: az egyikben a *build szkriptek*et tartom, ha létrehozom őket, a másikba pedig a tulajdonképpeni installációk történnek. Valahogy így:

- /work (a munkaterülete)
- /work/scripts (itt tárolom a szkripteket, hogy szükség esetén könnyen tudjam upgradelni őket)
- /work/builds (ide installálom a programokat)

Annak, hogy megőrzöm a szkripteket, több oka van – részben az, hogy könnyebben tudjak upgradelni egy-egy csomagot, ha új verziót fordítok belőle, és el tudjam helyezni őket a csomagjaimban. A csomagok upgradelése során próbáljunk következetesen maradni, azaz például mindig pontosan ugyanúgy történjen az installálásuk, hogy működjön velük az *upgradepkg*, és ne tegyenek tönkre más programokat.

Nézzünk tehát egy mintafordítást:

Az első lépés természetesen az, hogy megszerezzük és kitömörítjük a programot a munkaterületen. *tar xzf whatever.tar.gz*. A következő lépés természetesen az, hogy belépünk a program könyvtárába, és megnézzük, hogy működik a konfigurációs scriptje. Általában a *./configure --help*-vel kezdek, és megvizsgálom az opciókat. Ezáltal meg tudom határozni, hogy be vagy ki kell-e kapcsolni valamit a konfigurálás során, illetve hogy át kell-e helyeznem a *conf* fájlt vagy a *pid* fájlokat valahová máshová. Egyes programokkal közölni kell bizonyos fájlok vagy library-k helyét, és előfordulhat az is, hogy a fordítás előtt be kell állítani néhány dolgot. Ekkor határozzuk meg azt is, hogy hová

történjen majd az installálás. Az esetek 90 százalékában valami ilyesmire lesz szükség:

```
/configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
```

Ez természetesen az adott programtól függ. A fenti sorra elsősorban azért lehet szükség, mert – amint azt bizonyára megfigyeltük már – a legtöbb program alaphelyzetben a `/usr/local` könyvtárba települne. Ezt viszont nem akarjuk, mert azt szeretnénk, hogy mások is tudják használni a csomagot, a `/usr/local` pedig alapvetően nem jöhet szóba a mások által is használható csomagok kapcsán. Ennek a részletei a fájlrendszer Szabványban (FHS <http://www.pathname.com/fhs/>) található meg. Kikerüljük tehát a `/usr/local`-t, noha nem is oda fogjuk installálni a programot a fordítás során. A konfigurálást követően következhet a program lefordítása. Ismét figyeljünk oda arra, milyen követelményeket támaszthat a program. Ha a Slackware-nek nem része egy library, amit a program használ, akkor a program nem fog működni a többi felhasználónál. Az a legjobb, ha a fordításokat mindig „tisztá telepítésű” gépen végezzük, az idő múlásával ugyanis egyre kevésbé fogjuk tudni, mint telepítettünk fel az alaprendszeren kívül. Ezt a problémát kétféleképpen lehet áthidalni. Készíthetünk egy másik csomagot, ami a szükséges library-t tartalmazza (ez a jobb), vagy beletehetjük a library-t a program csomagjába (ami nem feltétlenül szerencsés, mivel később problémákat okozhat). A program lefordításához egyszerűen adjuk ki a `make` parancsot. Kövessük figyelemmel a fordítás folyamatát, ugyanis néha előfordul, hogy a könyvtárak, útvonalak explicit módon szerepelnek a program kódjában, és megtörténhet, hogy egy konfigurációs fájl nem az általunk a `prefix`-ben megadott útvonalra kerül. Ilyenkor valahogyan oldjuk meg, hogy ez ne így legyen. Megfordult már nálam néhány program, aminek a forrását kellett módosítani ahhoz, hogy elkerüljem a fenti problémát. A program lefordítása után következik még néhány lépés, mellyel a helyes jogosultságokat állíthatjuk be még a csomag elkészítése előtt. A következő parancsok kijavítják a forrásban esetleg rosszul szereplő jogosultságokat, hogy amikor a csomagba másoljuk a dokumentációt és egyébeket, akkor már a helyes jogosultságokkal kerüljenek a helyükre.

```
chown -R root.root .
```

```
find . -perm 777 -exec chmod 755 {} \;
```

```
find . -perm 555 -exec chmod 755 {} \;
```

```
find . -perm 444 -exec chmod 644 {} \;
```

```
find . -perm 666 -exec chmod 644 {} \;
```

```
find . -perm 664 -exec chmod 644 {} \;
```

A program lefordítása után átadunk neki még egy argumentumot, amivel ténylegesen is áthelyezzük a munkaterületünkre. A legjobb a `DESTDIR=` alkalmazása, ám ezt nem minden `make fájl` szereti. Én három lehetőséget ismerek: `DESTDIR=/work/builds`, `ROOT=/work/builds` és `prefix=/work/builds/usr`.

```
make install DESTDIR=/work/builds
```

Ez a legtöbb programmal működni fog, előfordulhatnak azonban esetek, amikor nem. A `make install` során figyeljük, hogy minden a `/works/builds/usr` alá került-e, nem pedig a `/usr` alá – néha előfordul, hogy a program egyes részei a megfelelő helyre kerülnek, más részei viszont a `/usr` alá. A `make install` végeztével nézzük meg a gyökérkönyvtárat és a `/usr`-t, hogy tényleg nem került-e semmi beléjük – én általában a fájlok dátumát ellenőrzöm. Ha találunk valamit a `/usr`-ben, egyszerűen helyezzük át a `/work/builds/usr`-be – de ellenőrizzük, hogy ott ugyanúgy helyezkedik el a `/work/builds/usr`-hez képest, mint korábban a `/usr`-ben a `/usr`-hez képest. (Azaz ha a `/usr/bin`-ben

volt, akkor a /work/builds/usr/bin-be kerüljön, stb. – *a ford.*). Ha ez megvan, létre kell hoznunk a dokumentáció helyét. Ez mindig a /work/builds/usr/doc lesz. Hozzuk létre ezt a könyvtárat, majd hozzunk benne létre egy újabb alkönyvtárat a program nevével és verziószámával. Mintának nézzük meg a /usr/doc könyvtárat. A program forrásából másoljuk a /work/builds/usr/doc-ba a dokumentációt, a licenszeket (GPL, stb.) és a readme/olvasfeljlokat.

Az installálás folyamatának figyelésére remek eszköz a linuxpackages.net-en megtalálható *installwatch* program, ami két változatban létezik. Ha terjeszteni szeretnénk a csomagunkat, ne azt használjuk, ami el is készíti a csomagot, ugyanis nem helyezi el benne a dokumentációt és még egy-két dolgot. Az *installwatch* naplózza az installált fájlokat, és a naplófájlban könnyen ellenőrizhetjük, hogy minden a helyére került-e. Akkor is hasznos segédeszköz, amikor a *make install prefix=* nem működik – ami nagyon gyakori a KDE programok esetében. Ha tudjuk, hogy hová kerültek az állományok az installálás során, könnyűszerrel átmásolhatjuk őket a munkaterületünkre. Az áthelyezés közben azonban figyeljünk oda a jogosultságokra és tulajdonosokra.

Van még néhány fájl, amit a csomag /install könyvtárában kell elhelyezni: a *doinst.sh* és a *slack-desc*. Nem minden csomagban lesz *doinst.sh*. A *doinst.sh*-t létrehozza a *makepkg* is, ha szimbolikus linkek vannak a csomagban. Nem kötelező eltávolítani ezeket a linkeket, de mindenképpen jó ötlet. Ha az installálás után bármilyen különleges parancsot kell végrehajtani, ezeket a parancsokat a *doinst.sh*-ban kell elhelyezni. A *doinst.sh* fájl használatának részletes, példákkal illusztrált leírása a [doinst.sh](#) oldalon található. Minden csomagban szerepelnie kell a *slack-desc* fájlnek, ugyanis ez látja el a csomagkezelő-rendszert a csomaggal kapcsolatos információkkal. A *slack-desc* leírása a [slack-desc fájl](#) oldalon található.

OK, tehát ott van a program a /work/builds-ben – készítsünk csomagot belőle! Ezen a ponton sokan elkövetik azt a hibát, hogy nem állítják be a munkakönyvtár tulajdonosának a root-ot, és nem állítják be a megfelelő jogosultságokat. Ne feledjük, hogy a csomag a fájlrendszer gyökerébe fog kerülni. A legbiztosabb megoldás az, ha a /work/builds könyvtárba belépve kiadjuk a *chown root.root* . és a *chmod 755* . parancsokat. Ezután írjuk be: *makepkg név-verzió-arch-build.tgz* parancsot, ami a könyvtár tartalmából létrehozza a csomagot. Ügyeljünk arra, hogy a szimbolikus linkek helyesek, és a nem a /work/builds/usr-be mutatnak. Amikor a *makepkg* felkínálja, hogy törli és újra létrehozza a szimbolikus linkeket, válaszoljunk igennel. Válaszoljunk igennel a tulajdonos és a csoport jogosultságok beállítására vonatkozó kérdéssel is, hacsak nem ragaszkodunk az általunk beállított értékekhez. Ezzel mindössze annyi a probléma, hogy a /usr/bin könyvtár és a benne lévő fájlok tulajdonosának elméletileg a bin csoportnak kellene lennie. Ha igent mondunk, ezek a jogosultságok megváltoznak. A *makepkg* lefutása után elkészül a csomagunk. A legegyszerűbben úgy tesztelhetjük, hogy a *pkgtool*-al installáljuk. Ha működik, akkor elkészítettük első csomagunkat, és megoszthatjuk a világgal. A csomag belsejébe is belenézhetünk: a *tar tzvf program-verzió-arch-build.tgz* parancs kilistázza a csomagban található fájlokat tulajdonosaikkal és jogosultságaikkal együtt.

## Fájlnevek

Használjuk következetesen a fájlneveket. A felhasználók így kényelmesen használhatják az *upgradepkg*-t a csomagjain frissítéséhez, ha új változatot készítünk belőlük. Ellenkező esetben teleshírdelhetjük az egész rendszert, mivel az új installálások nem fogják eltávolítani a régiakat. Néha nem is kell teljesen eltávolítani a korábbi változatokat, például ha meg akarjuk őrizni a konfigurációs fájlokat. Ezért érdemes a szabványos névadási konvencióhoz ragaszkodni. A helyes formátum: *alkalmazás-verzió-i386-lxxx.tgz*. További információ a [tökéletes csomag](#) oldalon található.

## Függőségek

Amint azt korábban már említettük: ha egy csomagnak olyan library-kre van szüksége, melyek nem részei a Slackware alaprendszernek, akkor vagy készítsünk azokból külön csomagot, vagy helyezzük el őket a program csomagjában. Mindig érdemesebb önálló csomagot készíteni a library-kból, hacsak nem kizárólag az az egy program használja őket, amikhez tartoznak. Ha egy csomagnak olyan library-re van szüksége, ami nem része a standard Slackware-installációnak, viszont opcionálisan installálható, akkor ezt és minden egyéb információt tüntessük fel egy readme/olvassal fájlban, és helyezzük el a csomag mellett, ha feltesszük valahová letöltésre. Emellett akárhová is tesszük fel a csomagot, a GPL szerint mindig el kell helyezni mellette a forrását is. Győződjünk meg arról is, hogy a program használata nem ütközik törvénybe. Végezetül értesítsük a program szerzőit arról, hogy elkészítettük a csomagot és elhelyeztük valahol.

Végezetül: a legjobb kezdés az, ha fejest ugrunk a közepébe. A Slackware forráscsomagokban található *build szkriptek*ből tanulhatunk a legtöbbet. Használhatjuk a [linuxpackages.net](http://linuxpackages.net) szkript-archívumát is, melyben a Slackware szkriptek módosított változatai találhatóak. Nézzük meg, ezek hogyan valósítják meg a csomagkészítést. És ne feledjük: tesztelni, tesztelni, tesztelni. Nyilván a barátaink lesznek a legjobb laboratóriumi egerek. Nem szeretnénk odajutni, ahová az RPM-világ jutott – láthatjuk, hogy nagyon sok RPM nem működik, mert létrehozóik nem követték a szabványos előírásokat.