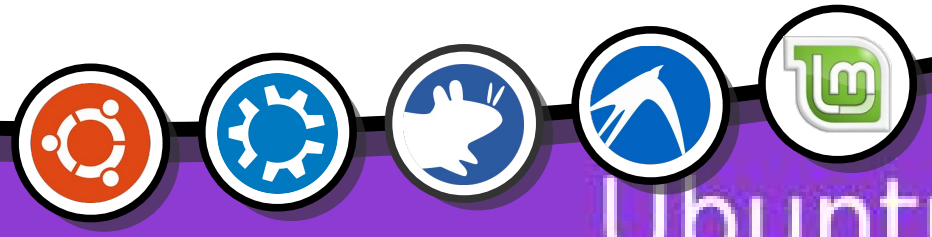
Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

UBUNTU FEJLESZTÉS SOROZAT KÜLÖNKIADÁS

UBUNTU FEJLESZTÉS SOROZAT KÜLÖNKIADÁS



EGY RÉSZES ÖSSZEÁLLÍTÁS A MAGAZIN 49.-52. SZÁMAIBÓL

A Full Circle Magazin különkiadása

Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

Üdvözöllek egy újabb “egyetlen témáról szóló különkiadásban”

Válaszul az olvasók igényeire, néhány sorozatként megírt cikk tartalmát összegyűjtjük dedikált kiadásokba.

Most ez a **"Hogyanok-Az Ubuntu fejlesztése"** című 4 részes sorozat Daniel Holbach tollából. (a magazin 49.-52. számaiból)

Kérlek, ne feledkezz meg az eredeti kiadási dátumról. A hardver és szoftver jelenlegi verziói eltérhetnek az akkor közöltektől, így ellenőrizd a hardvered és szoftvered verzióit, mielőtt megpróbálod emulálni/utánozni a különkiadásokban lévő ismertetőket. Előfordulhat, hogy a szoftver későbbi verziói vannak meg neked, vagy érhetők el a kiadásod tárolóiban.

Jó szórakozást!

Bevezetés az Ubuntu fejlesztésbe

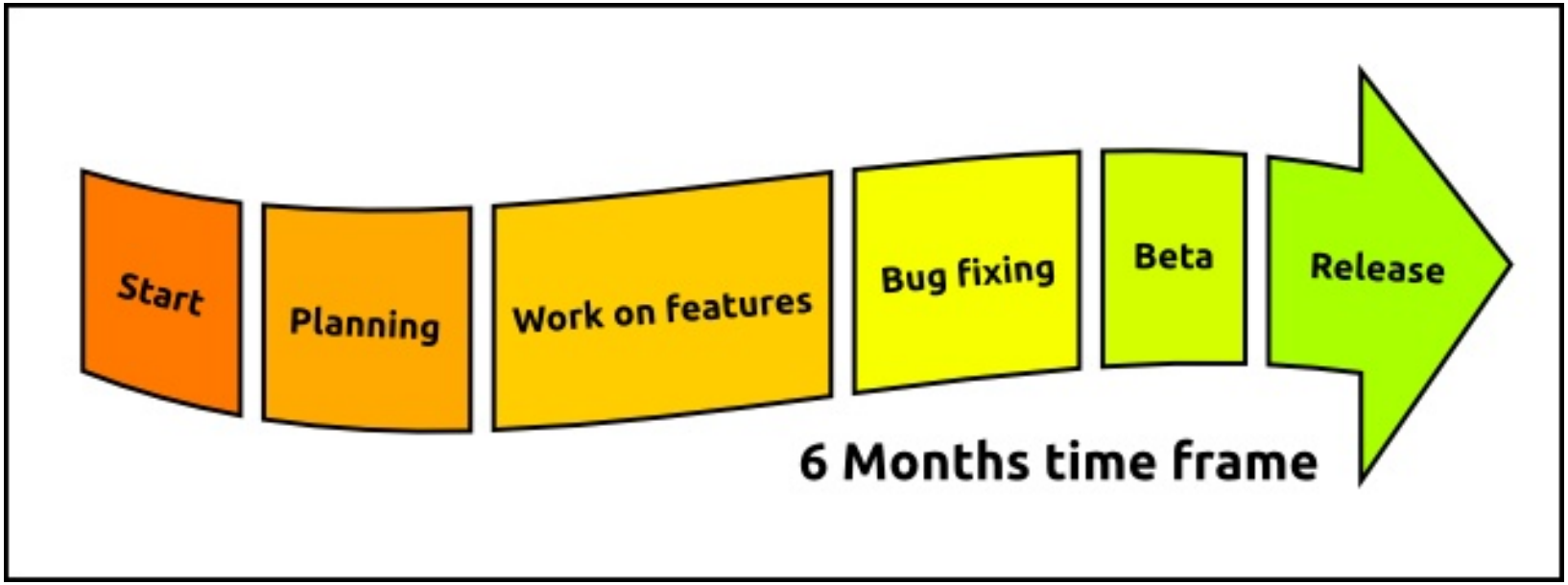
Írta Daniel Holbach

Az Ubuntu különböző programozási nyelven megírt különböző komponensek ezreiből készült. Minden komponens - szoftverkönyvtár, eszköz vagy grafikus alkalmazás - forráscsomagként elérhető. A forráscsomagok a legtöbb esetben két részből állnak: a tényleges forráskódból és a metaadatból. A metaadat tartalmazza a csomag függőségeit, a copyright és licenc információkat és a csomag fordításához szükséges utasításokat. Ha ezt a forráscsomagot lefordítottuk, a fordítási folyamat bináris csomagokat nyújt, ezek azok a .deb fájlok, amelyeket a felhasználók telepíthetnek.

Mindig, amikor egy alkalmazás új verzióját kiadják, vagy valaki módosítja az Ubuntuba kerülő forráskódot, a forráscsomagot fel kell tölteni a fordítógépekre a fordításhoz. Az eredményként létrejövő bináris csomagokat elosztják az archívumba és a különböző országokban lévő tükreire. Az /etc/apt/sources.list-ben lévő URL-címek archívumra vagy tükörre mutatnak. Minden nap fordítanak CD képeket a különböző Ubuntus sajátosságok választékához. Az Ubuntu Desktop, Ubuntu Server, Kubuntu és más ablakkezelők meghatározzák a CD-re kerülő igényelt csomagok listáját. Ezeket a CD képeket aztán telepítési tesztekhez használják és visszajelzést adnak a további kiadási tervekhez.

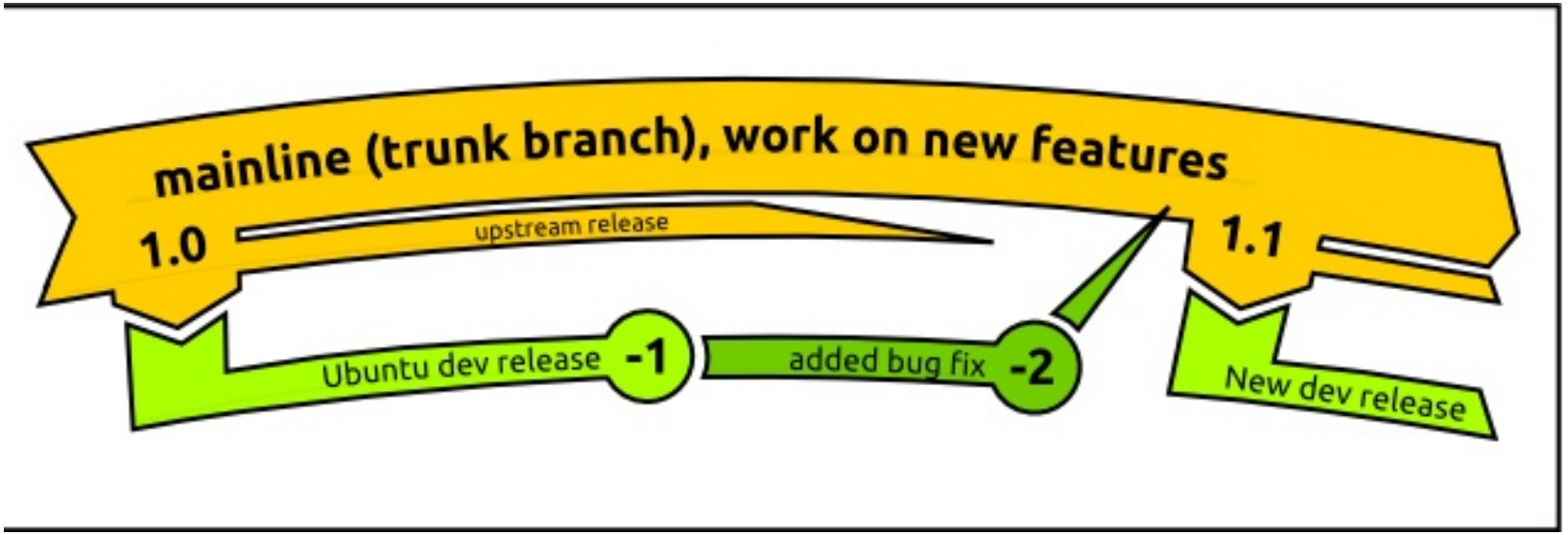
Az Ubuntu fejlesztése erősen függ a kiadási ciklus aktuális szakaszától. Új Ubuntu verziót minden hatodik hónapban adunk ki, ami csak azért lehetséges, mert szigorú fagyasztási dátumokat állapítottunk meg. Minden elért fagyasztási dátummal azt várjuk a fejlesztőktől, hogy kevesebb, kevésbé tolakodó módosításokat hajtsanak végre. A Feature Freeze az első nagy fagyasztási dátum, miután a ciklus első fele letelt. Ebben a szakaszban a funkciók többségét meg kell valósítani. A ciklus hátralévő részének feladata a hibajavításra való fókuszálás. Ezután a felhasználói interfészt, majd a dokumentációt, a kernelt, stb. fagyasztják, majd közzéteszik a béta kiadást, amely sok tesztelést kap. A béta kiadástól kezdve csak a kritikus hibákat javítják és egy kiadásra jelölt kiadást készítenek, és ha az nem tartalmaz súlyos hibát, ez válik végleges kiadássá.

A forráscsomagok ezrei, a kódsorok billiói és a közreműködők százai sok kommunikációt és tervezést igényelnek, hogy fenntartsák a magasszintű minőséget. Minden egyes kiadási ciklus kezdetén megtartjuk az Ubuntu Developer Summit csúcstalálkozót, ahol a fejlesztők és közreműködők összegyűlnek, hogy megtervezzék a következő kiadások funkcióit. Minden funkciót az érintettjei vitatnak meg és írnak egy specifikációt, amely részletes információkat tartalmaz a követelményekről, implementációról, a más helyeken szükséges módosításokról, a tesztelés módjáról, stb. Ezt nyitott és átlátható módon teszik, így ha személyesen nem is tudsz részt venni az eseményen, távolról részt vehetsz és streamcast-ot hallgathatsz, csetelhetsz a résztvevőkkel és feliratkozhatsz a specifikáció-módosításokra, hogy



naprakész légy.

Így sem tudnak minden egyes módosítást megvitatni a találkozón, főleg azért, mert az Ubuntu támaszkodik a más projektekben végrehajtott módosításokra. Ezért a közreműködők állandóan kapcsolatban maradnak. A legtöbb csapat vagy projekt dedikált levelezőlistákat használ, hogy elkerülje a túl sok oda nem illő zajt. A közvetlenebb koordinációhoz a fejlesztők és közreműködők az IRC-et használják. Minden vita nyitott és nyilvános.



A másik fontos kommunikációs eszköz a hibajelentés. Ahányszor hibát találnak egy csomagban vagy az infrastruktúra egy darabjában, rögzítenek egy hibajelentést a Launchpad-en. Minden információt összegyűjtenek abban a jelentésben, és szükség szerint frissítik a fontosságát, állapotát és az illetékes személyt. Ez hatékony eszközzé teszi a csomagban vagy projektben található hibák követéséhez és a munkaterhelés szervezéséhez.

Az Ubuntun keresztül elérhető szoftverek többségét nem maguk a fejlesztők írják. A legtöbbjüket más nyílt forrású projektek fejlesztői írják, majd integrálják az Ubuntuba. Ezeket a projekteket “Upstream”-eknek nevezzük, mivel a forráskódjuk beáramlik az Ubuntuba, ahová mi “csak” integráljuk. Az Upstream-ekkel való kapcsolat kritikusan fontos az Ubuntu számára. Nemcsak a kód, amit az Ubuntu kap az Upstream-ektől, de azok a felhasználók, hibajelentések és foltozások is, amit az Upstream-ek kapnak az Ubuntutól (és más disztribúcióktól).

Az Ubuntu számára legfontosabb Upstream a Debian. A Debian az a disztribúció, amelyen az Ubuntu alapul, és a csomagolási infrastruktúrát tekintve sok tervezési döntés innen származik. A Debiannak hagyományosan mindig minden egyes csomaghoz voltak kijelölt karbantartói vagy karbantartó csapatai. Az Ubuntuban olyan csapatok vannak, amelyeknek a csomagok egy részhalmazában is érdekeltek és természetesen minden fejlesztőnek van speciális szakterülete, de a részvétel (és a feltöltési jogok) mindenki számára nyitottak, aki erre képességet és hajlandóságot mutat.

Új közreműködőként Ubuntura váltani nem olyan ijesztő, mint amilyennek tűnik és nagyon kifizetődő tapasztalattá válhat. Ez nemcsak valami új és izgalmas dolog megtanulásáról szól, hanem a megoldás megosztásáról és a külső felhasználók milliónyi problémájának megoldásáról is.

A Nyílt forrású Fejlesztés különböző célokkal és különböző fókuszponti területekkel rendelkező elosztott világban történik. Például előfordulhat, hogy egy bizonyos Upstream-et érdekelheti egy nagy funkción való munka, míg az Ubuntut a szigorú kiadási ütemezés miatt abban lehet érdekelt, hogy csak egy kiegészítő hibajavítással ellátott stabil verziót szállítson. Ezért használjuk az “Elosztott Fejlesztést”, ahol a kódon több ágon dolgoznak, amelyeket a kód ellenőrzései és elegendő vita után egyesítenek egymással.

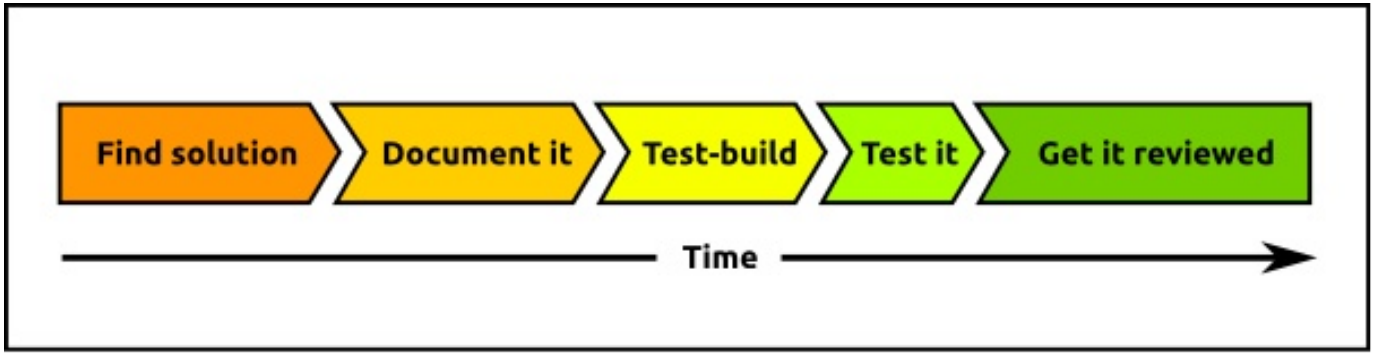
A fent említett példában annak lenne értelme, ha az Ubuntut a projekt létező verziójával szállítanák, hozzáadnák a hibajavítást, behelyeznék a következő kiadás Upstream-jébe és azt (ha alkalmas) a következő Ubuntu kiadásban szállítanák. Ez lenne a lehető legjobb kompromisszum és egy olyan helyzet, amin mindenki nyer.

Egy Ubuntuban lévő hiba kijavításához először megkapod a cso-

mag forráskódját, dolgozol a javításon, dokumentálod, hogy könnyen érthető legyen más fejlesztők és felhasználók számára, majd lefordítod a csomagot a teszteléshez. Miután letesztelted, könnyedén javasolhatod, hogy a módosítást tegyék bele a jelenlegi Ubuntu fejlesztési kiadásba. Egy feltöltési joggal rendelkező fejlesztő átnézi neked, majd integrálják az Ubuntuba. Amikor próbálunk megoldást találni, jó ötlet megnézni egy Upstream-et és azt, hogy a probléma (vagy egy lehetséges megoldás) ismert-e már és ha nem, adj bele mindent, hogy koncentrálva megoldd azt. További lépés lehet egy régebbi, de még támogatott verziójú Ubuntura visszaportolt módosítás megszerzése és áttétele az Upstreamre.

Az Ubuntu fejlesztésben elérhető siker legfontosabb feltétele, hogy rájöjjünk annak a nyitjára, "hogyan tehetünk dolgokat működőképessé" úgy, hogy nem félünk dokumentációt olvasni és kérdezni, csapattagok vagyunk és élvezzük a detektívmunkát.

A kérdéseid feltevéséhez jó hely az ubuntu-motu-mentors@lists.ubuntu.com és az **#ubuntu-motu** az irc.freenode.net-en. Könnyen találsz majd sok új barátot és olyan embereket, akikkel közös a szenvedélyetek: a világ jobbá tétele jobb Nyílt forrású szoftver készítésével.



Ubuntu Fejlesztés 2. rész – Set Up

Írta: Daniel Holbach

Sok dolgot kell tenned ahhoz, hogy elkezdhess Ubuntura fejleszteni. Ezt a cikket arra terveztük, hogy beállítsd a számítógépedet úgy, hogy elkezdhess csomagokkal dolgozni és a csomagjaidat feltölthesd a Launchpadra. Erről lesz szó:

• Csomagkezelő szoftver telepítése. Ez a következőket tartalmazza:
• Ubuntus csomagoló segédprogramok
• Titkosító szoftver, amivel a munkád ellenőrizheted, amint elkészülsz
• További titkosító szoftver, amivel biztonságosan tudsz fájlokat átvinni
• Launchpad-fiók létrehozása és beállítása
• A fejlesztőkörnyezeted beállítása úgy, hogy segítsen a csomagok helyi fordításában, a többi fejlesztővel való együttműködésben és a módosítások Launchpad-on történő közzétételében.

Megjegyzés: Javasoljuk, hogy a csomagolási munkát az Ubuntu jelenlegi fejlesztői verziójával végezd. Így lehetővé teszi, hogy ugyanabban a környezetben teszteld a módosításokat, ahol azokat valóban alkalmazni és használni fogják.

De ne aggódj, az Ubuntu fejlesztői kiadás wiki oldala (https://wiki.ubuntu.com/UsingDevelopmentReleases) több módszert is bemutat ahhoz, hogy magabiztosan használd a fejlesztői kiadást.

Alap csomagkezelő szoftver telepítése

Sok olyan eszköz van, amely Ubuntu fejlesztőként megkönnyíti az életedet. Később az útmutatóban találkozol ezekkel az eszközökkel. A legtöbb előírt eszköz telepítéséhez ezt a parancsot futtasd:

```
sudo apt-get install gnupg pbuilder ubuntu-dev-tools bzr-builddeb apt-file
```

Ez a parancs a következő szoftvert telepíti:

gnupg – GNU Privacy Guard olyan eszközöket tartalmaz, amik kellenek a titkos kulcs készítéséhez, amellyel a Launchpadra feltöltendő fájlokat fogod aláírni.
pbuilder – eszköz a csomag tiszta és elszigetelt környezetben végezhető, másolható fordításaihoz.
ubuntu-dev-tools (és devscripts, közvetlen függőség) – olyan eszköztár, amely könnyíti a csomagolási feladatokat.
bzr-builddeb (és bzr, függőség) – elosztott verziókezelő eszközök, amelyek sok fejlesztőnek megkönnyítik az ugyanazon kódon végzett munkát, míg az egymás munkájának egyesítése nyilvánvaló.
apt-file könnyű módszert nyújt adott fájlt tartalmazó bináris csomag keresésére.
apt-cache (az apt csomag része) még több információt nyújt az Ubuntus csomagokról.

A GPG kulcsod létrehozása

A GPG a GNU Privacy Guard rövidítése és az OpenGPG szabványt valósítja meg, amely az üzenetek és fájlok aláírását és titkosítását teszi lehetővé. Ez több célra használható. Esetünkben az a fontos, hogy a kulcsoddal fájlokat tudsz aláírni, így azok a te munkádként azonosíthatók. Ha feltöltesz egy forráscsomagot a Launchpadra, csak akkor fogadja el a csomagot, ha egyértelműen meg tudja határozni, ki töltötte fel azt.

Új GPG kulcs generálásához futtasd:

```
gpg --gen-key
```

A GPG először megkérdezi, milyen kulcsot akarsz generálni. Az alapértelmezett (RSA és DSA) kiválasztása jó lesz. Azután a kulcsméretet kérdezi. Az alapértelmezett (jelenleg 2048) jó, de a 4096 biztonságosabb. Utána megkérdezi, hogy akarod-e, hogy lejárjon a kulcs valamelyik fázisban. A „0” választása biztonságos, amely azt jelenti, hogy a kulcs soha nem jár le. Az utolsó kérdések a nevedet és e-mail címedet kérik. Itt olyan címet válassz, amelyet az Ubuntus fejlesztéshez fogsz használni, később még adhatsz hozzá címeket. Megjegyzés megadása nem szükséges. Majd be kell állítanod egy kulcsmondatot. Válassz biztonságosat.

A GPG most készít neked egy kulcsot, amely rövid időt vesz igénybe; véletlenszerű bájtokat igényel, ha adsz egy kis munkát a gépnek, az jó lesz. Mozgasd a kurzort.

Ha elkészült, a következőhöz hasonló üzenetet fogsz kapni:

```
pub  4096R/43CDE61D 2010-12-06
Key fingerprint = 5C28 0144 FB08 91C0 2CF3  37AC 6F0B F90F 43CD E61D
uid                  Daniel Holbach <dh@mailempfang.de>
sub  4096R/51FBE68C 2010-12-06
```

Ebben az esetben a 43CDE61D a kulcs ID.

Ezután fel kell töltened a kulcsod nyilvános részét egy kulcsszerverre, így tiedként fogják azonosítani az üzeneteket és fájlokat. Ehhez gépeld be:

```
gpg --send-keys <KEY ID>
```

Ez egyetlen kulcsszerverre küldi fel a kulcsodat, de a kulcsszerverek hálózata automatikusan szinkronizálja a kulcsokat egymás között. Ha ez a szinkronizáció befejeződött, az aláírt nyilvános kulcsod készen áll arra, hogy világszerte ellenőrizze a közreműködéseidet.

Az SSH kulcsod létrehozása

Az SSH a Secure Shell rövidítése, és olyan protokoll, amely lehetővé teszi a hálózaton keresztüli biztonságos adatcserét. Az SSH-t használhatjuk arra, hogy egy másik géphez hozzáférjünk és megnyissunk egy shellt, és biztonságos fájlátvitelre használjuk. A céljainkra főleg SSH-t fogunk használni, hogy biztonságosan kommunikáljunk a Launchpaddal.

Az SSH kulcs generálásához gépeld be:

```
ssh-keygen -t rsa
```

Az alapértelmezett fájlnév többnyire értelmes, így is hagyhatod. Biztonsági okokból erősen ajánlott, hogy kulcsmondatot használj.

A pbuilder beállítása

A pbuilder megengedi, hogy a saját gépeden fordítsd a csomagokat. Ez több célt szolgál:

• A fordítást minimális és tiszta környezetben végzed. Ez segít annak ellenőrzésében, hogy a fordításod sokszorosítva is működni fog, de nem módosítja a helyi rendszeredet.
• Nem kell minden szükséges fordítási függőséget helyben telepíteni.
• Több példányt is beállíthatsz a különböző Ubuntu és debian kiadásokhoz.

A pbuilder beállítása nagyon könnyű. Szerkesztésre nyisd meg az ~/.pbuilderrc állományt és add hozzá a következő sort:

```
COMPONENTS="main universe multiverse restricted"
```

Ez biztosítja, hogy a fordítási függőségeket minden komponens használatával kielégíted. Majd futtasd:

```
pbuilder-dist <release> create
```

ahol <release> pl. a natty, maverick, lucid, vagy, Debian esetén a sid. Több időt vehet igénybe, mivel minden „minimális telepítéshez” szükséges csomagot letölt. Ezek viszont bekerülnek a gyorsítótárba.

Beállítás a Launchpaddal való működéshez

Az elemi helyi konfigurálás helyett az a következő lépésed, hogy beállítsd a rendszered a Launchpaddal való működéshez. Ez a szekció a következő témákra fókuszál:

• Mi az a Launchpad, és a Launchpad fiók létrehozása
• A GPG és SSH kulcsaid feltöltése a Launchpadra
• Bazaar beállítása a Launchpaddal való működéshez
• Bash beállítása a Bazaarral való működéshez

A Launchpadról

A Launchpad az Ubuntuban általunk használt infrastruktúra központi darabja. Nemcsak a csomagjainkat és kódunkat tárolja, hanem olyan dolgokat is, mint a fordítások, hibajelentések és az Ubuntun dolgozó emberekről és azok csapattagságairól szóló információkat is. Arra is fogod használni a Launchpadot, hogy közzétedd a javasolt javításaidat, és elérj más Ubuntu fejlesztőket, hogy átnézzék és támogassák.

Regisztrálnod kell a Launchpadra és meg kell adnod egy minimális mennyiségű információt. Ez lehetővé teszi, hogy kódot tölts le és fel, hibajelentést tegyél közzé, stb.

Ubuntu fejlesztés 2. Rész – Set Up

Launchpad fiók létrehozása

Ha nincs Launchpad fiókod, könnyen létrehozhatsz egyet (itt: https://launchpad.net/+login). Ha már van Launchpad fiókod, de nem emlékszel a Launchpad azonosítódra, kitalálhatod, ha meglátogatod a https://launchpad.net/people/+me címet, és megkeresed a ~ utáni részt az URL-ben.

A Launchpad regisztrációs folyamata kérni fogja, hogy válassz egy megjeleníthető nevet. Javasoljuk, hogy itt a valódi neved használd, hogy az Ubuntu fejlesztő kollégáid jobban felismerhessenek.

Amikor új fiókot regisztrálsz, a Launchpad küld egy emailt egy linkkel, amelyet a böngésződben meg kell nyitnod, hogy ellenőrizzék az email címed. Ha nem kapod meg, nézd meg a levélszemét mappádban.

Az új fiók súgóoldala (https://help.launchpad.net/YourAccount/NewAccount) a Launchpadon több információt nyújt a folyamatról és a kiegészítő beállításokról, amelyeket módosíthatsz.

A GPG kulcsod feltöltése a Launchpadra

Hogy kitaláld a GPG ujjlenyomatodat, futtasd:

```
gpg --fingerprint <email@address.com>
```

és ez valami ilyesmit fog kiírni:

```
pub  4096R/43CDE61D 2010-12-06
Key fingerprint = 5C28 0144 FB08 91C0 2CF3  37AC 6F0B F90F 43CD E61D
uid                  Daniel Holbach <dh@mailempfang.de>
sub  4096R/51FBE68C 2010-12-06
```

Menj a https://launchpad.net/people/+me/+editpgpkeys címre és másold ki a „Kulcs ujjlenyomat” utáni részt a szövegdobozba. A fenti esetben ez a 5C28 0144 FB08 91C0 2CF3 37AC 6F0B F90F 43CD E61D. Kattints a „Kulcs importálására”.

A Launchpad arra használja az ujjlenyomatot, hogy megkeresse az Ubuntu kulcsszerveren a kulcsodat és ha sikerült, küld neked egy titkosított emailt, amelyben megkérnek, hogy erősítsd meg a kulcsimportálást. Nézd meg a levelezési fiókodat és olvasd el a levelet, amelyet a Launchpad küld. Ha a levelező kliensed támogatja az OpenPGP titkosítást, kérni fogja a kulcshoz választott jelszót. Írd be a jelszót, majd kattints a linkre, hogy megerősítsd, tiéd a kulcs.

A Launchpad a nyilvános kulcsod használatával titkosítja a levelet, hogy biztos legyen, hogy a kulcs a tied. Ha a levelezőprogramod nem támogatja az OpenPGP titkosítást, másold ki a titkosított levél tartalmát, gépeld be a gpg-t a terminálodba, majd illeszd be az email tartalmára terminálablakba.

Visszatérve a Launchpad honlapra, használd a Megerősít gombot és a Launchpad befejezi az OpenPGP kulcsod importálását.

Több információt itt találhatsz: https://help.launchpad.net/YourAccount/ImportingYourPGPKey

Az SSH kulcsod feltöltése a Launchpadra

Nyisd meg a https://launchpad.net/people/+me/+editsshkeys honlapot a böngésződben és a ~/.ssh/id_rsa.pub fájlt a szövegszerkesztőben. Ez az SSH kulcsod nyilvános része, így biztonságosan megosztható a Launchpaddal. Másold ki a fájl tartalmát és illeszd be a honlapon lévő szövegdobozba, amelynek felirata „SSH kulcs hozzáadása”. Most kattints a „Nyilvános kulcs importálása” gombra.

Ha több információt szeretnél a folyamatról, látogasd meg az SSH kulcspár készítése lapot (https://help.launchpad.net/YourAccount/CreatingAnSSHKeyPair) a Launchpadon.

A Bazaar beállítása

A Bazaar olyan eszköz, melyet arra használunk, hogy a kódmódosításokat logikus módon tároljuk, kicseréljük és egyesítsük a javasolt változásokat akkor is, ha a fejlesztés egyidejű. Hogy közöld a Bazaarral, ki vagy, egyszerűen futtasd:

```
bzr whoami "Bob Dobbs <subgenius@example.com>"

bzr launchpad-login subgenius
```

A whoami megmondja a Bazaarnak, milyen nevet és email címet kell használnia a megerősítő üzeneteidhez. A launchpad-loginnal beállítod a Launchpad ID-det. Így a Launchpadon közzétett kódot hozzád társítják.

Megjegyzés: Ha nem emlékszel az ID-re, menj a https://launchpad.net/people/+me címre és nézd meg, hová irányít át. Az URL „~” utáni része a Launchpad azonosítód.

Ubuntu fejlesztés 2. Rész – Set Up

A parancsértelmeződ beállítása

A Bazaarhoz hasonlóan a Debian/Ubuntu csomagkezelő eszközeinek is szükségük van arra, hogy felismerjenek. Egyszerűen nyisd meg a ~/.bashrc fájlodat a szövegszerkesztőben és add hozzá ezt az aljához:

```
export DEBFULLNAME="Bob Dobbs"

export DEBEMAIL="subgenius@example.com"
```

Most mentsd el a fájlt, és vagy indítsd újra a terminált vagy futtasd a parancsot:

```
source ~/.bashrc
```

(Ha az alapértelmezettől (bash) eltérő shellt használsz, kérlek szerkeszd át a konfigurációs fájlt annak megfelelően.)

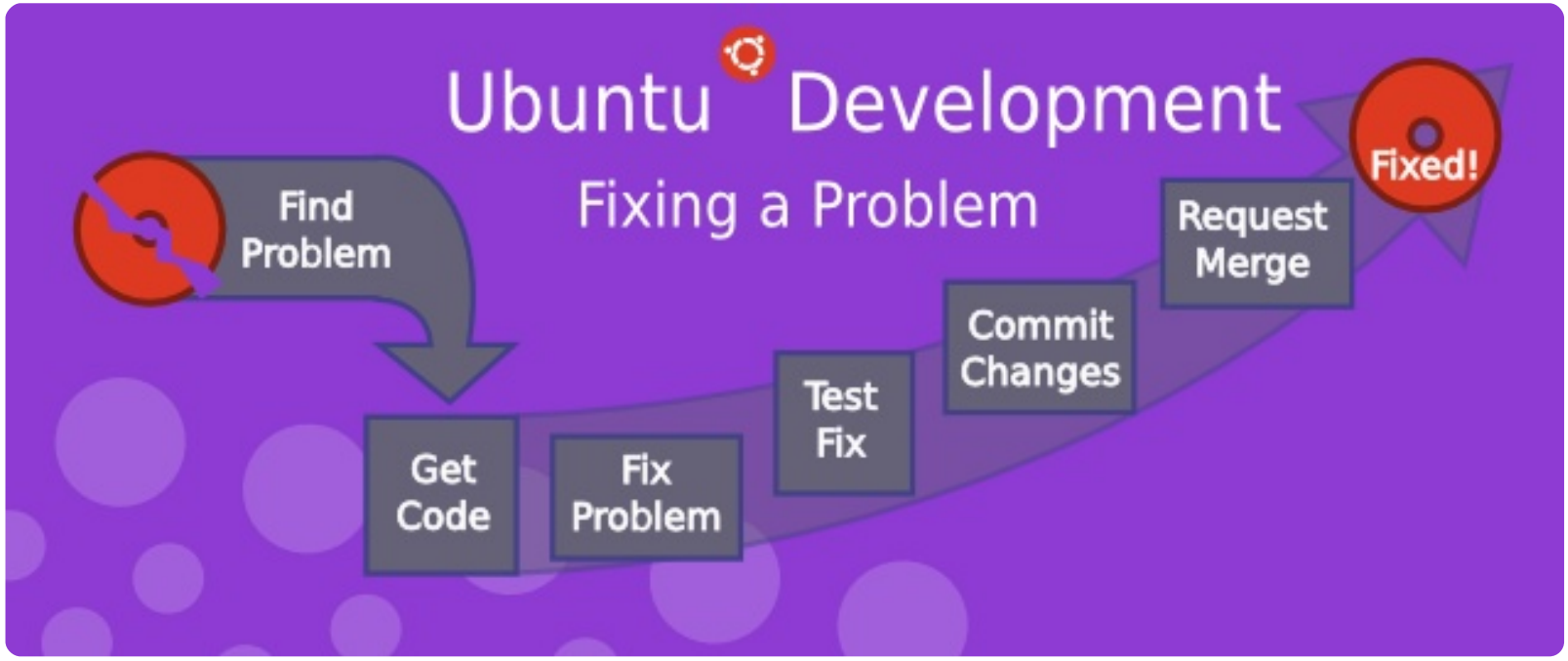
A KÖVETKEZŐ HÓNAPBAN: Javítsunk ki egy hibát!

Ubuntu fejlesztés – 3. rész – Hibajavítás

Írta: Daniel Holbach

Ha követted a Beállítás Ubuntu fejlesztéshez utasításait, akkor mindent beállítottál és készen állsz az indulásra.

Amint a fenti képen látod, nincs semmi meglepő az Ubuntu-hibák kijavításának folyamatában: találtál egy problémát, megkapod a kódot, dolgozol a javításon, teszteled, elküldöd a javításaidat a Launchpadra, és kéred azok ellenőrzését és beolvasztását. Ebben az útmutatóban egyenként végigmegyünk a szükséges lépéseken.



A probléma megtalálása

Sok különböző mód van arra, hogy megtaláld a javítandó dolgokat. Ez lehet egy olyan hiba jelentése, amivel találkoztál (amely jó alkalmat ad a javítás tesztelésére), vagy olyan probléma, amit máshol találtál, akár egy hibajelentésben.

A Harvest az a hely, ahol nyomon követjük az Ubuntu fejlesztéssel kapcsolatos különböző teendők listáit. Azokat a hibákat sorolja fel, amelyeket a forrásban vagy a Debianban már kijavítottak, a kis hibákat („falatkákat"), stb. Nézz körül és találd meg az első javítandó hibádat.

Találd ki, mi a hiba

Ha nem ismered a problémás kódot tartalmazó forráscsomagot, de tudod az érintett program útvonalát a rendszeredben, megkeresheted a forráscsomagot, amit ki kell javítanod.

Tegyük fel, hogy találtál egy hibát a Tomboy-ban, egy jegyzetkészítő asztali alkalmazásban. A Tomboy alkalmazás az /usr/bin/tomboy parancssori futtatásával indítható el. Az alkalmazást tartalmazó bináris csomag kereséséhez ezt a parancsot használd:

```
apt-file find /usr/bin/tomboy
```

Ennek kimenete a következő lesz:

```
tomboy: /usr/bin/tomboy
```

Jegyezd meg, hogy a kettőspontot megelőző rész a bináris csomag neve. Gyakori az az eset, hogy különbözik a forráscsomag és a bináris csomag neve. Az a leggyakoribb eset, amikor egyetlen forráscsomagot használnak több különböző bináris csomag fordításához. Adott bináris csomaghoz tartozó forráscsomag kereséséhez gépeld be ezt a parancsot:

```
apt-cache show tomboy | grep ^Source:
```

Ebben az esetben semmit nem ír ki, ami azt jelenti, hogy a bináris csomag neve is tomboy. A python-vigra egy példa arra, ahol a

forráscsomag és a bináris csomag neve eltér. Míg ez a bináris csomag neve, a forráskód neve valójában libvigraimpex és ezzel a paranccsal (és kimenettel) kereshető meg:

```
apt-cache show python-vigra | grep ^Source:

Source: libvigraimpex
```

A kód megszerzése

Ha már ismered a javítandó forráscsomagot, szeretnéd megkapni a kódjának a másolatát, hogy megkeresd benne a hibát. Ezt a forráscsomagnak megfelelő forráscsomag „elágaztatása" szerint végezheted el. A Launchpad minden Ubuntu csomaghoz forráscsomag ágakat tart fenn. Ha megkaptad a forráscsomag helyi ágát, megvizsgálhatod a hibát, készíthetsz egy javítást és egy Bazaar ág formájában feltöltheted a javasolt módosításodat a Launchpadra. Ha elégedett vagy a javításoddal, küldhetsz egy „beolvasztási kérelmet", amely felkéri a többi Ubuntu fejlesztőt a módosításod ellenőrzésére és jóváhagyására. Ha egyetértenek a módosításaiddal, egy Ubuntu fejlesztő feltölti a csomag új verzióját az Ubuntura, hogy mindenki élvezhesse a remek javításod előnyeit – és kapsz egy kis köszönetet. Most úton vagy afelé, hogy Ubuntu fejlesztővé válj! A következő szakaszokban leírjuk annak a sajátosságait, hogyan ágaztasd el a kódot, töltsd fel a javításodat és kérj ellenőrzést.

Munka a hibajavítással

Teljes könyveket írtak a hibák kereséséről, javításáról, teszteléséről, stb. Ha teljesen kezdő vagy a programozásban, próbálj meg először egyszerű hibákat, például nyilvánvaló elírásokat kijavítani. Próbáld minimalizálni, valamint érthetően dokumentáld a módosításaidat és a feltevéseidet.

Mielőtt dolgoznál egy javításon, győződj meg róla, hogy senki nem javította ki vagy nem dolgozik rajta.

Megfelelő források az ellenőrzéshez:
– Upstream (és Debian) hibakövető (megnyitott és lezárt hibák),
– Upstream javítástörténet (vagy újabb kiadás) kijavíthatta a problémát,
– Debian vagy más disztribúciók hiba- és csomag-feltöltései.

Ha találsz egy megoldást a probléma javítására, mondjuk egy hibajelentéshez csatolva, a forráskönyvtárban ezt a parancsot futtatva alkalmazni fogja a foltot:

```
patch -p1 < ../bugfix.patch
```

Fordulj a patch(1) man oldalhoz az opciók és argumentumok kereséséhez, mint például --dry-run, -p<num>, stb.

A javítás tesztelése

A tesztcsomag módosításaiddal való fordításához ezeket a parancsokat futtasd:

```
bzr bd -- -S -us -uc

pbuilder-dist <release> build ../<package>_<version>.dsc
```

Ez egy forráscsomagot fog készíteni az ág tartalmából (az -us -uc kihagyja a forráscsomag aláírásához vezető lépést) és a pbuilder-dist forrásból lefordítja a csomagot a választott kiadáshoz.

Ha a fordítás sikerül, telepítsd a csomagot a ~/pbuilder/<release>_result/ útvonalról (a sudo dpkg -i <package>_<version>.deb paranccsal). Majd teszteld, hogy megnézd, kijavult-e a hiba.

A javítás dokumentálása

Nagyon fontos, hogy elegendően dokumentáld a módosításodat, hogy a jövőben a kódot néző fejlesztőknek ne kelljen találgatniuk, mik voltak az okaid és feltevéseid. Minden Debian és Ubuntu csomag kódja tartalmazza a debian/changelog fájlt, ahol a feltöltött csomagok módosításait nyomon követik.

A legegyszerűbb mód ennek frissítésére:

```
dch -i
```

Ez létrehoz számodra egy szabványos változáslista-bejegyzést, és elindít egy szerkesztőt, ahol kitöltheted az üres mezőket. Példának használhatod:

```
specialpackage (1.2-3ubuntu4) natty; urgency=low
  * debian/control: updated description to include frobnicator (LP: #123456)
 -- Emma Adams <emma.adams@isp.com>  Sat, 17 Jul 2010 02:53:39 +0200
```

A dch előre kitölti számodra egy ilyen változáslista-bejegyzés első és utolsó sorát. Az első sor a forráscsomag nevéből, a verziószámból, az Ubuntu kiadásból, amire feltöltöd és a sürgősségből áll (amely

szinte mindig „alacsony"). Az utolsó sor mindig a nevet, e-mail címet és a módosítás időbélyegét tartalmazza (RFC 5322 formátumban).

Ezen kívül koncentráljunk magára a változáslista-bejegyzés tényleges tartalmára: nagyon fontos dokumentálni:
– hol történt a módosítás,
– mit módosítottunk,
– hol történt a módosítás megvitatása.

A (nagyon gyéren kitöltött) példánkban az utolsó pontot az (LP: #123456) fedi le, amely az 123456-os számú Launchpad hibára hivatkozik. A hibajelentések, levelezési listák szálai vagy a specifikációk általában jó információk ahhoz, hogy magyarázatot adjanak a módosításokra. Ráadásul ha a LP: #<number> megjegyzést használod a Launchpad hibáira, a hiba automatikusan lezárul, amikor a csomagot feltöltik az Ubuntura.

A javítás véglegesítése

A megírt és elmentett változáslista-bejegyzéssel csak futtasd ezt:

```
debcommit
```

és a módosítás (helyben) véglegessé válik véglegesítési üzenetként a változáslista-bejegyzéseddel együtt.

A Launchpadra való feltöltéshez a távoli ág neveként a következő nomenklatúrát kell használnod:

```
lp:~<yourlpid>/ubuntu/<release>/<package>/<branchname>
```

Ez például ilyen tartalmú lehet

```
lp:~emmaadams/ubuntu/natty/specialpackage/fix-for-123456
```

Így, ha csak ezt futtatod

```
bzr push lp:~emmaadams/ubuntu/natty/specialpackage/fix-for-123456
```

```
bzr lp-open
```

mindent beállítasz. A push parancs feltölti ezt a Launchpadra, a második parancs pedig megnyitja a böngésződben a távoli ág Launchpad oldalát. Ott keresd meg a „(+) Beolvasztás kérése" hivatkozást és kattints rá, hogy valaki átnézze a módosítást és feltöltse az Ubuntura.

A következő hónapban: a Debian könyvtárszerkezet áttekintése.

Ubuntu fejlesztés – 4. rész: A debian/ könyvtár általános áttekintése

Írta Daniel Holbach

Ez a cikk arról szól, hogy a debian/ könyvtárban található fájloknak mi a szerepe az Ubuntu csomagok csomagolásánál. A könyvtárban találhatók között a legfontosabbak a changelog, a control, a copyright és a rules fájlok, ezekre az összes csomagnak szüksége van. A csomagok testreszabásához és beállításához a debian/ könyvtárban további fájlok is használhatók, közülük néhányat ebben a cikkben is bemutatok, a teljesség igénye nélkül.

A changelog

Ahogy azt a neve is sugallja, ez a fájl a verziók közötti változtatásokat tartalmazza. Jól meghatározott formátuma van, amelyből kiolvasható a csomag neve, verziója, a disztribúció, a változtatások és az, hogy ez utóbbit ki és mikor követte el. Ha rendelkezel GPG kulccsal (lásd Getting set up), ugyanazt a nevet és email címet használd a changelog-ban is. Itt egy changelog sablon:

```
csomag (verzió) disztribúció;
urgency=sürgősségi szint

* a változtatások részletezése
 - további részletek a
változtatásokról
* még több részlet

-- karbantartó neve <email cím>[két
szóköz]  dátum
```

A formátum, különösen a dátumé, fontos. A dátumot RFC 5322 formátumban kell megadni, ezt a date -R paranccsal kérdezhetjük le. A changelog szerkesztésére a dch parancs használható, ez a dátumot automatikusan frissíti. Az alpontokat kötőjellel „-“, a magasabb szintű vázlatpontokat csillaggal „*” jelöljük. Ha a csomagolást a semmiből kezded, a dch --create paranccsal létrehozhatsz egy standard debian/changelog fájlt (a dch a devscripts csomagban található).

Itt egy changelog minta a hello programhoz:

```
hello (2.6-0ubuntu1) natty;
urgency=low

* New upstream release with lots of
bug fixes and feature improvements.
-- Jane Doe
<packager@example.com>  Thu, 21
Apr 2011 11:12:00 -0400
```

Figyeljük meg, hogy a verziószám kapott egy -0ubuntu1 toldalékot, ez a disztró-revízió, ami azért kell, hogy a csomagolás frissíthető legyen (például bugok javítása érdekében), miközben a program forráskódjának verziója nem változik.

Az azonos forráskód verzióval rendelkező csomagok összeakadásának elkerülése érdekében az Ubuntu és a Debian különbözően számozza a csomagokat. Ha egy Debian csomag Ubuntu alatt változott, a Debian változat végére egy ubuntuX jelölés kerül, ahol X az Ubuntu revízió száma. Ha tehát a hello 2.6-1 Debian csomag módosult Ubuntu alatt, a csomag új jelölése 2.6-1ubuntu1 lesz. Ha az alkalmazás csomagja Debian alatt nem létezik, akkor a Debian revízió egy 0 lesz (azaz 2.6-0ubuntu1).

További részletek a Debian Policy Manual changelog fejezetében (4.4 fejezet).

A control fájl

A control fájl a csomagkezelőt (pl. apt-get, synaptic, adept) látja el a szükséges információkkal, tartalmazza a csomagok közötti függőségeket, karbantartói információkat és még sok minden mást.

Az Ubuntu hello csomag control fájlja valahogy így néz ki:

```
Source: hello
Section: devel
Priority: optional
Maintainer: Ubuntu Developers
<ubuntu-devel-
discuss@lists.ubuntu.com>
XSBC-Original-Maintainer: Jane Doe
<packager@example.com>
Standards-Version: 3.9.1
Build-Depends: debhelper (>= 7)
Bzr-Vcs: lp:ubuntu/hello
Homepage:
http://www.gnu.org/software/hello/

Package: hello
Architecture: any
Depends: ${shlibs:Depends}
Description: The classic greeting,
and a good example
The GNU hello program
produces a familiar, friendly
greeting. It allows non-programmers
to use a classic computer science
tool which
would otherwise be unavailable to
them. Seriously, though: this is
an example of how to do a Debian
package. It is the Debian version of
the GNU Project's `hello world'
program (which is itself an example
```

```
for the GNU Project).
```

Az első bekezdés a forrás csomagot írja le, tartalmazza például a fordításhoz szükséges további csomagok listáját, és olyan meta-információkat, mint a karbantartó neve, vagy a csomag fordításához használt Debian Policy verziószáma, a csomagot előállító verziókövető tároló helye, valamint az upstream honlapja.

Az Ubuntu csomagoknál a karbantartó mezőbe egy általános címet szokás megadni, mert a csomagokat bárki módosíthatja (ebben eltér a Debiantól, ott a csomagok módosítására általában egyetlen személy vagy egy csapat jogosult). Az Ubuntu csomagok esetén a karbantartó mezőben általában a következő szerepel: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>. Ha ez a későbbiekben módosul, a régi értéket az XSBC-Original-Maintainer mezőbe kell írni. A módosítás az update-maintainer parancsfájllal automatikusan is elvégezhető, ez a fájl az ubuntu-dev-tools csomagban található. További információk az Ubuntu wikin a Debian Maintainer Field alatt.

Minden további bekezdés egy fordítandó bináris csomagot ír le.

További részletek a Debian Policy Manual control file section fejezetében.

A copyright fájl

Ez a fájl a a szerzői joggal kapcsolatos információkat tárolja az upstream forrásról és a csomagról. Az Ubuntu és a Debian Policy (12.5 fejezet) is megköveteli, hogy minden csomag telepítse a saját copyright és licensz adatainak másolatát a /usr/share/doc/$(package_name)/copyright helyre.

A szerzői jogi információk általában megtalálhatóak a program forráskönyvtárában a COPYING fájlban. Ebben a fájlban szerepel a szerző és a csomagoló neve, a forrás URL címe, a copyright sor évszámmal és a jogok tulajdonosának feltüntetésével, és itt szerepel persze maga a copyright szövege is. Itt egy példa sablon:

```
Format:
http://svn.debian.org/wsvn/dep/web/deps/dep5.mdwn?op=file&rev=166
Upstream-Name: Hello
Source:
ftp://ftp.example.com/pub/games

Files: *
Copyright: Copyright 1998 John Doe
<jdoe@example.com>
License: GPL-2+
This program is free software; you
can redistribute it
and/or modify it under the terms of
the GNU General Public
License as published by the Free
Software Foundation; either
version 2 of the License, or (at your
option) any later
version.
.
This program is distributed in the
hope that it will be
useful, but WITHOUT ANY
WARRANTY; without even the
implied
warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR
PURPOSE.  See the GNU General
Public License for more
details.
.
You should have received a copy of
the GNU General Public
License along with this package; if
not, write to the Free
Software Foundation, Inc., 51
Franklin St, Fifth Floor,
Boston, MA  02110-1301 USA
.
On Debian systems, the full text of
the GNU General Public
License version 2 can be found in the
file
`/usr/share/common-licenses/GPL-
2'.

Files: debian/*
Copyright: Copyright 1998 Jane Doe
<packager@example.com>
License: GPL-2+
```

Ez a példa a DEP-5: Machine-parseable debian/copyright javaslatát követi. Javasoljuk, hogy te is ezt a formátumot használd.

A rules fájl

Tekintsük át a rules fájlt is. Ez egy Makefile, ami az összes, csomagolással kapcsolatos munkát végzi. Az alkalmazás lefordítására és telepítésére, majd a telepített fájlokból a .deb fájl létrehozására használt célokat tartalmazza. Az összes lefordított fájl törlésére is van cél, a segítségével vissza tudunk térni a kezdeti állapotba: csak a forrásfájlok maradnak meg.

Itt egy egyszerűsített változata a rules fájlnak, amit a dh_make készített (a dh-make csomagban található):

```
#!/usr/bin/make -f
# -*- makefile -*-

# Uncomment this to turn on
verbose mode.
#export DH_VERBOSE=1

%:
	dh $@
```

Menjünk rajta végig. A kód minden, a debian/rules által lefordított célfájlt paraméterként továbbad a /usr/bin/dh-nak, az pedig meghívja az összes szükséges dh_* parancsot.

A dh debhelper parancsok sorozatát futtatja. A támogatott

sorozatok a debian/rules fájl céljainak felelnek meg: „build”, „clean”, „install”, „binary-arch”, „binary-indep” és „binary”. Hogy lássuk, hogy az egyes céloknál melyik parancsok futnak, futtassuk le ezt:

```
dh binary-arch --no-act
```

A binary-indep sorozatban a parancsoknak a „-i” kapcsoló adódik át, hogy csak a független bináris csomagokon fussanak, a binary-arch sorozatban lévő perancsoknak pedig a „-a” kapcsoló, hogy csak az architektúra-függő csomagokon működjenek.

Minden debhelper parancs a debian/package.debhelper.log-ban rögzül, amint sikeresen lefut (törölni a dh_clean paranccsal lehet). A dh tehát meg tudja mondani, hogy melyik parancs volt már futtatva, és hogy melyik csomagban, ezeknek a parancsoknak az újrafuttatását így át tudja ugorni. A dh minden futtatás során megvizsgálja a naplót, és megtalálja az utoljára naplózott parancsot az adott sorozathoz. A folytatásban a sorozatban szereplő következő parancsot futtatja le. Ezt a viselkedést az --until, --before, --after és --remaining kapcsolók felülírhatják.

Ha a debian/rules tartalmaz egy célt override_dh_command névvel, a sorozatban ehhez a parancshoz érve a dh a rules fájlból fogja futtatni a célt az adott parancs helyett. A felülíró cél ezután több paraméterrel futtathatja a parancsot, vagy egy teljesen másikat is futtathat. (Ennek a funkciónak a használatához dehelper 7.0.50-nek, vagy ennél újabb változatnak kell a Build-Dependben szerepelnie).

További példákért nézz körül a /usr/share/doc/debhelper/examples/ könyvtárban, vagy írd be: man dh. Ajánlom még a Debian Policy Manual Section 4.9 fejezetét (rules) is.

További fájlok

Az install fájl

Az install fájlt a dh_install használja a fájlok bináris csomagba történő telepítéséhez. Két esetben szokás használni:

1. Az upstream-beli fordítórendszer által nem kezelt fájlok saját csomagba történő telepítésére
2. Egy nagy forráscsomag több bináris csomagba történő szétválasztására.

Az első esetben az install fájl telepített fájlonként egy-egy sorból áll, amely megadja magát a fájlt és a telepítési könyvtárat is. Az alábbi példában az install fájl a forrás csomag gyökérkönyvtárában található foo parancsfájlt telepíti a usr/bin könyvtárba, valamint egy a debian könyvtárban lévő desktop fájlt a usr/share/applications alá:

```
foo usr/bin
debian/bar.desktop
usr/share/applications
```

Ha a forráscsomag több bináris csomagot hoz létre, a dh a debian/<csomag> helyett a debian/tmp-be telepít. A debian/tmp-be telepített fájlok ezután különálló bináris csomagokba mozgathatók a $package_name.install fájlok segítségével. Ezt nagy mennyiségű architektúra-független adat architektúrafüggő csomagokból történő ki-, majd azok Architecture: all csomagba való bepakolására is szokás használni. Ebben az esetben csak a telepítésre szánt fájl- és könyvtárnevek szükségesek, a telepítési könyvtár nem. Például a foo.install csak az architektúra-függő fájlokat tartalmazza:

```
usr/bin/
usr/lib/foo/*.so
```

A foo-common.install pedig csak az architektúra-független fájlokat:

```
/usr/share/doc/
/usr/share/icons/
/usr/share/foo/
/usr/share/locale/
```

Ez így két bináris csomagot hoz létre: foo és foo-common. A debian/control fájlban mindkettőnek saját bekezdésre van szüksége.

További részletek: man dh_install és a Debian New Maintainers' Guide 5.11 fejezete (install file).

A watch fájl

A debian/watch fájl segítségével automatikusan ellenőrizhetjük az új upstream változatokat, a devscripts csomag uscan eszközét használva. A watch fájl első sora a formátum verziószámát tartalmazza (jelenleg 3), míg az ezt követő sorok a feldolgozandó URL-eket tartalmazzák. Például:

```
version=3
http://ftp.gnu.org/gnu/hello/hello-(.*).tar.gz
```

A uscan a forrás gyökérkönyvtárában lefuttatva összehasonlítja a debian/changelog-ban talált upstream verziószámot a legújabb elérhető upstream verzióval. Ha egy új upstream verziót talál, automatikusan letölti. Például:

```
$ uscan
hello: Newer version (2.7) available
on remote site:
http://ftp.gnu.org/gnu/hello/hello-2.7.tar.gz
(local version is 2.6)
hello: Successfully downloaded
updated package hello-2.7.tar.gz
  and symlinked hello_2.7.orig.tar.gz
to it
```

További információ: man uscan és a Debian Policy Manual watch file fejezete (4.11).

A watch fájl jelentése alapján az upstream-től eltérő csomagok listája: Ubuntu External Health Status.

A source/format fájl

Ez a fájl jelöli ki a forráscsomag formátumát. Ha ez a fájl nem létezik, a csomag forrásformátumának jelenlegi alapértelmezett értéke 1.0. Ajánlott azonban az újabb, 3.0-s forrásformátum használata. Ebben az esetben a fájl egyetlen sorban jelöli a kívánt formátumot:

- 3.0 (native): Debian native csomagok (nincs upstream verzió) vagy
- 3.0 (quilt): különálló upstream tarball-lal rendelkező csomagok.

Ha valamiért mégis a régi formátumot szeretnéd használni, kérlek akkor is hozd létre ezt a fájlt, és írd bele az 1.0 jelölést, hogy egyértelmű legyen a forráscsomag verziója. Ezáltal később könnyebb lesz megszabadulni a jelenleg alapértelmezett 1.0 formátumtól.

A 3.0 formátumra való áttérés előnyeiről bővebben a http://wiki.debian.org/Projects/DebSrc3.0 címen olvashatsz.

További részletek: man dpkg-source és a Debian New Maintainers' Guide 5.21 fejezetében (source/format).

További források

Az itt tárgyalt fájlokról a Debian Policy Manual linkeken kívül további részletes leírás található a Debian New Maintainers' Guide-ban. A 4. fejezet (Required files under the debian directory) a control, changelog, copyright és rules fájlokról szól. Az 5. fejezet (Other files under the debian directory) a további fájlok használatát tárgyalja.

Közreműködnél?

Az olvasóközönségtől folyamatosan várjuk a magazinban megjelenítendő új cikkeket!
További információkat a cikkek irányvonalairól, ötletekről és a kiadások fordításairól a http://wiki.ubuntu.com/UbuntuMagazine wiki oldalunkon olvashatsz.
Cikkeidet az alábbi címre várjuk: articles@fullcirclemagazine.org
A **magyar fordítócsapat** wiki oldalát itt találod: https://wiki.ubuntu.com/UbuntuMagazine/TranslateFullCircle/Hungarian
A magazin eddig megjelent **magyar fordításait** innen töltheted le: http://www.fullcircle.hu
Ha **email**-t akarsz írni a magyar fordítócsapatnak, akkor erre a címre küldd: fullcirclehu@gmail.com

Ha **hírt** szeretnél közölni, megteheted a következő címen: news@fullcirclemagazine.org

Véleményed és Linux-os **tapasztalataidat** ide küldd: letters@fullcirclemagazine.org

Hardver és szoftver **elemzéseket** ide küldhetsz: reviews@fullcirclemagazine.org

Kérdéseket a 'Kérdések és Válaszok' rovatba ide küldd: questions@fullcirclemagazine.org

Az én asztalom képeit ide küldd: misc@fullcirclemagazine.org

... vagy látogasd meg **fórumunkat**: www.fullcirclemagazine.org

A FULL CIRCLE-NEK SZÜKSÉGE VAN RÁD!

Egy magazin, ahogy a Full Circle is, nem magazin cikkek nélkül. Osszátok meg velünk véleményeiteket, desktopjaitok kinézetét és történeteiteket. Szükségünk van a Fókuszban rovathoz játékok, programok és hardverek áttekintő leírására, a Hogyanok rovatban szereplő cikkekre (K/X/Ubuntu témával); ezenkívül, ha bármilyen kérdés, javaslat merül fel bennetek, nyugodtan küldjétek a következő címre: articles@fullcirclemagazine.org

A Full Circle Csapata

Szerkesztő - Ronnie Tucker
ronnie@fullcirclemagazine.org
Webmester - Rob Kerfia
admin@fullcirclemagazine.org
Kommunikációs felelős - Robert Clipsham
mrmonday@fullcirclemagazine.org
Podcast - Robert Catling
podcast@fullcirclemagazine.org

Full Circle Magazin
Magyar Fordítócsapat

Koordinátor:
Pércsy Kornél
Fordító:
Palotás Anna
Kiss Gábor
Lektorok:
Balogh Péter
Pércsy Kornélia
Szerkesztő:
Kiss László
Korrektor:
Kiss László

Nagy köszönet a Canonicalnek és a fordítócsapatoknak világszerte, továbbá **Thorsten Wilms**-nek a jelenlegi Full Circle logóért.