# Vékonykliensek használata az LTSP segítségével

Gergi Miklós
<mgergi@math.bme.hu>

Budapesti Műszaki és Gazdaságtudományi Egyetem
Matematika Intézet

**Kivonat**

Mit kezdjünk az elavult, régi számítógépeinkkel? Hogyan csináljunk magunknak csendes számítógépes munkahelyet? A két kérdés mindegyikére egy lehetséges jó válasz: használjunk vékonykliensenket.

A BME Matematika Intézetében 2004 őszén határoztuk el, hogy géptermeink felújításánál vékonykliens megoldást fogunk használni. Jelenleg mindkét géptermünkben teljesen csendes, mozgó alkatrészek nélküli gépek vannak elhelyezve, a felhasználók egyetlen gombnyomással, újraindítás nélkül választhatnak, hogy a Linuxot vagy Windows 2003-at futtató terminál-szerverünket szeretnének használni. Mindkét rendszer alatt lehetőségük nyílik USB-portos adathordozók használatára, és akár zenét is hallgathatnak.

A géptermek kialakítása mellett egyre több kiöregedett számítógépet is kliensként használunk.

A cikkem a rendszer kialakítását, a felmerült buktatókat és az ezekre talált megoldásokat mutatja be.

## Tartalomjegyzék

# 1. Bevezetés

A BME Matematika Intézetében 2004 végén döntöttünk arról hogy a hallgatói géptermeinket felújítjuk. Ennek során jelentős szempont volt, hogy minél értékállóbb megoldást találjunk, így mozgó alkatrész nélküli vékonykliens gépek beszerzése mellett döntöttünk. A mozgó alkatrészek hiánya várhatóan csökkenti a meghibásodások gyakoriságát, és további előny, hogy a géptermek teljesen csendessé váltak, és a hőtermelés is „elviselhetővé” vált (korábban a felhasználókat a nyári időszakban igencsak megviselte az a néhány plusz °C, amennyivel a gépteremben melegebb volt a többi helyiséghez képest).

Kulcsrakész vékonykliens megoldásokat sok cég kínál, hogy mi végül miért az LTSP rendszert választottuk, annak a legfőbb okai a következők:

1. Lehetőség van más beszerzésből származó, más gyártótól származó gépekkel bővíteni a kliensek körét, így gyártófüggetlenné válunk.
2. A régi, kiöregedett gépek is klienssé alakíthatóak.
3. Tud kapcsolódni Linuxhoz és Windowshoz is, egyetlen billentyűkombinációval, egy másodperc alatt válthatnuk a két operációs rendszer között.
4. Csak a saját tudásunk szab határt a megvalósítható lehetőségeknek.

Természetesen hátránya is van ennek a megoldásnak:

1. Nem egy kész, működő megoldást kapunk, hanem magunknak kell létrehoznunk a kívánt rendszert.
2. Ha nem üzemel, nincs hol reklamálni, legfeljebb a fórumokat bújhatjuk.
3. A dokumentáció rosszul karbantartott, hiányos, nem követi a frissülést. Sok esetben nem lehet tudni, hogy amit olvasunk, az éppen melyik verzióra igaz, és melyikre nem.

Az LTSP (Linux Terminal Server Project) egy Linux alá fejlesztett programcsomag, amit arra találtak ki, hogy megkönnyítse a kis teljesítményű számítógépek, vékonykliensek csatlakoztatását Linuxot és/vagy Windowst futtató terminál szerver gépekhez. A project fejlesztését 1999-ben kezdte Jim McQuillan és Ron Colcernian. Az aktuális verzió, valamint dokumentációk, wiki stb. elérhetőek a projekt honlapján [2].

# 2. Telepítés

## 2.1. Hardver- és szoftverigény

**Kliens-hardver** ♦ A fejlesztők szerint már egy 16 MB memóriával rendelkező 66 MHz-es 486-os gép is elegendően erős ahhoz, hogy kliensgépként használjuk, 24 MB memóriánál és P133 processzornál jobbra pedig nincs is szükség. Processzor szempontjából ez valóban így is van, viszont ha egy gépet egyidejűleg Linux és Windows kliensként is akarjuk használni, hanggal, nyomtatóval, és ha nem akarunk swapot használni (amihez használhatunk helyi merevlemezt, de lehetőség van hálózaton keresztüli swapolásra is), akkor szükség lehet 64 MB memóriára is.

A vásárlás előtt több gyártó többféle termékét kipróbáltuk, és még a kapható leggyengébb modellek is elegendőnek bizonyultak. Problémák csak az az AMD Geode processzorral szerelt kliensgépeknél merültek fel, ahol az X.Org nem támogatja az integrált monitorvezérlőt, a gyártó pedig csak egy X.Org verzióhoz adott ki patchet [1].

Eddig csak x86 architektúrájú gépeket teszteltünk kliensként. Néhány további architektúrához található ugyan régebbi LTSP változat, például PPC-hez vagy SPARC-hoz [3], a több vékonykliens által is használt AMD Alchemy processzorcsaládot (MIPS architektúra) az LTSP még nem támogatja.

**Szerver-szoftver** ♦ Amennyiben diskless klienseket akarunk létrehozni, akkor szükség van a PXE (Preboot Execution Environment) bootoláshoz egy dhcp- és egy tftp-szerverre. Debian Linux alatt több tftpd csomagot találunk, sajnos pont az alapértelmezett *tftpd* csomag nem megfelelő az LTSP számára, helyette a *tftpd-hpa* csomagot kell használnunk. A *tftpd-hpa* tud futni daemonként, vagy meghívhatja az inetd. Mivel viszonylag ritkán van szükség a futására, ezért használhatjuk nyugodtan az inetd által meghívva. Installálnunk kell egy nfs-szervert is, erre a célra mi az *nfs-kernel-server* csomagot használjuk. Szükség van még egy X Display Managerre, amit tetszőlegesen választhatunk (*gdm*, *kdm*, *wdm*, *xdm* . . . ). Szükség lehet még

egy fontszerverre, különösen, ha olyan alkalmazásaink is vannak, amelyek saját fontokat használnak, mint pl. a Mathematica. Erre a célra tökéletesen megfelel az *xfs* csomag. Ha hangokat is szeretnénk hallani a kliensünkön, akkor a terminál szerverünkre az *esound-clients* csomagot telepítsük.

**Szerver-hardver** ♦ Alapértelmezett telepítés során elegendő egy darab Linux rendszert futtató gép, ami egyrészt ellátja a kliensgépek működtetéséhez szükséges feladatokat, valamint terminál szerverként üzemel, vagyis valójában erre jelentkeznek be a felhasználók, ezen futtatják a programjaikat. Ennek méretezésekor gondolnunk kell arra, hogy ezt a gépet egy időben több felhasználó is használja, így ennek megfelelő mennyiségű memóriával lássuk el. Hasonló okokból többprocesszoros gép ajánlott.

A felsorolt feladatokat azonban érdemes két gépre szétbontani, amiből az az egyik egy kis teljesítményű gép, és pusztán a kliensek működtetéséhez szükséges feladatokat (dhcp, tftp, nfs, syslog, swap) látja el, a másik gép pedig csak terminál szerverként működik. Ebben az esetben a linuxos terminál szerver meghibásodása esetén még mindig tudjuk a kliensgépeinket Windows kliensként használni. Sőt, mivel az első szervergépünk egy egyszerű, olcsó gép, így kevés ráfordítással tudunk belőle HA-clustert építeni, ami tovább javítja a vékonykliens-rendszerünk megbízhatóságát.

A Linuxos szervereink mellé lehetőség van beállítani egy Windows 2003 vagy Windows 2000 szervert is, amihez szintén tudnak csatlakozni a klienseink.

## 2.2. Telepítés az `ltspadmin` és `ltspcfg` segítségével

Ha az xdm szerver, és a klienseket egyéb szempontból kiszolgáló (dhcp, tftp, nfs) szerver ugyanaz a gép, akkor egy kényelmes telepítési mód az *ltsp-utils* csomagban található `ltspadmin` parancs használata, ami letölti és kicsomagolja az ltsp rendszer fájljait, majd az `ltspcfg` programot meghívva annak bekonfigurálásában is segít. Fontos, hogy mindig a legfrissebb *ltsp-utils* csomagot használjuk! Ha a disztribúciónk által felkínált csomag nem a legfrissebb, akkor töltsük le az új verziót az LTSP honlapjáról!

Telepítsük fel csomagkezelőnkkel a szükséges programokat (*dhcpd*, *tftpd-hpa*, *nfs-kernel-szerver*).

Az `ltspadmin` parancsot elindítva a menüből válasszuk ki a `Configure the installer options` pontot, ahol megadhatjuk, hogy honnan kívánjuk letölteni az ltsp-csomagokat, melyik könyvtárat akarjuk a kliensek gyökérkönyvtáraként használni, és beállíthatjuk a proxynkat is, ha van.

Ezt követően indítsuk el az `Install/Update LTSP Packages` menüpontot, amiben kiválaszthatjuk, hogy az LTSP mely komponenseit telepítsük. Általában az *ltsp_core*, *ltsp_kernel*, *ltsp_libusb*, *ltsp_localdevs*, *ltsp_rdesktop*, *ltsp_x_core* komponensekre lesz szükségünk. Ha nem használunk fontszervert, akkor az *ltsp_x_addtl_fonts*-ot is érdemes telepíteni. Jó tudni, hogy *xfs* konfigurálásában az `ltspcfg` nem nyújt segítséget!

Végül indítsuk el a `Configure LTSP` menüpontot. Ekkor az `ltspcfg` leteszteli, hogy fut-e az összes szükséges szolgáltatás, majd a `Configure the services manually` pont alatt egyenként beállíthatjuk ezeket.

## 2.3. Telepítés kézzel

Ha kézi telepítést választjuk, akkor egyenként kell a különböző konfigurációs fájlokat megszerkesztenünk. Ez egy kicsit több tapasztalatot igényel, mint az `ltspcfg` hasznlata, de nagyobb rálátást nyújt a rendszer működésére, és finomabb beállításokra ad lehetőséget.

**DHCP** ♦ A DHCP daemon konfigurációs fájljában, a `/etc/default/dhcpd` fájlban tudjuk megadni, hogy melyik hálózati interfészen fusson a dhcp szolgáltatás. Ezután pedig a `/etc/dhcpd.conf` fájlt kell létrehoznunk. Ahhoz, hogy a klienseink távolról is egyértelműen azonosíthatóak legyenek, fontos, hogy fix IP-címet kapjanak. Egy jó konfigurációs fájl például a következő:

```
option subnet-mask         255.255.255.0;
option broadcast-address   192.168.1.255;
option routers             192.168.1.1;
option domain-name-servers 192.168.1.1;
option domain-name         "ltsp";
allow bootp;
allow booting;
```

```
use-host-decl-names         on;
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.200 192.168.1.250;
  next-server                192.168.1.1;
  filename                   "lts/2.6.17.8-ltsp-1/pxelinux.0"
  option root-path           "192.168.1.1:/usr/local/ltsp/i386"
}
host labor1 {
  hardware ethernet 00:40:63:DA:C3:76;
  fixed-address 192.168.1.11;
}
host labor2 {
  hardware ethernet 00:40:63:DA:C3:76;
  fixed-address 192.168.1.12;
}
```

Az első sorokban beállítjuk a hálózat alapvető paramétereit. A `bootp` és a `booting` a hálózatról való bootoláshoz szükséges jogosultságok. A mostani dhcp-szerverekben alapértelmezésben engedélyezve vannak, de a régi rossz tapasztalatok miatt jobb határozottan engedélyezni. A `use-host-decl-names` bekapcsolása azért fontos, mert így a fix IP-címet használó kliensgépek a nevüket is megkapják dhcp-n keresztül.

Létrehozunk egy subnetet, amiben a 192.168.1.200 és a 192.168.1.250 tartományban dinamikusan osztunk IP-címeket. Az új, fix IP-címet még nem kapó kliensek ebből a tartományból fognak IP-t kapni. A `next-server` a tftp-szerver címét addja meg, a `filename` pedig azt, hogy mit is kell arról a gépről bebootolni. A `root-path` paraméterben pedig az nfs-en exportált gyökérkönyvtár címét állítjuk be.

**TFTP** ♦ Mivel a tftpd-t az inetd indítja, ezért a beállítását a `/etc/inetd.conf` fájlban lehet elvégezni az alábbi sorral:

```
tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /tftpboot
```

**NFS** ♦ Az NFS-szerver konfigurációját a `/etc/exports` fájlban módosíthatjuk. Mindössze egyetlen sorra van szükség benne, mit osztunk meg, melyik gépeknek, és milyen paraméterekkel:

```
/usr/local/ltsp 192.168.1.1/255.255.255.0(ro,no_root_squash,sync)
```

**XDM** ♦ Ha gdm-et használunk,akkor a `/etc/gdm/gdm.conf` fájlban kell megkeresnünk az `[xdmcp]` fejezetet és ott:

```
[xdmcp]
Enable=true
MaxSessions=50
```

Az utóbbi sorral az egyidejűleg engedélyezett kapcsolatok számát növeljük meg, ha az kevesebb a szükségesnél.

Ötödik lépésként konfiguráljuk a fontszerverünket. Ennek a beállításai a `/etc/X11/fs/config` fájlban találhatóak. Itt a legfontosabb feladatunk a TCP-porton való hallgatózás tiltásának megszüntetése, valamint a fontokat tartalmazó könyvtárak felsorolása:

```
#no-listen = tcp
catalogue = /usr/lib/X11/fonts/misc/,
     /usr/lib/X11/fonts/100dpi/:unscaled,
     /usr/lib/X11/fonts/75dpi/:unscaled,
     /usr/lib/X11/fonts/Type1/,
     /usr/lib/X11/fonts/Speedo/,
     /usr/lib/X11/fonts/100dpi/,
     /usr/lib/X11/fonts/75dpi/,
     /usr/local/Wolfram/Mathematica/5.2/SystemFiles/Fonts/AFM/,
     /usr/local/Wolfram/Mathematica/5.2/SystemFiles/Fonts/BDF/,
     /usr/local/Wolfram/Mathematica/5.2/SystemFiles/Fonts/Type1/
```

Végül vegyük fel a kliensgépeinket a `/etc/hosts` fájlba, és az `ltspadmin` programot használva az előző fejezetben leírt módon töltsük le és csomagoljuk ki az ltsp szükséges komponenseit a `/usr/local/ltsp` könyvtárba. Ellenőrizzük, hogy a kernel, az initramfs, és minden a hálózatrol bootoláshoz szükséges fájl az `/tftpboot/lts` könyvtárban, a megfelelő helyen található.

## 2.4. Konfigurálás

A kliensek konfigurálása a nfs-en exportált gyökérkönvtárban levő `etc/lts.conf` szerkesztésével történik. A konfigurációs fájlban megadhatunk globális, minden kliensre vonatkozó beállításokat, mint például a terminálszerverek, vagy a fontszerver IP-címe, és megadhatunk kliensfüggő beállításokat, mint pl. a monitor felbontása, vagy a klienshez csatlakozó egér típusa.

A kliensgépeket nevük, IP-címük, vagy hardvercímük alapján azonosíthatjuk. Egy egyszerű konfigurációs fájl az alábbi módon néz ki:

```
[Default]
  SERVER             = 192.168.1.1
  RDP_SERVER         = 192.168.1.2
  XSERVER            = auto
  USE_XFS            = Y
  X_MODE_0           = 1280x1024
  X_MOUSE_PROTOCOL   = ImPS/2
  SCREEN_01          = telnet
  SCREEN_02          = telnet
  SCREEN_03          = telnet
  SCREEN_04          = telnet
  SCREEN_05          = telnet
  SCREEN_06          = rdesktop
  SCREEN_07          = startx
[192.168.1.18]
  XSERVER            = ati
  X_HORZSYNC         = 30-83
  X_VERTREFRESH      = 56-75
  X_MODE_0           = 1024x768
  SCREEN_01          = shell
```

Értelmezzük ezt a konfigurációt! Az alapértelmezett szervergép minden kliens számára a 192.168.1.1. Ezt a gépet használják xdm-szervernek, xfs-szervenek, ndb-szervernek, erre küldik a syslogot, és a telnetek is ehhez a géphez csatlakoznak. Mivel az rdp szervert külön megadtuk, így az rdesktopok a 192.168.1.2 géphez próbálnak csatlakozni.

Az `XSERVER=auto` beállítással megadtuk, hogy az induló X automatikusan próbálja meg eldönteni, melyik driver való a kliens monitorvezérlő kártyájához. Ez általában jól működik, de ha valamelyik kártyánál mégis hibásan dönt, akkor megadhatjuk közvetlenül a driver típusát, mint ahogy ezt a 192.168.1.18 gépnél tettük. A `USE_XFS=Y` jelzi, hogy szeretnénk fontszervert használni. Az `X_MODE_0` értéke adja meg, hogy mi a monitor elsődleges felbontása. Ha valami szokatlan felbontást szeretnénk használni, akkor az `X_MODE_0` értékének megadhatunk egy teljes ModeLine sort is. További felbontásokat a `X_MODE_1` és az `X_MODE_2` változókkal definiálhatunk. Mint a 192.186.1.18 gépnél láthatjuk, ha gondban vagyunk a monitor beállításával, akkor a vertikális és horizontális szinkron frekvenciákat is beállíthatjuk. Az `X_MOUSE_PROTOCOL` az egér típusát határozza meg az X konfigurációjához. Az ImPS/2 a legtöbb mai kétgombos-egygörgős egérnek megfelel.

A beállítások talán leglényegesebb része a `SCREEN` változók megadása. Ezek a változók határozzák meg, hogy melyik virtuális terminálon milyen program fog elindulni. Négy lehetőség közül választhatunk mindegyik screen esetében.

1. shell: az adott virtuális terminálon egy root-shell fog elindulni. Általában hibakeresési célra használjuk, és csak egy-két kliensgépen engedélyezzük.

2. telnet: ez egy telnet kapcsolatot nyit a `TELNET_HOST` változóban, vagy annak hiányában a `SERVER` változóban megadott gépre.

3. startx: csatlakozik az `XDM_SERVER` változóban, vagy annak hiányában a `SERVER` változóban megadott Linux terminálszerverhez.

4. rdesktop: csatlakozik az `RDP_SERVER` változóban, vagy annak hiányában a `SERVER` változóban megadott Windows terminálszerverhez.

A `/etc/lts.conf.readme` fájlban további konfigurációs lehetőségeket is találunk, azonban ez a fájl nem követi az LTSP fejlődését, így egyrészt vannak benne időközben megszűnt konfigurációs változók, és hiányoznak bizonyos újabb lehetőségek. Ahhoz, hogy megismerjük az összes kínálkozó lehetőséget, talán a leghatásosabb módszer az, ha végignézzük, hogyan is indul egy a kliens, és az indítófájlokban megtaláljuk a „titkos” változókat.

# 3. Hogyan működik?

1. A kliens tftp-n keresztül megkapja a kernelt és az initial ramdisket. (Ha a hálókártya nem támogatja a hálózatrol bootolást, akkor a kernelt és az initrd-t elhelyezhetjük egy lokális meghajtón is, és valamilyen bootmanagerrel betölthetjük.) Ezekkel elindul a bootolás. Gyökérkönyvtárként az nfs által exportált könyvtárrendszert csatolja fel.

2. Lefut az initial ramdiszken levő `init` fájl. Ez többek között létrehoz egy ramdiszket, ami a továbbiakban a rendszer gyökérkönyvtára lesz, ennek a /nfsroot könyvtárába felcsatolja az exportált könyvtárrendszert, majd linkeket csinál a legfontosabb könyvtárakra.

3. Lefut a `/etc/rc.early_sysinit`, ami felcsatolja a /proc és a /sys könyvtárakat, elindítja a udev daemont, és létrehozza az `lts.conf` alapján az inittab fájlt.

4. Elindul az init program. Az init indítja a `/etc/rc.sysinit` szkriptet.

5. A `/etc/rc.sysinit` az `lts.conf`-ban beállítottak alapján elindítja a hálózaton keresztüli swapolást, betölti a szükséges kernelmodulokat, létrehoz sok szükséges könyvtárat, elindítja a syslogot, az xinetd-t, végrehajtja a `/etc/rc.local` fájlt, elindítja az snmp, a sound, és a localdev szkripteket.

6. Az init elindítja az ltspinfod daemont, a screen-eket, és nyomtató szervert. Ha ezek közül bármelyik leáll, azt azonnal újraindítja.

# 4. Extrák

## 4.1. Billentyűzet

Furcsának tűnhet, hogy a billentyűzetet az extra opciók között látjuk, de ha Windows Terminal Serverhez is szeretnénk csatlakoztatni a klienseinket, akkor sajnos billentyűzet területén is van tennivalónk. Az LTSP rendszerben lévő rdesktop nem támogatja ugyanis a magyar billentyűzetkiosztásnál szinte elengedhetetlenül szükséges AltGr módosító billentyűt, és ezt a problémát sajnos nem is tudjuk pusztán konfigurálással megoldani. Egy Szabó Péter által megírt patch alkalmazásával újra kell fordítanunk az rdesktopot, és az új rdesktop csomaggal már lehetőség van a magyar billentyűzet használatára is [4]. A részletes megoldás megtalálható a „Készítsünk saját LTSP-t!” című fejezetben a 25. oldalon.

## 4.2. Nyomtató

A klienshez csatlakoztatott nyomtatókat hálózati nyomtatóként tudjuk használni a terminálszervereinkről, vagy egyéb gépekről. Ennek a beüzemeléséhez az `lts.conf` fájlt kell módosítanunk.

```
PRINTER_0_DEVICE = /dev/lp0
PRINTER_0_TYPE = P
```

```
PRINTER_0_PORT = 9100
PRINTER_1_DEVICE = /dev/usb/lp0
PRINTER_1_TYPE = U
PRINTER_1_PORT = 9101
```

A fenti példában két nyomtatót konfiguráltunk be a kliensen, a printer_0 egy párhuzamos portra kötött (`TYPE=P`) nyomtató, amit a 9100-as tcp porton keresztül lehet elérni, a második nyomtatónk pedig egy USB porton csatlatozó (`TYPE=U`) nyomtató, amit a 9101-es tcp porton tettünk elérhetővé.

Ha egy nyomtatót megosztunk a fent leírt módon a 192.168.1.28 gép 9100-as portján, és azt CUPS-on keresztül szeretnénk elérni, akkor a Device URI megadásakor a `socket://192.168.1.28:9100` értéket kell beállítani.

## 4.3. Hang

Az LTSP 4.2-es verziójában áttértek a 2.6-os kernelre, aminek következtében az eddigi OSD hangrendszer helyett az ALSA hangrendszert támogatja a kernel. Nem kerültek bele viszont az LTSP-be az ehhez szükséges programok, (alsa-utils, alsamixer, stb...), így ezeket külön kell telepítenünk a klienseket kiszolgáló nfs-szerveren. Ehhez az LTSP-esd-alsa csomagra lesz szükségünk, amit a `http://prdownloads.sourceforge.net/symbiont/LTSP-esd-alsa.tgz?download` címről lehet letölteni. A letöltött csomagot kitömörítve az install paranccsal telepíthetjük fel.

Ezek után az `lts.conf` fájlban a megfelelő klienshez a `SOUND=Y` bejegyzést kell beírni.

A Windows 2003 szerver alapbeállításban tiltja a hang átirányítását a kliensre. A tiltást a Microsft Management Console (mmc.exe) segítségével kapcsolhatjuk ki a biztonsági házirend megfelelő beállításával. (Számítógép konfigurációja / Felügyeleti sablonok / Windows-összetevők / Terminálszolgáltatások / Ügyfél/kiszolgáló adatai átirányításának tiltása / Hangátirányítások engedélyezése) [5].

Az rdesktop a `-r sound:local` kapcsolómegadásával veszi át a hangot a terminálszervertől. Ezt az `lts.conf` fájlban az `RDP_OPTIONS` változóban adhatjuk meg.

A Windows hangerőszabályzója nem működik az rdesktopon keresztül, ezért érdemes egy külön screent létrehozni, amiben az alsamixert futtatjuk.

## 4.4. USB-storage

Internet hozzáféréssel nem rendelkező kollégák számára igen fontos dolog, hogy az eléjük rakott kliensgép alkalmas legyen valamilyen módon az otthonról hozott állományok feltöltésére, illetve a terminál szerveren létrehozott állományok lementésére. Ennek megoldására az LTSP fejlődése során rengeteg ötlet és próbálkozás volt (például NFS-en vagy Samba-n megosztották az automaikusan felcsatolt adathordozókat.) Az LTSP jelenlegi megoldása egy saját hálózati fájlrendszer, az ltspfs, és hozzá kapcsolódó L-BUS rendszer.

Az `ltspfsd` (LTSP FileSystem Daemon) a kliens gépen fut, és annak könyvtárait elérhetővé teszi a terminál szerver számára. Az terminál szerveren az `ltspfs` paranccsal csatolhatjuk fel a klienshez csatlakoztatott adathordozót. Az `ltspfsd` igyekszik a helyi adattárolókat lecsatolt állapotban tartani. Ha 2 másodpercen keresztül semmiféle fájlművelet nem történik, akkor az adathordozót leválasztja, majd ha újabb fájlműveletre kap utasítást, akkor újra mountolja. Ezzel behelyezett írható adathordozók eltávolítását könnyíti meg, hiszen a fájlműveletektől eltekintve többnyire szinkronban van a fájlrendszer, sőt, fel sincsen csatolva.

Az L-BUS rendszer két daemonból áll, az egyik a kliensen futó `lbuscd` (L-BUS Client Daemon), a másik a terminál szerveren a felhasználó nevében futó `lbussd` (L-BUS Server Daemon). Egy új eszköz csatlakoztatásakor a `lbuscd` jelez az `lbussd`-nek, ami elindítja a `lbus_event_handler.sh` szkriptet. Ez a szkript létrehoz a felhasználó home könyvtárán belül egy könyvtárat az új adathordozónak, és az `ltspfs` segítségével oda felcsatolja. Ha az ablakkezelő támogatja, akkor még egy ikont is feldob az asztalra. Az adathordozó eltávolításánál is ugyanilyen módon hajtódik végre az umount, a célkönyvtár törlése, és az ikon eltávolítása.

Mit kell tennünk, hogy mindez működjön?

Először is kapcsoljuk be a helyi adathordozók támogatását a klienseken. Ehhez a `/etc/lts.conf` fájlba kell beírnunk a `LOCAL_DEVICES=Y` sort. Ellenőrizzük, hogy a kliensen a `hostname` parancsot kiadva ugyanazt a nevet kapjuk, mint amit terminál szerverre a kliensről belépve a `$DISPLAY` változóban találunk. Ha a

két név nem egyezik, akkor nézzük át a `/etc/hosts` fájlt és ellenőrizzük a dhcpd konfigját, hogy átadja-e a klienseknek a hostname-et.

Ha ezzel megvagyunk, akkor a kliensünk már készen is van. A terminál szerveren szükségünk van a kernelben a FUSE támogatásra és szükségünk van a *fuse-utils* csomagra. Adjuk hozzá a felhasználóinkat a fuse csoporthoz. Telepítsük a *libx11-protocol-perl* csomagot, amire a `lbussd`-nek van szüksége, majd töltsük le és telepítsük az LTSP localdev support csomagot, amit a `http://ltsp.mirrors.tds.net/pub/ltsp/utils/` címen találunk *ltsp-server-pkg* néven. Ez a csomag tartalmazza az `lbussd`-t, és az `ltspfs`-t.[1] Végül ellenőrizzük, hogy a `/etc/X11/Xsession.d` könyvtárban létrejött-e a `51lbus-start` fájl, ami arra szolgál, hogy bejelentkezéskor automatikusan elindítsa a felhasználónak az `lbussd` daemont. Ha ez a fájl hiányzik, , mert például tgz-ből telepítettünk, akkor az `xsession-lbus-start` állományt linkeljük be ezen a néven.

## 4.5. Monitor lekapcsolása

A kliensgépek, különösen a kis fogyasztású modern vékonykliensek általában napi 24 órában futnak, viszont ennek jelentős részében üresjáratban vannak. Jogos elvárás tehát, hogy a montitor energiagazdálkodási lehetőségeit kihasználják. Ennek beállításához ismét az lts.conf fájlt kell szerkesztenünk:

```
X_DPMS = Y
X_DPMS_STANDBYTIME = 3
X_DPMS_SUSPENDTIME = 6
X_DPMS_OFFTIME = 10
```

Ezekkel a beállításokkal, ha a klienst nem használjuk, a monitor 3 perc után standby, 6 perc után suspend módba kapcsol, majd további 4 perc múlva kikapcsolja magát.

## 4.6. Saját screen-ek használata

Az LTSP négy beépített screen típussal rendelkezik (shell, telnet, startx, rdesktop), ami mellé saját egyedi screen típusokat írhatunk. A hangkezelésnél láttuk, hogy hasznos, ha a felhasználók tudják használni a kliens `alsamixer` parancsát, vagy kirakhatunk egy `top`-ot valamelyik virtuális terminálra, vagy indíthatunk egy `tcd`-t[2], amivel az audio CD-ket lehet lejátszani.

Az alsamixerhez hozzunk létre egy alsamixer nevű fájlt a `/etc/screen.d` könyvtárban az alábbi tartalommal:

```
#!/bin/bash
/bin/alsamixer
/bin/sleep 1
```

A `sleep`-re azért van szükség, hogy ha kilépnek a programból, akkor az kis várakozással induljon újra, és ne lehessen leterhelni a rendszert a túlságosan gyakori újraindítással.

Ugyanez a top-hoz:

```
#!/bin/bash
/bin/cp /nfsroot/root/.toprc /.toprc
/usr/bin/top -s
/bin/sleep 1
```

Érdemes megfigyelni, hogy a `top` a gyökérkönyvtárban keresi a konfigurációs állományát, amit oda kell másolni a számára, valamint, hogy a `top` parancsot `-s` paraméterrel indítjuk, így bár rendszergazda tulajdonú process, mégsem lehet leállítani vele egy process futását, vagy módosítani annak prioritását.

Az itt említett top és tcd programok sajnos nem szerepelnek az LTSP-ben, ezért vagy statikusan fordított binársként kell elhelyeznünk a /usr/bin könyvtárban, vagy saját csomagot fordítunk belőlük.

[1] Ebben a csomagban található meg a network block device-on keresztüli swapoláshoz szükséges `ltspswapd` program is.
[2] A `tcd` egy karakteres alapú audió CD lejátszó alkalmazás. (`http://www.nongnu.org/tcd/`)

## 4.7. Monitorozás, távoli adminisztrálás

**ltspinfo** ♦ A kliensgépek állapotát távolról is nyomon követhetjük, sőt megfelelő beállítások esetén még némi beavatkozásra is lehetőségünk nyílik. Az LTSP erre kifejlesztett alapértelmezett eszköze a kliensen futó ltspinfod daemon, és az ehhez kapcsolódó `ltspinfo` program, ami az *ltsp-utils* csomag része. Az `ltspinfo` paranccsal ellenőrizni tudjuk a kliensgép által használt konfigurációs változókat:

```
ltspinfo -h 192.168.1.32 -cfg XSERVER
ltspinfo -h 192.168.1.14 -cfg RDP_OPTIONS
```

Lehetőség van a kliensgép /proc könyvtárában lévő fájlok kiíratására:

```
ltspinfo -h 192.168.1.32 -proc cpuinfo
ltspinfo -h 192.168.1.32 -proc meminfo
```

Kikapcsolhatjuk, vagy újraindíthatjuk a kliensgépet:

```
ltspinfo -h 192.168.1.11 -s
ltspinfo -h 192.168.1.12 -r
```

A /proc könyvtár távoli olvasásához az `ALLOW_PROCREAD=Y`, a távoli leállításhoz és újraindításhoz pedig az `ALLOW_SHUTDOWN=Y` megadása szükséges az lst.conf fájlban.

**snmp** ♦ A monitorozás egy másik lehetséges módja, ha a kliensgépünkön snmp daemont indítunk. Ehhez az `lts.conf` fájlba kell beírnunk, hogy `SNMPD=Y`. Az elindított snmp daemonhoz csatlakozhatunk közvetlenül az `snmpwalk` programmal, vagy készíthetünk a kinyert adatokból grafikont az *mrtg* vagy a *munin* használatával.

A kliens *load average* értékeinek lekérdezése:

```
snmpwalk -Os -c public -v 1 192.168.1.32 laLoad
```

**ssh** ♦ Az LTSP tartalmaz egy sshd daemont is, viszont az alapbeállításokkal ez csak akkor indul el, ha a `LOCAL_APPS=Y` beállítást használjuk az `lts.conf` fájlban. Ez viszont elindít mindent, ami a kliensen helyileg futó alkalmazások használatához szükséges (nfs-home, NIS, stb.), amikre viszont nekünk nincsen szükségünk. A megoldás a `/etc/rc.locale` fájl átírása. Ez a fájl alapértelmezésben nem is létezik, most viszont hozzuk létre, és töltsük fel az alábbi tartalommal:

```
# Start sshd
if [ "${SSHD}" = "Y" ] ; then
  /bin/cp /nfsroot/root/.ssh/ssh_host_*sa_key /tmp/
  /bin/chmod 600 /tmp/ssh_host_*sa_key
  echo "Starting sshd..."
  sshd
fi
```

Módosítanunk kell még a kliensek `etc/ssh/sshd_config` állományát is:

```
HostKey /tmp/ssh_host_rsa_key
HostKey /tmp/ssh_host_dsa_key
SyslogFacility AUTH
LogLevel INFO
PermitRootLogin yes
StrictModes yes
PubkeyAuthentication yes
AuthorizedKeysFile      %h/.ssh/authorized_keys
PasswordAuthentication no
KeepAlive yes
```

A kliensgép által használandó ssh_host kulcsokat másoljuk be az exportált gyökérkönyvtár `root/.ssh` könyvtárába. A `/tmp`-be való kimásolásra azért van szükség, mert a fájl jogosultságai csak így állíthatóak be a szükséges értékre.

Ezzel elértük, hogy az `lts.conf` fájlban az `SSHD=Y` beállításával a kliensgépen is tudunk ssh daemont indítani. Mivel a kliens rendszergazdáját nem lehet jelszó alapján autentikálni, ezért kulcs alapú autentikációt kell megvalósítanunk. Ehhez helyezzük el a publikus ssh-kulcsunkat a kliensgép root/.ssh könyvtárában az `authorized_keys` fájlba.

## 5. Biztonság

Biztonsági szempontból az LTSP sok támadási pontot nyújt. Csak tűzfallal szeparált lokális hálózatban használjuk. Szükséges a megfelelő policy-k megalkotása és betartása, bizonyos esetekben pedig akár egyes szolgáltatások leállítása is indokolt lehet.

Javaslom a szervergépek valamelyikén futtatni az arpwatch programot, aminek segítségével azonnali értesítést kapunk, ha a hálózatba új IP-cím–hardvercím pár jelenik meg. A bejövő leveleket procmaillel feldolgozva némi szkriptelés árán akár azonnali védelmi folyamatot indíthatunk ebben az esetben, például módosíthatjuk a tűzfal beállításait.

Vegyük szemügyre az LTSP fő biztonsági kockázatait:

**telnet** ♦ A biztonság egyik legégetőbb pontja a telnet használata. A loginnév/jelszó páros kódolatlanul utazik a hálózaton, nagyon könnyen lehallgatható. Ha nem tudjuk biztosítani, hogy a hálózatban csak általunk felügyelt gépek legyenek, akkor inkább ne is használjuk.

**XDMCP** ♦ Bár kevésbé rettegett protokoll, mint a telnet, valójában az xdmcp is ugyanúgy lehallgatható mint a telnet, a jelszavunk szintén kódolás nélkül utazik!

**nyomtatás** ♦ A klienseink 9100-as, 9101-es és 9102-es tcp portján várják a nyomtatni valót. Mivel semmiféle autentikálás nincsen, ezért bárki, aki ezekre a portokra adatot tud küldeni, az tud nyomtatni a klienshez csatlakozó nyomtatóra. Ha például postscript formátumot hardveresen tudó nyomtatónk van, akkor pusztán a cat és a telnet parncsokkal is nyomtathatnak az ügyesebb felhasználók:

```
cat print.ps | telnet 192.168.1.28 9100
```

**hang** ♦ az esd démon a kliensünk 16001-es tcp portján figyel, és a nyomtatáshoz hasonlóan itt sincsen semmiféle autentikáció. Bárki, aki erre a portra adatot tud küldeni, az „megszólaltathatja” a kliensünket. Ezt néhány esetben akár hasznos is lehet, például a saját mikrofonunkon bejövő jelet a kliens hangfalára továbbíthatjuk, így meglephetjük az órai munka helyett mással foglalkozó diákot. Általában ha a klienseink külön tűzfallal védett hálózatban vannak, akkor sem a hang, sem a nyomtató nem okoz gondot, amíg a felhasználóink megbízhatóak.

**ltspinfod** ♦ Nagyszerű dolog, hogy kliensgépeinket távolról leállíthatjuk, vagy újraindíthatjuk, de ha ezt az ltspinfo használatával akarjuk megoldani, akkor számítanunk kell arra, hogy ez sem autentikált módon történik, így gyakorlatilag bármelyik felhasználónk, vagy bárki, aki az adott portot (tcp/9200) eléri és kellő ismerettel rendelkezik, szintén leállíthatja vagy újraindíthatja a klienseinket. Ezt az opciót kapcsoljuk ki; ha nagyon szükséges az újraindítás, használjuk az SSH-t!

**SSH** ♦ Az előző fejezetekben láttuk, hogyan lehet megoldani, hogy a kliensgépeinken ssh daemon fusson. Azt is láttuk, milyen trükközéssel járt, hogy a daemon biztonságosan letároltnak gondolja a `host_key` fájlokat. Természetesen ez egyáltalán nem így van, és mivel a `host_key` gyakorlatilag szabadon elérhető, ha egy idegen gép felveszi a valamelyik kliensünk IP-címét, és ezt a `host_key`-t használja, akkor fel sem tűnik, hogy más gépre sikerült bejelentkeznünk.

## 6. Készítsünk saját LTSP-t!

Előbb vagy utóbb minden komolyabb LTSP-üzemeltető eljut arra a pontra, hogy további lehetőségekkel szeretné bővíteni a kiépített vékonykliens rendszerét. Ilyenkor először statikusan linkelt bináris programokkal

rakja teli az exportált /bin és /usr/bin könyvtárakat, majd egyre inkább library-ket is másol a fájlrendszerbe, végül elérkezik arra a pontra, hogy ezt így már nem lehet folytatni, és saját LTSP-t készít. Szerencsére, ez nem is olyan nehéz feladat, mint hinnénk.

## 6.1. Az LBE telepítése és használata

Ha további programokat szeretnénk a klienseken futtatni, vagy az alap LTSP rendszer valami miatt nem felel meg az igényeinknek (például patchelni kell az rdesktop programot a magyar billentyűzetért, vagy egy speciális videokártya miatt egy X.Org pathcre van szükségünk), akkor lehetőségünk van saját LTSP-t fordítani az LBE (LTSP Building Environment) segítségével. Ez az eszköz elsősorban az LTSP fejlesztők számára készült, de mi is használhatjuk saját LTSP rendszerünk létrehozására.

Bár az LBE-t letöltve és szétnézve a létrejött könyvtárrendszerben örömmel látjuk a `crosscomp-src` könyvtárat, sajnos a cross-compiling még nem működik, vagyis csak x86 architektúrán tudunk x86 alapú kliensek számára LTSP-t fordítani.

A fordításhoz szükségünk lesz kb. 5 GB helyre, egy 3.2 és 3.4 közötti verziójú *gcc*-re, *make*-re, *flex*-re, *bison*-ra, *autoconfig*-ra, *zlib1g-dev*-re és esetleg még néhány library-re, ami fordítás közben úgyis kiderül.

A saját LTSP rendszer elkészítéséhez először is töltsük le az LBE-t:

```
cvs -d :pserver:anonymous@cvs.ltsp.org:/usr/local/cvsroot checkout lbe
```

Ha ezzel megvagyunk, akkor lépjünk be az `lbe` könyvtárba, és töltsük le az aktuális LTSP forrást:

```
./build_all -fetch
```

Amikor ezzel is végeztünk, akkor jöhet a fordítás:

```
./build_all
```

Vagy ha többprocesszoros gépünk van, megadhatjuk a processzorok számát is:

```
./build_all -cpus=2
```

És itt most hátradőlhetünk, vagy elmehetünk ebédelni. Végül készítsük el a lefordított programokból az ltsp-csomagokat:

```
  ./build_all --makepkg
```

Az új csomagokat az `lbe/packages` könyvtárban találjuk, és akár rögtön innen telepíthetjük is őket az `ltspadmin` használatával.

## 6.2. Régi programok módosítása, új programok létrehozása

Az eddigiekben láttuk, hogyan tudjuk előállítani forrásból az eddig is használt LTSP rendszerünket, most végre foglalkozhatunk azzal a kérdéssel, hogy hogyan tudunk módosítani rajta, hogyan tudunk új programokat felvenni bele.

Nézzünk bele az `lbe/ltsp-src` könyvtárba! Minden egyes LTSP programcshoz találunk itt egy-egy saját kis könyvtárat. A program felépítését a `package.def` szabályozza.

Első példaként módosítsuk az rdesktop programot, hogy alkalmas legyen a magyar billentyűzetkiosztásban fontos szerepet kapó AltGr billentyű kezelésére. Lépjünk be az `rdesktop` könyvtárba, és töltsük le a szükséges patch-eket:

```
wget http://www.inf.bme.hu/~pts/pts-rdesktop-1.4.1-xkeymap-altgr-localstate.patch
wget http://www.inf.bme.hu/~pts/pts-xmodmap-raw-us.txt
wget http://www.inf.bme.hu/~pts/pts-rdesktop-keymap-raw.txt
```

Módosítsuk az `rdesktop.wrapper` fájlt az alábbi módon:

```
#!/bin/sh
/usr/X11R6/bin/xmodmap /usr/share/xmodmap.raw-us || exit
while true; do
  /usr/bin/rdesktop -k pts-raw "$@"
done
```

A `packages.def` fájlban be kell állítanunk, hogy az altgr-localstate patchet is szeretnénk használni, valamint az `INSTALL` változót kell módosítanunk, hogy a xmodmap és a keymap fájlok is belekerüljenek az LTSP csomagba:

```
PREPATCH2 = pts-rdesktop-1.4.1-xkeymap-altgr-localstate.patch
INSTALL = make DESTDIR=${LTSP_ROOT} prefix=/usr install && \
   cp ../rdesktop-screen ${LTSP_ROOT}/etc/screen.d/rdesktop && \
   cp ../rdesktop.wrapper ${LTSP_ROOT}/usr/bin/rdesktop.wrapper && \
   cp ../pts-xmodmap-raw-us.txt ${LTSP_ROOT}/usr/share/xmodmap/xmodmap.raw-us && \
   cp ../pts-rdesktop-keymap-raw.txt ${LTSP_ROOT}/usr/share/rdesktop/pts-raw
```

Természetesen lehetőség van egyetlen program újrafordítására is. Ehhez lépjünk az `lbe/ltsp-scr` könyvtárba, és futtassuk le a következő parancsokat:

```
./build --clean --only=rdesktop
./build --only=rdesktop --cpus=2
cd ..
./build_all --makekpkg
```

Ha egy új programból akarunk LTSP csomagot készíteni, akkor hozzunk létre számára egy könyvtárat az `lbe/ltsp-scr` könyvtárban, hozzuk létre a `package.def` fájlt a régebbi csomagoknál látható módon, és fordítsuk le a fent leírt parancsok segítségével a programunkat.

## Hivatkozások

[1] AMD Geode™ Processor Linux Drivers. URL `http://www.amd.com/us-en/ConnectivitySolutions/TechnicalResources/0,,50_2334_2452_11363,00.html`.

[2] Linux Terminal Server Project. URL `http://ltsp.org/`.

[3] LTSP clients on Ultra Sparc. URL `http://math.univ-lille1.fr/ltsp-sparc/`.

[4] Szabó Péter: rdesktop AltGr javítások, 2006. szeptember. URL `http://www.inf.bme.hu/~pts/pts-rdesktop-altgr-fixes.txt`.

[5] Microsoft Windows Server 2003 TechCenter: Ügyféleszköz-hozzárendelések beállításainak megadása, 2005. január. URL `http://www.microsoft.com/technet/prodtechnol/windowsserver2003/hu/library/ServerHelp/17d44d9a-cf4b-4a6a-94ec-093cb5f8b2b7.mspx?mfr=true`.