

A tökéletes csomag

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

Szerző: TG tg.nospam@nospam.linuxpackages.net

Fordította: Békési József (jbekesi@axelero.hu)

Pdf változat: Tuza László (tuza_laszlo@ujcolor.hu)

A tökéletes csomag elkészítéséhez be kell tartanunk néhány szabályt. Néhány írott és néhány íratlan szabályt. Az alábbiakban ismertetünk ezek közül néhányat, abban a reményben, hogy olvasóink követni fogják azokat saját csomagjaik elkészítése során.

- Fájlnév
- Telepítési útvonalak
- Jogosultságok és tulajdonosok
- A slack-desc fájl
- Dokumentáció
- Vegyes
- Opcionális dolgok

Fájlnév

Joggal tehető fel a kérdés, hogy miért olyan fontos a fájlnev. Nos azért, mert nagyon sok mindent elárul a csomagról. A fájl nevéből megtudjuk a program nevét, a verzióját, hogy milyen architektúrára fordították, a build verzióját, és azt, hogy ki készítette a csomagot. Gondoljunk ezekre az elemekre úgy, mint egy űrlap mezőire. Van névmező, verziómező, arch(itektúra) mező és build mező.

apache-1.2.1-i386-2jim.tgz

A fenti példában láthatjuk, hogy ez az apache (névmező) 1.2.1-es verziója (verziómező). Azt is láthatjuk, hogy Intel 386 platformra készült a csomag (architektúra mező). Ebben a mezőben a következő értékek szerepelhetnek:

i386
i486
i586
i686
x86_64
noarch (ami azt jelenti, hogy a csomag nem platformfüggő)
PPC

Mivel a Slackware esetében nincs sparc vagy egyéb támogatott platform, csak ezekre lesz szükségünk. A verziómezőn belül ne használjunk „-”-t, mert ezzel megnehezítjük az upgradelést: a csomagkezelő nem fogja tudni meghatározni a csomag verziószámát. Ne feledjük, hogy a Slackware csomagkezelője balról jobbra haladva olvassa a csomag nevét, és a „-”-t kezeli mezőhatárként.

A következő mező a build mező. Ez általában azt jelenti, hányadik alkalommal fordították le a csomagban lévő programot, vagy hányadik alkalommal állították össze a csomagot, s így ugyanannak a verziószámú programnak is létezhet egy másik változata. A fenti példában a build 2.

Ez jelezhet egy javítást, vagy azt, hogy valami megváltozott az -1 megjelenése óta, s ez egy update. A build számát követheti három karakter is, amivel a csomag készítője saját magát jelöli. Esetünkben ez jim. Ez meglehetősen fontos lehet a különböző csomagkezelők számára, és segít megkülönböztetni a hivatalos és a nem hivatalos csomagokat. Emellett segít a felhasználónak is tudatni a csomagkezelővel, hogy mi hivatalos és mi nem. Olyan ez, mintha márkajeggyel látnánk el a csomagokat.

Telepítési útvonalak

Nem fogom még egyszer áttekinteni a HOGYAN-ban már említett installációs könyvtárakat. Egyszerűen fogalmazva, a következőknek kell meglenniük:

```
./ (enélkül nem lehet rendesen eltávolítani a csomagot)
/install
/install/slack-desc
/install/slack-required (opcionális, lásd lentebb)
/install/slack-conflicts (opcionális, lásd lentebb)
/install/slack-suggests (opcionális, lásd lentebb)
/usr/doc/package-version/
```

És természetesen szükség van még azokra az alkönyvtárakra, amelyekbe a program kerül. A doc könyvtárban helyezük el a licenszet és dokumentációt, a readme-eket, stb.

Az alábbiakban röviden bemutatjuk azokat területeket és céljukat, amelyekre a fájlok nagy része kerül.

/etc

Ez a terület főleg a konfigurációs fájloké. A `-sysconfdir=/etc` konfigurációs paranccsal biztosítjuk, hogy a csomagban lévő konfigurációs fájlok ide kerüljenek. Ha egyszerűen csak a `./configure --prefix=/usr` parancsot futtatjuk, és a csomagnak vannak konfigurációs fájljai, akkor azok valószínűleg a `/usr/etc`-be fognak kerülni. Ezt pedig nem szeretnénk. Néha még az is előfordulhat, hogy a konfigurációs fájlokat a `/etc` valamelyik alkönyvtárában akarjuk elhelyezni. Megfigyelhetjük, hogy főleg a gtk és gnome csomagok teszik ezt. Példa: `/etc/sajatprogram`.

/usr

Ez a rendszer magja, ide kell installálni a legtöbb bináris fájlt. A legtöbb esetben a `--prefix=/usr` alkalmazása biztosítja, hogy a program ide kerüljön telepítéskor.

/var

A var terület fájlok, logok, pidek számára van fenntartva – azon fájlok számára, amiket a programok hoznak létre. Ennek a konfigurációs előtagja a `./configure --localstatedir=/var`

Felmerülhet a kérdés, hogy miért ilyen nagy gondot fordítani arra, hogy hová kerülnek telepítéskor a csomagok és a kiegészítő fájlok. Ennek számos oka van, melyek közül a legfontosabb az egységesség. Jó példa az, hogy szeretnék létrehozni egy olyan rendszert, ami szinte teljes egészében csak olvasható módban van csatolva. Ha betartom az egységességet megtartani hivatott szabályokat, akkor nagyon könnyen megtehetem ezt. Egyszerűen csak annyit kell tennem, hogy bizonyos területeket írható és olvasható, a rendszer magját pedig csak olvasható módban csatolom. Szeretnénk szabványos és egységes csomagokat készíteni a felhasználók számára, és ez néha azt

jelenti, hogy a megszokott *make install*-on kívül el kell még végeznünk egy-két feladatot. Ha azonban helyes argumentumokat adunk át a *./configure*-nak és használjuk például a *checkinstall*-t, akkor az esetek 99%-ában jó csomagot fogunk készíteni.

Jogosultságok és tulajdonosok

A jó csomagokhoz elengedhetetlenek a helyes jogosultságok. Ez magában foglalja a fájlok jogosultságait illetve felhasználó- és csoporttulajdonosait. A fájlrendszer egyes területeinek speciális csoport hozzáférési jogosultságai vannak, amiket nekünk is követnünk kell. Általában véve a könyvtáraknak 0755-nek (drwxr-xr-x) kell lennie. A különböző fájl típusok más és más jogosultságokkal rendelkeznek. Jól alkalmazható szabály az, hogy ha egy fájlt olvasni kell, akkor használjuk a 0644-et (-rw-r--r--) – ilyenek a dokumentumok, az ikonok és az egyéb kiegészítő fájlok. A futtatható állományoknak 0755-nek (-rwxr-xr-x) kell lenniük. Ez azonban csak általános szabály, nem feltétlenül igaz minden egyedi esetben. A csomagok fájljai általában a root felhasználó és a root csoport tulajdonai. Ez alól a bin terület elemei a kivételek. Ezen a területen minden fájlnak és könyvtárnak a root felhasználóhoz és a bin csoporthoz kell tartoznia. Ezek a területek:

```
/usr/bin  
/bin  
/sbin  
/usr/sbin  
/usr/X11R6/bin
```

A jogosultságok és a tulajdonosok fontos részei a csomagoknak. Ha rászánjuk az időt a helyes beállításukra, jó csomagot fogunk készíteni.

A slack-desc fájl

A slack-desc fájl azért nagyon fontos, mert ez látja olyan információkkal a csomagadatbázist, amiket mi is látunk. Emellett bizonyos eszközöknek a csomag azonosításában is segít. A formátumának leírása itt található: [slack-desc](#). A linuxpackages.net-en van egy [online segédeszköz](#) is, ami segít létrehozni a tökéletes slack-desc fájlt.

Dokumentáció

A létező licenzek szinte mindegyike előírja, hogy a csomag tartalmazza a licenst is. Ezért követeljük meg a /usr/doc/csomag alkönyvtárat. Hiszen szabad szoftverekről van szó, és mindössze néhány szabályt kell betartanunk! Ebben a könyvtárban emellett lennie kell valamilyen információnak a program használatáról is, már ha ez részét képezte magának a programnak. Ez az alkönyvtár megfelelő hely a minta konfigurációs állományok elhelyezésére is, ha a programnak szüksége van rájuk. A dokumentáció kerülhet a /usr/share/doc/csomag könyvtárba is, mivel ez egy a /usr/doc-ra mutató szimbolikus link. Egyes programok automatikusan ide helyezik a dokumentációjukat, és ez el is fogadható így.

Vegyes

Van néhány dolog, ami hosszú távon sokat segíthet mind a tökéletes csomagok készítésében, mind a rendszerünk üzemeltetésében. Az egyik fontos, napjainkban egyre gyakrabban felmerülő kérdés a fordítási opcióké. Ne fordítsuk a programokat i686 opcióval, mert azt szeretnénk, hogy mindenki használja őket, márpedig nem mindenkinek van i686-ja. A Slackware esetében jelenleg az i486 az

alapértelmezett beállítás. Ugyanakkor az, hogy egy i686-on fordítunk le valamit, nem jelenti automatikusan azt, hogy a csomagunk i686 lesz optimalizálva. A fordítás optimalizálását manuálisan kell beállítanunk a gcc számára. Ezt sokan nem értik meg. Még egy megjegyzés ezzel kapcsolatban: a legtöbb esetben nincs emberi szemmel érzékelhető különbség egy i486 és i686 optimalizálás között. Mi több, a net-en rengeteg arra utaló információ található, hogy a túlzott optimalizálás akár le is lassíthatja az alkalmazást. A tökéletes csomag elkészítéséhez pedig kövessük az eddigiekben ismertetett irányelveket. A csomag létrehozása után a *tar* paranccsal ellenőrizzük, nézzük át, és győződjünk meg róla, hogy legalább a jogosultságoka és a struktúra rendben van – pl: *tar tfvz csomagnev.tgz*. Aztán szánjunk még egy kis időt a következőkre:

gzip-peljük a man oldalakat

gzip-peljük az info oldalakat, távolítsuk el a csomagól a */usr/info/dir* fájlt, ha tartalmazza. Ez az info rendszer fő indexfájlja, és nem szabad lecserélnünk a csomagéra. Ha ki akarjuk egészíteni, a linuxpackages.net-en található doinst-hogyan tájékoztat az info rendszerről és az új info fájlok felvételéről.

strip-peljük a binárisokat és az archívumokat (root-ként, még a *makepkg* előtt adjuk ki a következő parancsokat):

```
find . | xargs file | grep "current ar archive" | cut -f 1 -d : | xargs strip --strip-debug
```

```
find . | xargs file | grep "executable" | grep ELF | cut -f 1 -d : | xargs strip --strip-unneeded
```

```
find . | xargs file | grep "shared object" | grep ELF | cut -f 1 -d : | xargs strip --strip-unneeded
```

Ellenőrizzük a jogosultságokat és a tulajdonosokat.

Opcionális dolgok

Először is, annak ellenére, hogy a következőkben tárgyaltak nem kötelezőek, a linuxpackages.net is úgy írta meg a backendjét, hogy információt szerezzen ezekből a fájlokból a csomagokról. Ez volt a leglogikusabb megoldás, hiszen nem futthatjuk le az *ldd*-t minden csomagon, hogy megszerezzük ezeket az információkat. Két fontosabb eszköz van forgalomban, mindkettő a függőségek ellenőrzését támogatja. A *swaret* beépített ellenőrzőrendszert használ, ami a honlapján található adatbázisra támaszkodik. A *swaret*-nek nincs szüksége semmilyen különleges módosításra ahhoz, hogy elvégezze a feladatát. Ha azonban azt szeretnénk, hogy a csomagunkat minden eszköz (és a linuxpackages.net backendje is) kezelni tudja, kisebb módosításokat kell végrehajtani a csomagon. A */install* könyvtárban helyezhetjük el a *slack-required*, *slack-conflicts* és *slack-suggests* fájlokat. Ezekre akkor van szükség, ha a program valami olyan library-t vagy más programot használ, ami nem része a szabványos Slackware rendszernek. Ugyanakkor ha a szóban forgó program vagy library nem szabványos, vagy a hivatalosnál magasabb verziószámú, akkor fel kell tüntetni. Ez a függőségekhez való egyértelmű hozzáállás, hiszen sok esetben nem csak library-k képezhetnek függőséget, hanem más alkalmazások is, mely utóbbiak viszont nem generálnak szabványos hiányzó library hibát. A */install*-ban elhelyezhetjük még a *slack-conflicts* fájlt is, amiben azokat a csomagokat sorolhatjuk fel, melyek ütköznek az elkészített csomaggal. Például attól függően, hogy hogyan telepítjük őket, a *firefox* és a *mozilla* ütközésbe kerülhet, mivel felülírhatják egymás létfontosságú library-jeit. Az utolsó fájl, amit a */install*-ban elhelyezhetünk, a *slack-suggests*, ami egyszerűen felsorolja azokat az opcionális csomagokat, amik működhetnek a programmal. Például az *evolution* nem követeli a *gnome-spell*t, de ha fent van a rendszeren, akkor használja, és az alkalmazások képesek lesznek a helyesírás-ellenőrzésre. Megfigyelhetjük, hogy a csomagok ilyenén

való megközelítése a debian-féle megközelítéshez hasonlít. Nem akarjuk megváltoztatni a Slackware csomagkezelő eszközeit, ezért ilyen kerülőutas megoldásokat használunk. És épp ezt teszik lehetővé ezek az opcionális fájlok.

slack-required

A fájl formátuma: egy bejegyzés per sor, a következő alakban:

```
csomag_név  
vagy  
[csomag_név][feltétel][verzió]  
ahol a [feltétel]  
=, >=, <=, < vagy > (a <= és a >= egyaránt működik).
```

Például:

```
man >= 1.51-i386-1
```

Figyeljük meg, hogy a verziószámot, az architektúrát és a build számot is feltüntettük, ez ugyanis segít a verziók ellenőrzésében. A feltétel pedig gondoskodik mindenről, ha megjelenik egy újabb változat.

slack-conflicts

A fájl formátuma ugyanaz, mint a slack-required fájlé.

slack-suggests

Ebben a fájlban pusztán az alkalmazás nevét kell feltüntetni, semmi mást.

```
gnome-spell
```

Nekünk, a csomagkészítőknek mindenki másnál jobban kell tudnunk, hogy mi kell egy csomaghoz. És ne feledjük: lehetőleg egy tiszta Slackware rendszeren készítsük a csomagjainkat, ne használjuk a *-current* ágat vagy a dropline gnome-ot, különben a csomagjaink szinte biztos, hogy nem fognak működni egy hivatalos release-en – hacsak nem vagyunk teljesen biztosak abban, hogy milyen függőségeket támaszt a csomagunk.

Sok e-mailt kaptunk a buid szkriptekkel kapcsolatban, ezért fűztük hozzá ehhez a dokumentumhoz a következő kiegészítést. Mivel pillanatnyilag ezeknek a szkripteknek nincs előírt és egységes helye, sokakat megkérdeztünk a közösségben, és arra a döntésre jutottunk, hogy a build szkripteket a `/usr/src/slackbuilds` alkönyvtárban helyezzük el. Ha a csomagunkat build szkript segítségével készítettük el, helyezzük el a csomagban a szkriptet is, a `/usr/src/slackbuilds/csomagneve` alkönyvtárban. Mivel ezek a szkriptek általában kicsik, nem fogják jelentősen megnövelni a csomag méretét.

Hát ennyit a tökéletes csomagról. Minden kiegészítést és véleményt örömmel fogadok.