Viola Zoltán
violazoli@gmail.com

Hogyan készíthetsz saját Linux disztribúciót?

Verzió: 7.4-2

A verziószám jelentése: A kötőjel előtti rész azt jelenti, melyik LFS Book kiadásra épül a leírás, a kötőjel utáni rész pedig ezen leírás változatszáma!

Figyelmeztetés

Mindenekelőtt egy **FONTOS FIGYELMEZTETÉS!**

E dokumentumot bár igyekeztem a legnagyobb gondossággal elkészíteni (tartalmilag... A stílus és a külalak totál hidegen hagy, mert szerintem stílusa a karatemestereknek van, a külalakkal meg foglalkozzanak a divatdiktátorok és úriszabók...), ennek ellenére jó, ha tudod kedves Olvasóm, hogy én egy *teljesen felelőtlen fickó vagyok*, azaz **egyáltalán nem vállalok felelősséget semmiért se!** Még magamért se, nemhogy annak helyességéért, amit itt összehablatyoltam!

Ezt jó lesz, ha állandóan észben tartod majd, „keep in mind” ahogy az angol mondja, mert ez NEM VICC: E leírás a disztróépítésről szól, eközben neked rendszergazdai jogosultságokkal kell tényked ned, s igen veszélyes műveleteket végrehajtanod, azaz NEM TRÉFÁLOK, amikor azt állítom, hogy elég egyetlen akárminek a kihagyása vagy elgépelése, és emiatt TELJESEN ELCSESZHETED A RENDSZEREDET, sajnos nemcsak azt, amit javában építesz, de az úgynevezett „host” rendszert is akár, azt, amit mindennap használsz mindenféle más célra is!

Az sem kizárt, hogy én írtam le e doksiban valamit rosszul. Ha így esik és kiderül, azt nagyon fogom szégyellni, DE AKKOR SE FIZETEK SEMMI KÁRTÉRÍTÉST, ezt előre bejelentem! (Nem is lenne miből, hehehe...) E doksit komoly és fáradságos munkával alkottam meg, ellenben INGYEN TESZEM ELÉRHETŐVÉ neked

is, bárkinek is, semmi hasznom az egészből, tehát OLYAN AMILYEN, a felhasználása a saját felelősséged! Amit olvasol itt, azt mind ÉSSZEL KELL HASZNÁLNOD, mielőtt kiadod a parancsot, előbb háromszor is elgondolkodnod rajta, s egyáltalán: e leírás NEM A KEZDŐ LINUXOSOKNAK VAN, akiknek még a fenekükön a virtuális tojáshéj! Igaz hogy nem is a profiknak, mert azoknak nincs szükségük az én fecsegésemre.

A fenti figyelmeztetést annál is inkább illik véresen komolyan venned, mert ezennel bejelentem és írásba adom, hogy ÉN NEM VAGYOK PROFI! Nem tanultam a Linuxot, sőt általában a számítástechnikát soha sehol semmiféle szervezett oktatás keretén belül, nincs egyetemi diplomám, sőt, jelenleg nem is a számítástechnikából élek, nem informatikai munkakörben dolgozom. Autodidakta vagyok, minden tudást, amivel rendelkezem, magamtól, önképzéssel csipegettem fel innen-onnan, ahonnan éppen tudtam, MERT ÉRDEKELT A TÉMA. Azaz: ÉN CSAK EGY LELKES AMATŐR VAGYOK. Maga a tény, hogy e módszerrel ilyen szintig eljutottam, hogy képes legyek saját disztrót építeni, ez mutatja, bizonyítja és igazolja, milyen messzire lehet jutni önerőből is, némi szorgalommal, ha az időnket nem a mulatságra, a hacacáréra pazaroljuk, holmi „társasági életre”, a haverokkal elpocsékolva azt, kocsmákban meg szajhák karjai közt, vagy épp a tévében az idióta valóságshowkat bámulva, hanem ahelyett hogy azt lesnénk, a Győzike kutyája mekkorát okádik a szőnyegre, az időnket valami sokkal hasznosabbal töltjük.

Bármilyen messzire jutottam is azonban, e módszer szükségszerűen hozza magával azt, hogy ismereteimben óriásiak lehetnek a fehér foltok. Azt a részt tanultam az informatikából, ami ÉRDEKELT és ugyanakkor amiről HOZZÁ IS JUTOTTAM információhoz könnyen. Ha e két feltétel egyike is nem volt biztosított számomra, az a terület sajnos kimaradt a tudásomból. Egyáltalán nem meglepő tehát, hogy a következő leírásban akármiféle olyasmi is szerepelhet, ami nálam semmi bajt nem okoz, de nálad esetleg okozhat valami gondot, de én ezt nem látom előre, nem figyelmeztetlek, nyíltan szólva: mert buta vagyok az adott kérdéskörhöz, azért. Tehát a fokozott óvatosság részedről nagyon is ajánlott, légy mélységesen bizalmatlan az itteni infókkal szemben, mert ezeket nem valami Nagy és Képzett Informatikai Gurutól kapod, nem a Csalatkozhatatlan Mestertől, hanem egy amatőrtől, aki biztos előrébb tart ugyan mint te, efelől semmi kétség, de az is biztos hogy nagyon messze van az igazi profizmustól, oly messze, hogy azt meg se tudja ugatni!

„Mindezek okából kifolyólagosan”, hogy „tudákosan” fejezzem ki magamat, még az se helyes, ha e művet egyáltalán „szakkönyvnek” tekinted. Ez NEM SZAKKÖNYV. Ez sokkal inkább csak afféle „komoly dologról szóló népszerűsítő iromány”. Annyi igaz, hogy az átlag tudománynépszerűsítő munkáknál azért magam is komolyabb színvonalúnak tartom amiatt, mert ez mégiscsak leír szinte lépésről-lépésre egy módszert, amivel valami hasznosat lehet alkotni, s nem csak általánosságban mutat be valamit. Ettől azonban még igenis sokkal közelebb van a népszerűsítő művekhez, mint a komoly tudományos szakkönyvekhez.

A lényeg az hogy nyugodtan olvass végig előbb itt MINDENT, s véletlenül se kezdj bele semminek a végrehajtásába amíg mindent el nem olvasol, de tényleg és igazán MINDENT, elejétől a végéig, s nem tudod, hogy mit vállalsz, minek kezdesz neki!

Sőt, miután elolvastad, előbb még aludj is rá egyet mielőtt eldöntöd, belevágsz-e. Még ha kivétel nélkül mindent jól írtam is le, sehol egy tévedés, és minden remekül megy a rendszereden, és te se fogsz sehol téveszteni soha semmit, ez akkor is HATALMAS MUNKA lesz a számodra! Gondold meg!

S amikor elolvastad és nekikezdenél, **ELŐBB CSINÁLJ EGY BACKUPOT azaz MENTÉST** a teljes eredeti rendszeredről, de mindenféleképpen legalább a te saját, értékes, pótolhatatlan adataidról, mint dokumentumok, családi fotók, zenék, filmek, akármik! Mert ha a rendszer maga megy tönkre, még hagyján, egy óra mondjuk és újratelepíted és egy újabb óra alatt beállítod, testreszabod. De ha a te adataid mennek tönkre, amikről nincs másolat, azokat soha sehonnan nem töltheted le az Internetről!

Ne feledd, hogy CSAK 2 FAJTA ADAT LÉTEZIK:

1. Amit elmentettek.
2. Ami **MÉG** nem veszett el...

A biztonsági mentés nem úgy értendő, hogy a cuccokat átmásolod egy külön könyvtárba a rendszereden belül. Nem is úgy, hogy ugyanazon merevlemez másik partíciójára másolod. Olyan helyre másold, ami fizikailag teljesen külön van szeparálva a számítógépedtől, nincs vele összeköttetésben! Mondjuk valami kellően nagy kapacitású pendrive, vagy USB merevlemez, amit aztán eltávolítasz róla!

A backup olyasmi, ami az esetek 99%-ában teljesen felesleges, értelmetlen idő- és energiapocséklás. Hanem az a maradék 1%, amikor mégis kell...! Hajaj, akkor de jól jön! Olyan jól, hogy bőven kárpótol a többi 99% felesleges voltáért!

Ember, ne mondd, hogy nem figyelmeztettelek!

Előszó

Aki kezdő linuxos, gyakran gondol vélhetőleg arra, milyen remek is volna, ha lenne egy SAJÁT disztribúciója! Olyan, amiben benne van minden program, amit ő szeret, és nincs benne semmi, amit feleslegesnek gondol. Mert ugye, mindenekelőtt az ilyesmi roppant kényelmes, másrészt meg mekkora „cool” dolog ám, ha elmondhatja magáról, hogy neki van egy saját disztrója, hogy már ő is „disztribútor”!

Igaz, általában véve sincsenek sokan azon emberkék a világban, akik Linuxot használnak. E sorok írásakor, 2014.01.04-én jött a friss hír, hogy bizonyos statisztikák szerint a desktop célokra Linuxot használók részaránya az összes desktopnak használt oprendszerek közt elérte az 1.75%-ot, s ezt mint hatalmas előrelépést ünnepelték a linuxosok. E hír forrása:
http://hup.hu/cikkek/20140103/netapplications_minden_idok_legnagyobb_desktop_linux_piaci_reszesedese_3?comments_per_page=9999

Azaz már eleve az a tény is, hogy te éppen Linuxot használsz, kiemel téged a szürke tömeg posványából, ezáltal különleges valaki vagy. Mégis, ez az 1.75% azt jelenti, hogy minden száz valakiből, aki számítógépet használ, majdnem 2 fő szintén linuxozik! Ez azért elég ciki. Kis túlzással mondható, hogy ma már majdnem minden IQ-negatív idióta a Linuxszal virít. Mert megteheti, mert vannak már olyan túlságosan is kezdőbarát rendszerek, mint az Ubuntu, Lubuntu, Kubuntu, Xubuntu, Manjaro, Linux Mint, meg a fene se tudja mi minden még. Ezekhez kb. annyi tudás kell csak, mint a Windowshoz. Ciki, na. Ezekből jószerivel hiányzik már a „geek-faktor”. Az igazi az volt, még régen, amikor a linuxozás kis túlzással azt jelentette, hogy ott ülsz a terminál előtt, titokzatos „varázsszavakat” pötyögsz be, erre elkezdenek rohangálni a képernyőn a kódsorok, természetesen fekete képernyőn zöld színnel, mert az olyan „Mátrixos feeling”, és senki nem érti, aki néz téged, hogy mit csinálsz és miért, csak azt tudják hogy megcsináltad, amit akartál, vagy amit ők akartak tőled, de ŐK bezzeg nem tudnák utánad csinálni! Nyilvánvaló persze, hogy miért nem: Ők ugye nem olyan okosak, mint TE! A Hacker!

Manapság már ez nem így van. Minden grafikus képernyőn megy, ahol ikonokra kell kattintgatni. Ezt tényleg tudja a hülye is. Ez nem igényel nagyobb intelligenciát, mint ami egy reményteljesebb makákótól is alapból elvárható. Kattintgatni tényleg tud egy majom is.

Mennyivel másabb lenne azonban, ha te nemcsak egy egyszerű, mezei, hétköznapi linuxos lennél, hanem olyasvalaki, aki annyira különleges disztrót használ, amit senki más se rajta kívül, vagy ha használják is azt rajta kívül akárhányan még, de azt a disztrót Ő MAGA KÉSZÍTETTE! Azaz te magad!

Képzeld csak el: annak a disztrónak te vagy a készítője! Garantáltan minden benne lesz amit bele akarsz rakni, és csak és kizárólag az lesz benne, pontosan olyan lesz, amilyennek szeretnéd, a legelső bájttól a legutolsóig, ismered kívülbelül, és még ha senki nem is használja majd rajtad kívül, akkor is tagadhatatlan tény lesz, hogy ezzel ország-világnak bebizonyítod egyszer s minden-

korra a linuxos hozzáértésedet, hogy te MÁS vagy, mint a többiek, természetesen jó értelemben, különb vagy, különböző, de jó értelemben különb, épp ezért e fogalomra illik is más szót használni, azt, hogy „külömb”. Ha ebbe belekötne valamely nyelvtannáci, nyugtassátok meg, hogy e szót már Jókai Mór is használta, amit meg egy ekkora irodalmi klasszikus használ, nem lehet helytelen. Sőt, inkább előkelő és választékos. Ha ezt nem tudja a nyelvtannáci, az rá nézve szégyen, ezzel ugyanis azt igazolja, hogy ő maga az irodalmilag műveletlen tahó, a kultúr-barbár, akit most szalasztottak az őserdőből, ahol a majmok és krokodilok közt szocializálódott.

Tehát:

`#define külömb jó értelemben különböző`

A http://distrowatch.com oldalon össze van gyűjtve szinte minden disztró, ezekből lehet úgy talán 300. Azért az nem lenne ám semmi, ha a világ 300 legelső embere közé számítódnál, akik tudnak saját disztrót gyártani...

Vannak is e cél, a disztrógyártás elérésére mindenféle könnyű módszerek. A dolog alapvetően nem is nehéz tényleg, úgy tűnik. Az Interneten fellelhető efféle leírások lényegében abból állnak, hogy telepítsünk fel egy már létező disztrót, ebben cseréljük ki a háttérképet, sőt akár magát az egész ablakkezelő rendszert (ehhez is csak néhány kattintás kell manapság), cseréljük ki a bootoláskor mutatott képet, találjunk ki neki valami jópofa nevet, no és természetesen a felhasználói programokból is töröljük le, ami nekünk nem kell, s telepítsünk helyettük pár másikat. Aztán indíts el egy erre a célra készült programot, ami neked ezt az egész egyéni kutyulmányt elmenti egy iso fájlba. E célra többnyire a „remastersys” programot ajánlják. Itt is van erről egy efféle leírás, íme:

https://uhubian.wordpress.com/2013/11/08/hogyan-keszitsunk-linux-disztribuciot-konnyeden/

Nos igen, így is lehet, de ez a módszer hogy nyersen fogalmazzak, olyasmi, amit egy gyermekkoromban elterjedt kifejezéssel jellemezhetek: „olcsó játék hülyegyerekeknek”.

E degradáló kifejezéssel nem azt akarom mondani, hogy egy efféle játszadozás teljesen haszontalan volna. Tényleg hasznos lehet, ha tudsz készíteni olyan telepítőt, amin rajta vannak a számodra fontos programok. Amikor kezdtem a linuxozást, ezelőtt bő 8 vagy tíz évvel (sőt talán régebben, nem emlékszem már rá pontosan) a fél életemet odaadtam volna érte, ha képes vagyok egy effélének a megalkotására! De akkoriban még nem voltak e célra könnyű módszerek. (Ennek ellenére, azért már akkor is megoldottam a dolgot, ha nem is linuxos korszakom legelején, de úgy az első év után... Nem volt könnyű, de megcsináltam.)

Manapság már vannak könnyű módszerek, és JÓ HOGY VANNAK, eszem ágában sincs ezt vitatni. Kifejezetten örülök e módszerek létezésének.

A gond ott van, hogy egy efféle játszadozás tényleg játék, akkor is, ha van valami haszna. Azért játék, mert nem kell hozzá semmi TUDÁS. Továbbá, ezáltal

az is nagyon kevés benne, amit „újat alkotásnak” nevezhetünk. Nincs benne számottevő „hozzáadott érték”. Ez ugyanis, amit létrehoztál, igazából NEM ÚJ DISZTRÓ. Kiindulsz mondjuk az Ubuntuból, letörölsz 10 programot, felteszel 15 újat, megváltoztatod a háttérképet, becsomagolod... Mondhatod, hogy ez egy új disztró, mondjuk „Freya” néven, de igazából akkor se az. Ez bizony akkor is csak egy Ubuntu. Hiszen az Ubuntu programjai vannak benne, annak a repóit használja, teljesen kompatibilis vele. Még attól se válik igazi új és más disztróvá, ha kicseréled benne a sokak által nem kedvelt Unity nevű ablakkezelőt mondjuk XFCE-re. Bárki megteheti a te „disztród” nélkül is könnyedén, ha jól emlékszem ehhez csak a következő parancs kell terminálban:

```
sudo apt-get install xfce-desktop
```

és kész. Marad minden Ubuntu. Az ablakkezelőd XFCE lesz, de attól ez még „lényegileg”, „ott a mélyben” egy Ubuntu. Olyan, mintha a feleségedet más ruhába bújtatnád. Megváltozik a kinézete? Igen. Új nő lesz-e ettől, mondhatod-e, hogy emiatt neked már más feleséged van? Persze hogy nem!

A distrowatch.com oldalon levő disztróknak legalább a fele efféle.

Persze ismétlem, ez mind nem baj, ezek is hasznosak lehetnek egyeseknek. Csak szeretném ha világosan látnád, hogy ha ilyesmit csinálsz, akkor se leszel disztribútor IGAZÁBÓL. Mondhatod, hogy az vagy, de magad is tudni fogod, hogy nem vagy az. Csak a dolgok felszínét kapirgálod, arról fogalmad sincs, mi van „ott a mélyben”. És ha váratlanul megszűnik a disztró, amit alapnak használtál, ott állsz megfürödve, nem tudod tovább folytatni. Ez olyan, hogy akármelyik „SzomszédPistike” letölthet valami torrentoldalról holmi kiberbűnözésre alkalmas programot, elindítja, betáplálja valamelyik weboldal linkjét, aztán vár, hátha sikerül a gépnek pár millió próbálkozás után kiderítenie a jelszót. És esetleg mázlija is lesz mondjuk minden huszadik esetben. Akkor ő oda belép, elhelyez ott valami öntömjénező üzenetet, hogy ő micsoda nagy hacker, mert feltörte a weboldalt. Pedig francokat törte ő azt fel! A weboldal fel lett törve az igaz, de ezt nem ő művelte, hanem az, aki azt a programot megírta, amit ő használt! Ő attól még marhára nem lett se hacker, se cracker, csak maradt egy hülye kiskölyök, aki egy kicsit eljátszadozott egy amúgy törvényellenes dologgal.

Ő majd akkor lesz hacker, ha maga is képes lenne megírni azt a programot...

Azaz: szerintem az nevezhető igazi disztrónak, ami valami **alapvető** dologban különbözik az összes többitől, vagy a többinek legalábbis legnagyobb részétől, és akkor vagy te magad igazi disztrókészítő szakember, ha képes vagy te magad létrehozni azt a legalapabb alapjaiból, mintegy összelegózni a megfelelő szoftveralkatrészekből!

Ennek módját szeretném most bemutatni neked, egy konkrét példán. Amit írok, azt nem kell szentírásnak tekinteni, amennyiben eltérhetsz az adott utasításoktól, de az a saját felelősséged, azaz ne végy mérget rá, hogy ha emiatt nem működik valami, akkor tudok is segíteni neked, ha emiatt hozzám fordulsz. Kezdjük azzal, van millió más és jobb dolgom is. (Tényleg van: saját programnyelvet írtam az utóbbi hónapokban, az a neve hogy „mau”, elérhető a web-

oldalamon: http://parancssor.info, bár már remekül működik de még mindig bővítem, emellett saját regexp kiértékelő függvényt akarok írni, s előbb említett mau nyelvemben egy text editort, ami van olyan jó mint a VIM... azaz van munkám bőven...) De ha van bőven időm, akkor se biztos, hogy válaszolok, mert a kezdőknek gyakran totál idióta kérdéseik vannak, de ha nem, akkor is könnyen meglelhetnék a választ rá egy kis dokumentáció-olvasással, amihez már lusták. Továbbá, a saját disztró készítésének nem az a legnagyobb haszna, hogy villoghatsz vele mindenki előtt, mert az nem is haszon, csak kellemetesség. A haszon az, ami tudásra közben szert teszel. Na most, ha csak eléd lököm a készet, abból marhára semmit se tanulsz.

Arról nem is beszélve, hogy még ott van egy minimális kis töredékszázalékkal az az undorító lehetőség is, hogy hátha nem is tudom a választ a problémádra. Mondjuk ennek esélye tényleg nem nagy, nem több mint legfeljebb talán 99.9%, de azért akkor is előfordulhat még ez is...

Mindemellett a dolgod ezerszer könnyebb, mint nekem volt. Amikor én kezdtem ezt, semmi magyar nyelvű leírás nem volt hozzá. Most van, ez. Lehet, hogy egyeseknek nem tetszik majd a stílusa, de akkor is magyar, s emiatt a legtöbbeknek könnyebben érthető, mintha angol lenne.

Vágjunk is tehát bele a dologba!

1. fejezet: Személyes és szociális feltételek

Ahhoz hogy disztrót csinálj az alapoktól kezdve, természetesen forrásból kell dolgokat fordítanod. Emiatt elengedhetetlenül szükséges az úgynevezett „terminál”, más néven „konzol”, azaz a „parancssor” használata. Ha neked még az is gondot okoz, hogy „abban a randa fekete ablakban” elnavigálj egy akármelyik másik könyvtárba, ha úgy érzed, létezni se tudsz grafikus fájlkezelő program (Nautilus, Krusader, stb.) nélkül, akkor teljesen reménytelen az ügy. Akkor ne is olvasd e doksit tovább, mert kárbaveszett idő a számodra. Ez olyan, hogy aki most ismerkedik az első osztály anyagával az általános iskolában, az nem biztos, hogy jól teszi, ha parciális differenciálegyenletek megoldásába kezd bele.

Tovább megyek: illik hogy tisztában légy az olyan alapvető linuxos fogalmakkal is, mint a shell, szkript, szimbolikus link, más néven symlink vagy magyarosabban „szimlink”, tudnod kell mi a különbség egy program forráskódja és a futtatható binárisa között, mik azok, hogy „jogosultságok”, s ezek csak a legfontosabb dolgok, mert még egy rakás olyasmi merülhet fel e doksi írása közben, amit most az elején, amikor e sorokat írom, én se látok előre, de ÉN alapdolognak tartom, s ezért eszem ágában se lesz őket elmagyaráznom. Teljesen nyíltan bevallom, hogy e leírás nem a Windowsról most Linuxra áttérő teljesen kezdőknek szól. Az olyan még ne akarjon saját disztrót készíteni, előbb tanulja meg az alapokat.

Teljesen nyíltan bevallom azt is, ha e doksi olvasója olyan kérdéssel fordul majd hozzám, amiről az a véleményem hogy alapszintű, akkor egyszerűen nem is válaszolok majd neki. Drága az időm. Nem izgat, ha emiatt megsértődik,

engem is sértettek meg már sokszor. És ez a részemről még mindig jelentős UDVARIASSÁG, mert hasonló esetben a legtöbb általam ismert linuxos fórumon nem az van, hogy nem válaszolnak neki, hanem nagyon is válaszolnak, de úgy, hogy elküldik a legdurvább arroganciával a búbánatos francba (sőt ennél még ocsmányabb válaszokat is kaphat).

Ennek ellenére ÉRTELMES kérdéseket lehet feltenni nekem is, amikre TALÁN válaszolok, de az Internet virtuális sikátorainak bármely bugyrában találjanak is meg engem, ezekre se válaszolok ha azokat nem a SAJÁT fórumomon szegezik nekem, mely az én doménemen van, aminek címe:

http://parancssor.info

A regisztráció ott adminisztrátori jóváhagyáshoz van kötve a sok troll miatt (az admin természetesen én vagyok), ezért a regisztráció után várni kell egy picit, míg észlelem hogy regelés történt. Előre szólok, hogy ha a regiző user nickneve vagy a mailcíme nevetséges, vagy nyilvánvalóan troll szándékú, akkor nem hagyom jóvá. Akkor se ha magyar torokkal kiejthetetlen betűhalmaz, például tksomahpr, vagy qwxxyjckrg. Az ilyenekre tojok, mert simán robotnak nézem, de ha nem robot, az se érdekel, mert azt akarom, ha kommunikálok valakivel, meg tudjam jegyezni magamban az illető nickjét, de efféléket megjegyezni képtelen vagyok. Ez okból azon nicknevek is „invalidok” a számomra, amik sok számjegyet tartalmaznak, sőt azok is, melyekben valamely betű valami hozzá hasonló számjeggyel van behelyettesítve, pld h31y35 a „helyes” helyett. Efféle trükköket alkalmazzanak a jelszavaik megjegyzésére, de ne nicknévként. Mindez érvényes a regiző emailcímére is, akinek ugyanis nem normális a mailcíme, arról feltételezem, hogy épp akkor kreálta, s csak trollkodási szándékból.

Továbbá itt és most bejelentem, hogy ragyogóan megvagyok akkor is, ha marhára nem lesz senki látogatója a fórumomnak. Azaz, ha valaki bejut oda és elkezd trollkodni, igen hamar kirúgom onnan. Agresszív dúvad vagyok, morózus vén fa$kalap, akinek rém alacsony az ingerküszöbe. Ott nálam igenis én vagyok az ISTEN, csupa nagybetűvel, nekem mindig igazam van, akkor is, ha mindenki számára szemmel láthatóan abszolút nincs igazam. Ott nálam önkényuralom van, diktatúra és zsarnokság, teljesen nyíltan bevallom tehát, hogy ott nemcsak demokrácia nincs, de még annak a LÁTSZATA SEM, sőt még a demokrácia látszatának a látszatát se igyekszem megteremteni! És erről semmiféle vitát nem vagyok hajlandó nyitni soha, sehol, senkivel sem.

Valójában már kezdem bánni, hogy miután megcsináltam a magam disztróját, ezzel eldicsekedtem egy fórumon. Mert erre jött egy lelkes érdeklődő, s most illik megcsinálnom neki e rém hosszú leírást. Baromi nagy meló. Eszem ágában sincs ezen felül más gondokat is felvállalni, meg holmi trollokkal kínlódni. Akinek nem tetszik ott a pofája, kirúgom, és ha nem marad senki, nekem az is jó, mert akkor is marad valami: a nyugalmam!

Ennyit tehát a figyelmeztetésekről.

Ha valaki idáig eljutott az olvasásban, esetleg úgy gondolhatja, hogy őt érdekelné a saját disztró készítése, de kevés még ehhez a tudása, ezért úgy véli, igényelne néhány linket, ahol e tudást megszerezheti, s azon helyeken kimű-velődve visszatérne ide, a „Nagy Kalandot” folytatandó!

Nos ennek érdekében ajánlom először is e linket:

http://parancssor.info/dokuwiki/doku.php?id=fossilcodger_a_linux_bemutatasa

A fenti link meglátogatása mindenkinek hasznos lehet, mert ott megismerkedhet honlapom filozófiájával, s azzal az elvvel is, amit e mostani leírásban követek majd a disztró megalkotásakor. Valamint ott élvezetes stílusban megtanulhatja, mi is az a fogalom, hogy egy program „**függősége**”, márpedig ez **alapvető fontosságú** fogalom egy disztró megalkotásakor, de EZT SE MAGYARÁZOM EL a továbbiakban! Aki nem tudja, olvassa el a fenti link alatt.

Egyéb infók érdekében tájékozódjon különböző más linuxos fórumokon a kedves érdeklődő.

Egy konkrét irodalmi ajánlat még a részemről:

Büki András: Unix/Linux héjprogramozás

című könyve. Nagyon melegen ajánlom, rendkívül hasznos, én is nagyon sokat tanultam belőle, magasan megéri az árát! Megrendelhető innen:

http://www.kiskapu.hu/index.php?BODY=BookInfo&OP=details&ID=33247

2. fejezet: Szoftverkörnyezet

Ha disztrót akarunk készíteni, mindenekelőtt kell egy már létező disztró hozzá — természetesen nem azért, hogy ezt alakítgassuk és farigcsáljuk nekünk tetszőre, hanem azért, hogy legyen egy olyan szoftverkörnyezet, amiben működnek az alapvető fejlesztőeszközök, mint például a fordítóprogramok. Ennek érdekében tehát tegyünk fel valami friss Linux kiadást, vagy ha van már nekünk telepítve a gépünkre valamilyen disztró (ami ugye erősen valószínű) akkor frissítsük azt a legutóbbi változatra! Az Ubuntu vagy valamelyik más olyan, amelynek a neve úgy végződik hogy „buntu”, feltehetőleg jó lesz, azok többnyire elég frissek.

Ezután telepítsük fel rá a legfontosabb fejlesztőeszközöket:

```
sudo apt-get update

sudo apt-get install build-essential cmake mc bash binutils bison bzip2
coreutils diffutils findutils gawk gcc glibc grep gzip m4 make patch perl sed
tar texinfo wget xz
```

A fenti lista java része csak biztonsági okokból van beleírva, egy ubuntu alapú rendszeren többnyire ez mind fenn van, s ki is írja majd neked a rendszer, hogy ez meg az a program már a legfrissebb. Többnyire elég volna telepíteni azt, hogy

build-essential cmake texinfo mc

De ugye biztos, ami biztos...

Az „mc”, azaz a Midnight Commander, tulajdonképpen nem létfontosságú, de melegen ajánlom, hogy telepítsük, és aki nem ismerné, tanulja meg a kezelését, mert magasan megéri, nagyon megkönnyíti a munkánkat!

Miután ezek fenn vannak, térjünk át egy olyan ablakkezelőre, amin van legalább 4 különböző virtuális munkaasztal. Ennyit lényegében mindegyik tud, bár van, ahol külön be kell ezek számát állítani. A Lubuntu esetén pld alapból csak kettő van, de beállítható több is. Én 9-re állítottam be, jól is jött...

Az 1. munkaasztalon lesz nyitva a Firefox, amiben ezt a doksit olvashatod. Vagy ha letöltötted a doksit, akkor valami más kütyüvel olvasod ezen a munkaasztalon.

A 2. munkaasztalon nyitva lesz neked egy terminál, amiben fut az MC, ennek segítségével csomagolod majd ki a forrásokat arra a partícióra, ahol a rendszeredet építed, s törlöd vele a neked már nem szükséges könyvtárakat (ugye, miután lefordítottad és telepítetted a szoftvert, azután nem kell már neked a kicsomagolt forráskönyvtár).

A 3. munkaasztalon is egy terminál lesz neked nyitva, ebben történik a tulajdonképpeni fordítási folyamat, az „igazi” rendszerépítés, itt építed fel előbb az úgynevezett „ideiglenes rendszert”, majd később ebbe chroottal belépve, itt kezded el építeni a „végleges”, az „igazi” rendszert.

A 4. munkaasztal meg jó lesz neked mindenféle másra, mondjuk, hogy munka közben zenét hallgass itt, vagy akármi váratlan esetre.

A munka kezdetén be kell szereznünk az összes szükséges program forrását. Ehhez mi az úgynevezett „LFS Book”-ot vesszük alapul. Ez kifejezetten épp azt írja le – sajnos angolul – hogyan lehet a „semmiből”, pontosabban „darabokból” felépíteni egy alap Linux rendszert. Mi ennek leírását követjük majd, illetve lesz ahol nem követjük, mert „egyénieskedünk” majd. Pontosan megmondom, miben térünk majd el ettől az alapműtől, „kvázi-szabványtól”. Mindenesetre, ha nem akarunk is szabványosak lenni, akkor is kell szabvány, jó, ha az van, mert pontosan kell tudnunk, mennyire vagyunk NEM szabványosak.

A megfelelő LFS Book erről az oldalról tölthető le:

http://www.linuxfromscratch.org/

E sorok írásakor a legfrissebb a 7.4 verziójú, én abból indultam ki.

E könyvnek az egyik fejezete közli a forrásokra vonatkozó linkeket, a direktlink ezen oldalra:

http://www.linuxfromscratch.org/lfs/view/stable/chapter03/packages.html

Töltsük le a szükséges patcheket is innen:
http://www.linuxfromscratch.org/lfs/view/stable/chapter03/patches.html

Mindezeket szerencsére nem kell egyesével letöltenünk. Elég, ha letöltjük e listát:

http://www.linuxfromscratch.org/lfs/view/stable/wget-list
majd csinálunk egy könyvtárat (rootként a gyökérkönyvtárba):

```
mkdir /Archives
```

s ezután kiadjuk e parancsot:

```
wget -i wget-list -P /Archives
```

Miután letöltöttünk mindent, csináljunk egy partíciót, ahol majd felépítjük az új rendszerünket! Ennek menetét nem írom le, ezt illik tudnia annak, aki ilyesmire adja a fejét. A partíció legyen ext3-ra formázva, esetleg ext4-re, és melegen ajánlott rá legalább 10 giga méret.

Ezután csináljunk a gyökérkönyvtárba egy könyvtárat ekképp:

```
mkdir -pv /Mount/RAM
```

Ehhez természetesen root-nak kell lenned. Minthogy Ubuntuban alapból nem-igen van olyan, hogy root, illetve van, de még sincs { :) } emiatt tégy róla, hogy permanensen is root maradhass, mert rém nehézkes lesz egymás után milliószor sudózgatni és a jelszavadat pötyörészni:

```
sudo passwd root
```

itt add meg a magad jelszavát, aztán a root user jelszavát. (Amit aztán nehogy elfelejts...!)

Ezután lépj be root-ként így:

```
su – root
```

Nagyon fontos az a mínuszjel!

A „RAM” azért ram, mert én ramdiszken hajtottam végre az alaprendszer létre-hozását, de neked, mint kezdőnek ezt nem ajánlom. Ez azonban ne aggasszon téged, attól, hogy teneked nem lesz ramdiszked, még nyugodtan meghagy-hatod ezt a könyvtárnevet, nem lesz belőle semmi bajod.

Ezután a gyökérkönyvtárban állva add ki e parancsot:

```
mount /dev/sdX /Mount/RAM
```

Ezzel felcsatoltad a megfelelő könyvtárba a frissen csinált partíciódat. Termé-szetesen az sdX helyére a partíció megfelelő számát kell írnod.

Ezután:

```
mkdir /Mount/RAM/sources
mkdir /Mount/RAM/tools
```

Ezután még mindig a gyökérkönyvtárban:

```
ln -s /Mount/RAM/tools tools
ln -s /Mount/RAM/sources sources
```

Ezzel két symlinket hoztál létre.

Ezután:

```
export LFS=/Mount/RAM
```

A fenti sort jó, ha belevesszük a root felhasználó **$HOME/.bashrc** fájljába is, hogy legközelebb is „kéznél legyen”... Ha ezt nem tesszük s mégis kiadunk egy parancsot, ami a $LFS változóra hivatkozik, miközben annak a tartalma nincs beállítva, nem kizárt, hogy tönkreteheti a rendszerünket. Nem azt, amelyiket készítjük, hanem sajnos azt, amit épp használunk hostként, azaz gazdarendszerként...

Ezután biztonsági okokból létre kell hoznunk egy **lfs** nevű felhasználót a rendszerünkben. Ennek menete részletesen itt van leírva az LFS Bookban:

http://www.linuxfromscratch.org/lfs/view/stable/chapter04/addinguser.html

Emiatt itt nem is értekezek róla bővebben, csak a szükséges parancsokat másolom be gyorsan:

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
passwd lfs
chown -v lfs $LFS/tools
chown -v lfs $LFS/sources
```

Ezután az lfs felhasználó tulajdonába adjuk a forráskönyvtárat:

```
chown -R lfs /Archives
```

Készítsük el az lfs nevű felhasználónk HOME könyvtárába e fontos fájlokat valami nekünk tetsző módon e tartalommal, ahogy az én példámon látszik:

```
root@Csiszilla:/home/lfs# cat .bash_profile
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash

root@Csiszilla:/home/lfs# cat .bashrc
set +h
umask 022
LFS=/Mount/RAM
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
```

Adjuk e fájlokat is az lfs tulajdonába:

chown -R lfs /home/lfs

és belépünk, mint az lfs felhasználó:

su - lfs

Ismétlem az a mínusz jel nagyon fontos a fenti utolsó sorban!

Be kell állítsunk egy fontos paramétert, ami meghatározza a fordításaink sebességét. Ez a „MAKEFLAGS” értéke. Ez attól függ, hány magos géppel rendelkezel. Egymagos gépen ennek értéke általában akkor jó, ha 2, s ezt ezzel az utasítással állíthatod be:

export MAKEFLAGS='-j 2'

Ha kétmagos a géped, akkor a 2 helyett azt írd, hogy 3. Nekem a gépem egy Lenovo ThinkPad T530, CORE i7 procival, ami ugyan elméletileg csak 2 magos, de ez a „HyperThreading”-nak nevezett trükk miatt 4-nek látszik, ezért nálam ezt így a jó beállítani:

export MAKEFLAGS='-j 5'

A lényeg az, hogy a szám a „j” után eggyel nagyobb legyen, mint a gépedben levő magok száma, akkor van a legoptimálisabban kihasználva a géped sebessége a fordításnál. Nem történik baj, ha kisebbre állítod, csak akkor lassúbb lesz a fordítás.

Ha nem tudod, a géped hány magos, általában úgy tekintheted, hogy annyi mag van benne, ahány kis grafikus linuxpingvin („Tux”) jelenik meg bootoláskor a képernyőd tetején. Egymagos gépnél egy, kétmagosnál kettő, és így tovább.

Na és akkor most kezdődnek az úgynevezett „ideiglenes rendszer” programjainak lefordításai! Ezt nevezi úgy az LFS Book angol nyelven, hogy „Temporary system”. Ez kerül majd az új partíciód /tools könyvtárába. Később, ha már kész lesz a végleges rendszered, e könyvtár letörölhető.

A fordítás lépéseit itt most mind leírom. Illetve mégsem mind, csak azt, ami mindegyiknél különböző, vagy elméletileg az lehet. Ez azt jelenti, hogy minden program esetén ez a következőképp van jelezve, mint az itt alább a BinUtilsnál van elmagyarázva:

```
================  5.4. Binutils-2.23.2 - Pass 1
```

A fenti sok egyenlőségjelből álló sor nem parancs, amit ki kell adnod, hanem azt jelzi, hogy itt kezdődik egy új program fordításának leírása. A jobb oldalon látod a program nevét, e név bal oldalán meg van adva még az is, az LFS Book melyik fejezetében találhatóak róla részletesebb információk. Ha baj van a fordításával, akkor OTT KELL UTÁNANÉZNI, nem engem nyaggatni. Már amiatt is, mert bár később jelentősen eltérünk majd az LFS előírásaitól, keresztülgázo-

lunk majd mint a megvadult bivalycsorda minden szentnek tartott linuxos szabványon és ajánláson, DE AZ AZ IDEIGLENES RENDSZER LÉTREHOZÁSA **UTÁN** LESZ CSAK, most azonban még az ideiglenes rendszert készítjük, és itt rém szabványosak vagyunk. Annyira azok vagyunk, hogy komolyan gondolkoztam rajta, egyáltalán belevegyem-e az ideiglenes rendszer elkészítésének leírását a könyvembe, hiszen ez alig több, mint az LFS Book valamiféle rövidített, kivonatolt magyarra fordítása. Írhattam volna ehelyett egyszerűen azt is, hogy „Csináld meg az ideiglenes rendszert úgy, ahogy az LFS írja, s ha ennyit se értesz angolul, hogy felfogd az ott közölt infókat, akkor hagyd a csudába!”

Aztán mégis leírtam itt a dolgokat nagyjából, mert én olyan rendes és jó gyerek vagyok. De a parancsokat nem fogom egyesével magyarázgatni, akit ez mélyebb részleteiben érdekel, forduljon tényleg az LFS Bookhoz.

Ha tehát egy efféle „sokegyenlőségjeles” sorhoz érsz a leírásban, akkor a dolgod a következő:

Egy másik terminálban be kell legyél jelentkezve, mint az lfs felhasználó. Itt ahogy neked tetszik (de az MC-vel a legkényelmesebb) ki kell csomagold az /Archives könyvtárban levő fájlokból azt a tömörített állományt, mely az épp fordítandó programé. Jelen esetben ugyebár a binutils-valahányas_verziójáé. A kicsomagolás azt jelenti, hogy a $LFS/sources könyvtárba, azaz az új partíciód sources könyvtárába kell kerüljön a fordítandó program könyvtára, mely a forrásfájlokat tartalmazza. Ezt nem fogom leírni minden lépésnél külön.

Ezután végig kell nézd a leírás további sorait, s ha látsz benne valami olyan sort, amiben szerepel a „patch” szó, akkor meg kell keresd az /Archives könyvtárban a megfelelő nevű patchet is. De azt nem kell kicsomagolnod, hanem úgy, ahogy van, bemásolni a $LFS/sources könyvtárba!

FIGYELEM! A sources könyvtárba kell másolnod, nem abba, ami a sourcesben van és az épp fordítandó forrásfájlokat tartalmazza!

Azaz, ha van neked ott a kicsomagolás után egy mondjuk superprogi-0.36.2 könyvtárad, tehát ennek helye a $LFS/sources/superprogi-0.36.2 és ehhez kell egy superprogi-patch-0.36.2-v1.0.patch fájl, akkor e fájlt úgy ahogy van kicsomagolás nélkül a $LFS/sources könyvtárba kell másolnod, és **nem** a $LFS/sources/superprogi-0.36.2 könyvtárba!

Ezután válts arra a terminálra (munkaasztalra), ahol a fordítást végzed. Itt is mint az „lfs” nevű felhasználó kell bejelentkezve legyél, és itt be kell lépned abba a könyvtárba, amit az imént hoztál létre a sources könyvtár alatt, ami tehát a fordítandó program forrásfájljait tartalmazza.

```
sed -i -e 's/@colophon/@@colophon/' \
       -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo
```

Ha ilyen sorokat látsz, amikből egy vagy több a visszaperjellel (\) van lezárva, azokat mind egy egységként kell a parancssorba másolnod, tehát az összes sort ami \ jellel végződik, ÉS AZT A SORT IS, AMI A LEGUTOLSÓ \ JELLEL VÉGZŐDŐT KÖVETI!

```
mkdir -v ../binutils-build
cd ../binutils-build

../binutils-2.23.2/configure   \
    --prefix=/tools            \
    --with-sysroot=$LFS        \
    --with-lib-path=/tools/lib \
    --target=$LFS_TGT          \
    --disable-nls              \
    --disable-werror

make

case $(uname -m) in
  x86_64) mkdir -v /tools/lib && ln -sv lib /tools/lib64 ;;
esac

make install
```

A fenti sorokat úgy, ahogy van, mind be kell pötyögnöd, illetve copy-pastevel beillesztened a terminálba.

Miután elvégezted az összeset s eljutsz egy újabb programot jelző sokegyenlőségjeles címsorig, akkor a teendőd – bár külön ezt se írom majd soha – mindig az, hogy kilépsz az adott munkakönyvtárból, erre a legbiztosabb módszer az, ha kiadod a

```
cd $LFS/sources
```

parancsot. Ezután egy másik terminálban, célszerűen abban, ahol a forrás kicsomagolását is végezted, TÖRÖLD LE a

$LFS/sources

könyvtárban azt a könyvtárat, ami az épp telepített program forrásfájljait tartalmazza! TÖRÖLD LE TELJESEN! (Ne a sourcest, hanem ami abban van, azt...!) Ha vannak a sources könyvtárban patch fájlok, AZOKAT IS!

Tudom, hogy szemfülesek észreveszik, hogy van az úgy, hogy egynél többször kell lefordítani majd egyes programokat, például a GCC vagy Glibc nevűeket, s arra gondolnak, mi az ürdüngnek töröljék le a forráskönyvtárt, nem ártogat az ott a sourcesben, s így megspórolják az újabb kicsomagolást!

Az igaz, hogy nem árt semmit, ha ott van. Akkor árt, ha amikor újra kell fordítani azt a programot az LFS szerint, akkor te nem egy frissen kicsomagolt példányon hajtod végre azt a fordítást, hanem ezen, amit egyszer már használtál. Még örülhetsz, ha a fordítás csak simán nem sikerül, mert rosszabb, ha sikerül, telepíted, de aztán nem fog megfelelően működni. Szóval ez a spórolás RÉM ROSSZ ÖTLET, igenis töröld le mindig a telepítés után a teljes forráskönyvtárat, és ha a fordításhoz kellett csinálni valami ideiglenes könyvtárat, aminek a neve úgy végződik hogy „build”, akkor AZT IS!

Ezután kezdhetsz neki a következő program kicsomagolásának majd fordításának.

Eddig tartott tehát a BinUtils program lefordítása. Ezt nem írom le újra, mert fent már megvan részletesen. Jön sorra a többi a következő fejezetben.

3. fejezet: Az ideiglenes rendszer létrehozása

A legelső program, amit le kell fordítanunk az ideiglenes rendszerhez, az a BinUtils. Ezt az előző fejezet végén írtam le, ha kihagytad volna, most pótold, mert enélkül nem működik a többi!

Ezután jön sorra a szükséges összes egyéb program:

========== 5.5. GCC-4.8.1 - Pass 1

Mielőtt a GCC lefordításába kezdenénk, előbb másoljuk be a $LFS/sources könyvtárba e három fájlt is az /Archives könyvtárból (kicsomagolás nélkül!):

mpfr-3.1.2.tar.xz
gmp-5.1.2.tar.xz
mpc-1.0.1.tar.gz

```
tar -Jxf ../mpfr-3.1.2.tar.xz
mv -v mpfr-3.1.2 mpfr
tar -Jxf ../gmp-5.1.2.tar.xz
mv -v gmp-5.1.2 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc

for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done

sed -i '/k prot/agcc_cv_libc_provides_ssp=yes' gcc/configure

mkdir -v ../gcc-build
cd ../gcc-build

../gcc-4.8.1/configure                                  \
    --target=$LFS_TGT                                   \
    --prefix=/tools                                     \
    --with-sysroot=$LFS                                 \
    --with-newlib                                       \
    --without-headers                                   \
    --with-local-prefix=/tools                          \
    --with-native-system-header-dir=/tools/include      \
    --disable-nls                                       \
```

```
    --disable-shared                                    \
    --disable-multilib                                  \
    --disable-decimal-float                             \
    --disable-threads                                   \
    --disable-libatomic                                 \
    --disable-libgomp                                   \
    --disable-libitm                                    \
    --disable-libmudflap                                \
    --disable-libquadmath                               \
    --disable-libsanitizer                              \
    --disable-libssp                                    \
    --disable-libstdc++-v3                              \
    --enable-languages=c,c++                            \
    --with-mpfr-include=$(pwd)/../gcc-4.8.1/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs

make

make install

ln -sv libgcc.a `$LFS_TGT-gcc -print-libgcc-file-name | sed 's/libgcc/&_eh/'`

================================== 5.6. Linux-3.10.10 API Headers

make mrproper

make headers_check

make INSTALL_HDR_PATH=dest headers_install

cp -rv dest/include/* /tools/include

============================ 5.7. Glibc-2.18

if [ ! -r /usr/include/rpc/types.h ]; then
  su -c 'mkdir -p /usr/include/rpc'
  su -c 'cp -v sunrpc/rpc/*.h /usr/include/rpc'
fi

sed -i -e 's/static __m128i/inline &/' sysdeps/x86_64/multiarch/strstr.c

mkdir -v ../glibc-build
cd ../glibc-build

../glibc-2.18/configure                             \
      --prefix=/tools                               \
      --host=$LFS_TGT                               \
      --build=$(../glibc-2.18/scripts/config.guess) \
      --disable-profile                             \
      --enable-kernel=2.6.32                        \
      --with-headers=/tools/include                 \
      libc_cv_forced_unwind=yes                     \
      libc_cv_ctors_header=yes                      \
      libc_cv_c_cleanup=yes

make

make install

echo 'main(){}' > dummy.c
$LFS_TGT-gcc dummy.c
readelf -l a.out | grep ': /tools'
```

A jó válasz:

```
  [Requesting program interpreter: /tools/lib64/ld-linux-x86-64.so.2]
```

A fenti sor 64 bites rendszerek esetén a jó válasz csak! 32 bites rendszerek esetén a jó válasz ez vagy valami ehhez nagyon hasonló kell legyen:

```
  [Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

```
================================== 5.8. Libstdc++-4.8.1

mkdir -pv ../gcc-build
cd ../gcc-build

../gcc-4.8.1/libstdc++-v3/configure \
    --host=$LFS_TGT                 \
    --prefix=/tools                 \
    --disable-multilib              \
    --disable-shared                \
    --disable-nls                   \
    --disable-libstdcxx-threads     \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/4.8.1

make

make install

====================== 5.9. Binutils-2.23.2 - Pass 2

sed -i -e 's/@colophon/@@colophon/' \
       -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo

mkdir -v ../binutils-build
cd ../binutils-build

CC=$LFS_TGT-gcc                \
AR=$LFS_TGT-ar                 \
RANLIB=$LFS_TGT-ranlib         \
../binutils-2.23.2/configure   \
    --prefix=/tools            \
    --disable-nls              \
    --with-lib-path=/tools/lib \
    --with-sysroot

make

make install

make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin

====================== 5.10. GCC-4.8.1 - Pass 2
```

Mielőtt a GCC lefordításába kezdenénk, előbb másoljuk be a $LFS/sources könyvtárba e három fájlt is a /Archives könyvtárból (kicsomagolás nélkül!):

mpfr-3.1.2.tar.xz
gmp-5.1.2.tar.xz
mpc-1.0.1.tar.gz

```
cat gcc/limitx.h gcc/glimits.h gcc/limity.h > \
  `dirname $($LFS_TGT-gcc -print-libgcc-file-name)`/include-fixed/limits.h

cp -v gcc/Makefile.in{,.tmp}
sed 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in


for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
  -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done

tar -Jxf ../mpfr-3.1.2.tar.xz
mv -v mpfr-3.1.2 mpfr
tar -Jxf ../gmp-5.1.2.tar.xz
mv -v gmp-5.1.2 gmp
tar -zxf ../mpc-1.0.1.tar.gz
mv -v mpc-1.0.1 mpc

mkdir -v ../gcc-build
cd ../gcc-build

CC=$LFS_TGT-gcc                                     \
CXX=$LFS_TGT-g++                                    \
AR=$LFS_TGT-ar                                      \
RANLIB=$LFS_TGT-ranlib                              \
../gcc-4.8.1/configure                              \
    --prefix=/tools                                 \
    --with-local-prefix=/tools                      \
    --with-native-system-header-dir=/tools/include  \
    --enable-clocale=gnu                            \
    --enable-shared                                 \
    --enable-threads=posix                          \
    --enable-__cxa_atexit                           \
    --enable-languages=c,c++                        \
    --disable-libstdcxx-pch                         \
    --disable-multilib                              \
    --disable-bootstrap                             \
    --disable-libgomp                               \
    --with-mpfr-include=$(pwd)/../gcc-4.8.1/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs

make

make install
```

```
ln -sv gcc /tools/bin/cc

echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

A jó válasz:

```
  [Requesting program interpreter: /tools/lib64/ld-linux-x86-64.so.2]
```

A fenti sor 64 bites rendszerek esetén a jó válasz csak! 32 bites rendszerek esetén a jó válasz ez vagy valami ehhez nagyon hasonló kell legyen:

```
  [Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

```
==================== 5.11. Tcl-8.6.0

sed -i s/500/5000/ generic/regc_nfa.c

cd unix
./configure --prefix=/tools

make

make install

chmod -v u+w /tools/lib/libtcl8.6.so

make install-private-headers

ln -sv tclsh8.6 /tools/bin/tclsh

================= 5.12. Expect-5.45

cp -v configure{,.orig}
sed 's:/usr/local/bin:/bin:' configure.orig > configure

./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include

make

make SCRIPTS="" install

========================= 5.13. DejaGNU-1.5.1

./configure --prefix=/tools
make install
make check

============================== 5.14. Check-0.9.10

./configure --prefix=/tools
make
make install

============================== 5.15. Ncurses-5.9

./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

```
make

make install

============================== 5.16. Bash-4.2

patch -Np1 -i ../bash-4.2-fixes-12.patch

./configure --prefix=/tools --without-bash-malloc

make

make install

ln -sv bash /tools/bin/sh

================================ 5.17. Bzip2-1.0.6

make

make PREFIX=/tools install

============================== 5.18. Coreutils-8.21

./configure --prefix=/tools --enable-install-program=hostname

make

make install

============================== 5.19. Diffutils-3.3

./configure --prefix=/tools

make

make install

================================ 5.20. File-5.14

./configure --prefix=/tools

make

make install

============================== 5.21. Findutils-4.4.2

./configure --prefix=/tools

make

make install

================================ 5.22. Gawk-4.1.0

./configure --prefix=/tools

make

make install
```

```
================================== 5.23. Gettext-0.18.3

cd gettext-tools
EMACS="no" ./configure --prefix=/tools --disable-shared

make -C gnulib-lib
make -C src msgfmt

cp -v src/msgfmt /tools/bin

================================ 5.24. Grep-2.14

./configure --prefix=/tools

make

make install

================================ 5.25. Gzip-1.6

./configure --prefix=/tools

make

make install

================================ 5.26. M4-1.4.16

sed -i -e '/gets is a/d' lib/stdio.in.h

./configure --prefix=/tools

make

make install

================================ 5.27. Make-3.82

./configure --prefix=/tools

make

make install

================================ 5.28. Patch-2.7.1

./configure --prefix=/tools

make

make install

================================ 5.29. Perl-5.18.1

patch -Np1 -i ../perl-5.18.1-libc-1.patch

sh Configure -des -Dprefix=/tools

make

cp -v perl cpan/podlators/pod2man /tools/bin
```

```
mkdir -pv /tools/lib/perl5/5.18.1
cp -Rv lib/* /tools/lib/perl5/5.18.1
```

```
================================== 5.30. Sed-4.2.2
```

```
./configure --prefix=/tools
```

```
make
```

```
make install
```

```
================================== 5.31. Tar-1.26
```

```
sed -i -e '/gets is a/d' gnu/stdio.in.h
```

```
./configure --prefix=/tools
```

```
make
```

```
make install
```

```
================================== 5.32. Texinfo-5.1
```

```
./configure --prefix=/tools
```

```
make
```

```
make install
```

```
================================== 5.33. Xz-5.0.5
```

```
./configure --prefix=/tools
```

```
make
```

```
make install
```

```
==========================================
```

Ezzel elkészült az „ideiglenes rendszer”! Hogy kisebb legyen a mérete, eltávolítjuk a binárisokból és a libekből a nekünk szükségtelen debug-infókat, ezzel jelentősen csökken a mérete, ami nem mindegy ha kevés a helyünk. Továbbá töröljük a nekünk szükségtelen dokumentációkat is belőle. Ez úgyis csak az ideiglenes rendszer, nem a végleges, azaz ezek akkor se kellenek most, ha később a végleges rendszerben meg akarjuk tartani a doksikat majd.

A sztrippelés során gyakran ír majd ki olyasmit (angolul persze), hogy nem ismeri a fájlformátumot, ez ne zavarjon. Azok többnyire mindenféle szkriptek csak.

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
rm -rf /tools/{,share}/{info,man,doc}
```

```
==========================================
```

A továbbiakban root-ként kell tényked nünk, azaz FOKOZOTT ÓVATOSSÁG AJÁNLOTT!

Mindenekelőtt lépjünk ki az lfs felhasználó shelljéből az

```
exit
```

paranccsal, majd adjuk a root tulajdonába a tools könyvtárat:

```
chown -R root:root $LFS/tools
```

4. fejezet: Milyen legyen a végleges rendszerünk?

„Elkészült hát a nagy mű, igen”, hogy a klasszikust idézzem... vagy mégsem? Hát ugye, legalábbis az ideiglenes rendszer készen van, s ez óriási szó! Na most előre sejthető, hogy te, kedves Olvasóm vélhetőleg sokszor elszúrod majd a végleges rendszered építését! Lehet, hogy jópárszor neki kell kezdened majd. Sőt lehet, hogy ha sikerül is, de több, különböző koncepciókat megvalósító végleges rendszert is óhajtasz majd építeni. Emiatt azt javaslom, mielőtt tovább lépsz, MENTSD EL A TELJES **tools** KÖNYVTÁRAT! Azt, ami a $LFS/tools. Legegyszerűbb ha mondjuk csinálsz a root könyvtárában egy BACKUP nevű könyvtárat, s egyszerűen bemásolod abba a tools könyvtárat mindenestül. Persze, ha van erre elég helyed, csak akkor. Nyilván persze tömörítheted is, ha akarod. Ez esetben akárhogy elcseszed is az építendő végleges rendszert, vagy akár azt is, ami a /tools -ban van, nincs komolyabb baj, mert INNEN, e fejezettől folytathatod, illetve újrakezdheted: Letörölsz a partícióról mindent, az ottani /tools könyvtárat is mert hátha abban is történt valami gond, kicsomagolod oda a tools-t a biztonsági mentésből (vagy ha nem tömörítetted be, akkor ki se kell csomagolni, csak odamásolni) és INNEN, e fejezettől folytathatsz mindent, nem a legelejétől! Hatalmas időspórolás!

Lényeg az, hogy ugye a könyv (vagy legalábbis doksi...) elején arról elmélkedtem, akkor van értelme egy disztrónak, ha valami újat is nyújt, nemcsak a meglévők mellé egy n+1-edik, amiben csak a háttérkép más meg mondjuk féltucat program.

Milyen legyen hát a disztribúciónk?

Hogy a TIED milyen legyen, azt neked kell eldöntened, én csak azt írhatom le, ÉN miként döntöttem.

Döntésem nem volt nehéz, mert nem csinálok titkot belőle, hogy eleve azért kezdtem bele ebbe az egész intellektuális kalandba, mert kedvenc disztróm, a GoboLinux, az már csak „egykori”, „néhai”, ugyanis immár nem fejlesztik. De ha fejlesztenék, akkor is muszáj lett volna ebbe belevágnom, mert 64 bites gépre tértem át, s a gobónak soha nem volt 64 bites változata.

A GoboLinux legfőbb tulajdonsága az volt, hogy benne minden program a maga külön könyvtárába van telepítve, azon belül egy verziószámmal meghatározott alkönyvtárba, például:

/Programs/Bash/4.2
/Programs/BinUtils/2.23.2

stb.

Ez nekem nagyon tetszett, effélét szerettem volna megvalósítani. Semmi értelmét nem látom ugyanis – legalábbis nem azok számára akik otthon, maguknak, desktop célokra használják a Linuxot – hogy a csomagokat alkotó fájlokat szétszórjuk milliónyi különböző helyre, állítólag úgymond „funkciók szerint”, de még ezt a koncepciót se tartja be következetesen a „hagyományos” disztrók egyike sem, mert például a végrehajtható állományoknak sem csak 1 könyvtár van fenntartva, hanem egy egész sereg:

/bin
/sbin
/usr/bin
/usr/sbin
/usr/local/bin

sőt vannak programok, amik az /opt -ba kerülnek vagy máshova.

Aztán szegény user totózza ki, melyik fájl melyik programhoz tartozik, és hol van éppen!

Persze, aki valami bináris disztrót használ, amit már készre megcsináltak neki, ezzel a dologgal többnyire nem foglalkozik, nem érdekli, melyik fájl hol van, s ha mégis, majd lekérdezi valahogy a csomagkezelőből. A baj ezzel az, ha mégis forrásból tesz fel dolgokat, az a program nem kerül bele a csomagkezelő információi közé, annak adatbázisába.

Továbbá, arról volt szó, hogy TE állítólag olyan disztrót szeretnél használni, amit teljesen személyre szabhatsz a magad igényei szerint, amibe csak az kerül bele, amit te szeretnél, semmi se, amit nem szeretnél, stb. Ez esetben nem engedheted meg magadnak azt a luxust, amit a „mezei” linuxuserek, a „tudatlanság luxusát”, azaz neked igenis MUSZÁJ tudnod, melyik fájl melyik programhoz tartozik, különben azt se tudod, mi van a rendszeredben, mi miért van ott, s mi vált esetleg feleslegessé. Erre kell valami módot találni.

Ezen elmélkednünk cseppet se felesleges, mert ha elolvasod az LFS Bookot, abban egy egész fejezet van e témának szentelve, épp azon a részen, ahova mostanára eljutottunk mi is a munkálkodásunk közepette! Ez a fejezet az:

http://www.linuxfromscratch.org/lfs/view/stable/chapter06/pkgmgt.html

Mindenképp át kell gondolnunk a csomagkezelés kérdéskörét, mielőtt még egyet is feltelepítenénk a végleges rendszerünkbe!

Én a GoboLinuxnak akartam létrehozni egy friss verzióját, de mert azt már nem fejlesztik, nem láttam nélkülözhetetlennek, hogy kompatibilis maradjak vele. Még talán úgy is döntöttem volna hogy kompatibilis maradok, de sajnos az a helyzet, hogy annak legalapvetőbb fejlesztőeszközei, például a csomagkezelést

megvalósító összes szkript Python nyelven van írva, holott az LFS Book szerinti sorrendben feltéve a programokat, a Python olyan későn jön, hogy benne sincs az LFS-ben, csak annak folytatásában, a BLFS-ben!

Úgy döntöttem tehát, efféle programtelepítési metódust valósítok meg, de megírom magam a szükséges szkripteket hozzá. Ez tehát azt jelenti, hogy nekem nem lesz valami nagyranőtt külön csomagkezelő programom a rendszerben, hanem a csomagkezelő, az néhány viszonylag egyszerű szkript lesz, a csomagkezelő ADATBÁZISA pedig az megintcsak nem valami speciális bináris fájl lesz, hanem maga a FÁJLRENDSZER. Miért is ne, eredetileg épp azért jött létre annak idején a fájlrendszer, azért találták ki, hogy RENDSZERT vigyen a programok s más fájlok elhelyezésébe, rendet tartson köztük! Már a neve is erre utal: fájl**RENDSZER**!

Igen ám, de a jelenlegi linuxos ökoszisztéma programjai nem ilyen rendszerre vannak felkészítve. Ők azt várják, legyen egy $PATH változó, amiben néhány könyvtár szerepel, s azok valamelyikében megtalálják majd a megfelelő bináris programot vagy szkriptet, amit végre kell hajtaniuk. Hasonlóképp vannak változói a header fájloknak, a libraryknak, s egyebeknek is.

Nincs más megoldás hát, ha ragaszkodunk hozzá, hogy minden program a maga külön könyvtárába legyen telepítve, valamiképp emulálnunk/szimulálnunk kell a hagyományos könyvtárstruktúrát nekik, s e célra a Linux operációs rendszer „szimbolikus link”-nek nevezett lehetőségét vetjük be. Azaz, lesz egy könyvtár, ami arra szolgál, hogy benne össze legyen gyűjtve a telepített programok minden végrehajtható állományáról egy-egy arra mutató symlink. Hasonlóképp járunk el a header fájlokkal, librarykkal, mindennel. Ezt így csinálja a GoboLinux is, illetve csinálta, amíg létezett. Eképp egy-egy ilyen symlinkeket tartalmazó könyvtár tulajdonképpen megfeleltethető egy „lekérdezésnek” a hagyományos adatbáziskezelők fogalmai szerint, mely lekérdezés a rendszerbe telepített programok egy bizonyos fontos tulajdonságára vonatkozik, hogy az végrehajtható állomány-e, vagy headerfájl, vagy library, stb.

Nyilván persze mazochistáknak való munka lenne egyesével symlinkelgetnünk a fájlokat minden telepítés után, ezért erre szkriptet kell írni.

Ha azonban már úgyis symlinkelgetünk, miért is ne lehetne magát az egész telepített rendszert is egyetlen symlinktől függővé tenni?! Azaz elérni, hogy egyetlen partícióra akárhány releaset is telepíthessünk a mi disztrónkból, úgy, hogy ne zavarják egymást?! Ez nagyon jó lenne, mert gyakori lehet, hogy ki szeretnénk próbálni valamit, de nincs már rá szabad partíciónk, ami még teljesen üres. Ám volna hely rá valamelyik másik partíción bőven, csak ott már van valami, amit nem törölhetünk le...

Ez aztán tényleg valami abszolút új PLUSZ a disztrómban, jelentős vagy legalábbis izgalmas „hozzáadott érték” mely szignifikánsan megkülönbözteti az összes létező disztrótól, de még a valaha létezett bármelyiktől is! Mert ha jelenleg nincs is olyan disztró, melyben a programok külön könyvtárakba vannak telepítve, de régen volt: A GoboLinux. Igaz, 64 bitesben az se létezett soha. De olyan, hogy az egész disztró maga is egyetlen könyvtárba legyen telepítve, olyan még nem volt soha sehol se!

Tehát, a következő könyvtárrendszert találtam ki. Ideírom a könyvtárakat, magyarázattal együtt, mi mire szolgál:

A gyökérkönyvtárban ezt látjuk:

```
lrwxrwxrwx  1 root root     13 dec   17 04.10 _ -> Releases/2014
lrwxrwxrwx  1 root root      7 dec   17 04.10 bin -> _/S/L/E
lrwxrwxrwx  1 root root     15 dec   17 04.10 boot -> _/S/Kernel/Boot
lrwxrwxrwx  1 root root     18 dec   17 04.10 dev -> _/S/Kernel/Devices
lrwxrwxrwx  1 root root      5 dec   17 04.10 etc -> _/S/O
lrwxrwxrwx  1 root root     10 dec   17 04.10 home -> _/U/Common
lrwxrwxrwx  1 root root      7 dec   17 04.10 lib -> _/S/L/Y
lrwxrwxrwx  1 root root      7 dec   17 04.10 lib64 -> _/S/L/Y
drwx------  2 root root  16384 dec    6 10.01 lost+found
lrwxrwxrwx  1 root root      7 dec   17 04.10 media -> _/Mount
lrwxrwxrwx  1 root root      7 dec   17 04.10 mnt -> _/Mount
lrwxrwxrwx  1 root root      7 dec   26 00.31 Mount -> _/Mount
lrwxrwxrwx  1 root root     17 dec   17 04.10 proc -> _/S/Kernel/Status
drwxr-xr-x  3 root root   4096 dec   17 04.07 Releases
lrwxrwxrwx  1 root root      8 dec   17 04.10 root -> _/U/root
lrwxrwxrwx  1 root root      9 dec   17 04.10 run -> _/S/V/run
lrwxrwxrwx  1 root root      7 dec   17 04.10 sbin -> _/S/L/E
drwxr-xr-x  3 root root   4096 jan    4 13.32 sources
lrwxrwxrwx  1 root root      3 dec   17 04.10 srv -> _/F
lrwxrwxrwx  1 root root     18 dec   17 04.10 sys -> _/S/Kernel/Objects
lrwxrwxrwx  1 root root      9 dec   17 04.10 tmp -> _/S/V/tmp
lrwxrwxrwx  1 root root      5 dec   17 04.10 usr -> _/usr
lrwxrwxrwx  1 root root      5 dec   17 04.10 var -> _/S/V
```

Megjegyzendő, hogy ha 32 bites rendszert építesz, akkor a fenti listában szereplő lib64 nevű könyvtár felesleges neked.

A többi könyvtár tartalma:

```
/Releases :

drwxr-xr-x 8 root root 4096 dec   18 05.06 2014
```

Ez arra szolgál, hogy e Releases könyvtárba rakjuk disztrónk különböző verziójú telepítéseit. Ebben jelenleg csak egy van, a 2014 verziószámot viselő. A gyökérkönyvtárban levő /_ symlink mutat az aktuális verzióra, amit végrehajt, működtet a gépünk, amikor e partícióra bebootolunk. Jelen esetben tehát e /_ link a /Releases/2014 könyvtárra mutat. Minthogy a gyökérkönyvtárban levő összes többi bejegyzés csak symlink, és mindegyik kivétel nélkül e /_ symlinken át mutat valami tényleges könyvtárra, ezért új releasera áttérni, vagy onnan a régire visszatérni rémségesen egyszerű: Mindössze a /_ symlinket kell átírnunk ahhoz, hogy valamely új (vagy régi) release könyvtárára mutasson!

Megjegyzendő, hogy még az se kötelező, hogy a release könyvtára (jelen esetben a 2014 nevű tartalomjegyzék) a Releases könyvtárban helyeztessék el. Akárhova teheted, akár valami félreeső sokadik altartalomjegyzék mélyére is, 36 subdirectory mélységbe is, mindegy neki. A lényeg, hogy a /_ symlink az ő könyvtárára mutasson! Ha akarod, bármikor át is nevezheted a könyvtárát,

lehet a neve 2014 helyett az is, hogy NaEztAFuraAgybajtKiKellProbalnom - teljesen mindegy, csak a /_ link mutasson rá, e linket ne felejtsd el aktualizálni!

A tartalomjegyzékben levő linkek természetesen azt a célt szolgálják, hogy emulálják az esetleg rosszul megírt, mert fixen „belehuzalozott” útvonalakat tartalmazó programok számára a hagyományos könyvtárszerkezetet.

```
/Releases/2014 :

drwxr-xr-x   3 root root  4096 dec   20 21.27 F
lrwxrwxrwx   1 root root     1 dec   17 04.07 home -> U
drwxr-xr-x   6 root   root  4096 jan    4 19.57 Mount
drwxr-xr-x 330 root root 12288 jan    3 23.06 P
lrwxrwxrwx   1 root root     6 dec   17 04.07 root -> U/root
drwxr-xr-x   6 root root  4096 dec   18 05.06 S
drwxr-xr-x   4 root root  4096 nov   25 23.31 U
drwxr-xr-x   3 root root  4096 jan    3 23.04 usr


/Releases/2014/U :

drwxr-xr-x  4 root root 4096 dec   24 18.05 Common
drwxr-x--- 14 root root 4096 jan    2 23.19 root
```

Ez az U könyvtár tartalmazza a rendszerünkben levő összes user HOME könyvtárát. Itt van mindenekelőtt a root, azaz annak könyvtára a /Releases/2014/U/root helyen van. Van itt azonban egy Common nevű könyvtár is, ebben vannak a „közönséges”, nem rendszergazdai jogú userek könyvtárai is, tehát ezek a /Releases/2014/U/Common könyvtárban vannak. Ennek az az értelme, hogy meg lehet csinálni, hogy csak a root van a helyi gépen, s a többi user esetleg egy másik partíción, vagy akár egy hálózati könyvtárban.

```
/Releases/2014/F :

drwxrwxrwx 17 root root 4096 dec   21 00.44 X11
```

Ezen X11 könyvtárba lesznek majd telepítve a fontkészletek.

A Mount könyvtárban hozunk majd létre csatolási pontokat a különböző eszközöknek, partícióknak.

A P könyvtárba lesznek telepítve a programok, mindegyik a maga könyvtárába. Ez az, ami megfelel az egykori GoboLinux /Programs könyvtárának.

Az S könyvtár felel meg a GoboLinux „System” könyvtárának. Ennek felépítése:

```
/Releases/2014/S :

drwxr-xr-x  6 root root 4096 dec    5 19.42 Kernel
drwxr-xr-x 11 root root 4096 jan    2 23.16 L
drwxr-xr-x 27 root root 4096 jan    1 22.41 O
drwxr-xr-x 14 root root 4096 dec   27 14.31 V
```

A Kernel könyvtár felépítése:

```
/Releases/2014/S/Kernel :

drwxr-xr-x   3 root root 4096 dec   18 22.02 Boot
drwxr-xr-x  14 root root 3420 jan    4 19.56 Devices
dr-xr-xr-x  12 root root    0 jan    4 12.01 Objects
dr-xr-xr-x 107 root root    0 jan    4 12.01 Status
```

A Boot könyvtárban van egy **grub** nevű alkönyvtár.

Az L könyvtár tartalmazza a szimbolikus linkeket. Felépítése:

```
/Releases/2014/S/L :

drwxr-xr-x   2 root root 12288 jan    3 22.29 C
drwxr-xr-x   5 root root  4096 dec   13 05.16 D
drwxr-xr-x   3 root root 53248 jan    3 23.53 E
drwxr-xr-x 138 root root 20480 jan    3 22.29 H
drwxr-xr-x 100 root root  4096 jan    2 01.27 T
drwxr-xr-x  12 root root  4096 dec   31 23.33 X
drwxr-xr-x  70 root root 65536 jan    3 22.43 Y
```

A könyvtárak jelentése:

C : pkgconfig fájlok linkjei. Ezek azok, amik úgy végződnek, hogy „.pc”.
D : dokumentumok linkjei.
E : végrehajtható fájlok linkjei.
H : Header (include) fájlok linkjei.
X : Libexec fájlok linkjei.
Y : Libraryk linkjei
T : az úgynevezett „shared” fájlok linkjei, de én csak úgy nevezem magamban hogy „Trash”, azaz „szemét”, mert ez az a kuka, ahova a „minden egyéb” kerül.

A D könyvtár tartalma:

```
/Releases/2014/S/L/D :

drwxr-xr-x 86 root root 4096 jan    3 22.43 doc
drwxr-xr-x  2 root root 4096 jan    3 22.14 I
drwxr-xr-x 42 root root 4096 dec   27 14.24 M
```

A doc-ba kerülnek a doksik, az I-be az info fájlok, az M-be pedig a man-oldalak, azaz a manuálok.

Az Y könyvtárban kell legyen egy könyvtár **services** néven, s egy **lsb** nevű symlink, ami e services könyvtárra mutat!

A T könyvtárban kell legyen egy **man** nevű symlink, ami a

```
../D/M
```

könyvtárra mutat! Továbbá e T könyvtárban kell legyen egy **fonts** nevű alkönyvtár, s ebben egy X11 nevű symlink, ami a /_/F/X11 könyvtárra mutat!

A /Releases/2014/S/O könyvtár felel meg a hagyományos disztróknál levő etc könyvtárnak. Fogd fel úgy, hogy az „O” annak rövidítése, hogy „Options”.

A /Releases/2014/S/V tartalma:

```
drwxrwxr-x 2 root root 4096 dec    2 21.11 cache
drwxr-xr-x 2 root root 4096 dec    2 21.11 db
drwxr-xr-x 3 root root 4096 dec    2 21.11 empty
drwxr-xr-x 7 root root 4096 dec    2 21.11 lib
drwxr-xr-x 2 root root 4096 okt   27 06.12 local
lrwxrwxrwx 1 root root    8 dec    6 01.54 lock -> run/lock
drwxr-xr-x 2 root root 4096 dec    2 21.11 log
drwxr-xr-x 2 root root 4096 nov   24 02.52 mail
drwxr-xr-x 2 root root 4096 okt   27 06.12 opt
drwxr-xr-x 3 root root 4096 dec    2 21.38 run
drwxr-xr-x 2 root root 4096 okt   27 06.07 spool
drwxrwxrwx 2 root root 4096 dec    2 21.10 tmp
```

Mint látható, itt a változó adatok vannak (V=Variables). Ezek közül azok, amik kezdetben nem üresek:

```
empty:
drwxr-xr-x 2 root root 4096 dec    2 21.11 dev

lib:
drwxr-xr-x 2 root root 4096 okt   29 02.00 hwclock
drwxr-xr-x 2 root root 4096 okt   30 04.28 locate
drwxr-xr-x 2 root root 4096 2010 máj    7 misc
drwxr-x--- 2 root root 4096 2007 ápr   21 slocate
drwx------ 2 root root 4096 okt   31 02.28 sudo

run:
drwxr-xr-x 2 root root 4096 dec    2 21.38 lock
```

A /Releases/2014/usr tartalma :

```
lrwxrwxrwx 1 root root    8 dec   17 04.07 bin -> ../S/L/E
lrwxrwxrwx 1 root root   12 dec   17 04.07 doc -> ../S/L/D/doc
lrwxrwxrwx 1 root root    8 dec   17 04.07 include -> ../S/L/H
lrwxrwxrwx 1 root root    8 dec   17 04.07 lib -> ../S/L/Y
lrwxrwxrwx 1 root root    8 dec   17 04.07 lib64 -> ../S/L/A
lrwxrwxrwx 1 root root    8 dec   17 04.07 libexec -> ../S/L/X
lrwxrwxrwx 1 root root    1 dec   17 04.07 local -> .
lrwxrwxrwx 1 root root   10 dec   17 04.07 man -> ../S/L/D/M
lrwxrwxrwx 1 root root    8 dec   17 04.07 sbin -> ../S/L/E
lrwxrwxrwx 1 root root    8 dec   17 04.07 share -> ../S/L/T
drwxr-xr-x 2 root root 4096 okt   20 01.55 src
lrwxrwxrwx 1 root root   10 dec   17 04.07 tmp -> ../S/V/tmp
```

Hozzuk most létre a fenti könyvtárszerkezetet! Hogy ezt miként, azt rád bízom, használhatod az MC programot, vagy csak úgy kézzel...

5. fejezet: Az alapvető szkriptek, device node-ok és symlinkek létrehozása

Mielőtt chroottal belépnénk az elkészült ideiglenes rendszerbe, létre kell hoznunk pár alapvető úgynevezett „device node”-t a kernel számára! Hogy ez mi, azt most fedje jótékony homály, szerencsére a dologgal a későbbiekben nem kell foglalkoznod. A lényeg, hogy add ki e parancsokat:

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

Na és akkor most gondoljunk arra, hogy bár kb. azon a helyen tartunk a folyamatban, ahol is az LFS Book azt írja, hogyan kell felmountolni az ideiglenes rendszert a chroot számára, de nekünk ez nem lesz jó. Ugyanis eszméletlenül sok programot telepítünk majd, ami soká tart. Kell, hogy abba tudjuk hagyni néha, kikapcsolni a gépet, majd folytatni azt!

Azaz megalkotjuk a magunk szkriptjeit e célra. Meg azokat a szkripteket is, amik majd segítenek bennünket a magunk speciális könyvtárszerkezetét kezelni, karbantartani.

Mindenekelőtt: Én úgy neveztem el a rendszeremet, a „disztrómat”, hogy **KISS-Linux**. A „KISS”, az természetesen egy betűszó, azt a linuxos berkekben közismert elvet fejezi ki, amit angolul úgy mondanak, hogy:

Keep **I**t **S**imple, **S**tupid!

Azaz hogy tartsuk meg a rendszerünket egyszerűnek és butának. Illetve tulajdonképpen nem a rendszerünket, hanem a benne levő programokat. Összességében tudhat sokat a rendszer, de az egyes programok ne: azok mindegyike egy-egy jól meghatározható részfeladatot lásson el csak, ne tudjon sokat, de azt az egyet nagyon jól! Azaz a megvalósított funkciók egymással legyenek „gyengén kapcsoltak”, tehát ne nagyra nőtt progikat írjunk, amik úgymond „mindent tudnak”.

Na, ha ez KISS-Linux, akkor legyen először is egy „KISS” nevű szkript, amivel belépünk majd a chroot környezetbe, ahányszor csak akarunk! E szkriptet a host rendszer rootjának $HOME könyvtárába mentsük el.

Tehát, az elkészült ideiglenes rendszerbe e szkripttel lépek be:

```
root@Csiszilla:~# cat KISS
#!/bin/bash

mount -v --bind /dev $LFS/dev
mount -vt devpts devpts $LFS/dev/pts -o gid=5,mode=620
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
if [ -h $LFS/dev/shm ]; then
  link=$(readlink $LFS/dev/shm)
  mkdir -p $LFS/$link
  mount -vt tmpfs shm $LFS/$link
  unset link
```

```
else
  mount -vt tmpfs shm $LFS/dev/shm
fi

chroot "$LFS" /tools/bin/env -i \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='\u:\w\$ '              \
    PATH="/_/S/L/E:/tools/bin:/tools/sbin" \
    LD_LIBRARY_PATH="/_/S/L/Y:/tools/lib" \
C_INCLUDE_PATH="/_/S/L/H:/tools/include" \
PKG_CONFIG_PATH="/_/S/L/C:/tools/lib/pkgconfig" \
   /tools/bin/bash --login +h
```

A szkript utolsó sorának végén a +h nagyon fontos – ez biztosítja, hogy a shell nem jegyzi meg a végrehajtott programok helyét annak érdekében, hogy később gyorsabban hajtsa végre őket, hanem minden alkalommal végigvizsgálja a $PATH változó könyvtárait, megkeresendő őket! Márpedig figyeljük csak meg: Minden olyan rendszerváltozónk, ami arra végződik, hogy „PATH”, úgy van beállítva, hogy előbb szerepelnek bennük a leendő végleges rendszerünk könyvtárai, s csak azután a /tools rendszerben levő ideiglenes könyvtárak! Ez biztosítja, hogy amint telepítettünk egy programot a végleges rendszerünkbe, azt fogja azontúl használni a rendszerünk, nem azt, ami az ideiglenes rendszerben van, ugyanis a végleges rendszerbe bekerültet előbb találja meg.

A készítendő rendszer root userjének HOME könyvtárában is be kell állítanunk 2 fontos fájl tartalmát:

Az egyik szkript:

```
root@Csiszilla:/Mount/RAM/root# cat .bash_profile
# /etc/skel/.bash_profile

# This file is sourced by bash for login shells.  The following line
# runs your .bashrc and is recommended by the bash info pages.
[[ -f ~/.bashrc ]] && . ~/.bashrc
```

A másik szkript:

```
root@Csiszilla:/Mount/RAM/root# cat .bashrc
export PS1="\[\033[1;31m\]\u\[\033[1;36m\]@\[\033[0;36m\]\h \[\033[1m\]\w\
[\033[1;31m\]=>\[\033[0m\]"

export EDITOR="mcedit"

export MAKEFLAGS='-j 5'

alias x=exit
alias make=kolormake

alias mc='. /_/S/L/X/mc/mc-wrapper.sh'

alias KISS="cd /_/P/KISS/-/sources"
alias P="cd /_/P"
alias S="cd /_/S/O"
alias L="cd /_/S/L"
alias A="cd /Archives"
alias s="cd /sources"
```

```
#####################################################################x

kicsomagol () {
    if [ -f $1 ] ; then
    case $1 in
        *.tar.bz2)   tar xvjf $1        ;;
        *.tar.gz)    tar xvzf $1        ;;
        *.bz2)       bunzip2 $1         ;;
        *.rar)       unrar x $1         ;;
        *.gz)        gunzip $1          ;;
        *.tar)       tar xvf $1         ;;
        *.tbz2)      tar xvjf $1        ;;
        *.tgz)       tar xvzf $1        ;;
        *.zip)       unzip $1           ;;
        *.Z)         uncompress $1      ;;
        *.z)        7z x $1            ;;
        *.arj)       arj x $1           ;;
        *.xz)        tar -xJvf $1           ;;

        *)           echo "A(z) '$1' nem kicsomagolható" ;;
    esac
        else
        echo "'$1' nevű állomány nem létezik"
        fi
            }
```

Eddig a két szkript. A MAKEFLAGS értékénél az 5 helyére azt írd, ami neked a legmegfelelőbb.

A „kicsomagol” függvény nagyon hasznos, mert így a chroot környezetben is tudsz kicsomagolni forrásokat, nem muszáj mindig visszatérned ehhez a host rendszerbe, azaz munkaasztalt váltani.

Készítsünk egy **UKISS** nevű szkriptet is, a host rendszer rootjának HOME könyvtárába, aminek tartalma:

```
#!/bin/bash
umount -v $LFS/dev/pts

if [ -h $LFS/dev/shm ]; then
  link=$(readlink $LFS/dev/shm)
  umount -v $LFS/$link
  unset link
else
  umount -v $LFS/dev/shm
fi

umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

E fenti szkripttel csatolhatjuk le az épített rendszer partíciójának dev könyvtárába felcsatolt könyvtárakat, hogy azután umountolni tudjuk azt, ha óhajtjuk.

Ha tehát valahol abba akarjuk hagyni a rendszerépítést, annyi a dolgunk, hogy egy „**exit**” paranccsal kilépünk a chroot környezet rootjának shelljéből, ekkor a host rendszer rootjának a shelljében leszünk. Ezután

```
cd
```

hogy biztosan a /root könyvtárban legyünk, majd:

```
./UKISS
```

erre lecsatolja a könyvtárakat. Ezután — amennyiben más virtuális munkaasztalokon nem vagyunk belépve arra a partícióra — lecsatolhatjuk az egész partíciót is az

```
umount $LFS
```

paranccsal.

Ha legközelebb folytatni akarjuk a fejlesztést, mindenekelőtt csatoljuk a partíciót a

```
mount /dev/sdX $LFS
```

paranccsal (sdX helyére a megfelelő eszköznevet írva, s ehhez persze már root kell legyünk), aztán

```
cd
```

hogy biztos a /root könyvtárban legyünk, majd

```
./KISS
```

s erre máris csatolódott minden, s chrootban vagyunk, folytathatjuk a fejlesztést!

Ezután ne feledjük el kiadni mindenesetre a

```
cd /sources
```

parancsot, különben nem találjuk majd a fordítandó programokat...

Lépjünk is be most a chrootba a KISS szkripttel, s hozzunk létre bizonyos alapvető symlinkeket! Sajnos akadnak olyan megátalkodott genyó programok, amik ragaszkodnak hozzá hogy okvetlenül egy bizonyos helyen keressenek egyes más programokat. Ezek igényeit ki kell elégítenünk:

```
ln -sv /tools/bin/{bash,cat,echo,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib
sed 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
ln -sv bash /bin/sh

ln -sv /proc/self/mounts /etc/mtab

touch /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/log/lastlog
```

```
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

Ezek után épp itt az ideje, hogy megalkossuk a magunk szkriptjeit, melyek elvégzik a tulajdonképpeni csomagkezelést nekünk!

E szkriptek természetesen szintén a /_/P fában lesznek, egy ottani KISS nevű könyvtárban. Ezt létre kell hoznunk:

```
mkdir -pv /_/P/KISS/0.0/sources
mkdir -pv /_/P/KISS/0.0/bin
```

A sourcesbe kerülnek a forrásaink, amennyiben C nyelvű programot is írunk néha, a bin-be pedig a végrehajtandó szkriptek illetve a C programjaink futtatható binárisai.

E sources könyvtárba mindenekelőtt csináljunk egy **colored.c** nevű fájlt, ami ezt tartalmazza:

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>
#include <stddef.h>

#define MAXCHAR 32767

#define COLOR_RED     "\x1b[31m"
#define COLOR_GREEN   "\x1b[32m"
#define COLOR_YELLOW  "\x1b[33m"
#define COLOR_BLUE    "\x1b[34m"
#define COLOR_MAGENTA "\x1b[35m"
#define COLOR_CYAN    "\x1b[36m"
#define COLOR_WHITE    "\x1b[37m"
#define COLOR_BRIGHT  "\033[01m"
#define COLOR_UNDERLINE "\033[04m"

#define COLOR_RESET   "\x1b[0m"

struct cser {
const char *regi;
const char *uj;
};

struct cser csere[] = {
{"error", COLOR_RED COLOR_BRIGHT "error" COLOR_RESET},
{"Error", COLOR_RED COLOR_BRIGHT "Error" COLOR_RESET},
{"broken", COLOR_RED COLOR_BRIGHT "broken" COLOR_RESET},
{"Broken", COLOR_RED COLOR_BRIGHT "Broken" COLOR_RESET},
{"ERROR", COLOR_RED COLOR_BRIGHT "ERROR" COLOR_RESET},
{"hiba", COLOR_RED COLOR_BRIGHT "hiba" COLOR_RESET},
{"Hiba", COLOR_RED COLOR_BRIGHT "Hiba" COLOR_RESET},
{"HIBA", COLOR_RED COLOR_BRIGHT "HIBA" COLOR_RESET},
{"MISSING", COLOR_RED COLOR_BRIGHT "MISSING" COLOR_RESET},
{"Missing", COLOR_RED COLOR_BRIGHT "Missing" COLOR_RESET},
{"missing", COLOR_RED COLOR_BRIGHT "missing" COLOR_RESET},
{"failure", COLOR_RED COLOR_BRIGHT "failure" COLOR_RESET},
{"Failure", COLOR_RED COLOR_BRIGHT "Failure" COLOR_RESET},
```

```
{"FAILURE", COLOR_RED COLOR_BRIGHT "FAILURE" COLOR_RESET},
{"bad", COLOR_RED COLOR_BRIGHT "bad" COLOR_RESET},
{"Bad", COLOR_RED COLOR_BRIGHT "Bad" COLOR_RESET},
{"BAD", COLOR_RED COLOR_BRIGHT "BAD" COLOR_RESET},
{"File exists", COLOR_RED COLOR_BRIGHT "File exists" COLOR_RESET},
{"ln: failed to create symbolic link", COLOR_RED COLOR_BRIGHT "ln: failed to
create symbolic link" COLOR_RESET},
{"executing", COLOR_RED "executing" COLOR_RESET},
{"rm ", COLOR_RED COLOR_BRIGHT COLOR_UNDERLINE "rm " COLOR_RESET},
{"warning", COLOR_MAGENTA "warning" COLOR_RESET},
{"Warning", COLOR_MAGENTA "Warning" COLOR_RESET},
{"WARNING", COLOR_MAGENTA "WARNING" COLOR_RESET},
{"unknown", COLOR_MAGENTA "unknown" COLOR_RESET},
{"Unknown", COLOR_MAGENTA "Unknown" COLOR_RESET},
{"UNKNOWN", COLOR_MAGENTA "UNKNOWN" COLOR_RESET},
{"invalid", COLOR_MAGENTA "invalid" COLOR_RESET},
{"Invalid", COLOR_MAGENTA "Invalid" COLOR_RESET},
{"INVALID", COLOR_MAGENTA "INVALID" COLOR_RESET},
{"gcc ", COLOR_YELLOW "gcc " COLOR_RESET},
{"g++ ", COLOR_YELLOW "g++ " COLOR_RESET},
{"ld ", COLOR_YELLOW "ld " COLOR_RESET},
{" CC ", COLOR_YELLOW " CC " COLOR_RESET},
{"CC ", COLOR_YELLOW "CC " COLOR_RESET},
{"\tCC", COLOR_YELLOW "\tCC" COLOR_RESET},
{"checking ", COLOR_YELLOW "checking " COLOR_RESET},
{"check ", COLOR_YELLOW "check " COLOR_RESET},
{"Checking ", COLOR_YELLOW "Checking " COLOR_RESET},
{"Check ", COLOR_YELLOW "Check " COLOR_RESET},
{" yes", COLOR_BLUE COLOR_BRIGHT " yes" COLOR_RESET},
{"\tyes", COLOR_BLUE COLOR_BRIGHT "\tyes" COLOR_RESET},
{" Yes", COLOR_BLUE COLOR_BRIGHT " Yes" COLOR_RESET},
{"succeeded", COLOR_BLUE COLOR_BRIGHT "succeeded" COLOR_RESET},
{"Succeeded", COLOR_BLUE COLOR_BRIGHT "Succeeded" COLOR_RESET},
{"SUCCEEDED", COLOR_BLUE COLOR_BRIGHT "SUCCEEDED" COLOR_RESET},
{"\tYes", COLOR_BLUE COLOR_BRIGHT "\tYes" COLOR_RESET},
{"Symlinking", COLOR_YELLOW COLOR_BRIGHT "Symlinking" COLOR_RESET},
{"install", COLOR_YELLOW COLOR_BRIGHT "install" COLOR_RESET},
{"Install", COLOR_YELLOW COLOR_BRIGHT "Install" COLOR_RESET},
{"INSTALL", COLOR_YELLOW COLOR_BRIGHT "INSTALL" COLOR_RESET},
{"linking", COLOR_YELLOW COLOR_BRIGHT "linking" COLOR_RESET},
{"ln -s", COLOR_YELLOW COLOR_BRIGHT "ln -s" COLOR_RESET},
{"Linking", COLOR_YELLOW COLOR_BRIGHT "Linking" COLOR_RESET},
{"installing", COLOR_YELLOW COLOR_BRIGHT "installing" COLOR_RESET},
{"Installing", COLOR_YELLOW COLOR_BRIGHT "Installing" COLOR_RESET},
{"install:", COLOR_YELLOW COLOR_BRIGHT "install:" COLOR_RESET},
{"Install:", COLOR_YELLOW COLOR_BRIGHT "Install:" COLOR_RESET},
{"Moving", COLOR_YELLOW COLOR_BRIGHT "Moving" COLOR_RESET},
{"creating", COLOR_YELLOW COLOR_BRIGHT "creating" COLOR_RESET},
{"Creating", COLOR_YELLOW COLOR_BRIGHT "Creating" COLOR_RESET},
{"No such file or directory", COLOR_RED COLOR_BRIGHT "No such file or directory"
COLOR_RESET},
{" no", COLOR_RED COLOR_BRIGHT " no" COLOR_RESET},
{"\tno", COLOR_RED COLOR_BRIGHT "\tno" COLOR_RESET},
{" No", COLOR_RED COLOR_BRIGHT " No" COLOR_RESET},
{"\tNo", COLOR_RED COLOR_BRIGHT "\tNo" COLOR_RESET},
{"failed", COLOR_RED COLOR_BRIGHT "failed" COLOR_RESET},
{"Failed", COLOR_RED COLOR_BRIGHT "Failed" COLOR_RESET},
{"szimbolikus link létrehozása meghiúsult: A fájl már létezik", COLOR_RED
COLOR_BRIGHT "szimbolikus link létrehozása meghiúsult: A fájl már létezik"
COLOR_RESET},
{"FAILED", COLOR_RED COLOR_BRIGHT "FAILED" COLOR_RESET},
{"==>", COLOR_CYAN COLOR_BRIGHT "==>" COLOR_RESET},
```

```
{"->", COLOR_CYAN COLOR_BRIGHT "->" COLOR_RESET},
{"skipped", COLOR_CYAN "skipped" COLOR_RESET},
{"SKIP:", COLOR_CYAN "SKIP:" COLOR_RESET},
{"mkdir:", COLOR_CYAN "mkdir:" COLOR_RESET},
{"ln:", COLOR_CYAN "ln:" COLOR_RESET},
{"Command for ", COLOR_YELLOW COLOR_BRIGHT "Command for " COLOR_RESET},


};

int MAXCSERE = sizeof(csere)/sizeof(struct cser);

char *replace_str(const char *str, const char *old, const char *new)
{
char *ret, *r;
const char *p, *q;
size_t oldlen = strlen(old);
size_t count, retlen, newlen = strlen(new);

if (oldlen != newlen) {
for (count = 0, p = str; (q = strstr(p, old)) != NULL; p = q + oldlen)
count++;
/* this is undefined if p - str > PTRDIFF_MAX */
retlen = p - str + strlen(p) + count * (newlen - oldlen);
} else
retlen = strlen(str);

if ((ret = malloc(retlen + 1)) == NULL)
return NULL;

for (r = ret, p = str; (q = strstr(p, old)) != NULL; p = q + oldlen) {
      /* this is undefined if q - p > PTRDIFF_MAX */
      ptrdiff_t l = q - p;
      memcpy(r, p, l);
      r += l;
      memcpy(r, new, newlen);
      r += newlen;
    }
      strcpy(r, p);

    return ret;
}

// ========================================================================


char inputpuffer[256];

size_t hossz=0;

int i;

char *elso;
char *masodik;
char ideiglenes[MAXCHAR];

int main(int argc,char **argv) {

while(fgets(inputpuffer,100,stdin)) { ; // beolvassuk a standard inputról jövő
stringet
inputpuffer[255]=0;
```

```
strcpy(ideiglenes,inputpuffer);
elso=ideiglenes;

// *********************************
for(i=0;i<MAXCSERE;i++) {
masodik=replace_str(elso,csere[i].regi,csere[i].uj);
strcpy(ideiglenes,masodik);
free(masodik);
elso=ideiglenes;
};
// *********************************

// Most az elso-ben illetve a ideiglenesben van a jo szoveg

printf("%s",elso);
} // while vége
//free(inputpuffer);

exit(EXIT_SUCCESS);
}
```

E fenti program fogja majd nekünk úgymond „kiszínezni az outputot”. Azaz, rengetegszer fogunk fordítani más programokat, s ekkor nagyon hasznos, ha a make kimenetében vörösen világítanak mondjuk az „error” szavak... Rögtön tudjuk, hol van a baki, hova kell nézni!

Ezt tehát mentsük el a /_/P/KISS/0.0/sources könyvtárba, majd fordítsuk le:

```
gcc colored.c -o colored
```

Az elkészült **colored** nevű binárist pedig mozgassuk át a /_/P/KISS/0.0/bin könyvtárba!

Létre kell hoznunk egy **kolormake** nevű szkriptet is, aminek ez a tartalma:

```
#!/bin/bash
#

make "$@"  2>&1 | colored

exit ${PIPESTATUS[0]}
```

Ezt természetesen rögtön a /_/P/KISS/0.0/bin könyvtárba mentsük el.

Abban az esetben, ha a majd elkészült rendszerünkben meg akarunk szabadulni valami már nem szükséges programtól, ugyebár csak letöröljük mindenestől az ő alkönyvtárát, ami a /_/P fában található. Igen ám, de mi lesz a /_/S/L fában található symlinkekkel, amik ekkor már nem létező állományokra mutatnak, s így töröttek lesznek?! Azokat is törölni kell! Erre is kell nekünk egy szkript. Ennek az lesz a neve hogy **BROKEN**, és így néz ki:

```
#!/bin/bash

if [[ $1 == "" ]] ; then
echo "Használat:"
echo "BROKEN link vagy könyvtár"
```

```
echo "A program kilistázza a megadott symlinke(ke)t, AKKOR és CSAK akkor, ha a
link törött."
echo "vagy:"
echo "BROKEN link vagy könyvtár --delete"
echo "Az utóbbi esetben törli is a törött linkeket"

exit
fi

if [[ $2 == "--delete" ]] ; then

find -L $1 -type l -print -delete

else
echo "Broken links:"

find -L $1 -type l

fi
```

Eddig a szkript. Ezt is a /_/P/KISS/0.0/bin könyvtárba kell másoljuk természetesen. És úgy használjuk, hogy miután letöröltünk valamit, kiadjuk e parancsot:

```
BROKEN /_/S/L --delete
```

Ezután meg kell alkotnunk egy **sztrip** nevű szkriptet. Ez csak annyit csinál, hogy ha megadunk neki egy könyvtárat, akkor az abban meg az annak alkönyvtáraiban található összes fájlt „sztrippeli”, már ha tudja, azaz eltávolítja belőle a debug infókat. Minek az nekünk, mi úgyse debuggoljuk majd mások programját (még azt se valószínű, amit mi magunk írunk...), ellenben nem is ritkaság az, hogy a debuggolást segítő mindenfélék eltávolítása után a program mérete akár HARMADÁRA IS csökken! Nem harmadá**VAL**, hanem harmadá**RA**!

A szkriptet szintén a /_/P/KISS/0.0/bin könyvtárba mentsük, s a tartalma:

```
#!/bin/bash

KONYVTAR=$1

if [[ $KONYVTAR == "" ]] ; then
echo "Nincs megadva könyvtárnév!"
exit
fi

if [ ! -d $KONYVTAR ]; then
echo "A megadott könyvtár nem létezik!"
echo "==> $KONYVTAR"
exit
fi

echo "A könyvtár összmérete a strippelés előtt:"
du $KONYVTAR
ELOTT=`du -s $KONYVTAR  | cut -f1`

find $KONYVTAR -type f  -exec strip 2>/dev/null --strip-debug '{}' ';'

echo "A könyvtár összmérete a strippelés után:"
du $KONYVTAR
UTAN=`du -s $KONYVTAR  | cut -f1`
```

```
KULONBSEG=`expr $ELOTT - $UTAN`
HANYAD=`echo $UTAN/$ELOTT | bc -l`

echo "A kulonbseg $KULONBSEG ;"
echo "Sztrippelés után ami maradt az az eredeti $HANYAD része."
exit
```

Amint látható, e szkript a hányadszámításhoz a **bc** nevű parancssori kalkulátor-programot használja. Na most ez eleinte nem áll rendelkezésünkre majd az építendő rendszerünkben, emiatt a hányadot jelző szám helyére semmit se fog kiírni. Egészen addig nem, míg nincs a bc telepítve. De ez ne aggasszon minket, attól a sztrippelés még rendben végrehajtódik majd.

Most már nagyon ideje megalkotnunk azt a szkriptet is, ami a tulajdonképpeni belinkelést végzi majd. Nos, ez nem is egyetlen szkript lesz, hanem három. Mindhármat szintén a /_/P/KISS/0.0/bin könyvtárba mentsük el. Az első, az amit mindig közvetlenül meghívunk majd, nagyon rövid lesz, és a **sy** névre hallgat. A tartalma:

```
#!/bin/bash
#

sym "$@"  2>&1 | colored

exit ${PIPESTATUS[0]}
```

Mint látható, ez nem csinál mást, csak meghív egy másik, egy **sym** nevű szkriptet, s annak kimenetét továbbküldi a korábban lefordított „colored” nevű programnak, ami majd azt a megfelelőképp kiszínezi. A lényeg tehát a sym nevű szkript. Elárulom azonban, a sym nevű szkript meg egy megint másik, egy LINK nevű programot hív majd meg nem is egyszer, ezért előbb azt mutatom be. E program nagyon buta, csak annyit tud, hogy ha megadunk neki 2 könyvtárat, akkor az egyikből belinkel mindent a másikba. E LINK nevű szkript tartalma:

```
#!/bin/bash

LN="-sv"

if [[ $1 == "" ]] ; then
echo "Használat:"
echo "LINK ide ezt"
exit
fi

if [[ $2 == "" ]] ; then
echo "Használat:"
echo "LINK ide ezt"
exit
fi

if [ ! -d "$1" ]; then
echo "ERROR: This directory not exist: $1"
exit
fi
```

```
if [ ! -d "$2" ]; then
echo "ERROR: This directory not exist: $2"
exit
fi

if [[ $3 != "" ]] ; then
PARAMETER=$3
if [[ $3 == "-F" ]] ; then
LN="-sfv"
else
echo "Invalid 3th parameter!"
exit
fi
fi

# ----------------------------------------
BIFS="$IFS"
#adjuk meg, hogy csak az enter legyen mezoelvalaszto a for ciklus szamara
IFS="
"
# ----------------------------------------

for KONYVTAR in `find $2 -maxdepth 1 -type d`
do
NEVE=$(basename $KONYVTAR)
IDELINKEL=$1/$NEVE

if [[ $KONYVTAR != $2 ]]; then
    if [ ! -d "$IDELINKEL" ]; then
    mkdir -v $IDELINKEL
    fi
fi
done
# for KONYVTAR vége

for REGULARFILE in `find $2 -maxdepth 1 -type f`
do
NEVE=$(basename $REGULARFILE)
IDELINKEL=$1/$NEVE
ln $LN $REGULARFILE $IDELINKEL
done
# for REGULARFILE vége

for symlink in `find $2 -maxdepth 1 -type l`
do
NEVE=$(basename $symlink)
IDELINKEL=$1/$NEVE
ln $LN $symlink $IDELINKEL
done
# for REGULARFILE vége

# ----------------- REKURZIÓ! -----------------------------

for KONYVTAR in `find $2 -maxdepth 1 -type d`
do
NEVE=$(basename $KONYVTAR)
IDELINKEL=$1/$NEVE
      if [[ $KONYVTAR != $2 ]]; then

# Meghívja rekurzívan önmagát !!!!!!!!!!!!!!!!!!!!!!
LINK $IDELINKEL $KONYVTAR $3
```

```
# ---------------------------- !!!!!!!!!!!!!!!!!!!!
      fi
done
# for KONYVTAR vége
# --------------- REKURZIÓ VÉGE --------------------


#utana allitsuk vissza az IFS-t mert meg meglep minket
IFS="$BIFS"
```

Fontos alaposan kirészleteznünk a fenti LINK nevű szkript működését, mert ha nem értjük, nem vagyunk tisztában rendszerünk, disztrónk alapjaival. LÉNYEGÉBEN EZ UGYANIS AZ EGÉSZNEK A MAGVA. Márpedig honlapom jelszava is — az, hogy „**Az ENYÉM, mert ÉRTEK HOZZÁ!**” — azt jelenti, hogy csak azt érezhetjük a magunkénak, amihez értünk, aminek tisztában vagyunk a működésével. Persze ritka az olyasmi, amit tényleg teljesen átlátunk, abszolút minden részletében, de ez mindegy, a lényeg ugyanis az, hogy MINÉL JOBBAN értjük a működését, minél inkább a legalapabb szinten értjük, ANNÁL JOBBAN érezhetjük a magunkénak, annál magabiztosabban tudjuk használni, testre/személyre szabni, beállítani, stb. Akkor tehát mi uralkodunk rajta, s nem ő mirajtunk! Vagyis, bár abszolút mindent soha nem érthetünk egyetlen disztróban sem (ahhoz például az kéne, hogy újraírjuk magát az egész kernelt...), de törekednünk kell ezt az eszményi, ideális állapotot minél jobban megközelíteni! Ezt mondták régen úgy, hogy „Jó pap holtig tanul”. Vagy hogy egy nem ennyire régi mondást idézzek: Borisz Jelcin mondta egyszer valakinek, s e mondása elterjedt az egész világban:

—„Nem ihatsz meg te egymagad minden jó piát, és nem vagy képes megdugni minden jó nőt. DE TÖREKEDNED KELL RÁ!”

Na tehát a fenti szkript végzi a tényleges linkelést, bár normális esetben mi ezt nemigen fogjuk meghívni, mert akkor már nagy gáz van ha őt kell hívogatnunk közvetlenül. 2 paramétert vár, két könyvtárnevet. Először lecsekkolja, e könyvtárak egyáltalán léteznek-e. Ezután a linkelendő könyvtárat végigjárja a find paranccsal, de csak 1 mélységben, és ha a talált állomány könyvtár, ellenőrzi, van-e ilyen nevű könyvtár abban a könyvtárban, ahova linkelnie kell. Ha nincs, létrehozza. FONTOS MEGÉRTENI, hogy tehát e szkript NEM SYMLINKEL KÖNYVTÁRAKAT, hanem a könyvtárakat ténylegesen, fizikailag létrehozza abban a könyvtárban is, ahova symlinkelnie kell!

Ennek mély nagyon az értelme. Arról van szó, hogy sajnos vannak olyan galád programok, amik úgy lettek megírva, hogy a maguk cókmókjait – főleg olyasmiket, amik a hagyományos linuxos könyvtárrendszerben a „share” könyvtárba kerülnének – azokat oda is akarják pakolni, valami olyan nevű alkönyvtárba, amit egy másik program is használna ugyanolyan néven! Például van valami, amit be szeretne pakolni az /usr/share/locale könyvtárba. Addig nincs is baj amíg ő az első telepítendő program, s tegyük fel, lenne neki egy /_/P/progineve/verzió/share/locale könyvtára a telepítés után, amit a mi szkriptünk annak rendje s módja szerint besymlinkel. Igen ám, de jön a következő progi, s az is az /usr/share/locale könyvtárba akarna tenni valamit! Neki is létrejön egy ilyen könyvtár, s amikor ezt a mi szkriptünk besymlinkelné, akkor megtalálja a régi symlinket, ami az első program megfelelő locale könyvtárára mutat, s erre a

buta szkript oda helyezi el a symlinket, ami a másik program ezen könyvtárára mutat! Szóval ilyesmiből, könyvtárak linkelgetéséből óriási kavarodás támadhat. A helyes megoldás az, hogy könyvtárakat egyáltalán nem linkelünk, legalábbis automatikusan nem, csak speciális esetekben kézzel, ha nagyon muszáj. A „default behavior”, azaz alapértelmezett működés és viselkedés a szkriptjeink részéről, hogy a /_/S/L fában a könyvtárak szigorúan követik a programok könyvtárszerkezetét, amennyiben ha itt még nem volt megfelelő nevű könyvtár, akkor létrehozza neki a szkript, s ebbe belelinkel minden közönséges állományt.

Miután tehát a szkriptünk létrehozta az esetleg hiányzó könyvtárakat, egy újabb ciklussal végigvizsgálja a könyvtárat, s minden közönséges fájlt (és az esetleges symlinkeket is) belinkel oda, ahova kell.

Ezután egy harmadik ciklus jön, ami megint a könyvtárakat járja végig, s ha könyvtárra lel, akkor MEGHÍVJA ÖNMAGÁT REKURZÍVAN arra az alkönyvtárra, azaz azzal ugyanezt játssza el.

Természetesen efféle brutális aljasságot, mint önmagunk rekurzív meghívása, nagyon gondosan kell megtervezni és kiemelni, ezért is van ilyen feltűnő módon jelölve az a rész a szkriptben.

És most jön akkor az a **sym** nevű szkript, ami ezt a **LINK** nevű szkriptet hívja, s amelyet a **sy** szkript hív:

```
#!/bin/bash

LN="-sv"
FORCED=""

if [[ $1 == "" ]] ; then
echo "Használat:"
echo "sym programnév verziószám"
exit
fi

if [[ $2 == "" ]] ; then
echo "Használat:"
echo "sym programnév verziószám"
exit
fi

ELSOPARAMETER=$1

ELSOHAROM=${ELSOPARAMETER:0:3}

ELSOKARAKTER=${ELSOPARAMETER:0:1}
if [[ $ELSOKARAKTER == "-" ]] ; then

MASODIKKARAKTER=${ELSOPARAMETER:1:1}

if [[ $MASODIKKARAKTER == "F" ]] ; then
# set to forced overwriting
LN="-sfv"
FORCED="-F"
shift
```

```
fi
fi

KURRENT="/_"
PROGRAMS="$KURRENT/P"

PGM="$PROGRAMS/$1/$2"
CURRENTPGM="$PROGRAMS/$1/-"

if [ ! -d "$PGM" ]; then
echo "ERROR: This directory not exist: $PGM"
echo "The available installed versions of this program:"
ls -1 $PROGRAMS/$1
exit
fi

BIFS="$IFS"
#adjuk meg, hogy csak az enter legyen mezoelvalaszto a for ciklus szamara
IFS="
"


# ===================================== stripping

if [[ $3 != "--nostrip" ]] ; then
echo "Stripping executables and libraries..."
SZTRIP=`sztrip $PGM`
echo
fi

echo "Creating Current link (Removing the old Current link, if exist):"
rm $PROGRAMS/$1/- 2>/dev/null
ln -s $2 $CURRENTPGM
ls -l $CURRENTPGM
echo

# ================================= share moving
if [ -d "$PGM/share" ]; then
echo "Renaming the \"share\" directory to \"Shared\""
mv $PGM/share $PGM/Shared
echo "Creating $PGM/share symlink to the \"share\" directory to $KURRENT/S/L/T"
ln -s $KURRENT/S/L/T $PGM/share
fi

# ================================= Localedatas

# Locales

if [ -d "$PGM/Shared/locale" ]; then
echo "Moving locale directory:"
echo "$PGM/Shared/locale ===> $PGM/locale"
echo
mv $PGM/Shared/locale $PGM
fi

if [ -d "$PGM/locale" ]; then
echo "Symlinking localedatas:"
echo
LINK $KURRENT/S/L/T/locale $CURRENTPGM/locale $FORCED
echo
fi
```

```
# ==================================  Executables

if [[ $ELSOHAROM == "X--" ]] ; then

    if [ -d "$PGM/bin" ]; then
    mv -v $PGM/bin $PGM/xbin
    fi

    if [ -d "$PGM/sbin" ]; then
    mv -v $PGM/sbin $PGM/xsbin
    fi

    if [ -d "$PGM/lib" ]; then
    mv -v $PGM/lib $PGM/xlib
    fi

    if [ -d "$PGM/include" ]; then
    mv -v $PGM/include $PGM/xinclude
    fi

    if [ -d "$PGM/xlib/pkgconfig" ]; then
    echo "Moving xlib/pkgconfig directory:"
    echo "$PGM/xlib/pkgconfig ===> $PGM/pkgconfig"
    echo
    mv -v $PGM/xlib/pkgconfig $PGM
    echo
    fi


fi

# ================================= X

# Executables

if [ -d "$PGM/bin/x86_64-linux" ]; then
echo "Symlinking executables (bin/x86_64-linux):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/bin/x86_64-linux $FORCED
echo

fi
if [ -d "$PGM/bin" ]; then
echo "Symlinking executables (bin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/bin $FORCED
echo
fi

if [ -d "$PGM/sbin" ]; then
echo "Symlinking executables (sbin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/sbin $FORCED
echo
fi

if [ -d "$PGM/usr/bin" ]; then
echo "Symlinking executables (usr/bin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/usr/bin $FORCED
echo
fi
```

```
if [ -d "$PGM/usr/sbin" ]; then
echo "Symlinking executables (usr/sbin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/usr/sbin $FORCED
echo
fi

if [ -d "$PGM/usr/local/bin" ]; then
echo "Symlinking executables (usr/local/bin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/usr/local/bin $FORCED
echo
fi

if [ -d "$PGM/usr/local/sbin" ]; then
echo "Symlinking executables (usr/local/sbin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/usr/local/sbin $FORCED
echo
fi

if [ -d "$PGM/xbin" ]; then
echo "Symlinking X executables (xbin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/xbin $FORCED
echo
fi

if [ -d "$PGM/xsbin" ]; then
echo "Symlinking X executables (xsbin):"
echo
LINK $KURRENT/S/L/E $CURRENTPGM/xsbin $FORCED
echo
fi

# ======================================== Pkgconfig
# Pkgconfig

if [ -d "$PGM/lib/pkgconfig" ]; then
echo "Moving lib/pkgconfig directory:"
echo "$PGM/lib/pkgconfig ===> $PGM/pkgconfig"
echo
mv $PGM/lib/pkgconfig $PGM
echo
fi

if [ -d "$CURRENTPGM/pkgconfig" ]; then
echo "Symlinking pkgconfig files:"
echo
LINK $KURRENT/S/L/C $CURRENTPGM/pkgconfig $FORCED
echo
fi

if [ -d "$PGM/Shared/pkgconfig" ]; then
echo "Moving Shared/pkgconfig directory:"
echo "$PGM/Shared/pkgconfig ===> $PGM/pkgconfig-shared"
echo
mv $PGM/Shared/pkgconfig $PGM/pkgconfig-shared
echo
fi
```

```
if [ -d "$CURRENTPGM/pkgconfig-shared" ]; then
echo "Symlinking pkgconfig-shared files:"
echo
LINK $KURRENT/S/L/C $CURRENTPGM/pkgconfig-shared $FORCED
echo
fi

if [ -d "$CURRENTPGM/usr/lib/pkgconfig" ]; then
echo "Symlinking usr/lib/pkgconfig files:"
echo
LINK $KURRENT/S/L/C $CURRENTPGM/usr/lib/pkgconfig $FORCED
echo
fi

if [ -d "$CURRENTPGM/usr/share/pkgconfig" ]; then
echo "Symlinking usr/share/pkgconfig files:"
echo
LINK $KURRENT/S/L/C $CURRENTPGM/usr/share/pkgconfig $FORCED
echo
fi

if [ -d "$CURRENTPGM/usr/local/lib/pkgconfig" ]; then
echo "Symlinking usr/local/lib/pkgconfig files:"
echo
LINK $KURRENT/S/L/C $CURRENTPGM/usr/local/lib/pkgconfig $FORCED
echo
fi

# ======================================== Libraries

# Libraries

if [ -d "$PGM/lib" ]; then
echo "Symlinking libraries (lib):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/lib $FORCED
echo
    if [ "$(ls -A "$PGM/lib")" ]; then
    echo "The \"lib\" directory not empty, I keep it."
    else
    echo "The \"lib\" directory is empty, I remove it."
    rm -rf $PGM/lib
    fi
    echo
fi

if [ -d "$PGM/usr/lib" ]; then
echo "Symlinking libraries (usr/lib):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/usr/lib $FORCED
echo
fi

if [ -d "$PGM/usr/local/lib" ]; then
echo "Symlinking libraries (usr/local/lib):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/usr/local/lib $FORCED
echo
fi

if [ -d "$PGM/xlib" ]; then
echo "Symlinking libraries (xlib):"
```

```
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/xlib $FORCED
echo
fi


# Lib64

if [ -d "$PGM/lib64" ]; then
echo "Symlinking libraries (lib64):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/lib64 $FORCED
echo
fi

if [ -d "$PGM/usr/lib64" ]; then
echo "Symlinking libraries (usr/lib64):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/usr/lib64 $FORCED
echo
fi

if [ -d "$PGM/usr/local/lib64" ]; then
echo "Symlinking libraries (usr/local/lib64):"
echo
LINK $KURRENT/S/L/Y $CURRENTPGM/usr/local/lib64 $FORCED
echo
fi

# ================================== Libexec
# Libexec

if [ -d "$PGM/libexec" ]; then
echo "Symlinking libexec:"
echo
LINK $KURRENT/S/L/X $CURRENTPGM/libexec $FORCED
echo
fi

if [ -d "$PGM/usr/libexec" ]; then
echo "Symlinking usr/libexec:"
echo
LINK $KURRENT/S/L/X $CURRENTPGM/usr/libexec $FORCED
echo
fi

if [ -d "$PGM/usr/lib/libexec" ]; then
echo "Symlinking usr/lib/libexec:"
echo
LINK $KURRENT/S/L/X $CURRENTPGM/usr/lib/libexec $FORCED
echo
fi

# ====================================== Manuals
# Manuals

if [ -d "$PGM/Shared/man" ]; then
echo "Moving manpages:"
echo "$PGM/Shared/man ===> $PGM/man"
echo
mv $PGM/Shared/man $PGM
echo
```

```
fi

if [ -d "$PGM/man" ]; then
echo "Symlinking manuals:"
echo
LINK $KURRENT/S/L/D/M $CURRENTPGM/man $FORCED
echo
fi

# ======================================== Info
# Infos

if [ -d "$PGM/Shared/info" ]; then
echo "Moving infos:"
echo "$PGM/Shared/info ===> $PGM/info"
echo
mv $PGM/Shared/info $PGM
echo
fi

if [ -d "$PGM/info" ]; then
LINK $KURRENT/S/L/D/I $CURRENTPGM/info $FORCED
echo
fi
# ======================================== doc
# docs

if [ -d "$PGM/Shared/doc" ]; then
echo "Moving docs:"
echo "$PGM/Shared/doc ===> $PGM/doc"
echo
mv $PGM/Shared/doc $PGM
echo
fi

if [ -d "$PGM/doc" ]; then
LINK $KURRENT/S/L/D/doc $CURRENTPGM/doc $FORCED
echo
fi

# ================================== Shared
# Shared

if [ -d "$PGM/Shared" ]; then
echo "Symlinking shared files and dirs:"
echo
LINK $KURRENT/S/L/T $CURRENTPGM/Shared $FORCED
echo

    if [ "$(ls -A "$PGM/Shared")" ]; then
    echo "The \"Shared\" directory not empty, I keep it."
    else
    echo "The \"Shared\" directory is empty, I remove it."
    rm -rf $PGM/Shared
    fi
    echo
fi

# ======================================== Headers
# Headers

if [ -d "$PGM/include" ]; then
```

```
echo "Symlinking headers (include):"
echo
LINK $KURRENT/S/L/H $CURRENTPGM/include $FORCED
echo
fi

if [ -d "$PGM/xinclude" ]; then
echo "Symlinking headers (xinclude):"
echo
LINK $KURRENT/S/L/H $CURRENTPGM/xinclude $FORCED
echo
fi

if [ -d "$PGM/usr/include" ]; then
echo "Symlinking headers (usr/include):"
echo
LINK $KURRENT/S/L/H $CURRENTPGM/usr/include $FORCED
echo
fi

# ========================================== etc
# etc

if [ -d "$PGM/etc" ]; then
echo "Symlinking setting:"
echo
LINK $KURRENT/S/O $CURRENTPGM/etc $FORCED
echo
fi

# ========================================== X

if [[ $ELSOHAROM == "X--" ]] ; then

    if [ -d "$PGM/xbin" ]; then
    ln -sv $KURRENT/S/L/E $PGM/bin
    fi

    if [ -d "$PGM/xsbin" ]; then
    ln -sv $KURRENT/S/L/E $PGM/sbin
    fi

    if [ -d "$PGM/xlib" ]; then
    ln -sv $KURRENT/S/L/Y $PGM/lib
    fi

    if [ -d "$PGM/xinclude" ]; then
    ln -sv $KURRENT/S/L/H $PGM/include
    fi

fi

# ========================================== X

#utana allitsuk vissza az IFS-t mert még meglep minket
IFS="$BIFS"

echo
MARADT=`echo $SZTRIP | cut --delimiter=";" -f2`
echo $MARADT
echo
```

Eddig a **sym** szkript. Bár tartok tőle, hogy láttán a kezdő linuxosok elszörnyednek, elárulom, hogy ez lehet ugyan kissé hosszúnak tekinthető, de programozástechnikailag nem egy nagy durranás. Mást se csinál, csak ellenőrzi a paramétereit, néhány könyvtárat átmozgat, aztán meg mindenféle variációkban meghívogatja a LINK nevű szkriptet.

A szkript működését nem elemzem részletesen, minden benne van a forrásában, aki akarja, kivesézheti. Annyit jegyzek csak meg, hogy a feltelepített program share könyvtárát átnevezi úgy, hogy Shared, s ebből átmozgat bizonyos alkönyvtárakat egy szinttel feljebb. Amennyiben emiatt a Shared könyvtár üressé válik, azt törli is. Ha volt share könyvtár, amit átnevezett Sharedre, akkor létrehoz a program könyvtárában egy symlinket, ami a /_/S/L/T könyvtárra mutat, share néven.

Ennek az az oka, hogy egyes programok olyan dolgokat keresnek a share könyvtárban, amiket nem is ők pakoltak oda, ellenben feltételezik, hogy e share könyvtár onnan nyílik, ahova ők maguk lettek telepítve...

Sajnos, hasonló ocsmányságok fordulnak elő a drága jó Xorg csomagot alkotó egyes programok esetében is, csak ott ezt az őt alkotó rész-csomagok nem elsősorban a share könyvtárral játsszák el, hanem a binárisokkal, libekkel és header fájlokkal. Nem totóztam ki, konkrétan melyek vannak ilyen szarul megírva, ehelyett, amint rájöttem, hogy némelyiküknél ilyesmi előfordulhat, úgy döntöttem, az Xorgot alkotó programok egyszerűen úgy lesznek elnevezve, hogy mindegyiknek a neve úgy kezdődik majd, hogy „X--”, s ha a sym szkript azt látja, hogy egy effélét kell belinkelnie, akkor a bin, lib stb. könyvtárakat átnevezi úgy, hogy xbin, xlib stb., ezeket symlinkeli be, s helyettük létrehoz egy megfelelő nevű symlinket abba a könyvtárba, ami a /_/S/L fa megfelelő könyvtárára mutat.

A share könyvtárból egyes alkönyvtárak egy szinttel feljebb mozgatása tulajdonképpen nem volna muszáj, de így áttekinthetőbbnek tartom a struktúrát, mintha minden egy helyre lenne behányva.

Ehelyütt kell megküzdenem azzal az ellenérvvel is, ami azt állítja, hülyeség egybetűs könyvtárneveket használnom, meg mi ez az idiótaság, hogy van a gyökérben nekem egy _ nevű symlinkem! Sőt, elnézve a fenti szkript működését, pillanatok alatt kiderül, hogy az létrehoz a /_/P/progineve könyvtárba egy „**-**”, azaz „mínuszjel” nevű symlinket is, ez mutat az adott program aktuális verziójára, ez felel meg lényegében a GoboLinux „Current” nevű linkjének! Miért nem használok én is normális neveket?!

Több jó oka is van annak, hogy ezen, a rendszeremben létfontosságú könyvtárnevek egybetűsek.

1. Mindenekelőtt: így rengeteg gépeléstől megmenekülök. Ez igazán nem utolsó szempont.

2. Akadnak olyan utasítások, illetve parancsok – például a „cut” nevű – melyek csak 1 karaktert fogadnak el mezőelválasztóként. Márpedig, ha én szkripteket irogatok a disztróm bizonyos funkcióit megvalósítandó, akkor erősen valószínű,

hogy azon szkriptekben nekem szükséges lesz egyes fájlok útvonalainak különböző részeit meghatározni, s akkor nagyon jó, ha a fontosabb rendszerkönyvtárak mindössze egybetűsek, mert így ezeket a programokat is felhasználhatom az útvonal részekre vagdalására, szeparátorként a könyvtárnevem egyetlen betűjét (karakterét) megadva.

3. Ha valahol fórumokon vagy dokumentációkban szóba kerül a disztróm, s meg kell említsem benne e könyvtárakat, messze könnyebb azt írnom hogy az L könyvtár vagy a S könyvtár, mint kiírni mindig hogy System vagy Links.

4. Így szebbnek tartom a rendszeremet, azaz esztétikai okai is vannak.

5. Így a rendszerem egyénibb, különlegesebb. Gondoljunk csak bele: Minden útvonal úgy kezdődik benne, hogy */_/*
Ez pedig szinte olyan, mint valami titkosírás... Növeli a komolyság látszatát. Márpedig pszichológusok bebizonyították, hogy ha valaki azt látja, hogy egy másik ember olyasmit művel, amit ő, aki nézi, nem ért, de bonyolultnak látszik, akkor hajlamos tiszteletet érezni iránta. Na most nem ez a fő érv nálam az „egybetűsség” mellett, de valahol az érvek vége felé ez is ott szerepel, nem tagadom. (Mindenki hiú, csak sokan pofátlanul letagadják. Én legalább be merem vallani...) Az, hogy a gyökérkönyvtárban levő link pedig ne csak egykarakteres legyen, de épp az aláhúzásjel, azért van, mert ezen karakterben a vonás legalul van, s ez mintegy jelképe annak, hogy e link mindennek az ALAPJA, mert minden útvonal innen indul ki. A telepített programoknál viszont azért nem szintén aláhúzásjel mutat az aktuális verzióra, hanem egy mínuszjel, hogy megkülönböztethető legyen a mindenféle stringfeldolgozó funkciók számára épp az útvonalban szereplő aláhúzásjeltől, ami legelöl szerepel. Ez is megkönnyíti a szkriptek megírását.

A rendszerem fontosabb könyvtárainak neve azért kapta épp azt a karaktert névnek, mert a következő nevek rövidítései:

S : **S**ystem
L : **L**inks
P : **P**rograms
O : **O**ptions
V : **V**ariables
E : **E**xecutables
H : **H**eaders
Y : Librar**Y**
X : Libe**X**ec
C : pkg**C**onfig
D : **D**ocuments
I : **I**nfo
M : **M**anuals
T : **T**rash (Ez utóbbi némi gúny a részemről. Ez a share könyvtár állományait tartalmazza javarészt, ezt nevezem úgy, hogy „trash”, azaz „szemét”, mert ez a „minden egyéb”. Ugyebár nekem gyakran a fejemhez vágták, hogy hülye

vagyok, mert nekem nem jó a régi unixos könyvtárszerkezet, pedig az ANNYIRA logikus, hogy minden funkciók szerint van csoportosítva! Na ja, persze. Nézzünk csak rá a share könyvtárra, akkora kupleráj, hogy csak na, hol van abban funkciók szerinti csoportosítás, hogy mindent behányunk oda, mint egy szemeteskukába?! Sajnos, ezt a rendetlenséget én is kénytelen vagyok valamennyire emulálni a programok számára, amik ehhez szoktak hozzá, de a fájlok FIZIKAI elrendezése nálam legalább egy kétségkívül jól definiálható logikai rendszert követ: Az van egy könyvtárban, ami egyszerre lett telepítve, egy csomagban!)

Fontos lehet, hogy tudjuk, egy program épp hol található meg. Ennek érdekében persze átnyálazhatnánk a find paranccsal a teljes /_/P fát, de ez nem túl hatékony. Ha már a program telepítve van a rendszerünkben, egyszerűbb csak a /_/S/L fát végignézni, ebből is csak a megfelelő típust tartalmazó alkönyvtárat! Minthogy efféle keresgélést általában a végrehajtható állományokkal és a librarykkal kell majd folytatnunk, írjunk e funkciókra is egy-egy szkriptet, amik már rögvest megfelelőképp kiszínezve nyújtják nekünk a végeredményt! Íme a „hol” nevű szkript:

```
#!/bin/bash

BIFS="$IFS"
#adjuk meg, hogy csak az enter legyen mezoelvalaszto a for ciklus szamara
IFS="
"

for i in `ls -l /_/S/L/E/$1  2>&1`
do
KI=`echo $i | sed -e 's/\/_\/P\//\/_\/P\/\\\\e\[97m/g' \
-e 's/\/-/\\\\e\[39m\/-/g' \
-e 's/No such file or directory/\\\\e\[91mNo such file or directory\\\\e\
[39m/g'`

echo -e "$KI"
done

#utana allitsuk vissza az IFS-t mert meg meglep minket...
IFS="$BIFS"
```

A fenti szkript használatára két példa:

```
root@Csiszilla /Releases/2014/P/KISS/0.0/bin=>hol bash
lrwxrwxrwx 1 root root 20 dec   18 05.06 /_/S/L/E/bash -> /_/P/Bash/-/bin/bash
root@Csiszilla /Releases/2014/P/KISS/0.0/bin=>hol ba*
lrwxrwxrwx 1 root root 31 dec   18 05.06 /_/S/L/E/badblocks -> /_/P/E2fsprogs/-/sbin/badblocks
lrwxrwxrwx 1 root root 27 dec   18 05.06 /_/S/L/E/base64 -> /_/P/CoreUtils/-/bin/base64
lrwxrwxrwx 1 root root 29 dec   18 05.06 /_/S/L/E/basename -> /_/P/CoreUtils/-/bin/basename
lrwxrwxrwx 1 root root 20 dec   18 05.06 /_/S/L/E/bash -> /_/P/Bash/-/bin/bash
lrwxrwxrwx 1 root root 23 dec   18 05.06 /_/S/L/E/bashbug -> /_/P/Bash/-/bin/bashbug
lrwxrwxrwx 1 root root 30 dec   26 03.52 /_/S/L/E/battery -> /_/P/Szkriptjeim/-/bin/battery
```

Mint látható, nemcsak teljes programnévre lehet rákeresni. A fenti szkript a végrehajtható állományok közt keres. Ennek párja a **holl** szkript, ami a libraryk közt kotorászik:

```
#!/bin/bash

BIFS="$IFS"
#adjuk meg, hogy csak az enter legyen mezoelvalaszto a for ciklus szamara
```

```
IFS="
"

for i in `ls -l /_/S/L/Y/$1  2>&1`
do
KI=`echo $i | sed -e 's/\/_\/P\//\/_\/P\/\\\\e\[97m/g' \
-e 's/\/-/\\\\e\[39m\/-/g' \
-e 's/No such file or directory/\\\\e\[91mNo such file or directory\\\\e\
[39m/g'`

echo -e "$KI"
done

#utana allitsuk vissza az IFS-t mert meg meglep minket...
IFS="$BIFS"
```

Használata ugyanaz, mint a **hol** szkripté, csak mint mondtam, ez librarykat keres. Ezeket is mentsük el a /_/P/KISS/0.0/bin könyvtárba.

Fontos az is, hogy tudjuk, telepítve van-e egyáltalán valamely program a rendszerünkben! Ehhez természetesen a /_/P fát kell átnézni. Az erre a célra szolgáló szkript neve: „van”. Így néz ki:

```
#!/bin/bash

if [[ $1 == "-h" ]] ; then
echo "E szkript kilistázza, hogy rendszerünkbe mely programcsomagok mely verziói
vannak felinstallálva."
echo "Használata:"
echo "van        (nincs megadva paraméter)"
echo "Ekkor a rendszerbe telepített összes programcsomag összes verzióját
kilistázza."
echo "van névstring"
echo "Ekkor csak azon programcsomagokat listázza (verziószámmal együtt) melyek
neve tartalmazza a megadott névstringet."
echo "A keresés nem kisbetű-nagybetű érzékeny. (Nem case-sensitive)."
exit
fi

van=$1
BIFS="$IFS"
#adjuk meg, hogy csak az enter legyen mezoelvalaszto a for ciklus szamara
IFS="
"

if [[ $1 != "" ]] ; then

    for i in `ls -1 /_/P | grep -i $van`
    do

        for type in `ls -1 /_/P/$i`
      do

          if [[ $type != "-" ]] ; then
            echo $i/$type
          fi
      done
    done
```

```
else

    for i in `ls -1 /_/P`
    do

        for type in `ls -1 /_/P/$i`
      do

          if [[ $type != "-" ]] ; then

            echo $i/$type

          fi

        done

    done

fi

#utana allitsuk vissza az IFS-t mert meg meglep minket...
IFS="$BIFS"

exit
```

Használata bele van írva a szkriptbe, annak elejére.

Már csak egy szkriptet kell ismertetnem, s ez az, aminek a neve az, hogy „konfig”. Ez így néz ki:

```
#!/bin/bash
#

if [[ $1 == "" ]] ; then
echo "Használat:"
echo "konfig programnév verziószám"
exit
fi

if [[ $2 == "" ]] ; then
echo "Használat:"
echo "konfig programnév verziószám"
exit
fi

neve="/_/P/$1/$2"
shift
shift
tobbi="$@"

./configure --prefix=$neve $tobbi  2>&1 | colored

exit ${PIPESTATUS[0]}
```

Ez a következő miatt kell:

Ha beleolvasol az LFS Bookba, láthatod, hogy rengetegszer kezdődik úgy egy program telepítése, hogy:

./configure --prefix=...

Na most prefixnek nálunk mindig az lesz megadva, hogy

/_/P/progineve/verziószám

tehát ezt így kéne megadnunk:

```
./configure --prefix=/_/P/progineve/verziószám
```

de a fenének van kedve ennyit gépelni...

Hála a fenti szkriptnek, ez leegyszerűsödik erre:

```
konfig progineve verziószám
```

A verziószám után szóközzel elválasztva még akárhány más paraméter is jöhet.

Sajnos ez tényleg csak akkor működik, ha a configure szkript egyáltalán létezik, és meghívható az ő saját könyvtárából, azaz nem kell a fordításhoz külön build directoryt csinálni. De így is rengeteget nyerünk.

Na tehát a fenti szkripteket mind mentsük el a /_/P/KISS/0.0/bin könyvtárba, mindegyikre adjunk futási jogot e paranccsal:

```
chmod -v 755 /_/P/KISS/0.0/bin/*
```

majd a /_/P/KISS könyvtárban állva adjuk ki e parancsot:

```
ln -s 0.0 -
```

S erre létrejön a megfelelő „mínusz” nevű symlink, ami az aktuális verziót jelzi.

Ezután kétszer is ellenőrizzük le, hogy a chroot környezetben vagyunk-e! Ha igen, adjuk ki e parancsot:

```
/_/P/KISS/0.0/bin/LINK /_/S/L/E /_/P/KISS/-/bin
```

A fenti parancs azon ritka esetek egyike, amikor a LINK szkriptet közvetlenül is meghívjuk. Ez besymlinkeli e fontos szkriptjeinket (önmagát is) a /_/S/L/E könyvtárba, hogy ezután használhassuk őket.

Látható, hogy a létrejött symlinkek útvonala nem a verziószámon halad át közvetlenül, hanem a **-** (mínusz) nevű symlinken! Ez nem azonos a régi GoboLinux módszerével, mert ott a Current nevű symlink nem volt használva e célra. Én azonban úgy gondoltam, ha úgyis ott egy symlink, akkor használjuk is! Eképp ha több verzió is telepítve van a megfelelő könyvtárba egy programból, a verziók közti váltás nagyon egyszerű: Csak az ottani „mínusz” nevű symlinket kell átírni, hogy egy újabb verzióra mutasson (vagy régebbire, ha arra akarunk visszatérni).

Most létre kell hozni egypár szükséges fájlt, már amiatt is, mert olyan randa, hogy a shellünk mindig azt írja ki, hogy „I have no name!”:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF

cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

A fenti sorok közvetlenül az LFS Bookból vannak kimásolva. Ne zavarjon, hogy ezek szemmel láthatóan az /etc könyvtárba akarnak írni, s nem a /_/S/O -ba: Ez utóbbi könyvtárba kerülnek majd e bejegyzések, mert nálunk létezik etc nevű symlink a gyökérkönyvtárban, ami épp ezen /_/S/O könyvtárra mutat. Felesleges macera lenne módosítani a fenti parancsokat tehát.

Ezután javaslom, lépj ki a shellből az

```
exit
```

paranccsal, majd

```
cd
./UKISS
./KISS
```

s így visszajutottál a chroot könyvtárba, de már azt jelzi ki neked, hogy root vagy, nem azt, hogy nincs neved.

Ki kell add ekkor e parancsokat is:

```
touch /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

Na most, e helyütt meg kell említsük, miként használhatóak a szkriptjeink. Alapesetben, a legegyszerűbb esetben tehát, egy program telepítése forrásból úgy kell végrehajtassék, hogy kiadjuk e parancsokat:

```
konfig progineve verziószám
make
make install
sy progineve verziószám
```

Amíg a sy parancs ki nincs adva, hiába van feltelepítve a program, a rendszerünk nem vesz tudomást róla, nem tudod használni. Olyan, mintha nem is lenne.

Ha már besymlinkelted a sy szkripttel, s rájössz, hogy mégsem kell a program, egyszerűen töröld le a könyvtárát a /_/P fából, majd add ki a

```
BROKEN /_/S/L --delete
```

parancsot.

Ha csak ideiglenesen akarod szüneteltetni a program használhatóságát, akkor töröld a

/_/P/progineve

könyvtárból az ottani „-” (mínusz) symlinket. Ettől még megmaradnak a /_/S/L fában a progira mutató symlinkek, de töröttek lesznek, a program nem tud működni. Olyan, mintha nem létezne a rendszeredben. Ha később vissza akarod állítani a program használhatóságát, egyszerűen hozd létre újra e „mínusz” symlinket, ami a megfelelő verzióra kell mutasson. Ez természetesen csak addig megoldható, amíg a BROKEN szkripttel nem törlöd a törött symlinkeket.

Ha azokat törölted, **nehogy** elkövess olyat, hogy a /_/P fába telepített, s egyszer már besymlinkelt programot ÚJRA besymlinkelteted a sy szkripttel**!!!! Ez iszonyú rossz ötlet!** A sy szkript létrehoz ott mindenféle symlinkeket is, amiket ekkor újra létre akarna hozni. Meg átnevez és átmozgat könyvtárakat. Teljesen összezagyválná a könyvtárrendszeredet. Ne tévesszen meg téged az se, hogy a korábban látott szkriptjeimet alaposan átnézve találhatsz bennük „forced” opciót. Jobb, ha nem használod! Az még nálam erősen és igen-igen nagyon kísérleti stádiumban van, de annyira, hogy nem kizárt hogy a későbbiekben nem is lesz benne a szkriptekben és máshogy oldom meg a kérdést. Mert e probléma megoldandó természetesen, ezt elismerem, ez egy nem implementált, de meglehetősen szükséges feature a disztrómban, csak még nem volt időm foglalkozni a kérdéssel. Meg van több ötletem is rá, s még nem tudom melyik mellett kötelezzem el magamat, melyik lesz a gyümölcsözőbb. Szerencsére, erre a funkcióra tapasztalatom szerint nem gyakran volna szükség. (De ismétlem, ettől még elismerem, hogy e kérdés megoldásra szorul).

Mindenesetre, ha egyszer a már telepített program symlinkjeit akármiért is, akármi módszerrel is de eltávolítottad a /_/S/L fából, akkor utána, ha a program neked mégis kell, az egyetlen biztonságos módszer az, ha letörlöd és újratelepíted forrásból!

Bocs. Egyelőre ez van. A későbbiekben e hiányosságot természetesen orvosolom majd. De nem megy minden egyszerre. Sorry! A rendszerem most, e sorok írása idején csak kb. nem egészen 1 hónapja létezik. Igazán nem csodálható, hogy nincs még felszerelve mindennel, „mi szem-szájnak ingere”!

6. fejezet: A minimális végleges rendszer létrehozása

Ez hosszú fejezet lesz... **:(**

Tapasztalatból állítom, amit biztos te is tapasztaltál már: egy otthoni rendszertelepítésnek sosincs vége! Akármikor rábukkanhatsz valami cuki programra, amit szeretnél feltenni. Az is egyéni ízlés kérdése, mit tekintünk „minimális” rendszernek. Általában az a vélekedés ezzel kapcsolatban, hogy ehhez nem szükséges grafikus felület. Az LFS Book, amit mi azért úgy nagyjából mégis követünk, azon a véleményen van, hogy minimális rendszer címén megelégszik gyakorlatilag csak egy működő kernellel, s a legislegalapabb programok azon készletével, amik épp csak elegendőek azon dolgokhoz, hogy újabb programokat telepíthessünk forrásból a rendszerünkbe, s természetesen ehhez szükségesek a fájlrendszer kezelését biztosító szoftverek, meg azok, amik a tömörített forrásfájlok kicsomagolását végzik.

Én is ezt tekintem e leírásban alaprendszernek, azt kivéve, hogy leírom itt azon szoftverek telepítését is, amik kellenek az MC-hez. Azért már egy MC progink csak legyen a leendő új rendszerben, ha bebootolunk oda... Mutassa már valami picikét látványos akármi, hogy MŰKÖDÜNK...

Ezen döntésem ellen nem érv, hogy ez ellentmondana a KISS elvnek, mert ugye az MC nem okvetlenül szükséges. Persze, hogy lehet nélküle élni, én se vitatom... De gondolkozzunk picit előbbre! Valamiféle grafikus környezetet azért vélhetőleg akarunk majd magunknak. Nyilván lesz igényünk fájlkezelőre is. Mi legyen a fájlkezelőnk? Logikusan olyan program, aminek nem sok függősége van. Az MC ilyen. Tehát ezt feltesszük minél előbb, hogy hamar segíthessen nekünk, aztán más fájlkezelőt nem is teszünk már fel, mert minek!

Felhívom Olvasóim figyelmét arra, hogy ellentétben az ideiglenes rendszer összeállításánál folytatott hozzáállásunkkal, innentől cseppet se követjük szentírásként az LFS előírásait! Az alábbiakban részletezem a fontosabb eltéréseket.

—Az LFS a programokat a /bin – /sbin – /lib könyvtárakba, azaz a „gyökérbe” telepíti, vagy az /usr -be. Mi mindent külön könyvtárba. Ezért, amikor ő olyasmiket ír, hogy

--prefix=/usr

akkor mi ehelyett a konfig utasítást (a korábban ismertetett szkriptünket) használjuk megfelelően felparaméterezve.

—Az LFS olykor megad paraméterként a configure szkriptnek sysconfigdir-t vagy más ilyesmit. Ezt mi többnyire kihagyjuk, nálunk nem baj, ha a

/_/P/progineve/verzió könyvtárban keletkezik egy etc nevű könyvtár, majd a sym szkriptünk besymlinkeli onnan a config fájlt a /_/S/O könyvtárba úgyis. Ugyanez van akkor is többnyire, ha mandirt ad meg a man oldalak számára.

—Az LFS írja, miként telepíthetjük a progik dokumentációját is. Én ezt a lépést is többnyire kihagyom, legfeljebb akkor nem, ha nem nagyon macerás. Többnyire sajnos az. Ekkor kihagyom, görcsöljön vele az, akinek két anyja van! Nincs is nagy jelentősége, mert egyrészt infók után az ember többnyire úgyis az Interneten át tájékozódik manapság már, másrészt így kisebb lesz a rendszerünk mérete is. Emiatt inkább csak a man oldalakkal tettem kivételt, mintegy megszokásból és hagyománytiszteletből, no meg mert azok általában nem igényeltek különösebb hackerkedést. Info oldalak akkor vannak telepítve, ha ezek telepítése nem igényelt különösebb gányolást, vagy épp külön szoftverfüggőséget, például valami extra programot a kicsomagolásukhoz vagy generálásukhoz. De még ez esetben is az van, hogy ezen info oldalaknak van valami „dir” nevű fájljuk, ami holmi tartalomjegyzék, s ez nem frissül automatikusan, legalábbis az én esetemben/rendszeremben nem. A sym szkript jelzi is, hogy ezt nem tudja besymlinkelni, mert már van ott egy symlink egy régebbi változat dir fájljára. Nos, bevallom, e problémára ballisztikus ívben tojtam. Soha nem volt szükségem info fájlokra, nem olvastam őket, leragadtam a man oldalaknál. Azt tudom, létezik valami megoldás e dir fájl tartalmának frissítésére/aktualizálására, de nem néztem utána mi az, vannak ennél milliószorta fontosabb feladataim is. Aki akarja, járjon utána maga! Mindenesetre a rendszer működőképessége szerencsére nem az info fájloktól vagy ezek tartalomjegyzékétől függ. De annyira nem, hogy halálnyugodtan letörölheted az összes info fájlt, ha neked azok nem kellenek, és spórolni akarsz a hellyel, rémségesen meg lennék lepve, ha emiatt a rendszered funkcionalitásában a továbbiakban bármi gond következne be.

Ki kell térnem a programok NEVÉRE. Jó ha tudod, erre hivatalosan az a szabály, hogy nincs szabály. Sőt nemcsak hogy szabály nincs rá, de azért nincs szabály, mert a programnak mindegy. Elvileg mondjuk a binutils programjait telepíthetnéd a /_/P/kiskutyafarka/nyamnyam123 könyvtárba is, akkor is ugyanúgy működnének. Azaz a név megválasztása, azon névé, ami nevű könyvtárba kerülnek a /_/P alatt, kizárólag a te ízléseden és óhajodon múlik, olyat érdemes választani, amit könnyen megjegyzel. A korábban említett okok miatt a kivételek az Xorg programjai, melyek nevei X-- karaktersorral kell kezdődjenek, de te még ezt is megváltoztathatod, csak akkor aszerint át kell írd a szkriptemet is.

Én azt az elvet követtem a nevek meghatározásakor, hogy:

1. Kivétel nélkül minden név NAGY kezdőbetűvel kezdődik
2. A néven belül ha valami olyan szórészlet található, ami értelmes (angol) szónak tűnik, vagy olyan szó részletének, az is nagybetűvel kezdődik
3. Ha a névrészlet szemlátomást rövidítés, betűszó, akkor minden betűje nagybetű.

De te halálnyugodtan dönthetsz másképp. Az is egy korrekt logika lenne, hogy az első betű nagy, az összes többi mindig kicsi. Ez nem befolyásolja a rendszer működését.

Egy jó tanács: A következőkben néha előfordul majd sajnos, hogy egy már meglevő szimbolikus linket „kézzel” kell módosítanunk, azaz azt csinálni, hogy a link maga mint bejegyzés megmarad, nem törlődik, nem változik a link állományneve a könyvtárban, de meg kell változtatnunk azt a fájlnevet, ahova ő a link mutat! Ez nem lesz túl gyakori, de néha sajnos elkerülhetetlen. Na most ez a legkényelmesebb az MC progiban. Bár MC jódarabig nem lesz nekünk a készülő rendszerben, de a host rendszer alatt van MC. Na most a link tárgyának megváltoztatása MC alatt annyi, hogy ráállsz a panelben a linkre, majd megnyomod a Ctrl-X, Ctrl-S billentyűkombinációt, s beírod a felbukkanó ablakban, hogy ezentúl hova mutasson. Ezután a host rendszer alól a link többnyire töröttnek fog látszani, de ez ne zavarjon téged, mert ha jól csináltál mindent, a készülő rendszeredben, már chroot környezetben is, a link nem számít majd töröttnek. Továbbá, a host rendszer alól nézve az a link majdnem biztosan akkor is töröttnek látszik már eredetileg is, ha a chroot környezetben nyilvánvalóan nem törött.

Tanácsok a felmerülő esetleges gondok megoldására:

A felmerülő hibák oroszlánrésze abból adódik, hogy a fordító nem talál valamilyen állományt. Nos, akkor a következőkkel próbálkozz:

—Ha valami libraryt nem talál, akkor add ki az

```
ldconfig -v
```

parancsot, s ismételd meg a fordítást. Ez majdnem mindig segít. Ha mégsem, nézd meg, ami libraryt keres, annak neve úgy végződik-e, hogy

.so

Ha igen, akkor keress rá a /_/S/L/Y könyvtárban, létezik-e ott egy ugyanilyen nevű library, aminek a neve vége azonban olyan, hogy **.so.x** ahol is az „x” helyén valamilyen számjegy áll, többnyire egy 1-es. Ha találsz ilyet, akkor menj el a /_/S/L/Y könyvtárba, s ha a keresett library neve mondjuk valami.so és te olyat találtál, hogy valami.so.1, akkor e parancsot add ott ki:

```
ln -s valami.so.1 valami.so
```
majd:
```
ldconfig -v
```

Ezután a /sources könyvtárban megismételheted a fordítást, jó eséllyel sikerül majd.

Ha header fájlt nem talál, akkor a find paranccsal keress rá a hiányolt header fájl nevére a /_/S/L/H könyvtárban:

```
find /_/S/L/H | grep valami.h
```

Valószínűleg megtalálja neked, csak nem közvetlenül a /_/S/L/H könyvtárban, hanem annak valamely altartalomjegyzékében. Ez esetben semmi gond, csak az ottani header fileról csinálj egy symlinket a /_/S/L/H könyvtárba, s máris folytathatod a munkát.

Essünk hát neki...

============================== 6.7. Linux-3.10.10 API Headers

```
make mrproper

make headers_check

make INSTALL_HDR_PATH=dest headers_install
find dest/include \( -name .install -o -name ..install.cmd \) -delete
cp -rv dest/include/* /usr/include
```

Feltűnt-e neked, hogy itt nincs az utasítások legvégén meghívva a **sy** szkript?! Nos amiatt nem, mert ez KIVÉTEL. Ez ugyanis a Kernel része. A kernel nem kerül a /_/P fába, hanem egészen máshova, azt „hagyományosan” telepítem. Az ritkán cserélődik úgyis, és olyankor tök kényelmes, hogy az LFS előírásait követhetem kernelfordításkor szóról-szóra.

Lesz majd egy másik kivétel is, az Udev program, amit nem a /_/P fába telepítek. Azt szeretném odatenni, de egyelőre túl bonyolultnak tűnik. Azért annak, mert annak fejlesztői „összeolvadtak” a Systemd fejlesztőivel, de nekünk nem kell Systemd csak Udev, ezért annak csomagjából azt ki kell szedni. Itt egyszerűen követtem az LFS előírásait. Valamikor nagysokára majd talán visszatérek e gondra, ha lesz több időm. Ez a 2 kivétel van tehát a könyvemben: a kernel és az Udev.

============================== 6.8. Man-pages-3.53

A Makefile elején átírni e sort:
```
prefix?=/usr
```
erre:
```
prefix?=/_/P/Man-pages/3.53
```
Ezután:
```
make install
sy Man-pages 3.53
```

============================== 6.9. Glibc-2.18

```
sed -i -e 's/static __m128i/inline &/' sysdeps/x86_64/multiarch/strstr.c
```

A glibc forráskönyvtárában a localedata/locales/hu_HU fájlban átírtam ezt a sort:
```
decimal_point              "<U002C>"
```
erre:
```
decimal_point              "<U002E>"
```
azért, hogy a tizedes-szeparátor ne az a hülye vessző, hanem a pont legyen.

```
mkdir -v ../glibc-build
cd ../glibc-build

../glibc-2.18/configure --prefix=/_/P/Glibc/2.18 --disable-profile  --enable-
kernel=2.6.32

make

make -k check 2>&1 | tee glibc-check-log
```

A fenti parancs rohadtul sokáig dolgozik majd még egy NAGYON gyors gép esetén is, és látszatra sokáig leáll!

grep Error glibc-check-log

Eredménye az én esetemben:

```
make[2]: *** [/sources/glibc-build/stdlib/test-canon.out] Error 1
make[1]: *** [stdlib/tests] Error 2
make[2]: *** [/sources/glibc-build/dlfcn/bug-atexit3.out] Error 1
make[1]: *** [dlfcn/tests] Error 2
make[2]: [/sources/glibc-build/posix/annexc.out] Error 1 (ignored)
make[2]: *** [/sources/glibc-build/posix/tst-getaddrinfo4.out] Error 1
make[1]: *** [posix/tests] Error 2
make[2]: *** [/sources/glibc-build/nptl/tst-mutex8.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-mutexpi8.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cond7.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cond8.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cond22.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cond25.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cond-except.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust1.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust2.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust5.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust6.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robust7.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi1.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi2.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi5.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi6.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-robustpi7.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-once3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-once4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-key3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-sem11.out] Error 134
make[2]: *** [/sources/glibc-build/nptl/tst-sem12.out] Error 134
make[2]: *** [/sources/glibc-build/nptl/tst-basic3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-basic4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-join1.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-join5.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-join6.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-tsd5.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel1.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel2.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel5.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel6.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel7.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel8.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel9.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel10.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel11.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel12.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel13.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel14.out] Error 1
```

```
make[2]: *** [/sources/glibc-build/nptl/tst-cancel15.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel16.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel17.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel18.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel20.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel21.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel22.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel23.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel24.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancel25.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel-self.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel-self-cancelstate.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel-self-canceltype.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancel-self-testcancel.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cleanup0.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cleanup1.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cleanup3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cleanup4.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-exit2.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-exit3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx2.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx3.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx4.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx5.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx6.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx7.out] Error 1
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx8.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx9.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx10.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx11.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx12.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx13.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx14.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx15.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx16.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx17.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx18.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx20.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cancelx21.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cleanupx0.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cleanupx1.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cleanupx3.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-cleanupx4.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-oncex3.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-oncex4.out] Error 127
make[2]: *** [/sources/glibc-build/nptl/tst-fini1.out] Error 134
make[2]: *** [/sources/glibc-build/nptl/tst-execstack.out] Error 1
make[1]: *** [nptl/tests] Error 2
make[2]: *** [/sources/glibc-build/rt/tst-mqueue8.out] Error 1
make[2]: *** [/sources/glibc-build/rt/tst-cpuclock2.out] Error 1
make[2]: *** [/sources/glibc-build/rt/tst-mqueue8x.out] Error 127
make[1]: *** [rt/tests] Error 2
make[2]: [/sources/glibc-build/conform/run-conformtest.out] Error 1 (ignored)
make[2]: *** [/sources/glibc-build/debug/tst-chk4.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-chk5.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-chk6.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-lfschk4.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-lfschk5.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-lfschk6.out] Error 127
make[2]: *** [/sources/glibc-build/debug/tst-backtrace2.out] Error 1
make[2]: *** [/sources/glibc-build/debug/tst-backtrace3.out] Error 1
make[2]: *** [/sources/glibc-build/debug/tst-backtrace4.out] Error 1
```

```
make[2]: *** [/sources/glibc-build/debug/tst-backtrace5.out] Error 1
make[1]: *** [debug/tests] Error 2
make[2]: *** [check-abi-libc] Error 1
make[1]: *** [elf/tests] Error 2
```

Ha a te eredményed a fentitől FELTŰNŐEN eltér, akkor baj van. Ezzel kapcsolatban nem tudok segíteni neked, olvasd el az LFS megfelelő fejezetét. Lényeg az, hogy néhány hiba megjelenése elkerülhetetlen, ez amiatt van mert eltérőek a hardverek, meg akadnak mindenféle kísérleti funkciók is a Glibc-ben amikre úgysincs szükséged, ezek egy része nem is dokumentált, csak a fejlesztőknek szól, stb. Szóval, ha csak pár hiba van, pláne azok java része is egybeesik azzal, amit az én fenti listámon látsz, nem kell aggódni. De azért mondom, jobb, ha átszőrözöd ezzel kapcsolatban az LFS megfelelő fejezetét.

```
mkdir -p /_/P/Glibc/2.18/etc
touch /_/P/Glibc/2.18/etc/ld.so.conf

make install

cp -v ../glibc-2.18/sunrpc/rpc/*.h /_/P/Glibc/2.18/include/rpc

cp -v ../glibc-2.18/nis/rpcsvc/*.h /_/P/Glibc/2.18/include/rpcsvc

cp -v ../glibc-2.18/sunrpc/rpcsvc/*.h /_/P/Glibc/2.18/include/rpcsvc

make localedata/install-locales

cat > /_/P/Glibc/2.18/etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

Másold a tzdata2013d.tar.gz fájlt a /sources könyvtárba!

```
tar -xf ../tzdata2013d.tar.gz

ZONEINFO=/_/P/Glibc/2.18/share/zoneinfo

mkdir -pv $ZONEINFO/{posix,right}
for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward pacificnew solar87 solar88 solar89 \
          systemv; do
    zic -L /dev/null   -d $ZONEINFO       -y "sh yearistype.sh" ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix -y "sh yearistype.sh" ${tz}
    zic -L leapseconds -d $ZONEINFO/right -y "sh yearistype.sh" ${tz}
done
cp -v zone.tab iso3166.tab $ZONEINFO
```

```
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO

sy Glibc 2.18

cd /_/S/O
ln -s ../../P/Glibc/-/Shared/zoneinfo/America/New_York localtime
```

A fenti sornál az esetben, ha te merő véletlenségből nem New York környékén élnél, a magad időzónájának megfelelő fájlt linkeld be. Ha az emberiség azon jelentéktelen hányadához tartoznál, akik Magyarországon élnek, akkor neked e sor lesz a jó:

```
ln -s ../../P/Glibc/-/Shared/zoneinfo/Europe/Budapest localtime


cat > /_/P/Glibc/-/etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/_/S/L/Y
EOF

mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld

gcc -dumpspecs | sed -e 's@/tools@@g'                   \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' >      \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

INNENTŐL a 64 bites rendszerekre vonatkozik a leírás egészen a következő zöld színű csukó kapcsos zárójelig! Természetesen, akinek 32 bites rendszere van, az ezt a részt ugorja át, folytassa a 32 bites résznél: **{**

Ezután elmentem a
/tools/lib/gcc/x86_64-unknown-linux-gnu/4.8.1

könyvtárba, és ott:

```
cat specs | sed -e 's/\/usr\/include/\/_\/S\/L\/H/g' -e
's/\/usr\/lib/\/_\/S\/L\/Y/g' > spec2
```

majd töröltem a specs filet és a spec2 filet átneveztem specs-re.

```
cd /sources

echo 'main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

A jó válasz:

```
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]


cat dummy.log | grep "/_/S/L/Y/crt[1in].*succeeded"
```

A jó válasz:

```
attempt to open /_/S/L/Y/crt1.o succeeded
attempt to open /_/S/L/Y/crti.o succeeded
attempt to open /_/S/L/Y/crtn.o succeeded
```

grep -B1 '^ /_/S/L/H' dummy.log

Erre a jó válasz:

```
#include <...> search starts here:
 /_/S/L/H
```

grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'

Erre a jó válasz:

```
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

grep "/lib.*/libc.so.6 " dummy.log

Erre a jó válasz:

```
attempt to open /_/P/Glibc/2.18/lib/libc.so.6 succeeded
```

grep found dummy.log

Erre a jó válasz:

```
found ld-linux-x86-64.so.2 at /_/P/Glibc/2.18/lib/ld-linux-x86-64.so.2
```

}

################ Eddig vonatkozott csak a 64 bites rendszerekre!

INNENTŐL a 32 bites rendszerekre vonatkozik a leírás egészen a következő piros csukó kapcsos zárójelig! Természetesen, akinek 64 bites rendszere van, az fentebb már megcsinálta amit kellett, ezt a részt hagyja ki, ugorja át: **{**

Ezután elmentem a
/tools/lib/gcc/i686-pc-linux-gnu/4.8.1
könyvtárba, és ott:

cat specs | sed -e 's/\/usr\/include/\/_\/S\/L\/H/g' -e 's/\/usr\/lib/\/_\/S\/L\/Y/g' > spec2

majd töröltem a specs filet és a spec2 filet átneveztem specs -re.

cd /sources

echo 'main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'

A jó válasz:

```
      [Requesting program interpreter: /lib/ld-linux.so.2]
```

cat dummy.log | grep "/_/S/L/Y/crt[1in].*succeeded"

Erre a jó válasz:

```
attempt to open /_/S/L/Y/crt1.o succeeded
attempt to open /_/S/L/Y/crti.o succeeded
```

```
attempt to open /_/S/L/Y/crtn.o succeeded
```

```
grep -B1 '^ /_/S/L/H' dummy.log
```

Erre a jó válasz:

```
#include <...> search starts here:
 /_/S/L/H
```

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Erre a jó válasz:

```
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

```
grep "/lib.*/libc.so.6 " dummy.log
```

Erre a jó válasz:

```
attempt to open /_/P/Glibc/2.18/lib/libc.so.6 succeeded
```

```
grep found dummy.log
```

Erre a jó válasz:

```
found ld-linux.so.2 at /_/P/Glibc/2.18/lib/ld-linux.so.2
```

```
rm -v dummy.c a.out dummy.log
```

}

INNENTŐL ÚJRA VONATKOZIK EGYARÁNT A 32 ÉS 64 BITES RENDSZEREKRE IS !!!!

```
********************** NAGYON FONTOS !!!! **********************
```

```
ldconfig -v
```

Ezután ellenőrizzük le, jól lettek-e beállítva a locale értékek!

```
locale decimal_point
```

Ha erre nem egy „pont” karaktert ír ki, akkor elszúrtad a Glibc beállítását korábban.

```
*****************************************************************
```

```
============================== 6.11. Zlib-1.2.8
```

```
konfig Zlib 1.2.8
make
make install
sy Zlib 1.2.8
```

```
================================== 6.12. File-5.14
```

```
konfig File 5.14
make
make install
sy File 5.14
```

```
========================================== 6.13. Binutils-2.23.2
expect -c "spawn ls"
```
A jó válasz:
```
spawn ls
ldconfig -v

rm -fv etc/standards.info
sed -i.bak '/^INFO/s/standards.info //' etc/Makefile.in

sed -i -e 's/@colophon/@@colophon/' \
       -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo


mkdir -v ../binutils-build
cd ../binutils-build

../binutils-2.23.2/configure --prefix=/_/P/BinUtils/2.23.2 --enable-shared

make tooldir=/_/P/BinUtils/2.23.2

make check

make tooldir=/_/P/BinUtils/2.23.2 install

cp -v ../binutils-2.23.2/include/libiberty.h /_/P/BinUtils/2.23.2/include

sy BinUtils 2.23.2


============================== 6.14. GMP-5.1.2

ldconfig -v

konfig GMP 5.1.2 --enable-cxx
make
make check 2>&1 | tee gmp-check-log
awk '/tests passed/{total+=$2} ; END{print total}' gmp-check-log
```
A jó válasz:
```
185

make install
```
A doksikat nem installálom. Kicsi rá az 500 gigás merevlemezem...
```
sy GMP 5.1.2

========================================== 6.15. MPFR-3.1.2

konfig MPFR 3.1.2 --enable-thread-safe
make
make check
make install
```
doksikat itt sem installálok.
```
sy MPFR 3.1.2

================================== 6.16. MPC-1.0.1
```

```
konfig MPC 1.0.1
make
make check
make install

sy MPC 1.0.1
```

================================ 6.17. GCC-4.8.1

```
case `uname -m` in
  i?86) sed -i 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in ;;
esac

sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in

sed -i -e /autogen/d -e /check.sh/d fixincludes/Makefile.in
mv -v libmudflap/testsuite/libmudflap.c++/pass41-frag.cxx{,.disable}

mkdir -v ../gcc-build
cd ../gcc-build

../gcc-4.8.1/configure --prefix=/_/P/GCC/4.8.1  --enable-shared --enable-
threads=posix --enable-__cxa_atexit --enable-clocale=gnu  --enable-
languages=c,c++ --disable-multilib --disable-bootstrap --disable-install-
libiberty  --with-system-zlib


make

ulimit -s 32768
make -k check
../gcc-4.8.1/contrib/test_summary | grep -A7 Summ
```

Eredménye nálam:

```
                === g++ Summary ===

# of expected passes            54027
# of expected failures          290
# of unsupported tests          877
/sources/gcc-build/gcc/testsuite/g++/../../xg++  version 4.8.1 (GCC)

                === gcc tests ===
--
                === gcc Summary ===

# of expected passes            93335
# of expected failures          261
# of unsupported tests          1356
/sources/gcc-build/gcc/xgcc  version 4.8.1 (GCC)

                === libatomic tests ===
--
                === libatomic Summary ===

# of expected passes            54
                === libgomp tests ===
```

```
Running target unix

                === libgomp Summary ===

# of expected passes            1313
                === libitm tests ===



Running target unix

                === libitm Summary ===

# of expected passes            26
# of expected failures          3
# of unsupported tests          1
                === libmudflap tests ===


--
                === libmudflap Summary ===

# of expected passes            1428
                === libstdc++ tests ===


Running target unix

                === libstdc++ Summary ===

# of expected passes            9221
# of expected failures          45
# of unsupported tests          218

Compiler version: 4.8.1 (GCC)
Platform: x86_64-unknown-linux-gnu
```

make install

A /_/P/GCC/4.8.1/bin könyvtárba elmentem, és ott csináltam egy symlinket e paranccsal:

ln -s gcc cc

Töröld le a következő symlinkeket:

```
/_/S/L/Y/libgcc_s.so
/_/S/L/Y/libgcc_s.so.1
/_/S/L/Y/libstdc++.la
/_/S/L/Y/libstdc++.so
/_/S/L/Y/libstdc++.so.6
```

Ezek ugyanis a /tools fában levő régi változatokra mutatnak.

sy GCC 4.8.1

```
echo 'main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

INNENTŐL a 64 bites rendszerekre vonatkozik a leírás egészen a következő zöld színű csukó kapcsos zárójelig! Természetesen, akinek 32 bites rendszere van, az ezt a részt ugorja át, folytassa a 32 bites résznél: **{**

A jó válasz:

```
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

cat dummy.log | grep "succeeded" | grep crt

A jó válasz:

```
attempt to open /lib/crt1.o succeeded
attempt to open /lib/crti.o succeeded
attempt to open /Releases/2014/P/GCC/4.8.1/bin/../lib/gcc/x86_64-unknown-linux-
gnu/4.8.1/crtbegin.o succeeded
attempt to open /Releases/2014/P/GCC/4.8.1/bin/../lib/gcc/x86_64-unknown-linux-
gnu/4.8.1/crtend.o succeeded
attempt to open /lib/crtn.o succeeded
```

cat dummy.log | grep -B2 "/_/S/L/H"

A jó válasz:

```
#include "..." search starts here:
#include <...> search starts here:
 /_/S/L/H
```

cat dummy.log | grep "SEARCH_DIR("

A jó válasz:

```
SEARCH_DIR("/_/P/BinUtils/2.23.2/x86_64-unknown-linux-gnu/lib64");
SEARCH_DIR("/_/P/BinUtils/2.23.2/lib64"); SEARCH_DIR("/usr/local/lib64");
SEARCH_DIR("/lib64"); SEARCH_DIR("/usr/lib64");
SEARCH_DIR("/_/P/BinUtils/2.23.2/x86_64-unknown-linux-gnu/lib");
SEARCH_DIR("/_/P/BinUtils/2.23.2/lib"); SEARCH_DIR("/usr/local/lib");
SEARCH_DIR("/lib"); SEARCH_DIR("/usr/lib");
```

grep "/lib.*/libc.so.6 " dummy.log

A jó válasz:

```
attempt to open /_/P/Glibc/2.18/lib/libc.so.6 succeeded
```

grep found dummy.log

A jó válasz:

```
found ld-linux-x86-64.so.2 at /_/P/Glibc/2.18/lib/ld-linux-x86-64.so.2
```

rm -v dummy.c a.out dummy.log

Csináljunk egy könyvtárat a /_/P/GCC/4.8.1/Shared könyvtárba auto-load néven.
Ide másoljuk át a /_/P/GCC/4.8.1/lib64 könyvtárból az összes .py végű fájlt.
Ezt az auto-load könyvtárat symlinkeljük be a /_/S/L/T/gdb könyvtárba.

}

################ Eddig vonatkozott csak a 64 bites rendszerekre!

INNENTŐL a 32 bites rendszerekre vonatkozik a leírás egészen a következő piros csukó kapcsos zárójelig! Természetesen, akinek 64 bites rendszere van, az fentebb már megcsinálta amit kellett, ezt a részt hagyja ki, ugorja át: **{**

A jó válasz:

```
      [Requesting program interpreter: /lib/ld-linux.so.2]
```

cat dummy.log | grep "succeeded" | grep crt

A jó válasz:

```
attempt to open /lib/crt1.o succeeded
attempt to open /lib/crti.o succeeded
attempt to open /Releases/2014/P/GCC/4.8.1/bin/../lib/gcc/i686-pc-linux-
gnu/4.8.1/crtbegin.o succeeded
attempt to open /Releases/2014/P/GCC/4.8.1/bin/../lib/gcc/i686-pc-linux-
gnu/4.8.1/crtend.o succeeded
attempt to open /lib/crtn.o succeeded
```

cat dummy.log | grep -B2 "/_/S/L/H"

A jó válasz:

```
#include "..." search starts here:
#include <...> search starts here:
 /_/S/L/H
```

cat dummy.log | grep "SEARCH_DIR("

A jó válasz:

```
SEARCH_DIR("/_/P/BinUtils/2.23.2/i686-pc-linux-gnu/lib");
SEARCH_DIR("/_/P/BinUtils/2.23.2/lib"); SEARCH_DIR("/usr/local/lib");
SEARCH_DIR("/lib"); SEARCH_DIR("/usr/lib");
```

grep "/lib.*/libc.so.6 " dummy.log

A jó válasz:

```
attempt to open /_/P/Glibc/2.18/lib/libc.so.6 succeeded
```

grep found dummy.log

A jó válasz:

```
found ld-linux.so.2 at /_/P/Glibc/2.18/lib/ld-linux.so.2
```

rm -v dummy.c a.out dummy.log

Csináljunk egy könyvtárat a /_/P/GCC/4.8.1/Shared könyvtárba auto-load néven.
Ide másoljuk át a /_/P/GCC/4.8.1/lib könyvtárból az összes .py végű fájlt.
Ezt az auto-load könyvtárat symlinkeljük be a /_/S/L/T/gdb könyvtárba.

}

INNENTŐL ÚJRA VONATKOZIK EGYARÁNT A 32 ÉS 64 BITES RENDSZEREKRE IS !!!!

========== 6.18. Sed-4.2.2

```
konfig Sed 4.2.2
make
make html
make check
make install
make -C doc install-html
sy Sed 4.2.2
```

========== 6.19. Bzip2-1.0.6

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch

sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile

make -f Makefile-libbz2_so
make clean

make

make PREFIX=/_/P/Bzip2/1.0.6 install
cp -v bzip2-shared /_/P/Bzip2/1.0.6/bin
cp -av libbz2.so* /_/P/Bzip2/1.0.6/lib
```

A /_/P/Bzip2/1.0.6/lib könyvtárban állva:

```
ln -s libbz2.so.1.0 libbz2.so
```

Töröljük a /_/P/Bzip2/1.0.6/bin könyvtárból a bunzip2 és bzcat programokat, és csináljunk ugyanilyen neveken egy-egy symlinket a bzip2 programra!

```
sy Bzip2 1.0.6
```

========== 6.20. Pkg-config-0.28

```
konfig Pkg-config 0.28 --with-internal-glib --disable-host-tool

make
make check
make install
sy Pkg-config 0.28
```

========== 6.21. Ncurses-5.9

```
konfig Ncurses 5.9 --with-shared --without-debug --enable-pc-files --enable-
widec

make

make install
```

Elmenni a /_/P/Ncurses/5.9/lib könyvtárba, és ott:

```
ln -s libncursesw.a libncurses.a
ln -s libncurses++w.a libncurses++.a
```

```
ln -s libncursesw.so libncurses.so
ln -s libncursesw.so.5 libncurses.so.5
ln -s libncursesw.so.5.9 libncurses.so.5.9
ln -s libformw.a libform.a
ln -s libformw.so libform.so
ln -s libformw.so.5 libform.so.5
ln -s libformw.so.5.9 libform.so.5.9
ln -s libmenuw.so libmenu.so
ln -s libmenuw.so.5 libmenu.so.5
ln -s libmenuw.so.5.9 libmenu.so.5.9
ln -s libmenuw.a libmenu.a
ln -s libpanelw.a libpanel.a
ln -s libpanelw.so libpanel.so
ln -s libpanelw.so.5 libpanel.so.5
ln -s libpanelw.so.5.9 libpanel.so.5.9
echo "INPUT(-lncursesw)" > libcursesw.so
ln -s libncurses.so libcurses.so
ln -s libncursesw.a libcursesw.a
ln -s libncurses.a libcurses.a
```

MÁSOLJUK ÁT (Nem átmozgatjuk hanem MÁSOLJUK, azaz DUPLIKÁLJUK!) a
/_/P/Ncurses/5.9/include/ncursesw
könyvtár összes fájlját a
/_/P/Ncurses/5.9/include
könyvtárba!
Enélkül sajnos pár program később nem fordulna le... :(

```
sy Ncurses 5.9

============================== 6.22. Shadow-4.1.5.1

sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;

sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs

konfig Shadow 4.1.5.1

make
make install

sy Shadow 4.1.5.1

pwconv

grpconv
```

Ezután határozhatod meg a készülő új rendszeredben érvényes root jelszót:

```
passwd root

======================== 6.23. Util-linux-2.23.2

sed -i -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
     $(grep -rl '/etc/adjtime' .)

konfig Util-Linux 2.23.2 --disable-su --disable-sulogin --disable-login
```

```
make

make install
sy Util-Linux 2.23.2
```

============================== 6.24. Psmisc-22.20

```
konfig Psmisc 22.20

make
make install

sy Psmisc 22.20
```

============================ 6.25. Procps-ng-3.3.8

```
konfig Procps-ng 3.3.8 --disable-static --disable-skill --disable-kill

make
make install

sy Procps-ng 3.3.8
```

============================ 6.26. E2fsprogs-1.42.8

```
sed -i -e 's/mke2fs/$MKE2FS/' -e 's/debugfs/$DEBUGFS/'
tests/f_extent_oobounds/script

mkdir -v build
cd build

../configure --prefix=/_/P/E2fsprogs/1.42.8 --enable-elf-shlibs --disable-
libblkid --disable-libuuid  --disable-uuidd  --disable-fsck

make

ldconfig -v
```

(Másképp, ha kihagyod a fenti ldconfig parancsot, a „make check” hibát jelez, mert nem tudja betölteni a blkid libjét!)

```
make check
```

A végén ezt írja ki:

```
131 tests succeeded     5 tests failed
Tests failed: f_mmp f_mmp_garbage m_mmp t_mmp_1on t_mmp_2off
make[1]: *** [test_post] Error 1
make[1]: Leaving directory `/sources/e2fsprogs-1.42.8/build/tests'
make: *** [check-recursive] Error 1
```

```
make install

make install-libs

chmod -v u+w /_/P/E2fsprogs/1.42.8/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Doksikat nem installálok. Ha kell valami, úgyis a Zinterneten nézek utána.

```
sy E2fsprogs 1.42.8
```

========== 6.27. Coreutils-8.21

```
patch -Np1 -i ../coreutils-8.21-i18n-1.patch

FORCE_UNSAFE_CONFIGURE=1 konfig CoreUtils 8.21 --enable-no-install-
program=kill,uptime

make

make install

sy CoreUtils 8.21
```

A /_/S/L/E könyvtárban írjuk át a cat symlinkjét, hogy a /_/P/CoreUtils/-/bin/cat file-re mutasson
A /_/S/L/E könyvtárban írjuk át az echo symlinkjét, hogy a /_/P/Core-Utils/-/bin/echo file-re mutasson
A /_/S/L/E könyvtárban írjuk át az pwd symlinkjét, hogy a /_/P/CoreUtils/-/bin/pwd file-re mutasson
A /_/S/L/E könyvtárban írjuk át az stty symlinkjét, hogy a /_/P/CoreUtils/-/bin/stty file-re mutasson

(Ezek ugyanis eddig a /tools könyvtárban levő ideiglenes rendszer megfelelő programjaira mutattak).

========== 6.28. Iana-Etc-2.30

Átírni a Makefile-ben az elején ezt a sort:

```
ETC_DIR=/etc
```

erre:

```
ETC_DIR=/_/P/Iana-Etc/2.30/etc
```

Ezután:

```
make
make install
sy Iana-Etc 2.30
```

========== 6.29. M4-1.4.16

```
sed -i -e '/gets is a/d' lib/stdio.in.h

konfig M4 1.4.16

make

sed -i -e '41s/ENOENT/& || errno == EINVAL/' tests/test-readlink.h
make check

make install

sy M4 1.4.16
```

========== 6.30. Flex-2.5.37

```
sed -i -e '/test-bison/d' tests/Makefile.in

konfig Flex 2.5.37

make

make check

make install

cat > /_/P/Flex/2.5.37/bin/lex << "EOF"
#!/bin/sh
# Begin lex
exec /_/S/L/E/flex -l "$@"
# End lex
EOF

chmod -v 755 /_/P/Flex/2.5.37/bin/lex
sy Flex 2.5.37
```

================================ 6.31. Bison-3.0

```
konfig Bison 3.0

make
make check
make install

sy Bison 3.0
```

== 6.32. Grep-2.14

```
konfig Grep 2.14

make
make check
make install

sy Grep 2.14
```

================================ 6.33. Readline-6.2

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install

patch -Np1 -i ../readline-6.2-fixes-1.patch

konfig Readline 6.2

make SHLIB_LIBS=-lncurses
make install
sy Readline 6.2
```

Töröljük a bin directoryt, mert üres:

```
rm -rf /_/P/Readline/6.2/bin
```

============================ 6.34. Bash-4.2

```
patch -Np1 -i ../bash-4.2-fixes-12.patch
```

```
konfig Bash 4.2 --without-bash-malloc  --with-installed-readline

make

chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make tests"
```

A teszt végén ezt írja ki:

```
warning: there may be a message regarding a cat process dying due to a
warning: SIGHUP.  Please disregard.
```

Itt nyomni kell egy Ctrl-C -t, erre ezt írja ki és visszaadja a promptot:

```
^Cmake: *** [tests] Interrupt
```

Bevallom, ennek nem tudom mi az oka.

```
make install
```

A /_/P/Bash/4.2/bin könyvtárban:

```
ln -s bash sh

sy Bash 4.2
```

A /_/S/L/E könyvtárban írjuk át a bash és a sh symlinkeket, hogy a
/_/P/Bash/-/bin/bash
illetve a
/_/P/Bash/-/bin/sh
progikra mutassanak. Ezek is eddig ugyanis a /tools könyvtárban levő ideiglenes rendszer beli bash shellre mutattak.

```
================================== 6.35. Bc-1.06.95

konfig Bc 1.06.95 --with-readline

make

make install

sy Bc 1.06.95
```

Ezután a "sy" parancsunk már kiírja majd rendesen a sztrippelés utáni méretkülönbségeket.

```
================================ 6.36. Libtool-2.4.2

konfig LibTool 2.4.2

make
make check
make install

sy LibTool 2.4.2

ldconfig -v

======================================== 6.37. GDBM-1.10
```

```
konfig GDBM 1.10 --enable-libgdbm-compat

make
make check
make install
sy GDBM 1.10
```

========== 6.38. Inetutils-1.9.1

```
sed -i -e '/gets is a/d' lib/stdio.in.h

konfig InetUtils 1.9.1 --localstatedir=/_/S/V --disable-ifconfig   --disable-
logger --disable-syslogd --disable-whois  --disable-servers

make
make check
make install
sy InetUtils 1.9.1
```

Töröljük a /_/P/InetUtils/-/libexec könyvtárat, mert üres, s ezért felesleges.

========== 6.39. Perl-5.18.1

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"           \
       -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /usr/include|" \
       -e "s|LIB\s*= ./zlib-src|LIB        = /usr/lib|"         \
    cpan/Compress-Raw-Zlib/config.in


sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"          \
          -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /_/S/L/H|" \
          -e "s|LIB\s*= ./zlib-src|LIB        = /_/S/L/Y|"     \
          cpan/Compress-Raw-Zlib/config.in

sh Configure -des -Dprefix=/_/P/Perl/5.18.1 -Dvendorprefix=/_/P/Perl/5.18.1
-Dman1dir=/_/P/Perl/5.18.1/share/man/man1
-Dman3dir=/_/P/Perl/5.18.1/share/man/man3 -Dpager="/_/S/L/E/less -isR"
-Duseshrplib

make
make -k test
make install
sy Perl 5.18.1
```

A
/_/S/L/E/perl
linket írjuk át úgy, hogy a
/_/P/Perl/-/bin/perl
-re mutasson. Ezt EL NE FELEDJÜK, különben az ezután felrakandó Autoconf illetve Automake hibásan fog működni (vagy egyáltalán nem), annak ellenére, hogy hibamentesen lefordulnak majd neked!

========== 6.40. Autoconf-2.69

```
konfig Autoconf 2.69

make
make check
make install
```

```
sy Autoconf 2.69

============================ 6.41. Automake-1.14

patch -Np1 -i ../automake-1.14-test-1.patch

konfig Automake 1.14

make
make install

sy Automake 1.14

============================== 6.42. Diffutils-3.3

konfig DiffUtils 3.3

make
make check
make install

sy DiffUtils 3.3

================================== 6.43. Gawk-4.1.0

konfig Gawk 4.1.0

make
make check
make install
sy Gawk 4.1.0

=========================== 6.44. Findutils-4.4.2

konfig FindUtils 4.4.2 --localstatedir=/_/S/V/lib/locate

make
make check
make install

sy FindUtils 4.4.2

============================= 6.45. Gettext-0.18.3

konfig GetText 0.18.3

make
make check
make install
sy GetText 0.18.3

============================== 6.46. Groff-1.22.2

PAGE=A4 konfig Groff 1.22.2

make

mkdir -p /_/P/Groff/1.22.2/share/doc/groff-1.22/pdf
make install

A /_/P/Groff/1.22.2/bin mappában állva:
```

```
ln -s eqn qeqn
ln -s tbl qtbl
sy Groff 1.22.2

================================ 6.47. Xz-5.0.5

konfig Xz 5.0.5

make
make check
make pkgconfigdir=/_/P/Xz/5.0.5/pkgconfig install

sy Xz 5.0.5

================================== 6.48. GRUB-2.00
```

Megnyugtatásul előre bocsátom, ez, ami itt következik, NEM azt jelenti, hogy új Grub-ot telepítesz a merevlemezed rendszerbetöltő szektorába, hanem csak hogy a készülő rendszeredben lesz egy grub nevű program, amit ha akarsz, majd elindíthatsz valamikor később, hogy betelepedjék a merevlemezed megfelelő helyére! Szóval nyugodtan telepítheted így, ahogy itt le van írva, ez nem jelenti azt, hogy ezentúl ne tudnál netán bebootolni a régi gruboddal!

```
sed -i -e '/gets is a/d' grub-core/gnulib/stdio.in.h

konfig Grub 2.00 --disable-grub-emu-usb --disable-efiemu --disable-werror

make
make install

sy Grub 2.00

================================= 6.49. Less-458

konfig Less 458
make
make install
sy Less 458

================================== 6.50. Gzip-1.6

konfig Gzip 1.6

make
make check
make install
sy Gzip 1.6

============================ 6.51. IPRoute2-3.10.0

sed -i '/^TARGETS/s@arpd@@g' misc/Makefile
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile

make DESTDIR=/_/P/IProute2/3.10.0

make DESTDIR=/_/P/IProute2/3.10.0  install
```

A /_P/IProute2/3.10.0/usr könyvtárból mozgassuk át a share és lib könyvtárakat egy szinttel feljebb, a /_P/IProute2/3.10.0 könyvtárba, majd töröljük az üres usr könyvtárat.

```
sy IProute2 3.10.0
```

6.52. Kbd-1.15.5

```
patch -Np1 -i ../kbd-1.15.5-backspace-1.patch
sed -i -e '326 s/if/while/' src/loadkeys.analyze.l

sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' man/man8/Makefile.in

konfig Kbd 1.15.5 --disable-vlock

make
make install
sy Kbd 1.15.5
```

6.53. Kmod-14

```
konfig Kmod 14 --disable-manpages --with-xz --with-zlib
make
make check
```

A check 1 hibát jelez:

```
Testsuite summary for kmod 14
============================================================================
# TOTAL: 10
# PASS:  9
# SKIP:  0
# XFAIL: 0
# FAIL:  1
# XPASS: 0
# ERROR: 0
============================================================================
See ./test-suite.log
Please report to linux-modules@vger.kernel.org
============================================================================
make[3]: *** [test-suite.log] Error 1
make[2]: *** [check-TESTS] Error 2
make[1]: *** [check-am] Error 2
make: *** [check-recursive] Error 1
```

```
make pkgconfigdir=/_P/Kmod/14/lib/pkgconfig install
```

A /_P/Kmod/14/bin könyvtárban állva:

```
for target in depmod insmod modinfo modprobe rmmod; do
ln -sv kmod $target
done
ln -sv kmod lsmod

sy Kmod 14
```

6.54. Libpipeline-1.2.4

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig konfig LibPipeLine 1.2.4

make
make check
make install

sy LibPipeLine 1.2.4

========================================= 6.55. Make-3.82

patch -Np1 -i ../make-3.82-upstream_fixes-3.patch

konfig Make 3.82

make
make check
make install

sy Make 3.82

============================ 6.56. Man-DB-2.6.5

konfig Man-DB 2.6.5 --disable-setuid  --with-browser=/_/S/L/E/lynx --with-
vgrind=/_/S/L/E/vgrind  --with-grap=/_/S/L/E/grap

make
make check
make install

sy Man-DB 2.6.5

================================ 6.57. Patch-2.7.1

konfig Patch 2.7.1

make
make check
make install

sy Patch 2.7.1

============================== 6.58. Sysklogd-1.5

make
mkdir -p /_/P/Sysklogd/1.5/sbin
make BINDIR=/_/P/Sysklogd/1.5/sbin install
sy Sysklogd 1.5
mkdir /_/P/Sysklogd/-/etc
cat > /_/P/Sysklogd/-/etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# End /etc/syslog.conf
EOF
```

Ezt a sysklogd.conf fájlt symlinkeljük be a /_/S/O directoryba. Például ezzel a paranccsal:

```
LINK /_/S/O /_/P/Sysklogd/-/etc
```

========== 6.59. Sysvinit-2.88dsf

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' src/init.c

sed -i -e '/utmpdump/d' \
       -e '/mountpoint/d' src/Makefile
```

A sysvinit-2.88dsf/src/Makefile fájl elejére írjuk be e sort:

```
ROOT=/_/P/Sysvinit/2.88dsf
```

Ezután (a programcsomag főkönyvtárában állva):

```
make -C src

make -C src install
```

A /_/P/Sysvinit/2.88dsf/bin könyvtárban a pidof nevű symlinket írjuk át úgy, hogy a /_/S/L/E/killall5 -re mutasson.

```
sy Sysvinit 2.88dsf
```

========== 6.60. Tar-1.26

```
patch -Np1 -i ../tar-1.26-manpage-1.patch

sed -i -e '/gets is a/d' gnu/stdio.in.h

FORCE_UNSAFE_CONFIGURE=1 konfig Tar 1.26

make
```

A tesztet jobb, ha nem futtatjuk... Veszélyes... :(

```
make install

mkdir -p  /_/P/Tar/1.26/share/man/man1
perl tarman > /_/P/Tar/1.26/share/man/man1/tar.1

sy Tar 1.26
```

Töröljük a /_/P/Tar/1.26/sbin könyvtárat, mert üres.

========== 6.61. Texinfo-5.1

```
patch -Np1 -i ../texinfo-5.1-test-1.patch

konfig TexInfo 5.1
make
make check
make install

make TEXMF=/_/P/TexInfo/5.1/share/texmf install-tex

sy TexInfo 5.1
```

======= 6.62. Udev-206 (Extracted from systemd-206)

A fucked Udev-et pont úgy installáltam, ahogy az LFS írta, SEMMIT nem változtattam rajta!

Épp emiatt az nem is a /_/P alá van telepítve. :(

Vagyis, itt a systemd csomagot csomagoljuk ki a /sources könyvtárba, az udev-lfs-206-1.tar.bz2 csomagot meg csak úgy másoljuk be kicsomagolás nélkül a /sources könyvtárba.

A kicsomagolt systemd könyvtárában állva:

```
tar -xvf ../udev-lfs-206-1.tar.bz2

make -f udev-lfs-206-1/Makefile.lfs

make -f udev-lfs-206-1/Makefile.lfs install

build/udevadm hwdb --update

bash udev-lfs-206-1/init-net-rules.sh
```

A telepítés végén ellenőrizzük le a /_/S/L/Y könyvtárat, hogy az ottani **libudev.so** symlink jó helyre mutat-e! Ha nem a **libudev.so.1.3.2** fájlra mutatna netán, mondjuk mert törött link volna, írjuk át úgy, hogy arra mutasson!

================================= 6.63. Vim-7.4

```
echo '#define SYS_VIMRC_FILE "/_/P/Vim/-/etc/vimrc"' >> src/feature.h

konfig Vim 7.4 --enable-multibyte

make
```

A teszttel nem kínlódok, nagyon bonyolult. Majd ha felállt az új rendszer, elindítom a vimet és megnézem jól működik-e.

```
make install
```

A /_/P/Vim/7.4/bin könyvtárban állva adjuk ki e parancsot:

```
ln -s vim vi

mkdir /_/P/Vim/7.4/etc

cat > /_/P/Vim/7.4/etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif
```

```
" End /etc/vimrc
EOF


sy Vim 7.4
```

****

Eddig tartottak az „alap” LFS Book programjai. Innentől jön az, ami tulajdonképpen a BLFS tartalma, de én még a minimálrendszer részének tartom, mert kell az MC-hez, illetve a Check ahhoz kell, hogy további programtelepítéseknél a teszteket végrehajthassuk.

================= 5.14. Check-0.9.10

```
konfig Check 0.9.10
make
make install
sy Check 0.9.10
```

====================================

Kijelentkezem a chroot környezetből exit-tel:

```
exit
```

Törlöm a /tools könyvtárat. !!
(Ezt töröld valami neked tetsző módon. Akár MC-vel a host rendszer alól...)

Most csinálj a **KISS** szkript mintájára egy **kiss** nevű szkriptet, mert mostantól másképp lépek be a chroot környezetbe:

```
#!/bin/bash
mount -v --bind /dev $LFS/dev
mount -vt devpts devpts $LFS/dev/pts -o gid=5,mode=620
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
if [ -h $LFS/dev/shm ]; then
  link=$(readlink $LFS/dev/shm)
  mkdir -p $LFS/$link
  mount -vt tmpfs shm $LFS/$link
  unset link
else
  mount -vt tmpfs shm $LFS/dev/shm
fi

chroot "$LFS" /_/S/L/E/env -i \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='\u:\w\$ '              \
    LD_LIBRARY_PATH="/_/S/L/Y" \
    PATH="/_/S/L/E" \
C_INCLUDE_PATH="/_/S/L/H" \
PKG_CONFIG_PATH="/_/S/L/C" \
    /_/S/L/E/bash --login
```

A fejlesztés alatt álló rendszerem /root könyvtárában a .bashrc tartalma ez kell legyen:

```
export PS1="\[\033[1;31m\]\u\[\033[1;36m\]@\[\033[0;36m\]\h \[\033[1m\]\w\
[\033[1;31m\]=>\[\033[0m\]"

export C_INCLUDE_PATH="/_/S/L/H"
export LD_LIBRARY_PATH="/_/S/L/Y"
export PATH="/_/S/L/E"
export PKG_CONFIG_PATH="/_/S/L/C"

export EDITOR="mcedit"

export MAKEFLAGS='-j 5'

alias x=exit
alias make=kolormake

alias mc='. /_/S/L/X/mc/mc-wrapper.sh'

alias KISS="cd /_/P/KISS/-/sources"
alias P="cd /_/P"
alias S="cd /_/S/O"
alias L="cd /_/S/L"
alias A="cd /Archives"
alias s="cd /sources"

##########################################

kicsomagol () {
    if [ -f $1 ] ; then
    case $1 in
        *.tar.bz2)   tar xvjf $1        ;;
        *.tar.gz)    tar xvzf $1        ;;
        *.bz2)       bunzip2 $1         ;;
        *.rar)       unrar x $1         ;;
        *.gz)        gunzip $1          ;;
        *.tar)       tar xvf $1         ;;
        *.tbz2)      tar xvjf $1        ;;
        *.tgz)       tar xvzf $1        ;;
        *.zip)       unzip $1           ;;
        *.Z)         uncompress $1      ;;
        *.z)         7z x $1            ;;
        *.arj)       arj x $1           ;;
        *.xz)        tar -xJvf $1           ;;

        *)           echo "A(z) '$1' nem kicsomagolható" ;;
    esac
        else
        echo "'$1' nevű állomány nem létezik"
        fi
            }
```

Ezentúl tehát nem úgy lépsz be a host alól chroot környezetbe hogy

```
./KISS
```

hanem úgy, hogy

```
./kiss
```

Miután újra beléptem a chroot-ba, mindenekelőtt:

```
ldconfig -v
```

Ezután újra kell fordítani a **colored** programot, hogy az ne a már nem létező /tools libraryjait használja, hanem a végleges rendszerben levőket:

```
cd /_/P/KISS/-/sources
gcc colored.c -o colored
cp colored ../bin
```

============================== 8.3. Linux-3.10.10

...Azaz a kernelfordítás...

```
make mrproper

unalias make
```

(A fenti unalias parancs nagyon fontos, különben a menuconfig nem fog működni...)

```
make LANG=hu_HU.UTF-8 LC_ALL= menuconfig
```

Miután megvolt a menuconfig, visszaállíthatjuk az aliasunkat:

```
alias make='kolormake'
```

Ezután:

```
make
make modules_install

cp -v arch/x86/boot/bzImage /boot/vmlinuz-3.10.10-lfs-7.4
cp -v System.map /boot/System.map-3.10.10
cp -v .config /boot/config-3.10.10

install -d /usr/share/doc/linux-3.10.10
cp -r Documentation/* /usr/share/doc/linux-3.10.10
```

Mint látható, technikailag a kernelfordítás nem egy nagy ördöngösség. A gond abban rejlik, miként állítsd be a menuconfig menüjében a megfelelő paramétereket! Sajnos, ebben nem tudok segíteni neked, mert a te gépedet neked kell ismerned. Előtte mindenképpen ajánlatos elolvasni a BLFS Book Xorg driverekkel foglalkozó fejezetét, amire itt a link:

http://www.linuxfromscratch.org/blfs/view/svn/x/x7driver.html

azért, hogy beleforgasd a kernelbe azt, ami a te videokártyádhoz kell. Igaz, jó sokára jön még az a rész, ahol az X-et installáljuk, s addig a kerneled jó eséllyel működik majd e driverek nélkül is, de jobb most beleforgatnod a kernelbe, ami kell, különben forgathatsz új kernelt, ha majd grafikus felületet is akarsz magadnak, és az nem kellemes dolog. Nekem elhiheted, mert én is úgy jártam!

```
================= GPM-1.20.7

./autogen.sh

konfig GPM 1.20.7

make

make install

install -v -m644 conf/gpm-root.conf /_/P/GPM/1.20.7/etc

A /_/P/GPM/1.20.7/lib könyvtárban állva:
ln -sv libgpm.so.2.1.0 libgpm.so

sy GPM 1.20.7
```

A blfs-bootscripts-20130908 csomagból:

```
make install-gpm

cat > /etc/sysconfig/mouse << "EOF"
# Begin /etc/sysconfig/mouse

MDEVICE="/dev/psaux"
PROTOCOL="imps2"
GPMOPTS=""

# End /etc/sysconfig/mouse
EOF

==================================== PCRE-8.33

konfig PCRE 8.33 --enable-utf --enable-unicode-properties  --enable-pcre16
--enable-pcre32 --enable-pcregrep-libz --enable-pcregrep-libbz2  --enable-
pcretest-libreadline --disable-static

make
make check
make install
sy PCRE 8.33

============================ libffi-3.0.13

konfig LibFFI 3.0.13 --disable-static

make
```

(tesztet nem futtatunk, ahhoz kéne a DejaGNU 1.5.1)

```
make install
```

A /_/P/LibFFI/3.0.13/lib/libffi-3.0.13/include könyvtárat mozgassuk át, hogy a /_/P/LibFFI/3.0.13/include legyen a helye!

Töröljük a
A /_/P/LibFFI/3.0.13/lib/libffi-3.0.13 könyvtárat mert üres!

A /_/P/LibFFI/3.0.13/lib/pkgconfig könyvtárat mozgassuk át, hogy a /_/P/LibFFI/3.0.13/pkgconfig legyen a helye!

Töröljük a
/_/P/LibFFI/3.0.13/lib könyvtárat mert üres!

```
sy LibFFI 3.0.13

mkdir -p /_/P/LibFFI/3.0.13/lib/libffi-3.0.13
cd /_/P/LibFFI/3.0.13/lib/libffi-3.0.13
ln -sv /_/S/L/H include

================================================ attr-2.4.47

sed -i -e 's|/@pkg_name@|&-@pkg_version@|' include/builddefs.in

INSTALL_USER=root INSTALL_GROUP=root konfig Attr 2.4.47 --disable-static

make

make install install-dev install-lib

chmod -v 755 /_/P/Attr/2.4.47/lib/libattr.so

sy Attr 2.4.47

================================================ Zip-3.0

make -f unix/Makefile generic_gcc

make prefix=/_/P/Zip/3.0 -f unix/Makefile install

sy Zip 3.0

=========================================== UnZip-6.0

case `uname -m` in
  i?86)
    sed -i -e 's/DASM"/DASM -DNO_LCHMOD"/' unix/Makefile
    make -f unix/Makefile linux
    ;;
  *)
    sed -i -e 's/CFLAGS="-O -Wall/& -DNO_LCHMOD/' unix/Makefile
    make -f unix/Makefile linux_noasm
    ;;
esac

make prefix=/_/P/UnZip/6.0 install

sy UnZip 6.0

================================== Expat-2.1.0

konfig Expat 2.1.0 --disable-static

make
make install
sy Expat 2.1.0

============================== Python-2.7.5

konfig Python 2.7.5  --enable-shared  --with-system-expat --with-system-ffi
--enable-unicode=ucs4
```

```
make
make install
chmod -v 755 /_/P/Python/2.7.5/lib/libpython2.7.so.1.0
sy Python 2.7.5
```

Töröljük a /_/P/Python/2.7.5/lib/python2.7/site-packages könyvtárat.
Töröljük a /_/S/L/Y/python2.7/site-packages/README symlinket.
Hozzunk létre egy symlinket site-packages néven a /_/P/Python/2.7.5/lib/python2.7 könyvtárba, ami a /_/S/L/Y/python2.7/site-packages könyvtárra mutat!

Később újra kell majd forgatnunk a Pythont, hogy benne legyen a Tk támogatás is!

```
========================== GLib-2.36.4

touch -t 201306082300 gtk-doc.make
konfig GLib 2.36.4 --with-pcre=system
make
make install
sy GLib 2.36.4

======================================== MC-4.8.10

konfig MC 4.8.10 --enable-charset --disable-static  --with-screen=ncurses

make
make check
make install
cp -v doc/keybind-migration.txt /_/P/MC/4.8.10/share/mc
sy MC 4.8.10

================================================================================
================================================================================
================================================================================
================================================================================
================================================================================
```

Eddig tartott az, amit én az alaprendszernek tartok, más néven minimál-rendszernek. Most annyi a dolgod, hogy elolvasod rém alaposan az LFS Book hetedik és nyolcadik fejezetét, de tényleg **nagyon** alaposan (itt kezdődik: http://www.linuxfromscratch.org/lfs/view/stable/chapter07/introduction.html) azért, hogy jól megértsd, hogyan is kell inicializálnod a rendszeredet, beállítanod a boot szkripteket, stb. Sajnos, ezzel lehet hogy több időd vész el és több kínlódásodba kerül ez, mint az egész eddigi rendszertelepítés... velem mindenesetre így esett! Tudod, ilyen ügyek kapcsán derülnek ki az ember ismereteiben mutatkozó fehér foltok, hézagok, hiányosságok. Hát ezeket pótolni kell. Ez rém keserves folyamat lesz. Én megtettem, de még most is bőven van olyan terület errefelé, ahol saját véleményem szerint is síkhülye vagyok. Ez veled is így lesz egészen biztosan, tudniillik mert ha ezeket a dolgokat már tudnád, felesleges lenne neked a könyvem olvasása, rég abbahagytad volna, mert látnád, hogy olyasmiket magyarázok, amit te röhögve elintézel egy laza csukló-mozdulattal.

Szóval, a lényeg az hogy ha idáig minden jól sikerült neked, akkor most a rendszered elvileg okés, csak be kell állítanod helyesen a boot szkripteket, aztán...

Aztán ha a **/_/S/Kernel/Boot/grub/menu.lst** fájl is helyesen lett beállítva, akkor ellenőrizd, fel van-e csatolva (mountolva) az épülő rendszered partíciója, és tégy róla, hogy biztosan NE légy belépve a chroot környezetbe! Ezután root-ként a host rendszer alól add ki e parancsot:

```
update-grub
```

Erre a géped módosítja a telepített grub beállításait úgy, hogy abba bekerüljön az új rendszered is. Ekkor reboot, kiválasztod a megfelelő menüpontot, és még az se teljesen kizárt, hogy be fog bootolni a rendszered! Akkor jelentkezz be root-ként, és próbáld ki, működik-e az mc progi!

Ha igen, az nagy siker. Ez esetben folytathatod a rendszerépítést innentől.

==========
==========
==========
==========
==========

7. fejezet: Az XORG, vagyis a grafikus felület

Az X egy úgynevezett „meta-csomag”, sok apró programból meg más apró piszlicsári kis nyavalyából áll össze. Az effélét irtó nagy kínlódás feltelepíteni. Amikor elolvastam, a BLFS Book miként műveli ezt, egészen elszörnyedtem, mert borzasztó nehéznek tartottam, hogy az ottani módszert adaptáljam a magam könyvtárrendszeréhez.

Végül nem is tettem meg; saját módszert fejlesztettem ki. Ennek lényege az, hogy ha az X úgyis több apró csomagból áll, miért is lenne nekem muszáj ezeket egyetlen könyvtárba telepítenem?! Rendszerem eddig követett logikájához úgyis az illik, az konzisztens vele, hogy ha valamit a fejlesztő külön fejleszt, külön csomagnak tart, akkor az nálam is kerüljön külön könyvtárba! Így legalább külön is frissíthetem majd, a többitől függetlenül adott esetben. (Hm, még az se kizárt, hogy ezzel a hozzáállással a disztróm „rolling release” lesz, hehehe...)

Persze a dolognak vannak bökkenői is, ez esetben speciális nehézségekkel kell megküzdenem, mert e fránya csomagok mind meg kell találják egymás állományait! E megközelítést mégis reményteljesebbnek találtam. A nehézségeim egy részét ugyanis a BLFS Book se tudta kikerülni a maga telepítési metódusával, mert látható, hogy szinte minden csomag telepítése után le kell futtassa az ldconfig parancsot!

Mindenekelőtt adjuk ki e parancsot:

```
export X="--sysconfdir=/_/S/O --localstatedir=/_/S/V --disable-static"
```

E fenti sort jó, ha belevesszük a root user $HOME/.bashrc fájljába is.

A telepítési folyamat során a configure szkript sokszor ezt irogatja majd ki:

```
configure: WARNING: unrecognized options: --disable-static
```

de ez nem baj. Ezzel nem kell törődnünk.

Elnevezési konvencióként azt találtam ki, mint már említettem is korábban, hogy mindegyik csomag neve, ami az X része, azzal kezdődik majd, hogy

X--

Azt hogy az X-et alkotó librarykat, header fájlokat, appokat, egyebeket hogyan töltheted le, megtalálod a BLFS Book megfelelő fejezetében, itt nem ismertetem, ennyi kreativitás kell legyen benned, hogy ezt magad is megoldod.

Vágjunk hát bele ebbe is!

```
============================ util-macros-1.17.1

konfig X--Util-Macros 1.17.1 $X
make install
sy X--Util-Macros 1.17.1

==================================== Sudo-1.8.7

konfig Sudo 1.8.7 --with-timedir=/_/S/V/lib/sudo --with-all-insults --with-env-
editor
make
make install
sy Sudo 1.8.7

================================= OpenSSL-1.0.1e

patch -Np1 -i ../openssl-1.0.1e-fix_parallel_build-1.patch &&
patch -Np1 -i ../openssl-1.0.1e-fix_pod_syntax-1.patch
./config --prefix=/_/P/OpenSSL/1.0.1e shared zlib-dynamic
make
make MANDIR=/_/P/OpenSSL/1.0.1e/share/man MANSUFFIX=ssl install
sy OpenSSL 1.0.1e

==================================== Wget-1.14

patch -Np1 -i ../wget-1.14-texi2pod-1.patch
konfig Wget 1.14 --with-ssl=openssl
make
make install
sy Wget 1.14

========================== Xorg Protocol Headers
```

Mindegyiknek a neve azzal fog kezdődni, hogy

```
"X--Proto-"

konfig X--Proto-BigreqsProto 1.1.2 $X
make install
sy X--Proto-BigreqsProto 1.1.2

konfig X--Proto-CompositeProto 0.4.2 $X
make install
sy X--Proto-CompositeProto 0.4.2

konfig X--Proto-DamageProto 1.2.1 $X
```

```
make install
sy X--Proto-DamageProto 1.2.1

konfig X--Proto-DmxProto 2.3.1 $X
make install
sy X--Proto-DmxProto 2.3.1

konfig X--Proto-Dri2Proto 2.8 $X
make install
sy X--Proto-Dri2Proto 2.8

konfig X--Proto-FixesProto 5.0 $X
make install
sy X--Proto-FixesProto 5.0

konfig X--Proto-FontsProto 2.1.2 $X
make install
sy X--Proto-FontsProto 2.1.2

konfig X--Proto-GlProto 1.4.16 $X
make install
sy X--Proto-GlProto 1.4.16

konfig X--Proto-InputProto 2.3 $X
make install
sy X--Proto-InputProto 2.3

konfig X--Proto-KbProto 1.0.6 $X
make install
sy X--Proto-KbProto 1.0.6

konfig X--Proto-RandrProto 1.4.0 $X
make install
sy X--Proto-RandrProto 1.4.0

konfig X--Proto-RecordProto 1.14.2 $X
make install
sy X--Proto-RecordProto 1.14.2

konfig X--Proto-RenderProto 0.11.1 $X
make install
sy X--Proto-RenderProto 0.11.1

konfig X--Proto-ResourceProto 1.2.0 $X
make install
sy X--Proto-ResourceProto 1.2.0

konfig X--Proto-ScrnsaverProto 1.2.2 $X
make install
sy X--Proto-ScrnsaverProto 1.2.2

konfig X--Proto-VideoProto 2.3.2 $X
make install
sy X--Proto-VideoProto 2.3.2

konfig X--Proto-XcmiscProto 1.2.2 $X
make install
sy X--Proto-XcmiscProto 1.2.2

konfig X--Proto-XextProto 7.2.1 $X
make install
sy X--Proto-XextProto 7.2.1
```

```
konfig X--Proto-Xf86BigFontProto 1.2.0 $X
make install
sy X--Proto-Xf86BigFontProto 1.2.0

konfig X--Proto-Xf86dgaProto 2.1 $X
make install
sy X--Proto-Xf86dgaProto 2.1

konfig X--Proto-Xf86driProto 2.1.1 $X
make install
sy X--Proto-Xf86driProto 2.1.1

konfig X--Proto-Xf86VidModeProto 2.3.1 $X
make install
sy X--Proto-Xf86VidModeProto 2.3.1

konfig X--Proto-XineramaProto 1.2.1 $X
make install
sy X--Proto-XineramaProto 1.2.1

konfig X--Proto-XProto 7.0.24 $X
make install
sy X--Proto-XProto 7.0.24

============================ makedepend-1.0.5

konfig MakeDepend 1.0.5 $X
make
make install
sy MakeDepend 1.0.5

================================ libXau-1.0.8

konfig X--Lib-LibXau 1.0.8 $X
make
make install
sy X--Lib-LibXau 1.0.8

============================ libXdmcp-1.1.1

konfig X--Lib-LibXdmcp 1.1.1 $X
make
make install
sy X--Lib-LibXdmcp 1.1.1

========================== xcb-proto-1.8

konfig Xcb-Proto 1.8 $X
make install
sy Xcb-Proto 1.8

============================== libxml2-2.9.1

konfig LibXML2 2.9.1 --disable-static --with-history --with-python --with-lzma
--with-readline

make
make install
sy LibXML2 2.9.1

============================ libxslt-1.1.28
```

```
konfig LibXSLT 1.1.28 --disable-static
make
make install
sy LibXSLT 1.1.28

========================== libxcb-1.9.1

patch -Np1 -i ../libxcb-1.9.1-automake_bug-1.patch
sed -e "s/pthread-stubs//" -i configure.ac

autoreconf -fi

konfig X--Lib-LibXCB 1.9.1 $X --enable-xinput --enable-xkb

make
make install
sy X--Lib-LibXCB 1.9.1

============================== libpng-1.6.4

gzip -cd ../libpng-1.6.3-apng.patch.gz | patch -p1
konfig LibPNG 1.6.4 --disable-static
make
make check
make install
sy LibPNG 1.6.4

======================== Which-2.20

konfig Which 2.20
make
make install
sy Which 2.20

========================== FreeType-2.5.0.1

sed -i  -e "/AUX.*.gxvalid/s@^# @@" \
        -e "/AUX.*.otvalid/s@^# @@" \
        modules.cfg

sed -ri -e 's:.*(#.*SUBPIXEL.*) .*:\1:' \
        include/freetype/config/ftoption.h

konfig FreeType 2.5.0.1 --disable-static

make
make install
sy FreeType 2.5.0.1

============================== Fontconfig-2.10.93

konfig FontConfig 2.10.93 --localstatedir=/_/S/V
--docdir=/_/P/FontConfig/2.10.93/share/doc --disable-docs --disable-static
make
make check
make install
sy FontConfig 2.10.93

============ Xorg Libraries

ldconfig -v
```

Mindegyik X library neve úgy fog kezdődni, hogy: X--Lib-

Ezeket MUSZÁJ **éppen pontosan ebben a sorrendben feltenni**, különben nem fog sikerülni!

```
konfig X--Lib-Xtrans 1.2.7 $X
make
make install
sy X--Lib-Xtrans 1.2.7
ldconfig -v

konfig X--Lib-LibX11 1.6.2 $X  --enable-loadable-i18n
make
make install
sy X--Lib-LibX11 1.6.2
ldconfig -v

konfig X--Lib-LibXext 1.3.2 $X
make
make install
sy X--Lib-LibXext 1.3.2
ldconfig -v

konfig X--Lib-LibFS 1.0.5 $X
make
make install
sy X--Lib-LibFS 1.0.5
ldconfig -v

konfig X--Lib-LibICE 1.0.8 $X
make
make install
sy X--Lib-LibICE 1.0.8
ldconfig -v

konfig X--Lib-LibSM 1.2.2 $X
make
make install
sy X--Lib-LibSM 1.2.2
ldconfig -v

konfig X--Lib-LibXscrnSaver 1.2.2 $X
make
make install
sy X--Lib-LibXscrnSaver 1.2.2
ldconfig -v

konfig X--Lib-LibXt 1.1.4 $X --with-appdefaultdir=/_/S/O/X11/app-defaults
make
make install
sy X--Lib-LibXt 1.1.4
ldconfig -v

konfig X--Lib-LibXmu 1.1.2 $X
make
make install
sy X--Lib-LibXmu 1.1.2
ldconfig -v

konfig X--Lib-LibXpm 3.5.11 $X
make
```

```
make install
sy X--Lib-LibXpm 3.5.11
ldconfig -v

konfig X--Lib-LibXaw 1.0.12 $X
make
make install
sy X--Lib-LibXaw 1.0.12
ldconfig -v

konfig X--Lib-LibXfixes 5.0.1 $X
make
make install
sy X--Lib-LibXfixes 5.0.1
ldconfig -v

konfig X--Lib-LibXcomposite 0.4.4 $X
make
make install
sy X--Lib-LibXcomposite 0.4.4
ldconfig -v

konfig X--Lib-LibXrender 0.9.8 $X
make
make install
sy X--Lib-LibXrender 0.9.8
ldconfig -v

konfig X--Lib-LibXcursor 1.1.14 $X
make
make install
sy X--Lib-LibXcursor 1.1.14
ldconfig -v

konfig X--Lib-LibXdamage 1.1.4 $X
make
make install
sy X--Lib-LibXdamage 1.1.4
ldconfig -v

konfig X--Lib-LibFontEnc 1.1.2 $X
make
make install
sy X--Lib-LibFontEnc 1.1.2
ldconfig -v

konfig X--Lib-LibXfont 1.4.6 $X --disable-devel-docs
make
make install
sy X--Lib-LibXfont 1.4.6
ldconfig -v

konfig X--Lib-LibXft 2.3.1 $X
make
make install
sy X--Lib-LibXft 2.3.1
ldconfig -v

konfig X--Lib-LibXi 1.7.2 $X
make
make install
sy X--Lib-LibXi 1.7.2
```

```
ldconfig -v

konfig X--Lib-LibXinerama 1.1.3 $X
make
make install
sy X--Lib-LibXinerama 1.1.3
ldconfig -v

konfig X--Lib-LibXrandr 1.4.2 $X
make
make install
sy X--Lib-LibXrandr 1.4.2
ldconfig -v

konfig X--Lib-LibXres 1.0.7 $X
make
make install
sy X--Lib-LibXres 1.0.7
ldconfig -v

konfig X--Lib-LibXtst 1.2.2 $X
make
make install
sy X--Lib-LibXtst 1.2.2
ldconfig -v

konfig X--Lib-LibXv 1.0.10 $X
make
make install
sy X--Lib-LibXv 1.0.10
ldconfig -v

konfig X--Lib-LibXvMC 1.0.8 $X
make
make install
sy X--Lib-LibXvMC 1.0.8
ldconfig -v

konfig X--Lib-LibXxf86dga 1.1.4 $X
make
make install
sy X--Lib-LibXxf86dga 1.1.4
ldconfig -v

konfig X--Lib-LibXxf86vm 1.1.3 $X
make
make install
sy X--Lib-LibXxf86vm 1.1.3
ldconfig -v

konfig X--Lib-Libdmx 1.1.3 $X
make
make install
sy X--Lib-Libdmx 1.1.3
ldconfig -v

konfig X--Lib-LibPciAccess 0.13.2 $X
make
make install
sy X--Lib-LibPciAccess 0.13.2
ldconfig -v
```

```
konfig X--Lib-LibXkbFile 1.0.8 $X
make
make install
sy X--Lib-LibXkbFile 1.0.8
ldconfig -v

========================= xcb-util-0.3.9

konfig X--Xcb-Util 0.3.9 $X
make
make install
sy X--Xcb-Util 0.3.9

==================== xcb-util-image-0.3.9

konfig X--Xcb-Util-Image 0.3.9 $X
make
make install
sy X--Xcb-Util-Image 0.3.9

====================== xcb-util-keysyms-0.3.9

konfig X--Xcb-Util-KeySyms 0.3.9 $X
make
make install
sy X--Xcb-Util-KeySyms 0.3.9

====================== xcb-util-renderutil-0.3.8

konfig X--Xcb-Util-RenderUtil 0.3.8 $X
make
make install
sy X--Xcb-Util-RenderUtil 0.3.8

==================== xcb-util-wm-0.3.9

konfig X--Xcb-Util-WM 0.3.9 $X
make
make install
sy X--Xcb-Util-WM 0.3.9

============================ libdrm-2.4.46

sed -e "/pthread-stubs/d" -i configure.ac
autoreconf -fi
konfig LibDRM 2.4.46 --enable-udev
make
make install
sy LibDRM 2.4.46

========================= elfutils-0.156

konfig ElfUtils 0.156 --program-prefix="eu-"
make
make check
make install
sy ElfUtils 0.156

============================ LLVM-3.3
```

Na ez nagyon szopatós progi...

Készítsünk symlinkeket a
/_/P/GCC/-/include/c++/4.8.1/ könyvtár minden fájljáról
a
/_/S/L/H könyvtárba!

valamint a
/_/P/GCC/4.8.1/include/c++/4.8.1/x86_64-unknown-linux-gnu/bits
meg a
/_/P/GCC/4.8.1/include/c++/4.8.1/bits
könyvtár minden .h fájljáról
a
/_/S/L/H/bits könyvtárba!

A /_/P/GCC/4.8.1/include/c++/4.8.1/ext könyvtárat symlinkeljük be a
/_/S/L/H könyvtárba!
Meg ezt a directoryt is onnan:

debug

```
tar -xf ../cfe-3.3.src.tar.gz -C tools
tar -xf ../compiler-rt-3.3.src.tar.gz -C projects
mv tools/cfe-3.3.src tools/clang
mv projects/compiler-rt-3.3.src projects/compiler-rt

sed -e 's:\$(PROJ_prefix)/docs/llvm:$(PROJ_prefix)/share/doc/llvm-3.3:' \
    -i Makefile.config.in


CC=gcc CXX=g++ konfig LLVM 3.3 --enable-libffi --enable-optimized --enable-
shared --enable-targets=all --disable-assertions  --disable-debug-runtime
--disable-expensive-checks --enable-experimental-targets=R600

make
make install

for file in /_/P/LLVM/3.3/lib/lib{clang,LLVM,LTO,profile_rt}*.a
do
  test -f $file && chmod -v 644 $file
done

install -v -dm755 /_/P/LLVM/3.3/lib/clang-analyzer

for prog in scan-build scan-view
do
  cp -rfv tools/clang/tools/$prog /_/P/LLVM/3.3/lib/clang-analyzer/
done
mv -v /_/P/LLVM/3.3/lib/clang-analyzer/scan-build/scan-build.1
/_/P/LLVM/3.3/share/man/man1/
```

A /_/P/LLVM/3/lib/clang-analyzer/scan-build könyvtárban ezen állományokról:

```
c++-analyzer
ccc-analyzer
scan-build
set-xcode-analyzer
```

készítsünk symlinkeket a /_/P/LLVM/3.3/bin könyvtárba!

A /_/P/LLVM/3/lib/clang-analyzer/scan-view/scan-view progiról is készítsünk symlinket a /_/P/LLVM/3.3/bin könyvtárba!

```
sy LLVM 3.3

================================ libvdpau-0.7

konfig LibVDPAU 0.7 $X
make
make install
sy LibVDPAU 0.7

======================== MesaLib-9.2.0
```

A /_/S/L/E könyvtárban:

```
ln -s automake-1.14 automake-1.12
ln -s aclocal-1.14 aclocal-1.12
```

A /_/S/L/Y/pkgconfig könyvtár ezen fájljairól:
libudev.pc
udev.pc
csináljunk symlinkeket a /_/S/L/C könyvtárba!

```
autoreconf -fi

konfig Mesa 9.2.0 CFLAGS="-O2" CXXFLAGS="-O2" $X --enable-texture-float
--enable-gles1 --enable-gles2 --enable-openvg --enable-osmesa  --enable-xa
--enable-gbm --enable-gallium-egl --enable-gallium-gbm --enable-glx-tls --with-
llvm-shared-libs  --with-egl-platforms="drm,x11" --with-gallium-
drivers="nouveau,r300,r600,radeonsi,svga,swrast"

make
make install
sy Mesa 9.2.0

============================ GLU 9.0.0

konfig GLU 9.0.0 $X --disable-static
make
make install
sy GLU 9.0.0

======================== xbitmaps-1.1.1

konfig X--Xbitmaps 1.1.1 $X
make install
sy X--Xbitmaps 1.1.1

=========================================================
```

Jönnek az Xorg applications-ok! Mindegyiknek a neve úgy kezdődik majd, hogy: X--App-

```
konfig X--App-bdftopcf 1.0.4 $X
make
make install
sy X--App-bdftopcf 1.0.4

konfig X--App-iceauth 1.0.6 $X
make
```

```
make install
sy X--App-iceauth 1.0.6

konfig X--App-luit 1.1.1 $X
make
make install
sy X--App-luit 1.1.1

konfig X--App-mkfontdir 1.0.7 $X
make
make install
sy X--App-mkfontdir 1.0.7

konfig X--App-mkfontscale 1.1.1 $X
make
make install
sy X--App-mkfontscale 1.1.1

konfig X--App-sessreg 1.0.8 $X
make
make install
sy X--App-sessreg 1.0.8

konfig X--App-setxkbmap 1.3.0 $X
make
make install
sy X--App-setxkbmap 1.3.0

konfig X--App-smproxy 1.0.5 $X
make
make install
sy X--App-smproxy 1.0.5

konfig X--App-x11perf 1.5.4 $X
make
make install
sy X--App-x11perf 1.5.4

konfig X--App-xauth 1.0.7 $X
make
make install
sy X--App-xauth 1.0.7

konfig X--App-xbacklight 1.2.0 $X
make
make install
sy X--App-xbacklight 1.2.0

konfig X--App-xcmsdb 1.0.4 $X
make
make install
sy X--App-xcmsdb 1.0.4

konfig X--App-xcursorgen 1.0.5 $X
make
make install
sy X--App-xcursorgen 1.0.5

konfig X--App-xdpyinfo 1.3.1 $X
make
make install
sy X--App-xdpyinfo 1.3.1
```

```
konfig X--App-xdriinfo 1.0.4 $X
make
make install
sy X--App-xdriinfo 1.0.4

konfig X--App-xev 1.2.1 $X
make
make install
sy X--App-xev 1.2.1

konfig X--App-xgamma 1.0.5 $X
make
make install
sy X--App-xgamma 1.0.5

konfig X--App-xhost 1.0.6 $X
make
make install
sy X--App-xhost 1.0.6

konfig X--App-xinput 1.6.0 $X
make
make install
sy X--App-xinput 1.6.0

konfig X--App-xkbcomp 1.2.4 $X
make
make install
sy X--App-xkbcomp 1.2.4

konfig X--App-xkbevd 1.1.3 $X
make
make install
sy X--App-xkbevd 1.1.3

konfig X--App-xkbutils 1.0.4 $X
make
make install
sy X--App-xkbutils 1.0.4

konfig X--App-xkill 1.0.4 $X
make
make install
sy X--App-xkill 1.0.4

konfig X--App-xlsatoms 1.1.1 $X
make
make install
sy X--App-xlsatoms 1.1.1

konfig X--App-xlsclients 1.1.3 $X
make
make install
sy X--App-xlsclients 1.1.3

konfig X--App-xmessage 1.0.4 $X
make
make install
sy X--App-xmessage 1.0.4

konfig X--App-xmodmap 1.0.8 $X
```

```
make
make install
sy X--App-xmodmap 1.0.8

konfig X--App-xpr 1.0.4 $X
make
make install
sy X--App-xpr 1.0.4

konfig X--App-xprop 1.2.2 $X
make
make install
sy X--App-xprop 1.2.2

konfig X--App-xrandr 1.4.1 $X
make
make install
sy X--App-xrandr 1.4.1

konfig X--App-xrdb 1.1.0 $X
make
make install
sy X--App-xrdb 1.1.0

konfig X--App-xrefresh 1.0.5 $X
make
make install
sy X--App-xrefresh 1.0.5

konfig X--App-xset 1.2.3 $X
make
make install
sy X--App-xset 1.2.3

konfig X--App-xsetroot 1.1.1 $X
make
make install
sy X--App-xsetroot 1.1.1

konfig X--App-xvinfo 1.1.2 $X
make
make install
sy X--App-xvinfo 1.1.2

konfig X--App-xwd 1.0.6 $X
make
make install
sy X--App-xwd 1.0.6

konfig X--App-xwininfo 1.1.3 $X
make
make install
sy X--App-xwininfo 1.1.3

konfig X--App-xwud 1.0.4 $X
make
make install
sy X--App-xwud 1.0.4

---------------------- xcursor-themes-1.0.4

konfig X--XCursor-Themes 1.0.4 $X
```

```
make
make install
```

Hozzuk létre a /_/P/X--XCursor-Themes/1.0.4/share könyvtárat, s ide mozgassuk át a /_/S/L/T/icons könyvtárat!
majd:

```
sy X--XCursor-Themes 1.0.4
```

```
------------------------- Xorg Fonts
```

Csináljunk egy fontok nevű könyvtárat, másoljuk bele a font állományokat, a fontok szülőkönyvtárába pedig a font-7.7.md5 fájlt!

Ezután a fontok könyvtárban állva:

```
for package in $(grep -v '^#' ../font-7.7.md5 | awk '{print $2}')
do
  packagedir=${package%.tar.bz2}
  tar -xf $package
  pushd $packagedir
  ./configure --prefix=/_/P/XORG-Fonts/7.7 $X
  make
  make install
  popd
  rm -rf $packagedir
done
```

A /_/P/XORG-Fonts/7.7/share/X11 könyvtár alkönyvtáraiból a fontokat mozgassuk át a /_/F/X11 könyvtár megfelelő könyvtáraiba.
Töröljük a /_/P/XORG-Fonts/7.7/share/fonts könyvtárat!

```
sy XORG-Fonts 7.7
```

```
-------------------- Perl-modulok
```

Telepítünk 2 PERL-modult:

URI-1.60
és
XML-Parser-2.41

A telepítésük:

Kicsomagoljuk őket, majd a könyvtárukban:

```
perl Makefile.PL
make
make test
make install
```

```
------------------ Intltool-0.50.2
```

```
konfig IntlTool 0.50.2
make
make install
install -v -m644 -D doc/I18N-HOWTO /_/P/IntlTool/0.50.2/share/doc/intltool-
0.50.2/I18N-HOWTO
sy IntlTool 0.50.2
```

```
------------------- XKeyboardConfig-2.9
```

```
konfig X--XKeyboardConfig 2.9 $X --with-xkb-rules-symlink=xorg
make
make install
sy X--XKeyboardConfig 2.9

---------------------- Pixman-0.30.2

konfig Pixman 0.30.2 --disable-static
make
make install
sy Pixman 0.30.2

---------------------- acpid-2.0.19

konfig Acpid 2.0.19
make
make install
install -v -m755 -d /_/S/O/acpi/events
cp -r samples /_/P/Acpid/2.0.19/share/doc/acpid-2.0.19
sy Acpid 2.0.19

======================= Xorg-Server-1.14.3

patch -Np1 -i ../xorg-server-1.14.3-add_prime_support-1.patch

konfig X--XORG-Server 1.14.3 $X --with-xkb-output=/_/S/V/lib/xkb --enable-
install-setuid
make
make check
make install
sy X--XORG-Server 1.14.3
mkdir -pv /_/S/O/X11/xorg.conf.d

cat >> /_/S/O/sysconfig/createfiles << "EOF"
/tmp/.ICE-unix dir 1777 root root
/tmp/.X11-unix dir 1777 root root
EOF

------------------------- xorg driverek

konfig X--Driver-MtDev 1.1.4 --disable-static
make
make install
sy X--Driver-MtDev 1.1.4

konfig X--Driver-Xf86-Input-EvDev 2.8.1 $X
make
make install
sy X--Driver-Xf86-Input-EvDev 2.8.1

Xorg Fbdev Driver-0.4.3
sed -i -e "/mibstore.h/d" -e "/miInitializeBackingStore/d" src/fbdev.c
konfig X--Driver-Xf86-video-FBdev 0.4.3 $X
make
make install
sy X--Driver-Xf86-video-FBdev 0.4.3

Xorg Intel Driver-2.21.15
konfig X--Driver-Xf86-Video-Intel 2.21.15 $X --enable-kms-only --with-default-
accel=sna
make
```

```
make install
sy X--Driver-Xf86-Video-Intel 2.21.15

Xorg Nouveau Driver-1.0.9
konfig X--Driver-Xf86-Video-Nouveau 1.0.9 $X
make
make install
sy X--Driver-Xf86-Video-Nouveau 1.0.9

Xorg VESA Driver-2.3.3
konfig X--Driver-Xf86-Video-Vesa 2.3.3 $X
make
make install
sy X--Driver-Xf86-Video-Vesa 2.3.3

Xorg VMware Driver-13.0.1
konfig X--Driver-Xf86-Video-VMware 13.0.1 $X
make
make install
sy X--Driver-Xf86-Video-VMware 13.0.1

libva-1.2.1
konfig X--Driver-LibVA 1.2.1 $X
make
make install
sy X--Driver-LibVA 1.2.1

konfig X--Driver-LibVA-Intel-Driver 1.2.0 $X
make
make install
```

töröljük le a /_/P/X--Driver-LibVA/- symlinket, és a

```
/_/P/X--Driver-LibVA/-/bin
/_/P/X--Driver-LibVA/-/lib
/_/P/X--Driver-LibVA/-/include
```

symlinkeket,
a /_/S/L/Y/dri könyvtárból e két állományt:

```
i965_drv_video.la
i965_drv_video.so
```

mozgassuk át a

```
/_/P/X--Driver-LibVA/1.2.1/xlib/dri
```

könyvtárba, majd:

```
sy X--Driver-LibVA 1.2.1

=============== printproto-1.0.5

konfig X--Proto-PrintProto 1.0.5 $X
make
make install
sy X--Proto-PrintProto 1.0.5

================ libXp-1.0.2

konfig X--Lib-LibXp 1.0.2 $X
make
make install
```

```
sy X--Lib-LibXp 1.0.2

================== twm-1.0.8

sed -i -e '/^rcdir =/s,^\(rcdir = \).*,\1/etc/X11/app-defaults,' src/Makefile.in
konfig TWM 1.0.8 $X
make
make install
sy TWM 1.0.8

====================== xterm-297

sed -i '/v0/,+1s/new:/new:kb=^?:/' termcap &&
echo -e '\tkbs=\\177,' >>terminfo
TERMINFO=/_/S/L/T/terminfo konfig XTerm 297 $X --with-app-
defaults=/_/S/O/X11/app-defaults
make
make install
make install-ti
sy XTerm 297

cat >> /_/S/O/X11/app-defaults/XTerm << "EOF"
*VT100*locale: true
*VT100*faceName: Monospace
*VT100*faceSize: 10
*backarrowKeyIsErase: true
*ptyInitialErase: true
EOF

====================== xclock-1.0.7

konfig XClock 1.0.7 $X
make
make install
sy  XClock 1.0.7

================================ xinit-1.3.3

konfig X--Xinit 1.3.3 $X --with-xinitdir=/_/S/O/X11/app-defaults
make
make install
sy X--Xinit 1.3.3

============================ rxvt-unicode-9.15

konfig Rxvt-Unicode 9.15  --enable-256-color --enable-xft --enable-font-styles
--enable-rxvt-scroll --enable-perl --enable-selectionscrolling

make
make install
sy Rxvt-Unicode 9.15

cat >> /_/S/O/X11/app-defaults/URxvt << "EOF"
URxvt*perl-ext: matcher
URxvt*urlLauncher: firefox
URxvt.background: black
URxvt.foreground: green
URxvt*font: xft:Monospace:pixelsize=12
EOF

============================================
```

Újraforgatom az MC-t, hogy az X támogatása is benne legyen.
Ez egyszerűen úgy történik, hogy letörlöm a régit, mondjuk így:

```
rm -rf /_/P/MC
```

majd:

```
BROKEN /_/S/L --delete
```

s ezután pont úgy járok el, ahogy azt e doksi korábbi részében az MC telepítésekor leírtam.

```
====================== libunistring-0.9.3

konfig LibUniString 0.9.3
make
make install
sy LibUniString 0.9.3

================================== GC-7.2d

sed -i 's#AM_CONFIG_HEADER#AC_CONFIG_HEADERS#' configure.ac &&
sed -i 's#AM_CONFIG_HEADER#AC_CONFIG_HEADERS#' libatomic_ops/configure.ac &&
sed -i 's#pkgdata#doc#' doc/doc.am
autoreconf -fi
konfig GC 7.2 --enable-cplusplus --disable-static
make
make check
make install
mkdir -p /_/P/GC/7.2/share/man/man3
install -v -m644 doc/gc.man /_/P/GC/7.2/share/man/man3/gc_malloc.3
sy GC 7.2

======================= Guile-2.0.9

konfig Guile 2.0.9  --disable-static
make
make html
makeinfo --plaintext -o doc/r5rs/r5rs.txt doc/r5rs/r5rs.texi
makeinfo --plaintext -o doc/ref/guile.txt doc/ref/guile.texi
make install
sy Guile 2.0.9

================================
fc-cache -v -r
================================ Tcl-8.6.0

cd unix
konfig Tcl 8.6.0 --without-tzdata --mandir=/_/P/Tcl/8.6.0/share/man $([ $(uname
-m) = x86_64 ] && echo --enable-64bit)
make

sed -e "s@^\(TCL_SRC_DIR='\).*@\1/usr/include'@" \
    -e "/TCL_B/s@='\(-L\)\?.*unix@='\1/usr/lib@" \
    -i tclConfig.sh

make test
make install
make install-private-headers
```

A /_/P/Tcl/8.6.0/bin könyvtárban:

```
ln -s tclsh8.6 tclsh

chmod -v 755 /_/P/Tcl/8.6.0/lib/libtcl8.6.so
sy Tcl 8.6.0

============================== Tk-8.6.0

cd unix

konfig Tk 8.6.0 --mandir=/_/P/Tk/8.6.0/share/man $([ $(uname -m) = x86_64 ] &&
echo --enable-64bit)

make

sed -e "s@^\(TK_SRC_DIR='\).*@\1/usr/include'@" \
    -e "/TK_B/s@='\(-L\)\?.*unix@='\1/usr/lib@" \
    -i tkConfig.sh

make install
make install-private-headers
```

A /_/P/Tk/8.6.0/bin könyvtárban állva:

```
ln -s wish8.6 wish

chmod -v 755 /_/P/Tk/8.6.0/lib/libtk8.6.so
sy Tk 8.6.0

======================== xbindkeys-1.8.5

konfig XBindKeys 1.8.5
make
make install
sy XBindKeys 1.8.5

========================== DWM 6.0
```

A config.mk fájl elején így állítsuk be a változókat:

```
PREFIX = /_/P/DWM/6.0
MANPREFIX = ${PREFIX}/share/man

X11INC = /_/S/L/H/X11
X11LIB = /_/S/L/Y/X11

# Xinerama
XINERAMALIBS = -L${X11LIB} -lXinerama
XINERAMAFLAGS = -DXINERAMA

# includes and libs
INCS = -I. -I/_/S/L/H -I${X11INC}
LIBS = -L/_/S/L/Y -lc -L${X11LIB} -lX11 ${XINERAMALIBS}
```

Ezután:

```
make clean install

sy DWM 6.0
```

```
============================== Menu 0.0 (saját programom)

mkdir -p /_/P/Menu/0.0/bin
mkdir /_/P/Menu/0.0/sources

gcc menu.c -o menu
cp menu.c /_/P/Menu/0.0/sources
cp menu /_/P/Menu/0.0/bin
sy Menu 0.0

============================== dmenu 4.5
```

A config.mk fájlban a változókat így állítsuk be:

```
PREFIX = /_/P/DMENU/4.5
MANPREFIX = ${PREFIX}/share/man

X11INC = /_/S/L/H/X11
X11LIB = /_/S/L/Y/X11
```

majd:

```
make clean install

sy DMENU 4.5

======================================================================
======================================================================
======================================================================
```

Eddig tartott az Xorg felrakása. Elvileg ezek után van grafikus felületed, DWM ablakkezelővel. Sőt van Xbindkeys progid is, ami nagyon hasznos. Ahhoz, hogyan kell beállítani az X-et, hogy a DWM legyen az ablakkezelőd, s hogyan konfigolhatod azt agyon, hogy neked a legmegfelelőbb legyen, afelől tájékozódj az itt található MAGYAR NYELVŰ cikkekből:

http://parancssor.info/dokuwiki/doku.php?id=dwm

```
======================================================================
======================================================================
======================================================================
```

8. fejezet: Mindenféle egyéb programok felrakása

```
======================== alsa-lib-1.0.27.2

konfig Alsa-Lib 1.0.27.2
make
make install
sy Alsa-Lib 1.0.27.2

====================== ICU-51.2

cd source
CPPFLAGS="-I/_/S/L/H/c++/4.8.1 -I/_/S/L/H/c++/4.8.1/i686-pc-linux-gnu" konfig
ICU 51.2
make
make check
```

```
make install
sy ICU 51.2

==================== Wireless Tools-29
Wireless Tools-29

make
make PREFIX=/_/P/WirelessTools/29 INSTALL_MAN=/_/P/WirelessTools/29/share/man
install
sy WirelessTools 29

========================= Cairo-1.12.16

konfig Cairo 1.12.16 --disable-static --enable-xlib-xcb --enable-xml
make
make install
sy Cairo 1.12.16

========================== Harfbuzz-0.9.20

konfig Harfbuzz 0.9.20
make
make install
sy Harfbuzz 0.9.20

================== Pango-1.34.1

konfig Pango 1.34.1
make
make install
sy Pango 1.34.1
pango-querymodules --update-cache

============================== Pangox-compat 0.0.2

konfig Pangox-compat 0.0.2 --disable-static
make
make install
sy Pangox-compat 0.0.2

=================== NASM-2.10.09

konfig NASM 2.10.09
make
make install
sy NASM 2.10.09

===================== libjpeg-turbo-1.3.0

konfig LibJPEG-Turbo 1.3.0 --with-jpeg8 --disable-static
make
make install
sy LibJPEG-Turbo 1.3.0

====================== LibTIFF-4.0.3

konfig Tiff 4.0.3 --disable-static
make
make install
sy Tiff 4.0.3

=================== gdk-pixbuf-2.28.2
```

```
konfig Gdk-Pixbuf 2.28.2 --with-x11
make
make check
make install
sy Gdk-Pixbuf 2.28.2
gdk-pixbuf-query-loaders --update-cache

============================ ATK-2.8.0

konfig Atk 2.8.0
make
make install
sy Atk 2.8.0

=================== hicolor-icon-theme-0.12

konfig Hicolor-Icon-Theme 0.12
make install
sy Hicolor-Icon-Theme 0.12

================================ GTK+-2.24.20

sed -i 's#l \(gtk-.*\).sgml#& -o \1#' docs/{faq,tutorial}/Makefile.in &&
sed -i 's#.*@man_#man_#' docs/reference/gtk/Makefile.in

konfig GTK2 2.24.20
make
make install
sy GTK2 2.24.20
gtk-query-immodules-2.0 --update-cache

================================ D-Bus-1.6.14

groupadd -g 18 messagebus &&
useradd -c "D-Bus Message Daemon User" -d /var/run/dbus \
        -u 18 -g messagebus -s /bin/false messagebus

konfig D-BUS 1.6.14 --localstatedir=/_/S/V --with-console-auth-dir=/run/console/
--without-systemdsystemunitdir --disable-systemd --disable-static

make
make install

cat > /_/P/D-BUS/1.6.14/etc/dbus-1/session-local.conf << "EOF"
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>

  <!-- Search for .service files in /usr/local -->
  <servicedir>/usr/local/share/dbus-1/services</servicedir>

</busconfig>
EOF

sy D-BUS 1.6.14

dbus-uuidgen --ensure


============================ Cups-1.6.3
```

```
useradd -c "Print Service User" -d /var/spool/cups -g lp -s /bin/false -u 9 lp
groupadd -g 19 lpadmin
sed -i 's#@CUPS_HTMLVIEW@#firefox#' desktop/cups.desktop.in

patch -Np1 -i ../cups-1.6.3-blfs-1.patch

aclocal -I config-scripts &&
autoconf -I config-scripts

CC=gcc konfig CUPS 1.6.3 --with-rcdir=/_/S/V/tmp/cupsinit --with-system-
groups=lpadmin

make
make install
rm -rf /tmp/cupsinit
echo "ServerName /_/S/V/run/cups/cups.sock" >
/_/P/CUPS/1.6.3/etc/cups/client.conf

sy CUPS 1.6.3

==================================  LibreOffice
```

Most már lehet telepíteni a LibreOffice-ot! Bevallom, ezt nem forrásból fordítottam, hanem úgy jártam el, ahogy azt itt leírtam még régen: http://parancssor.info/dokuwiki/doku.php?id=libreoffice:libreoffice_telepitese_tetszoleges_koenyvtarba

azaz a hivatalos binárist raktam fel kicsit meghackelve. Persze értelemszerűen változtattam az ott leírtakon, nálam a /_/P/LibreOffice/verziószám könyvtárba került, a binárisait egy itt kreált bin nevű könyvtárba symlinkeltem, majd az egész LibreOffice-ot besymlinkeltettem a rendszerbe a sy szkripttel.

```
========================== dbus-glib-0.100.2

konfig DBus-GLib 0.100.2 --disable-static
make
make install
sy DBus-GLib 0.100.2

============================ dhcpcd-6.0.5

konfig DHCPCD 6.0.5 --dbdir=/run
make
make install

sed -i "s;/var/lib;/run;g" dhcpcd-hooks/50-dhcpcd-compat &&
install -v -m 644 dhcpcd-hooks/50-dhcpcd-compat
/_/P/DHCPCD/6.0.5/libexec/dhcpcd-hooks/

sy DHCPCD 6.0.5
```

A blfs-bootscripts csomagban állva:

```
make install-service-dhcpcd

cat > /etc/sysconfig/ifconfig.eth0 << "EOF"
ONBOOT="yes"
IFACE="eth0"
SERVICE="dhcpcd"
```

```
DHCP_START="-b -q"
DHCP_STOP="-k"
EOF

====================== Net-tools-CVS_20101030
```

A Makefile fájlban e sort:

```
BASEDIR ?= $(DESTDIR)
```

írjuk át erre:

```
BASEDIR = /_/P/Net-Tools/CVS_20101030
```

Ezután:

```
sed -i -e '/Token/s/y$/n/'        config.in
sed -i -e '/HAVE_HWSTRIP/s/y$/n/' config.in
yes "" | make config

make

make update

sy Net-Tools CVS_20101030

========================= libevent-2.0.21

konfig LibEvent 2.0.21 --disable-static

make
make install

sy LibEvent 2.0.21

====================== yasm-1.2.0

sed -i 's#) ytasm.*#)#' Makefile.in

konfig Yasm 1.2.0

make
make install
sy Yasm 1.2.0

============================ NSPR-4.10

cd nspr
sed -ri 's#^(RELEASE_BINS =).*#\1#' pr/src/misc/Makefile.in
sed -i 's#$(LIBRARY) ##' config/rules.mk

konfig NSPR 4.10 --with-mozilla --with-pthreads $([ $(uname -m) = x86_64 ] &&
echo --enable-64bit)

make
make install
sy NSPR 4.10
====================== libvpx-v1.2.0

mkdir ../libvpx-build
cd ../libvpx-build
```

```
../libvpx-v1.2.0/configure --prefix=/_/P/LibVPX/v1.2.0 --enable-shared
--disable-static

make

make install

sy LibVPX v1.2.0
================================ Firefox-23.0.1

cat > mozconfig << "EOF"
# If you have a multicore machine, firefox will now use all the cores by
# default. Exceptionally, you can reduce the number of cores, e.g. to 1,
# by uncommenting the next line and setting a valid number of CPU cores.
#mk_add_options MOZ_MAKE_FLAGS="-j1"

# If you have installed DBus-Glib comment out this line:
# ac_add_options --disable-dbus

# If you have installed wireless-tools comment out this line:
# ac_add_options --disable-necko-wifi

# If you have installed libnotify comment out this line:
ac_add_options --disable-libnotify

# Uncomment these lines if you have installed optional dependencies:
# GStreamer is necessary for H.264 video playback in HTML5 Video Player
#ac_add_options --enable-gstreamer
#ac_add_options --enable-system-hunspell
#ac_add_options --enable-startup-notification

# Uncomment this line if you compiled Cairo with --enable-tee switch and want
# to use it instead of the bundled one:
#ac_add_options --enable-system-cairo

# If you have not installed Yasm then uncomment this line:
#ac_add_options --disable-webm

# If you have installed xulrunner uncomment the next two ac_add_options lines
# and check that the sdk will be set by running pkg-config in a subshell
# and has not become hardcoded or empty when you created this file
#ac_add_options --with-system-libxul
#ac_add_options --with-libxul-sdk=$(pkg-config --variable=sdkdir libxul)

# Comment out following options if you have not installed
# recommended dependencies:
#ac_add_options --enable-system-sqlite
ac_add_options --with-system-libevent
ac_add_options --with-system-libvpx
ac_add_options --with-system-nspr
#ac_add_options --with-system-nss

# It is recommended not to touch anything below this line
ac_add_options --prefix=/_/P/Firefox/23.0.1
ac_add_options --enable-application=browser

ac_add_options --disable-crashreporter
ac_add_options --disable-installer
ac_add_options --disable-updater
ac_add_options --disable-debug
ac_add_options --disable-tests
```

```
ac_add_options --enable-optimize
ac_add_options --enable-strip
ac_add_options --enable-install-strip

ac_add_options --enable-gio
ac_add_options --enable-official-branding
ac_add_options --enable-safe-browsing
ac_add_options --enable-url-classifier

ac_add_options --enable-system-ffi
ac_add_options --enable-system-pixman

ac_add_options --with-pthreads

ac_add_options --with-system-bz2
ac_add_options --with-system-jpeg
ac_add_options --with-system-png
ac_add_options --with-system-zlib

mk_add_options MOZ_OBJDIR=@TOPSRCDIR@/firefox-build-dir
EOF

sed -i 's@ ""@@' browser/base/Makefile.in

export SHELL=/bin/bash

make -f client.mk

make -C firefox-build-dir/browser/installer

mkdir -pv /_/P/Firefox/23.0.1

tar -xvf firefox-build-dir/dist/firefox-23.0.1.en-US.linux-$(uname -m).tar.bz2 \
    -C /_/P/Firefox/23.0.1 --strip-components=1

chown -R -v root:root /_/P/Firefox/23.0.1

mkdir -pv /_/P/Firefox/23.0.1/include/npapi
cp -v dom/plugins/base/*.h /_/P/Firefox/23.0.1/include/npapi

mkdir -pv /_/P/Firefox/23.0.1/lib/mozilla/plugins

sy Firefox 23.0.1

================================= Sharutils-4.13.5

konfig SharUtils 4.13.5
make
make install
sy SharUtils 4.13.5

================================= Berkeley DB-6.0.20

cd build_unix

../dist/configure --prefix=/_/P/BerkeleyDB/6.0.20 --enable-compat185 --enable-
dbm --disable-static --enable-cxx

make

make docdir=/_/P/BerkeleyDB/6.0.20/share/doc/db-6.0.20 install
```

```
chown -v -R root:root /_/P/BerkeleyDB

sy BerkeleyDB 6.0.20

============================ Ruby-2.0.0

konfig Ruby 2.0.0-p247 --enable-shared
make
make install
sy Ruby 2.0.0-p247

============================ SQLite-3.8.0.2

CFLAGS="-g -O2 -DSQLITE_ENABLE_FTS3=1 -DSQLITE_ENABLE_COLUMN_METADATA=1
-DSQLITE_ENABLE_UNLOCK_NOTIFY=1 -DSQLITE_SECURE_DELETE=1" konfig SQLite 3.8.0.2
--disable-static

make
make install

sy SQLite 3.8.0.2

============================ Gperf-3.0.4

konfig GPerf 3.0.4
make
make install
sy GPerf 3.0.4

============================ libogg-1.3.1

konfig LibOGG 1.3.1 --disable-static
make
make install
sy LibOGG 1.3.1

============================ libvorbis-1.3.3

konfig LibVorbis 1.3.3 --disable-static

make
make install
install -v -m644 doc/Vorbis* /_/P/LibVorbis/1.3.3/share/doc/libvorbis-1.3.3
sy LibVorbis 1.3.3
============================ libtheora-1.1.1

sed -i 's/png_\(sizeof\)/\1/g' examples/png2theora.c
konfig LibTheora 1.1.1 --disable-static
make
make install
sy LibTheora 1.1.1

============================ CDParanoia-III-10.2

patch -Np1 -i ../cdparanoia-III-10.2-gcc_fixes-1.patch
CFLAGS=-fPIC konfig CDParanoia-III 10.2
make
make install
chmod -v 755 /_/P/CDParanoia-III/10.2/lib/libcdda_*.so.0.10.2
sy CDParanoia-III 10.2

============================ FLAC-1.3.0
```

```
konfig FLAC 1.3.0 --disable-thorough-tests
make
make install
sy FLAC 1.3.0

========================== libsndfile-1.0.25

konfig LibSndFile 1.0.25 --disable-static

make

make htmldocdir=/_/P/LibSndFile/1.0.25/share/doc/libsndfile-1.0.25 install

sy LibSndFile 1.0.25

============================ LAME-3.99.5

konfig LAME 3.99.5 --enable-mp3rtp --disable-static
make
make pkghtmldir=/_/P/LAME/3.99.5/share/doc/lame-3.99.5 install
sy LAME 3.99.5

======================== GStreamer-0.10.36

sed -i  -e '/YYLEX_PARAM/d'                                          \
        -e '/parse-param.*scanner/i %lex-param { void *scanner }' \
            gst/parse/grammar.y

konfig GStreamer 0.10.36 --disable-static

make
make install
sy GStreamer 0.10.36

======================== gst-plugins-base-0.10.36

konfig Gst-Plugins-Base 0.10.36 --disable-static

make
make install

sy Gst-Plugins-Base 0.10.36

====================== Speex-1.2rc1

konfig Speex 1.2rc1 --disable-static

make
make install
sy Speex 1.2rc1

====================== libsamplerate-0.1.8

konfig LibSampleRate 0.1.8 --disable-static
make
make htmldocdir=/_/P/LibSampleRate/0.1.8/share/doc/libsamplerate-0.1.8 install
sy LibSampleRate 0.1.8
================================ FAAC-1.28

patch -Np1 -i ../faac-1.28-glibc_fixes-1.patch
```

```
sed -i -e '/obj-type/d' -e '/Long Term/d' frontend/main.c

konfig FAAC 1.28 --disable-static
make
make install
sy FAAC 1.28

============================ FAAD2-2.7

patch -Np1 -i ../faad2-2.7-mp4ff-1.patch
sed "s@AM_CONFIG_HEADER@AC_CONFIG_HEADERS@g" -i configure.in
autoreconf -fi
konfig FAAD2 2.7 --disable-static
make
make install

sy FAAD2 2.7

============================ OpenJPEG-1.5.1

konfig OpenJPEG 1.5.1 --disable-static
make
make install
sy OpenJPEG 1.5.1

================================ XviD-1.3.2

cd build/generic

konfig XviD 1.3.2

make

sed -i '/libdir.*STATIC_LIB/ s/^/#/' Makefile

make install
chmod -v 755 /_/P/XviD/1.3.2/lib/libxvidcore.so.4.3

cd /_/P/XviD/1.3.2/lib
ln -sv libxvidcore.so.4.3 libxvidcore.so.4
ln -sv libxvidcore.so.4   libxvidcore.so
sy XviD 1.3.2

==================================== FFmpeg-1.2.2

export LIBRARY_PATH=/_/S/L/Y

konfig konfig FFmpeg 1.2.2 --enable-gpl --enable-version3 --enable-nonfree
--disable-static --enable-shared --enable-x11grab --enable-libfaac --enable-
libfreetype --enable-libmp3lame --enable-libopenjpeg --enable-libspeex --enable-
libtheora --enable-libvorbis --enable-libvpx --enable-libxvid --enable-openssl
--disable-debug

make

gcc tools/qt-faststart.c -o tools/qt-faststart

unset LIBRARY_PATH

make install

sy FFmpeg 1.2.2
```

```
================================= Jack-1.9.9.5

./waf --prefix=/_/P/Jack/1.9.9.5 configure
./waf build
./waf install
sy Jack 1.9.9.5

========================== alsa-plugins-1.0.27

konfig Alsa-Plugins 1.0.27

make
make install
sy Alsa-Plugins 1.0.27
============================= alsa-utils-1.0.27.2

konfig Alsa-Utils 1.0.27.2 --disable-alsaconf --disable-xmlto

make
make install
sy Alsa-Utils 1.0.27.2

touch /var/lib/alsa/asound.state
alsactl store
usermod -a -G audio vz
```

A blfs bootscriptből:

```
make install-alsa

========================== AudioFile-0.3.6

konfig AudioFile 0.3.6

make
make install
sy AudioFile 0.3.6

============================= Libao-1.1.0

konfig LibAO 1.1.0
make
make install
sy LibAO 1.1.0

============================ vorbis-tools-1.4.0

konfig Vorbis-Tools 1.4.0 --enable-vcut --without-curl

make
make install
sy Vorbis-Tools 1.4.0

======================== FreePats.20060219

mkdir -p -v /_/P/FreePats/20060219/share
mkdir -p -v /_/P/FreePats/20060219/etc/freepats
```

Másoljuk át a /_/P/FreePats/20060219/etc/freepats könyvtárba a crude.cfg és a freepats.cfg fájlokat!

Másoljuk át a /_/P/FreePats/20060219/share könyvtárba a Drum_000 és a Tone_000 könyvtárakat!

```
sy FreePats 20060219

=================== TiMidity++-2.13.2

konfig TiMidity 2.13.2 --enable-ncurses --enable-slang --enable-tcltk --enable-
xskin --enable-gtk --with-x

make
make install
sy TiMidity 2.13.2
```

Sajnos, ezek után a Timidity még nem fog működni. Telepítve van ugyan, de valami baj van a config fájljával. Még nem volt időm, hogy e bajnak utánanézzek. Akartam, de közbejött épp az, hogy meg kellett írnom e könyvet gyorsan egy lelkes érdeklődőnek... Természetesen igyekszem e bajt mielőbb orvosolni, addig ez egy „known bug”... A besorolása a szememben „semi-critical”.

```
============================ LibTiMidity-0.1.0

konfig LibTiMidity 0.1.0

make
make install
sy LibTiMidity 0.1.0

======================== libmad-0.15.1b

patch -Np1 -i ../libmad-0.15.1b-fixes-1.patch
sed "s@AM_CONFIG_HEADER@AC_CONFIG_HEADERS@g" -i configure.ac
touch NEWS AUTHORS ChangeLog
autoreconf -fi
konfig LibMad 0.15.1b --disable-static
make
make install

mkdir -pv /_/P/LibMad/0.15.1b/lib/pkgconfig

cat > /_/P/LibMad/0.15.1b/lib/pkgconfig/mad.pc << "EOF"
prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include

Name: mad
Description: MPEG audio decoder
Requires:
Version: 0.15.1b
Libs: -L${libdir} -lmad
Cflags: -I${includedir}
EOF

sy LibMad 0.15.1b

================================== libid3tag-0.15.1b

konfig LibID3tag 0.15.1b
make
```

```
make install
sy LibID3tag 0.15.1b

================================ CMake-2.8.11.2

./bootstrap --prefix=/_/P/CMake/2.8.11 --system-libs
--docdir=/_/P/CMake/2.8.11/share/doc/cmake-2.8.11.2
--mandir=/_/P/CMake/2.8.11/share/man --no-system-curl --no-system-libarchive

make
make install
sy CMake 2.8.11


========================== Taglib-1.8

mkdir build
cd build

cmake -DCMAKE_INSTALL_PREFIX=/_/P/TagLib/1.8 \
      -DCMAKE_BUILD_TYPE=Release  \
      ..

make
make install
sy TagLib 1.8

======================== Moc-2.5.0-beta1

konfig Moc 2.5.0-beta1
make
make install
sy Moc 2.5.0-beta1
```

Ezek után a /_/P/Moc/2.5.0-beta1/lib/moc/decoder_plugins könyvtárból a
libtimidity_decoder.la
libtimidity_decoder.so
fájlokat mozgassuk át valami ideiglenes helyre, hogy ne legyenek ott. Különben a Moc nem fog működni, mert e 2 fájl észleli a Timidity korábban említett hibáját. Majd ha azt kijavítottuk, ezeket visszamásolhatjuk a Moc könyvtárába, hogy tudja nekünk a midi fájlokat is lejátszani.

```
==================== UnRar-5.0.11

make -f makefile
mkdir -pv /_/P/UnRar/5.0.11/bin
install -v -m755 unrar /_/P/UnRar/5.0.11/bin
sy UnRar 5.0.11
============================ MPlayer-1.1.1

patch -Np1 -i ../MPlayer-1.1.1-giflib_fixes-1.patch
patch -Np1 -i ../MPlayer-1.1.1-live_fixes-1.patch

sed -i 's:libsmbclient.h:samba-4.0/&:' configure stream/stream_smb.c

konfig MPlayer 1.1.1 --enable-dynamic-plugins --enable-menu --enable-gui

make
make install
install -v -m644 etc/*.conf /_/P/MPlayer/1.1.1/etc/mplayer
sy MPlayer 1.1.1
```

```
tar -xvf ../Clearlooks-1.5.tar.bz2 -C /_/P/MPlayer/1.1.1/Shared/mplayer/skins
cd /_/P/MPlayer/-/Shared/mplayer/skins
ln -s Clearlooks default
```

Csináljunk egy default nevű symlinket a **/_/S/L/T/mplayer/skins** könyvtárba, ami a **/_/P/MPlayer/-/Shared/mplayer/skins/default** linkre mutat.

```
============================ IMake 1.0.6

konfig IMake 1.0.6
make
make install
sy IMake 1.0.6

============================ xdotool-2.20110530.1
```

A Makefile elején így állítsuk be a prefixet:

```
PREFIX?=/_/P/XDoTool/2.20110530.1
```

Ezután:

```
make
make showman
make install
sy XDoTool 2.20110530.1

============================ xfontsel-1.0.5

./autogen.sf

konfig X--App-XFontSel 1.0.5 $X

make
make install
sy X--App-XFontSel 1.0.5

============================== Vixie-Cron-4.1

mkdir -pv /_/P/Vixie-Cron/4.1/share/man/cat1
mkdir -pv /_/P/Vixie-Cron/4.1/share/man/cat5
mkdir -pv /_/P/Vixie-Cron/4.1/share/man/cat8
mkdir -pv /_/P/Vixie-Cron/4.1/bin
mkdir -pv /_/P/Vixie-Cron/4.1/sbin
```

A Makefile elejét így írjuk át:

```
DESTDIR = /_/P/Vixie-Cron/4.1
DESTROOT        =       $(DESTDIR)
DESTSBIN        =       $(DESTROOT)/sbin
DESTBIN         =       $(DESTROOT)/bin
DESTMAN         =       $(DESTROOT)/share/man
```

Ezután:

```
make all
make install

sy Vixie-Cron 4.1

================================ WCalc 2.4.1
```

```
konfig WCalc 2.4.1
make
make install
sy WCalc 2.4.1

============================== fetchmail

konfig FetchMail 6.3.26 --with-ssl
make
make install
sy FetchMail 6.3.26

=================================================
```

Az urxvt terminálemulátor karakterméretének beállítása:

Adjuk ki e parancsot:

```
urxvt --help 2>&1| sed -n '/:  /s/^ */! URxvt*/gp' >> ~/.Xresources
```

Ezután az ~/.Xresources fájlt szerkesszük úgy, hogy e sor elől elvesszük a felkiáltójelet:

```
! URxvt*font:               fontname
```

És átírjuk így:

```
 URxvt*font:               12x24
```

Ezután a $HOME/.xinitrc fájl legelejére írjuk be e sort:

```
[[ -f ~/.Xresources ]] && xrdb -merge ~/.Xresources

========================================== ImageMagick-6.8.6-9

konfig ImageMagick 6.8.6-9 --with-modules --with-perl --disable-static

make
make install

sy ImageMagick 6.8.6-9
```

Innentől már van képernyőképmentési lehetőségem is (PrintScreen, persze ezt be kell állítani a $HOME/.xbindkeysrc fájlban a megfelelőképpen)

```
==========================================  giflib-5.0.5

konfig GifLib 5.0.5 --disable-static
make
make install

sy GifLib 5.0.5 --disable-static

========================================== Imlib2-1.4.5

sed -i "/DGifOpen/s:fd:&, NULL:" src/modules/loaders/loader_gif.c

konfig ImLib2 1.4.5 --disable-static

make
make install
```

```
sy ImLib2 1.4.5
========================================== GibLib-1.2.4

konfig GibLib 1.2.4 --disable-static
make
make install
sy GibLib 1.2.4

========================================== Feh 2.9.3
```

A config.mk fájl elején így állítsuk be a prefixet:

```
PREFIX ?= /_/P/Feh/2.9.3
```

Azután:

```
make curl=0
sy Feh 2.9.3

================================================== dpkg-1.15.5

konfig DPKG 1.15.5

make
make install

sy DPKG 1.15.5

====================================== wxWidgets-3.0.0

./autogen.sh
konfig WXWidgets 3.0.0 --enable-compat24 --with-gtk=2 --enable-unicode --enable-
utf8 --enable-config --enable-plugins
make
make install
```

A progi lib/wx/include könyvtárból a gtk2-unicode-3.0 könyvtárat mozgassuk át a progi gyökérkönyvtárában levő include könyvtárba.

```
sy WXWidgets 3.0.0
ldconfig -v

========================================== LibSBSMS-2.0.1

konfig LibSBSMS 2.0.1 --enable-shared
make
make install
sy LibSBSMS 2.0.1

====================================== audacity-2.0.5
```

Debian csomag binárisából feltéve... :(
Annyi nyűgje volt, hogy bevallom, tele lett a hócipőm vele. De olyan igazán nem nagyon izgat a dolog, elismerem, mert a fő célom sosem az volt, hogy okvetlenül mindent forrásból fordítsak, hanem az, hogy minden a maga külön könyvtárába telepedjék. Nos, az Audacity is oda került, tehát olyan igazán nagy baj végül is nem történt.

```
====================================== xautomation-1.0.3
```

```
export LDFLAGS=-lX11
konfig XAutomation 1.03
make

=======================================  Poppler-0.24.1

konfig Poppler 0.24.1 --disable-static --enable-xpdf-headers --sysconfdir=/etc
--disable-libcurl --with-x --disable-gtk-test --enable-zlib --enable-poppler-
glib
make
make install
tar -xf ../poppler-data-0.4.6.tar.gz
cd poppler-data-0.4.6
make prefix=/_/P/Poppler/0.24.1 install
sy Poppler 0.24.1

============================================= mupdf-1.3

make prefix=/_/P/MuPDF/1.3 install

sy MuPDF 1.3
========================================== gsettings-desktop-schemas-3.8.2

konfig GSettings-Desktop-Schemas 3.8.2
make
make install
sy GSettings-Desktop-Schemas 3.8.2

================================================  Nettle-2.7.1

konfig Nettle 2.7.1
make
make install
chmod -v 755 /_/P/Nettle/2.7.1/lib/libhogweed.so.2.5
/_/P/Nettle/2.7.1/lib/libnettle.so.4.7
sy Nettle 2.7.1

================================================= GnuTLS-3.2.4

konfig GnuTLS 3.2.4 --disable-static --with-default-trust-store-
file=/etc/ssl/ca-bundle.crt
make
make install
sy GnuTLS 3.2.4

==============================================  glib-networking-2.36.2

konfig GLib-Networking 2.36.2  --with-ca-certificates=/etc/ssl/ca-bundle.crt
--disable-static
make
make install
sy GLib-Networking 2.36.2

==============================================  libsoup-2.42.2

konfig LibSOUP 2.42.2 --disable-static
make
make install
sy LibSOUP 2.42.2

================================================= w3c-libwww-5.4.0
```

```
konfig W3C-LibWWW 5.4.0
make
make install
sy W3C-LibWWW 5.4.0

================================================  WebKitGTK+-1.10.2

sed -i '/generate-gtkdoc --rebase/s:^:# :' GNUmakefile.in
sed -i '/parse-param/ a%lex-param {YYLEX_PARAM}' \
             Source/ThirdParty/ANGLE/src/compiler/glslang.y

patch -Np1 -i ../webkitgtk-1.10.2-fix_librt_linking-1.patch
konfig WebKitGTK 1.10.2 --with-gtk=2.0 --disable-webkit2 --disable-geolocation
make
make install
sy WebKitGTK 1.10.2

================================================  Git-1.8.4

konfig Git 1.8.4 --with-gitconfig=/etc/gitconfig
make
make install
sy Git 1.8.4

================================  babl-0.1.10

konfig Babl 0.1.10
make
make install
sy Babl 0.1.10

================================  gegl-0.2.0

sed -e '274cerr = avformat_open_input (&p->ic, o->path, NULL, NULL);' \
    -i operations/external/ff-load.c

konfig Gegl 0.2.0
make
make install
sy Gegl 0.2.0
================================  Gimp-2.8.6

konfig Gimp 2.8.6 --without-gvfs --disable-python

make
make install
sy Gimp 2.8.6
```

A Gimp könyvtárában törölni kell a share linket, és vissza kell nevezni a Shared könyvtárat share-re.

```
cd gimp-help-2.8.0
ALL_LINGUAS="en hu" konfig Gimp 2.8.6
make
make install
```

A Gimp könyvtárában visszanevezni a share könyvtárat Sharedre, és megcsinálni újra a share linket.

```
================================  cURL-7.32.0
```

```
PATH_SEPARATOR=':' konfig CUrl 7.32.0 --enable-threaded-resolver --with-ca-
path=/etc/ssl/certs

make
make install
sy CUrl 7.32.0
=============================================  Transmission-2.82

konfig TransMission 2.82
make
make install
sy Transmission-2.82
```

Megjegyzendő, hogy ezután még csak a transmission-cli nevű parancssoros változata fog működni neked, grafikus felülete nem lesz...

```
============================================== ccache-3.1.9

konfig CCache 3.1.9
make
make install
sy CCache 3.1.9
```

A /_/S/L könyvtárba készítsünk egy CCACHE nevű könyvtárat. Ide pedig csináljunk symlinkeket a következőképpen:

```
root@Csiszilla /Releases/2014/S/L/CCACHE=>ls -l

lrwxrwxrwx 1 root root 11 jan    2 23.18 c++ -> ../E/ccache
lrwxrwxrwx 1 root root 11 jan    2 23.17 cc -> ../E/ccache
lrwxrwxrwx 1 root root 11 jan    2 23.17 g++ -> ../E/ccache
lrwxrwxrwx 1 root root 11 jan    2 23.17 gcc -> ../E/ccache
```

Ezután a root userünk $PATH változóját írjuk át ekképpen a $HOME/.bashrc fájljában:

```
export PATH="/_/S/L/CCACHE:/_/S/L/E"
```

Ez biztosítja azt, hogy a forrásból fordításhoz szükséges fontos parancsokat amikor a rendszer keresi, azok helyett előbb a ccache programot találja meg.

```
============================================== Gsl-1.16

konfig Gsl 1.16 --disable-static
make
make install
sy Gsl 1.16

============================================== Mpg123-1.15.4
konfig Mpg123 1.15.4
make
make install
sy Mpg123 1.15.4
============================================== libmpeg2-0.5.1
sed -i 's/static const/static/' libmpeg2/idct_mmx.c
konfig LibMPEG2 0.5.1
make
make install
sy LibMPEG2 0.5.1
============================================== Transcode-1.1.7
```

```
sed -i 's|doc/transcode|&-$(PACKAGE_VERSION)|' \
       $(find . -name Makefile.in -exec grep -l 'docsdir =' {} \;)

patch -Np1 -i ../transcode-1.1.7-ffmpeg-2.patch

konfig TransCode 1.1.7 --enable-alsa --enable-libmpeg2 --enable-libavcodec
--enable-ffmpeg --enable-libavformat  --enable-libmpeg2convert  --enable-
freetype2  --enable-lame --enable-xvid --enable-ogg  --enable-vorbis  --enable-
theora --disable-libdvdread  --enable-libxml2  --enable-imagemagick  --enable-
libjpeg  --enable-iconv --disable-x86-textrels

make
make install
sy TransCode 1.1.7


======================================== Archive::Zip-1.30 perl modul
perl Makefile.PL
make
make test
make install

======================================== Boost-1.54.0

patch -Np1 -i ../boost-1.54.0-glibc-1.patch
./bootstrap.sh --prefix=/_/P/Boost/1.54.0
./b2 stage threading=multi link=shared
./b2 install threading=multi link=shared
sy Boost 1.54.0
```

========== libreoffice-4.2.0.2

Másoljuk a LO forrásfájljait kicsomagolás nélkül a /sources -be!

MAJD NEM ROOT-KÉNT :

```
tar -xf libreoffice-4.2.0.2.tar.xz --no-overwrite-dir
cd libreoffice-4.2.0.2
install -dm755 src
tar -xf ../libreoffice-dictionaries-4.2.0.2.tar.xz --no-overwrite-dir --strip-
components=1
ln -sv ../../libreoffice-dictionaries-4.2.0.2.tar.xz src/
ln -sv ../../libreoffice-help-4.2.0.2.tar.xz src/
ln -sv ../../libreoffice-translations-4.2.0.2.tar.xz src/

sed -e "/gzip -f/d"   \
    -e "s|.1.gz|.1|g" \
    -i bin/distro-install-desktop-integration
sed -e "/distro-install-file-lists/d" -i Makefile.in

chmod -v +x bin/unpack-sources
sed -e "s/target\.mk/langlist\.mk/"                \
    -e "s/tar -xf/tar -x --strip-components=1 -f/" \
    -e "/tar -x/s/lo_src_dir/start_dir/"           \
    -i bin/unpack-sources

./autogen.sh --prefix=/_/P/LibreOffice/4.2.0.2     \
             --sysconfdir=/etc          \
             --with-vendor="BLFS"       \
             --with-lang="en-US hu"     \
```

```
            --with-help               \
            --with-alloc=system       \
            --without-java            \
            --disable-gconf           \
            --disable-odk             \
            --disable-postgresql-sdbc \
            --with-system-boost       \
            --with-system-cairo       \
            --with-system-curl        \
            --with-system-expat       \
            --with-system-harfbuzz    \
            --with-system-icu         \
            --with-system-jpeg        \
            --with-system-libpng      \
            --with-system-libxml      \
            --with-system-mesa-headers \
            --with-system-openssl     \
            --with-system-poppler     \
            --with-system-zlib        \
            --with-parallelism=$(getconf _NPROCESSORS_ONLN)

make build
```

Kell hogy legyen működő internetkapcsolatunk, mert egy csomó mindent letöltöget majd magának, ami nincs feltelepítve a rendszerünkbe jóelőre.

```
make distro-pack-install

chown -cR 0:0 dictionaries/

mkdir -pv /_/P/LibreOffice/4.2.0.2/lib/libreoffice/share/extensions/dict-en

cp -vR dictionaries/en/*
/_/P/LibreOffice/4.2.0.2/lib/libreoffice/share/extensions/dict-en

mkdir -pv /_/P/LibreOffice/4.2.0.2/lib/libreoffice/share/extensions/dict-hu

cp -vR dictionaries/hu_HU/*
/_/P/LibreOffice/4.2.0.2/lib/libreoffice/share/extensions/dict-hu
```

Törlöm a
`/_/P/LibreOffice/4.2.0.2/lib/libreoffice/share/fonts`
könyvtárat, mert minden itteni font megvan már nekem.
A /_/P/LibreOffice/ könyvtárban állva:

```
ln -s 4.2.0.2 2-

LINK /_/S/L/E /_/P/LibreOffice/2-
```

És máris használhatom, s le se kellett törölnöm a régit...
Az swriter neve itt lowriter, erre ügyeljünk.

========== pciutils-3.2.0

```
make PREFIX=/_/P/PciUtils/3.2.0  \
     SHAREDIR=/_/P/PciUtils/3.2.0/share/misc \
     MANDIR=/_/P/PciUtils/3.2.0/share/man    \
     SHARED=yes                \
     ZLIB=no                   \
     all
```

```
make PREFIX=/_/P/PciUtils/3.2.0  \
     SHAREDIR=/_/P/PciUtils/3.2.0/share/misc \
     MANDIR=/_/P/PciUtils/3.2.0/share/man    \
     SHARED=yes               \
     ZLIB=no                  \
     install install-lib
chmod -v 755 /_/P/PciUtils/3.2.0/lib/libpci.so.3.2.0

sy PciUtils 3.2.0
```

========== libusb-1.0.9

```
konfig LibUSB 1.0.9 --disable-static
make
make install
sy LibUSB 1.0.9
```

========== usbutils-007

```
konfig USBUtils 007 --disable-zlib --datadir=/_/P/USBUtils/007/share/misc
make
```

A /_/**P**/**USBUtils**/**007**/**sbin** könyvtárban állva:

```
ln -s update-usbids.sh update-usbids
sy USBUtils 007
```

========== lshw-B.02.16

Az src/Makefile fájl elején írjuk át a PREFIX-et így:

```
PREFIX?=/_/P/LsHW/B.02.16
```

Ezután a forráskönyvtár gyökerében állva:

```
make all
make install

sy LsHW B.02.16
```

****
****
****

Hát egyelőre ennyi! Természetesen a rendszeredbe még rengeteg mindent bele lehet pakolni, de az már rajtad múlik... Egyelőre az a helyzet, hogy ha eddig eljutottál, akkor van grafikus felületed, Interneted, Firefoxod, LibreOffice-od, tudsz torrentezni, zenét hallgatni (ha nem is midi fájlokat...), zenei fájlokat vagdosni-szerkeszteni, tudsz filmeket nézni és átkonvertálni, képeket nézegetni (a Feh programmal) és szerkesztgetni azokat, és hát ennél sokkal többre egy „normál user” általában nem is nagyon használja a számítógépét. Vagy legalábbis ritkán. Azaz készen lett egy alapdolgokra jól használható rendszer, „disztró”.

Hogy ebből mit fejlesztesz ki tovább magadnak, az már csak rajtad múlik, azaz „rajtad a Világ szeme”!

Hogyan tovább?

Ha idáig eljutottál az olvasásban, vélhető hogy egészséges „hacker-szellem” és „geek-mentalitás” lakozik benned! Ez esetben ajánlom figyelmedbe a web-oldalamat, aminek linkje:
http://parancssor.info
Mint már a nevéből is kiderül, ez olyanoknak való, akik nem félnek a parancssor használatától. Sőt, szeretik azt. Még „sőttebben”, ez kifejezetten a „kemény-vonalas” hardcode/hardcore-linuxusereknek van szánva (nem is tolonganak a látogatók, bevallom...), tudod, akikről mások, a „szellemi barbárok/nyomorultak” úgy vélekednek, hogy „elvakult, elborult agyú parancssor-buzerátorok, akik dülledt szemmel lesik a képernyőt, s állandóan hexadecimális kódokat mormognak”, mindenesetre ott egy olyan közösséget szeretnék létrehozni, akik nem azt a manapság divatos és gyakran hangoztatott nézetet vallják a számítógépről, hogy „Nem értek hozzá, de nem is akarok érteni hozzá, egyszerűen csak használni akarom, működjön és kész”! Olyanokat várok, akik épp ellenkezőleg vélekednek erről a kérdésről: akik igenis ÉRTIK a számítógépet, a programokat, a programozást, vagy ha nem is értik még, de SZERETNÉK érteni, és ezért készek tanulni is!

Jelszavam, ami ki is van írva arra a honlapra, mert annak is ez a jelszava:

„Az ENYÉM, mert ÉRTEK HOZZÁ!”

Azaz, csak azt érezhetjük a magunkénak, aminek tisztában vagyunk a működésével. Természetesen soha semmit se ismerhetünk 100%-ig, de törekedni kell erre! Minél jobban ismerjük, annál inkább a „magunké”.

Azon a honlapon már találhatsz doksit még arról is, hogyan írhatsz SAJÁT PROGRAMNYELVET... Magyar doksi, én írtam azt is! Igen, írtam programnyelvet, remekül használható, használom a napi munkám során, egy szkriptnyelv, s az a neve hogy „mau”. Természetesen nyílt forráskódú, ingyenes, letölthető, tanulmányozható, stb... Részletesen dokumentálva van a megalkotás teljes folyamata is, ennek alapján tényleg írhatsz te is saját programnyelvet! Képzeld el, ha írnál egyet magadnak, micsoda „geek” dolog volna... Szóval, ha efféle közösségre vágysz, akik ilyesfélék alkotásában élik ki a teremtési vágyukat, akkor várlak oda!