Merevlemez nélküli linuxos X terminál

Hogyan indítsunk hálózatról és használjuk meghajtó nélküli X terminálként állandó háttértár nélküli Linuxot?

Az *X* terminál nem igazán új ötlet; az *NCD* és más cégek legalább 15 éve gyártanak ilyeneket. A vékony ügyfél ötlete az 1990-es években kezdett igazán divatba jönni, amikor a PC alkatrészek ára olyan mélyre zuhant, hogy nem volt tovább pénzügyi jelentősége az *X* termináloknak. Komoly viták kezdődtek a teljes bekerülési költségről (amibe az alkatrész és a karbantartási költségeket is beleszámolták) a vékony ügyfelek és a PC támogatói között, de ezt a vitát ez a cikk sem fogja eldönteni. Célunk mindössze annyi, hogy megmutassuk, hogyan lehet a rohamléptekkel fejlődő PC technológia nyomán hátramaradt elavult alkatrész halmainkból *X* terminálokat építeni.

Minden vékony ügyfél legfontosabb tulajdonsága, hogy egyáltalán nincs, vagy csak nagyon kevés állandó tára van. Általában az elve *X* terminálnak szánt terminálokban kis mennyiségű *NVRAM*-ot alkalmaznak, ahol a beállítási adatokon kívül semmi mást nem tárolnak. Gyakorlatban ezeket a az adatokat is áttehetjük a kiszolgálón tárolt beállításállományba, amit a terminál induláskor letölthet. Ebben a cikkben azt a letisztult elvet követjük, miszerint egy *X* terminálnak egyáltalán nincs szüksége tárhelyre.

PXE Indítás

A PC-nek nincs merev, hajlékony vagy CD lemeze úgyhogy az indítóprogram és az indítható állomány tárolásához valamilyen más eszközt kell találnunk. Az *X* terminálok a lakhelyükül szolgáló hálózat teremtményei, így magától értetődő választás lenne a hálózati csatolófelület kártya (*NIC*). Ennek megvalósításához a *NIC*-nek rendszerindító eszközként kell regisztrálnia magát a *BIOS*-ban. Ha kiválasztják, le kell tudnia tölteni a rendszerindítót a hálózatról. Ez nem igazán olyasmi, amit minden futószalagról lekerült *NIC* csuklóból tudna. Az *Intel* ugyanakkor kiadta a *NIC*-hez szánt indító *ROM*-ját, amelyet *PXE*-nek (*Preboot eXecution Environment,* kiejtve*: pixie*) keresztelt el, s amit aztán a cég és számos más gyártó be is épített bizonyos termékekbe. Sok újabb, beépített hálózati kártyával rendelkező alaplapban találunk *PXE* támogatást.

A cikkre készülés során öt különféle NIC-et teszteltem amelyek a hirdetések szerint támogatják a *PXE*-t: az *Intel PRO/100+ (PILA8460BNG1)*, a *3Com 3C905CX-TX-M*, a *D-Link DFE-550TX*, a *Linksys LNE100TX* és az *SMC 1255TX* (*Tulip* lapkakészlet) lapkát. Az ötből kizárólag a *3Com* kártya volt képes minden további nélkül elindulni. Az *SMC* kártyához tudtam külön szerezni indító *ROM*-ot így végül az is működött. A másik három kártyán feltűnő, ám üres foglalatot találtam az indító *ROM* helyén, amit alapértelmezés szerint nem szállítanak. Nem árt az óvatosság vásárláskor.

Amikor az alaplap *BIOS*-a a *PXE NIC BIOS*-t választja indítóeszközként, az kiad egy *DHCP* kérelmet a *LAN*-on, majd a visszakapott válaszokban a *PXE* kiterjesztéseket keres. Ha ilyen kiterjesztést tartalmazó választ talál, akkor visszajelez és elfogadja a választ. A kiszolgáló válaszában különösen a `next-server` és a `filename` paramétereket keresi. Ezek a paraméterek adják meg a *TFTP* kiszolgáló IP számát valamint az ügyfél által letöltendő és elindítandó rendszerbetöltő állomány nevét.

DHCP és TFTP

Az *Internet Software Consortium* 3.0-ás verziójú *DHCP* kiszolgálóját *PXE* kiterjesztések kezelésére is be lehet állítani. Szerencsére ezt a *DHCP* kiszolgálót kapjuk a legtöbb *Linux* terjesztéssel, ide értve a *Red Hat 8.0* és későbbi verzióit. Az *1. listában* egy olyan *DHCP* kiszolgáló beállításállományát (*dhcpd.conf*) olvashatjuk, amely *PXE* kiterjesztésekkel bővített *DHCP* válaszokat küld, amennyiben a *DHCP* ügyfél *PXE NIC* néven azonosítja magát.

Az itt olvasható beállítások szerint az ügyfél a *192.168.1.1* címen található *TFTP* kiszolgálóról tölti le a pxelinux.0 állományt. A beállításfájlban felsorolt kapcsolók listáját az *1. táblázatban* találjuk.

A `192.168.1.1` címen található kiszolgálónkat természetesen úgy kell beállítani, hogy képes legyen *TFTP* szolgáltatást nyújtani. Ezen kívül kell lennie rajta egy `pxelinux.0` rendszerbetöltő állománynak, mégpedig ott, ahol a *TFTP* kiszolgáló folyamatok keresik (általában a */tftpboot* könyvtárban). A *TFTP* kiszolgáló folyamatot általában valamelyik szuperkiszolgáló kezeli (*inetd* vagy *xinetd*) következésképpen ha be szeretnénk kapcsolni valamelyikük beállításfájljában (*/etc/inetd.conf* vagy */etc/xinetd.conf*) kell matatnunk.

A `pxelinux.0` nevű rendszerbetöltő állomány *H. Peter Anvin SYSLINUX* projektjéből származik. Az általános

1. táblázat *A dhcpd.conf állományban használható PXE-vonatkozású kódok leírása*

Code	Meaning
1	Az indító fájlkiszolgáló multicast IP címe.
2	UDP kapu melyen az ügyfél az MTFTP válaszokat figyelheti.
3	UDP kapu melyet az MTFTP kiszolgálók MTFTP kérelmek fogadására használnak.
4	Az ügyfélnek ennyi másodpercet kell aktivitásra várakoznia mielőtt új MTFTP átvitelt kezdeményezne.
5	Az ügyfél ennyi másodpercet vár, mielőtt újraindítaná az MTFTP átvitelt.

rendszerbetöltőkkel szemben mint amilyen a ***LILO*** vagy a ***GRUB***, a ***PXELINUX*** képes a ***PXE*** protokollt kezelésére és el tudja látni a szükséges hálózati feladatokat, így ezen a ponton már képes átvenni a rendszerindítást és ***TFTP***-n keresztül letölteni a rendszermagot valamint a tömörített ***RAM*** lemezt. Ugyanakkor a ***PXELINUX*** használatához bővített ***TFTP*** kiszolgálóra lesz szükségünk, amelyik képes megérteni a `TSIZE` kapcsolót (***RFC 2349***). Szerencsére, ***H. Peter Anvin*** a szabványos ***BSD TFTP*** démon ***tftp-hpa*** névre keresztelt, javított verzióját is elkészítette amelyik már képes e parancs értelmezésére. Könnyű dolgunk van: a ***/usr/sbin/in.tftpd*** könyvtárban található ***TFTP*** démont egyszerűen csak lecseréljük a ***tftp-hpa*** programra.

PXELINUX

A ***PXELINUX*** tudja, hogy a ***PXE*** indító ***ROM*** hová teszi a ***DHCP*** kiszolgálótól kapott hálózati paramétereket a memóriában, így képes felhasználni őket egy új ***TFTP*** folyamat megindításához amellyel saját beállításállományait tölti le a kiszolgálóról. A fentiek szerint beállított ***TFTP*** kiszolgálón az ügyfélen futó rendszerbetöltő először a ***/tftpboot/pxelinux.cfg/ethermac*** könyvtárban próbálja megtalálni a beállításállományait. Az ***ethermac*** szó itt az ügyfél kisbetűs hexadecimális kódban megadott Ethernet hardvercímének felel meg, ahol a nyolcadokat kötőjelek választják el, például: ***fe-ed-de-ad-be-ef***. Ha ezt nem tudja elérni, a rendszertöltő ***/tftpboot/pxelinux.cfg/iphex állományt keresi, ahol az iphex*** az ügyfél IP címe, nagybetűs hexadecimális kóddal megadva. Például, ha az ügyfél ***IP*** címe ***192.168.0.12***, a ***PXELINUX*** a ***/tftpboot/pxelinux.cfg/C0A8000C*** állományt keresné. Amennyiben ez a fájl sem létezik, a legkevésbé lényeges szakaszt kivágja a névből és újrakezdi a folyamatot. Így a fenti példában ha a ***C0A8000C*** állományt nem találja, a ***PXELINUX*** megpróbálkozik a ***C0A8000***, majd a ***C0A800*** állományokkal és így tovább. Ezáltal egyetlen beállításállományt használhatunk az egész alhálózathoz, feltéve hogy az alhálózat határait a szakaszokhoz igazítottak. A 2. *lista* a ***PXELINUX*** beállításállomány tartalmát mutatja be. Az első sor a letöltendő tömörített rendszermag állomány nevét adja meg. Minden elérési út a kiszolgáló

1. lista PXE ügyfeleket is támogató dhcpd.conf állomány minta

```
option space PXE;
option PXE.mtftp-ip
  code 1 = ip-address;
option PXE.mtftp-cport
  code 2 = unsigned integer 16;
option PXE.mtftp-sport
  code 3 = unsigned integer 16;
option PXE.mtftp-tmout
  code 4 = unsigned integer 8;
option PXE.mtftp-delay
  code 5 = unsigned integer 8;
option PXE.discovery-control
  code 6 = unsigned integer 8;
option PXE.discovery-mcast-addr
  code 7 = ip-address;
subnet 192.168.1.0 netmask 255.255.255.0 {
  class "pxeclients" {
    match if substring (option
      vendor-class-identifier, 0, 9) =
      ➥"PXEClient";
    option vendor-class-identifier "PXEClient";
    vendor-option-space PXE;
    # Legalább az egyik gyártófüggő PXE
    # kapcsolót be kell kapcsolnunk
    # hogy az ügyfél
    # indító ROM-ja felismerje, hogy PXE-kezelő
    # kiszolgálóval van dolga.
    # Mi a MCAST IP címet
    # állítottuk 0.0.0.0-ra jelezve
    # a ROM-nak, hogy
    # nem kezelünk multicast TFTP-t.
    option PXE.mtftp-ip 0.0.0.0;
    # Itt adjuk meg annak a fájlnak nevét
    # amit a ROM-ok
    # letölthetnek.
    filename "pxelinux.0";
    # A kiszolgáló neve amiről le kell
    # tölteniük.
    next-server 192.168.1.1;
  }
  pool {
    max-lease-time 86400;
    default-lease-time 86400;
    range 192.168.1.2 192.168.1.254;

   # Amennyiben ezt a sort beletesszük,
   # minden ügyfélhez
   # külön gépbejegyzéseket kell
   # készítenünk, ahol az
   # ethernet MAC címeket a IP címekkel
   # összerendelhetjük.
   # deny unknown clients;
  }
}
```

2. lista A PXELINUX beállításállománya határozza meg melyik tömörített rendszermag fájlt kell letölteni

```
DEFAULT vmlinuz
APPEND initrd=ramdisk.gz ramdisk=65536
root=/dev/ram rw
IPAPPEND 1
```

/tftpboot könyvtárához képest relatív kell legyen. A második sorban a rendszermagnak átadandó paramétereket soroljuk fel, jelezve, hogy a gyökér fájlrendszert 64MB-os memória lemezre kell felcsatolni írható/olvasható módon. Az utolsó sor hatására a ***PXELINUX*** egy további rendszermag paramétert hoz létre:

```
ip=client-ip:server-ip:gateway-ip:netmask
```

Ehhez azokat az adatokat használja fel, amit a ***PXE*** indító ***ROM*** a ***DHCP*** kiszolgáló válaszában kapott meg. Ez akkor lehet hasznos, ha a rendszermagot engedélyezett rendszermag-szintű IP hálózattal fordítottuk. Ilyenkor ugyanis a rendszermag ezeket a paramétereket felhasználja a hálózati csatolófelület beállításához, így később ezt már nem kell megtennünk az indítóparancsfájlban az `ifconfig` vagy `ifup` futtatásával.

Rendszermag készítése

A rendszermag szintű automata ***IP*** paraméter beállítás használatához a hálózati meghajtóeszköznek viszonylag hamar elérhetőnek kell lennie, még a root fájlrendszer felcsatolása előtt. Következésképpen nem tölthetjük be modulként. Minthogy a legtöbb terjesztéssel együtt érkező rendszermagok igen kiterjedten használnak modulokat, gyakorlatilag kénytelenek vagyunk új rendszermagot forgatni az ***X*** terminálunkhoz. Továbbá a saját rendszermagunknak kezelnie kell a ***RAM*** lemezeket és a indítási ***RAM*** lemezeket. A rendszermag szintű ***IP*** hálózat automata beállítása szintén kényelmes dolog. Persze nem kötelező semmilyen dinamikus módszert alkalmaznunk az ***IP*** címek megadásához (választhatjuk a ***DHCP***, ***BOOTP*** vagy ***RARP*** megoldásokat is), a ***PXELINUX*** beállításfájljában megadott ***IPAPPEND*** viszont biztosítja számunkra, hogy rendszermag biztosan a megfelelő IP paramétereket kapja meg. Végül ha az eszköz fájlrendszert automatikusan csatoljuk a ***devfs*** segítségével, jelentősen leegyszerűsíthetjük a RAM-gyökérlemezen található */dev* könyvtárunkat.

A RAM lemez root fájlrendszere

Egy lemez nélküli ***Linux X*** munkaállomás esetében legnehezebb feladatunk a gyökér fájlrendszer kialakítása és feltöltése lenne, ha nem jelent volna meg ***Richard Gooch*** eszköz fájlrendszere és Erik ***Anderson BusyBox*** kombinált végrehajtható állománya. Az eszközkezelő fájlrendszer automatikusan kezeli a */dev* könyvtárat, a modulok rendszermagba töltésekor szükség szerint hozza létre a megfelelő eszközvégpontokat. Ez két dolgot jelent: a könyvtárban nincsenek szükségtelen bejegyzések, és a ***RAM*** lemezes fájlrendszer készítőinek nem kell órákat töltenie az `mknod`-al, hogy létrehozza az összes szükséges végpontot. A ***BusyBox*** egyesített program olyan végrehajtható állomány amely indítástól függően változtatja személyiségét. Szokásos használata során a */bin/busybox* programra mutató közvetett hivatkozásokat hozunk létre a */bin/ls*, */bin/cat*, */bin/ps*, */sbin/mount* és a egyéb neveken amivel egy minimalista ***UNIX*** rendszerhez jutunk. Semmilyen egyéb futtatható állományra vagy programkönyvtárra nincs szükségünk; a ***BusyBox*** már mindent tartalmaz.
Úgy is fogalmazhatnánk, hogy az eszközfájlrendszer felügyeli a */dev* könyvtárat, a ***BusyBox*** a */bin* és */sbin* könyvtárakat; a rendszermag kezeli */proc* mappát; a csak olvasható ***NFS*** csatolás pedig az */usr* könyvtárért felelős; a */tmp* üres marad. Így egyetlen dologgal kell csak törődnünk, az */etc*-vel. Szerencsére az */etc* könyvtárba se kell túl sok minden, elég ha tartalmazza az */etc/fstab*, */etc/inittab* és */etc/init.d/rcS* állományokat, ez utóbbi az az indító-parancsfájl amit a ***BusyBox*** `init`-ként futva használ.
A ***BusyBox*** a beágyazott ***Linux*** világnak íródott és általában statikus futtatható állományként fordítják. ugyanakkor, az ***XFree86*** kiszolgáló önmagában is sok, a */lib* könyvtárban található osztott könyvtártól függ. A */usr*-t hálózati fájlrendszerként (***NFS***) csatoljuk, tehát az */usr/lib*-ben található osztott könyvtárak miatt nem kell aggódnunk, biztosítanunk kell viszont azokat, amelyeket a */lib*-ben keres majd az ***XFree86***. Emiatt érdemes kihasználhatunk egy helymegtakarítási lehetőséget, nevezetesen, hogy a ***BusyBox***-ot dinamikus végrehajtható állományként állítjuk be és fordítjuk le. Az ***XFree86*** futtatásához szükséges minimális */lib* programkönyvtárak kigyűjtését az

```
ldd /usr/X11R6/bin/XFree86
```

parancs kiadásával tudjuk elvégezni. Ezek a ***glibc*** (***libc.so*** és ***libm.so***), a PAM (***libpam.so*** és ***libpam_misc.so***) valamint maga a dinamikus töltő (***libdl.so*** és ***ld-linux.so***) lesznek.

XFree86 beállítása

Az ***XFree86*** végrehajtható állomány általában az */usr/X11R6/bin* könyvtárban található, amely a */usr* alkönyvtára. Így az ***X*** kiszolgálót nem kell ***RAM*** lemezünkön tartanunk, ehelyett az ***NFS*** csatolásról betölthetjük. Bár a moduláris ***XFree86*** kiszolgáló a 4.0-ás verzió óta már nem alkatrészfüggő, a beállításállománya bizony igencsak az. Amennyiben több különféle videó alkatrésszel rendelkező gépet kezelünk, nem használhatjuk valamennyi géphez ugyanazt az ***XF86Config*** állományt. Ezért inkább nem tartjuk a gyökér fájlrendszer ***RAM*** lemezén, ahol általában egyébként */etc/X11/XF86Config* néven szerepel. Ehelyett inkább terminálonkénti beállításállományokat tárolunk az */usr NFS* könyvtárban. Végül, a ***BusyBox*** `init` folyamatát úgy állítjuk be, hogy egyetlen sort tartalmazó parancsfájlt indítson újra folyamatosan:

```
/usr/X11R6/bin/XFree86 \
-xf86config /usr/X11R6/configs/iphex -query \
server
```

3. lista A terminálok XF86Config állományának részlete

```
Section "Files"
        RgbPath         "/usr/X11R6/lib/X11/rgb"
        ModulePath      "/usr/X11R6/lib/modules"
        FontPath        "tcp/192.168.1.1:7100"
EndSection
```

ahol az `iphex` az ügyfél ***IP*** címe hexadecimális kódban (a ***PXELINUX***-tól kölcsönzött jelölési rendszer szerint) a `server` pedig kiszolgáló ***IP*** címe ponttal elválasztott decimális kódban. Néhány ügyes ***awk*** trükk a ***/proc/cmdline***-on és teljesen elkerülhetjük a gépnevek és ***IP*** címek ***RAM*** lemezbe kódolását.
Alapszintű ***XFree86*** beállításállományt az `XFree86 -configure` parancs kiadásával hozhatunk létre a terminálon. Ez többnyire helyesen ismeri fel a videó alkatrészt az eredményül kapott beállításállomány pedig betölti a megfelelő ***XFree86*** modulokat. Érdemes megemlíteni azonban, hogy az alapértelmezett mutató eszköz a /dev/mouse lesz, ami általában nem szerepel az eszköz fájlrendszerrel létrehozott eszközök között. Például, a ***PS/2***-es egér nem itt, hanem a ***/dev/misc/psaux*** helyen található meg.

Beállítások a kiszolgálón

Amitől az ***X*** terminálunk tényleg ***X*** terminál lesz egyszerű grafikus kijelzővel felszerelt ***Linux*** gép helyett az a fenti ***XFree86*** parancssorunk `-query` ***server*** része. Hatására a terminálon futó ***XFree86*** kiszolgáló elindítja az ***XDMCP (X Display Manager Control Protocol)*** folyamatát és egy olyan kiszolgálót keres, amely kezelhetné a képernyőjét. Csakhogy erre nem minden kiszolgáló kapható.
Először is, a kiszolgálónak értelemszerűen hallgatnia kell a bejövő ***XDMCP*** kapcsolatokra. Az ***XDMCP*** általában a 177-es ***UDP*** kaput használja, és a legtöbb megjelenítésvezérlő (***xdm***, ***gdm***, ***kdm***) be is állítható úgy, hogy engedélyezze az ***XDMCP*** kérelmeket. Bár a legtöbb terjesztés alapértelmezés szerint grafikus felülettel indul, biztonsági megfontolásokból szinte mindegyik tiltja a bejövő ***XDMCP*** kérelmeket. Például a klasszikus ***xdm X*** megjelenítőkezelő beállításállományában (amely általában a ***/etc/X11/xdm/xdm-config***) a következő sort találjuk alapértelmezés szerint:

```
DisplayManager.requestPort: 0
```

Ezt a sort megjegyzésbe kell tennünk, ha azt szeretnénk, hogy az ***xdm*** fogadni tudja az ***XDMCP*** kérelmeket. Ezen felül az ***xdm*** beállítható úgy is, hogy gépenkénti vagy alhálózat alapon korlátozza saját elérhetőségét, amit a ***/etc/X11/xdm/Xaccess*** beállításállományban tehetünk meg(senkit ne zavarjon meg a ***/etc/X11/xdm/Xservers***, ami lényegében történelmi emlék). Például, ha az ***xdm***-et a ***192.168.1.0/24*** alhálózat termináljaira szeretnénk korlátozni, mindössze a `192.168.1.0/24` sort kell felvennünk a ***/etc/X11/xdm/Xaccess*** állomány végére.
Ezen felül jól jöhet, ha a kiszolgáló az ***xfs X*** fontkiszolgálón keresztül fontokat is tud nyújtani a terminálóknak.
Akárcsak az előbbi esetben, a legtöbb terjesztés saját fontkiszolgáló folyamatot futtat, mégis általában úgy van beállítva, hogy nem fogad bejövő kérelmeket. Például a az ***xfs*** beállításállományában, az ***/etc/X11/fs/config***-ban, általában a `no-listen = tcp` sort találjuk. Amennyiben ezt megjegyzésbe helyezzük, a terminálok ***XF86Config*** állományának (amelyet a kiszolgáló ***/usr/X11R6/configs/iphex*** állományában találunk) ***Files*** szakasza mindössze egyetlen ***FontPath*** bejegyzést kell tartalmazzon a szokásos fél tucat helyett, amint azt a *3. listában* olvashatjuk (ahol egyébként feltételezzük, hogy a kiszolgáló IP száma ***192.168.1.1***).
Végül a kiszolgálót be kell állítanunk, hogy támogassa ***/usr*** fájlrendszerének csak olvasható az ***NFS*** megosztását a terminálok részére, hiszen ez az a hely ahol a terminálok ***XFree86*** kiszolgálójukat keresik.

Néhány szó a biztonságról

Az ***X*** terminálok futtatása során számos biztonsági kérdésre kell ügyelnünk. Először is, az ***xdm*** és ***xfs*** beállításokban végzett változtatásaink elég egyértelműen visszafordítottak bizonyos dolgokat amit egyébként a kiszolgáló biztonsága növelése érdekében tettek. Ezen felül, a cikkben leírt változat semmilyen forgalmat sem titkosít. A terminál minden billentyűleütése kódolatlanul megy keresztül a hálózaton. Az ***X*** teminálokkal dolgozó egyetlen biztonságosnak mondható módszer, ha valamennyit olyan privát ***LAN*** hálózatra helyezzük, amelyet kizárólag az ***X*** terminálok használnak és nem lehet belőle az Internet-re kijutni. A terminálok és a kiszolgáló egyetlen kártyáján kívül senki más nem csatlakozhat erre a ***LAN***-ra.

Letölthető készletek

A nyomtatott anyag korlátozott mérete miatt ez a cikk csak magas szintű áttekintést nyújthatott a Linux gépek lemez nélküli indításáról és ***X*** terminállá alakításáról, így nem mentünk bele a pontos megvalósítás apró részleteibe. Az érdeklődő olvasók a szerző honlapjáról letölthetik az ***X Terminal Kit*** készletet. A készletben parancsfájlokat, ***Makefile*** állományokat és ***README***-ket találhatnak amelyek végigvezetnek a néha kicsit bonyolulttá váló folyamaton. Ezen felül a cikkben bemutatott program az Internet különféle helyeiről származik. Valamennyi hely igen részletes leírással rendelkezik saját csomagjait illetően. További útmutatást a hálózati forrásokban találunk.

Linux Journal 2005. február, 130. szám

A cikk forrásai:
➲ ***www.linuxjournal.com/article/7924***

Chip Coldwell
(coldwell@physics.harvard.edu)
A Harvard Egyetem Fizika Tanszékének rendszeradminisztrátora. Ha éppen nem valamilyen számítógéppel játszadozik, általában biciklin találjuk, vagy menyasszonyának, Cindynek társaságát élvezi.