Digitális mozgóképsorok saját kezűleg

MEncoder – kikövezett út a hatékony tárolás felé.

Sorozatunk előző részében próbát tettem a mozgóképtárolás területén szerzett ismeretanyagunk rendszerezésére, valamint az elméleti alapok lefektetésére. Kísérletem sikerességének reményében a jelen és a következő írásban sokkal inkább a tettek mezejére szeretnék lépni, és bemutatom az általam ismert gyakorlati lépéseket. Mondanivalóm gerincét most is az mplayeres vonal alkotja. Az előzőekben tett ígéretemhez híven, be kívánom mutatni az MPlayer-csomag szerves részét képező MEncodert, mely – mint nevéből is látszik – a lejátszás inverz folyamatát végzi, és a segítségével videókat készíthetünk és tömöríthetünk. Sokat gondolkodtam azon, vajon hogyan is érzékeltethetném számotokra azt a meglepődést, amelyet a program használata keltett bennem, így megmutattam az egyik – videós körökben nagy tiszteletnek örvendő – ismerősömnek, aki a redmondi operációs rendszerrel szerzett komoly tapasztalatokat a témával kapcsolatban. Ez az ismerősöm csak ennyit tudott mondani: „Nagyon profi!”

A MEncoder mint eszköz

A program igazából az MPlayer különleges változata, egyfajta kiterjesztése, ezért ezt az alapfelfogást szem előtt tartva célszerű képességeit, jellemzőit elemeznünk. A MEncoder tulajdonképpen nem más, mint egy olyan MPlayer, amelynek kimenete a monitor helyett más kodekek bemenetét képezi, tehát tisztán tükrözi a jó öreg unixos filozófiát, miszerint a programok láncba fűzve kiválóan képesek egymással kapcsolatot tartani. Az MPlayer dekódoló része a lánc első tagja, amely a kép és hang kicsomagolása után a megjelenítési réteg helyett újabb kodekeket fűz a folyamatba, és ezek valamilyen más formában újra bekódolják az adatfolyamot, majd visszaírják az adattárolóra. Ennek következményeképpen minden olyan fájl „átalakítására” képesek vagyunk, amelyet az MPlayer le tud játszani, tehát a bemeneti fájlformátumok halmaza igen nagy. A kimenet egyelőre sajnos csak AVI lehet, és az alkalmazható kép- és hangkodekek listája is szegényes, megvalósításuk jelenleg is folyik, ennek ellenére már most is kiválóan használhatók (a későbbiek folyamán azt is látni fogjuk, hogy miért).

Használatbavétel

A használatbavétel gyakorlatilag megegyezik az MPlayerről szóló cikkben leírtakkal (Linuxvilág 2002., 3. szám, 28. oldal). Mindenképpen érdemes elolvasni, hiszen nem szeretném ismételni magam, tehát néhány helyen feltételezem az előző anyag tartalmának ismeretét. Az előkészületek során szinte pontosan ugyanúgy kell eljárnunk, mint az MPlayer telepítésénél. A forrás letöltése után a fordítás során egyszerűen a `make` parancs kiadásával létrejön: mind az MPlayer, mind a MEncoder lefordul. Célszerű a CVS-változatot használni, mivel rengeteg újítás (amely természetesen a MEncoder mellett az MPlayer részre is vonatkozik) még nincs benne a 0.6-os üzembiztos változatban, így előfordulhat, hogy a cikk során olyan szolgáltatásokat is kihasználok, amelyek a régebbi változatokban még nem találhatók meg – remélem, ez nem vezet félreértésekhez. A kodekek telepítésére hasonlóképpen kell ügyelnünk, mint az eddigiekben, hiszen a csomag mindkét része ugyanazokra az alapokra épülve dekódol. A hatékony munka érdekében nagyon ajánlott a DivX4Linux kodek telepítése, mert a MEncoder a tömörítésre egyelőre ezt tudja igazán jól használni (a DivX4Linux kodek a ➲ http://www.divx.com címről tölthető le). Ha hangot is szeretnénk kódolni, szükségünk lesz a LAME MP3 kodekre, melynek forrását a ➲ http://www.sulaco.org/mp3/ címről tölthetjük le. A futtatható állomány nem elég, magunknak kell fordítani és telepíteni. A kodekek telepítése után természetesen legalább a MEncodert újra kell fordítanunk. Ha a fentiekkel megvolnánk, jöhet az igazi munka!

A legfontosabb gyakorlati jellemzők

Mint már említettem, az MPlayer által ismert összes fájlformátumból kedvünk szerint kódolhatunk: lehetőségünk nyílik például V4L-megfelelő TV tunerekről történő kódolásra is, valamint minden olyan lineáris adatfolyamot feldolgozhatunk, amelyet az MPlayer szabványos bemeneteként meg tudunk adni. A folyamat során használhatunk külső audiosávokat, beépített képátméretezést, alkalmazhatunk többmenetes DivX4-et, változó bitrátájú MP3-kódolást, de a kép- és hangsávok változtatások nélkül történő másolására is lehetőségünk van (direct stream copy). Alapvetően mindig a kapcsolók, lehetőségek két nagy csoportját adjuk meg a MEncoder számára: az egyik a bemeneti adatfolyamra és annak kezelésére vonatkozik, a másik a kimenettel kapcsolatos tulajdonságokat fekteti le.
A bemenetre vonatkozó megkötések teljesen megegyeznek az MPlayernél alkalmazottakkal: a bemeneti forrásra ugyanúgy hivatkozunk, kérhetjük külső audiofájl lejátszását, vagy épp a hang nélküli megjelenítést, megadhatjuk a dekódoláshoz alkalmazandó video- és hangkodekeket, kodekcsaládokat. Számunkra azonban a MEncoder kimenete és annak jellemzői az érdekesek. A kimenet tulajdonságai további két csoportra bonthatók: egyik része az audio-, másik a videosávokra vonatkozik.
A kimenetre alkalmazott videokodeket a `-ovc` *`kodek neve`*, a hangkodeket pedig a `-ovc` *`kodek neve`* kapcsolókkal választhatjuk ki, továbbá a rendelkezésre álló kodekek listája a `-ovc help`, illetve a `-oac help` kapcsolókkal kérdezhető le. Attól függően, hogy melyiket választottuk, további kapcsolók segítségével lehet megadni az egyes kodekekre jellemző tulajdonságokat.
A `-divx4opts` a DivX4, a `-lavopts` a libavcodec, a `-lameopts` kapcsoló után pedig a LAME mp3 kodek tulajdonságai adhatók meg. Az egyes kapcsolókat itt szóköz nélkül, kettőspontokkal elválasztva kell átadni. Nézzük meg a két legfontosabb kodekhez használható finombeállításokat! Nem térnék ki minden egyes kapcsolóra, inkább csak azokat írom le, amelyeket a mindennapos esetek során szoktunk használni (a teljes lista egyébként a MEncoder leírásában is megtalálható).

- A Divx4Linux legfontosabb kapcsolói (`-divx4opts`):
  `help`: ezzel kérdezhetjük le a

Kulcsképkockák

A mozgóképek tömörítésének alapja: az egymást követő képkockák csak kis mértékben különböznek egymástól, ezért elég csupán az egyes képekhez viszonyított különbséget tárolnunk. Azokat a képeket, amelyekhez képest csak a különbségeket tároljuk, kulcsképkockáknak nevezzük. (A többi kép nem is igazi kép, inkább csak válozásvektorok, amiből az eredeti kép kiszámítható.) Ez egészen addig megy így, amíg a különbségek már akkora méretűek, hogy célszerűbb beszúrni egy új képkockát, és az azt követő képeket pedig ahhoz viszonyítani. (Ezek általában vágásoknál, gyors mozgásoknál találhatók egy-egy filmben.) A valóságban ez természetesen bonyolultabb, a hatékonyabb méretcsökkentés érdekében általában a kulcshoz viszonyított különbségekhez viszonyított (különbségekhez viszonyított... és folytathatnám) képeket tárolnak a kulcsképkocka után.

kapcsolók listáját és jelentésüket.
`br=<szÆm>`: a tömörítési bitrátát (a tömörítés mértékét) határozza meg. A kódolásnál Alapértelmezetten változó bitrátát alkalmaz. Ez azt jelenti, hogy a mozgókép pillanatnyi jellemzőitől függően kisebb-nagyobb mértékben változtat a tömörítés mértékén, így például gyors mozgásoknál sem romlik a kép minősége.
`key=<szÆm>`: meghatározza, hogy legfeljebb hány képkocka követheti egymást anélkül, hogy kulcsképkockát helyeznénk el. A kulcsképkockák (keyframe) olyan képek, amelyekhez a tömörítés során az azt követő képet viszonyítják, ezenkívül összehangolási célokat is szolgálnak. Alapértelmezés szerint a kodek, ha szükségesnek érzi, ezeket a képkockákat önműködően szúrja be, ám például hosszú állóképeknél előfordulhat, hogy ez akár fél percig sem történik meg, ami később, a pozicionálásnál gondot okozhat, ráadásul hibaérzékennyé válik. E kulcsképkocka-mentes tartományok méretét szabályozhatjuk a fenti kapcsolóval.
`q=<1-5>`: segítségével a tömörítés minőségét határozhatjuk meg. Alapértelmezetten 5 – ez a legjobb minőségű és a leglassabb is egyben. Ahogy csökken a minőség, úgy nő a feldolgozási sebesség. Például:

```
-divx4opts
➥br=1800:q=3:key=250
```

- A LAME mp3 kodek kapcsolói (`-lameopts`):
  `help`: ezzel kérdezhetjük le a kapcsolók listáját és jelentésüket.
  `br=<szÆm>`: a tömörítési bitrátát (tömörítés mértékét) határozza meg. Ez a kodek alapértelmezetten állandó tömörítési arányt használ, a kimenetet a pillanatnyi bemenettől függetlenül ugyanakkorára tömöríti. A változó tömörítési arány használatára is lehetőség nyílik, ám ezzel könnyen állíthatunk elő olyan filmeket, amit aztán Windows alatt nem lehet majd lejátszani, épp ezért együttműködő képességének megőrzése végett nem árt tartózkodnunk tőle.
  `q=<szÆm>`: ugyanaz, mint a DivX4Linux-kodek esetében.
  `mode=<0-3>`: meghatározhatjuk a többcsatornás kimeneti MP3 (hangsáv) tulajdonságait. Négy lehetőség közül választhatunk: stereo, joint-stereo, dualchannel, mono. Például:

```
-lameopts br=192:q=4:mode=1
```

Arra is módunk van, hogy a kép- vagy hangsávokat feldolgozás nélkül csak átmásoljuk, ekkor a `-oac copy` vagy a `-ovc copy` kapcsolókkal kell a műveletet elindítanunk, és a kodekek jellemzőit nem adhatjuk meg, hiszen nincs is értelme. Ezt a lehetőséget akkor érdemes használni, ha például DVD-ről „mentjük le" filmjeinket, és meg szeretnénk őrizni az eredeti AC3-hangot. Ilyenkor a képkimenetnek megadjuk a divx-kodeket, a hangot pedig feldolgozás nélkül átmásoljuk. Ez természetesen nem az egyetlen alkalmazási lehetőség: ezzel a módszerrel a hibásan összefésült (összefésületlen) vagy a hibás indexű AVI-k is javíthatók. A kép és hangsávokat egyszerűen másoljuk át egy másik fájlba, közben önműködően összefésülődnek, és az index is újból helyesen jön létre.
A kimeneti kodekek kapcsolókkal való ellátásán kívül még számos olyan adatot adhatunk meg, amely befolyásolja az elkészült filmet. Mindenekelőtt a `-o` *`kimeneti fÆjl`* kapcsolót kötelező megadnunk. Általában szeretnénk tudni, hogy a megadott kapcsolókkal létrehozott AVI hogyan fog kinézni, még mielőtt nekikezdünk a fél napig tartó tömörítésnek. Ilyenkor csak egy rövid részletet célszerű kódolni, így ellenőrizve a filmet. Erre egyrészt a pozícionáló kapcsolók szolgálnak, másrészt megadhatjuk, hogy a program hány képkocka tömörítése után álljon le. A `-ss <  :pp:mp>` kapcsoló hatására a kimeneti fájl kezdete a megadott idő lesz, a `-endpos <  :pp:mp>` kapcsolóval megadott időpontnál pedig a bementi fájl feldolgozása befejeződik. A `-ss` kapcsoló után a `-frames` kapcsolóval – a képkockák számával – is megadhatjuk, hogy a tömörítés mikor érjen véget. Kellően rövid időintervallum megadásával könnyedén ellenőrizhetjük a kódolás eredményét. Ez természetesen csak egy lehetőség, a közvetlen kép- és hangsáv másolásával összekötve részleteket másolhatunk ki egy filmből, levághatunk belőle. A `-pass <1/2>` kapcsolóval határozhatjuk meg, hogy egy többmenetes divx-kódolás esetén épp melyik lépést alkalmazzuk. Ha túl nagy felbontású forrásunk van, a képet úgy átméretezhetjük, hogy a kimeneti fájl kisebb legyen, de akár nagyíthatunk is rajta, ha kedvünk tartja. Néha az átméretezés kifejezetten szükséges, mivel az MPEG-2 formátumok fizikai felbontása az esetek nagy többségében eltér a helyes képaránytól, így ezt a dolgot is a méretezéssel kell megoldanunk. A pontos értékeket a `-x <kimeneti szØlessØg>` és a `-y <kimeneti magassÆg>` kapcsolókkal adhatjuk meg. A `-sws <0-2>` kapcsolóval az átméretezési eljárást határozhatjuk meg, amely a kimeneti kép minőségét befolyásolja. Három lehetőség közül választhatunk: fast bilinear (ez a leggyorsabb), bilinear, bicubic (ez a legjobb minőségű). Egyelőre hiányolom a programból a filmek széléről történő keretek levágásának lehetőségét (cropping), ám a fejlesztők remélhetőleg ezt a szolgáltatást is hamarosan megvalósítják.
A következő cikkben részletesen szeretnék beszélni a többmenetes kódolásról, a szabványos bemenetről történő kódolásról, amellyel több fájl összefűzésére nyílik lehetőség és a különböző bemeneti egységekről történő közvetlen tömörítésről. Szeretném továbbá egy példán keresztül bemutatni egy DVD-lemez tartalmának áttömörítését, valamint az egyéb más fájlokból történő kódolást is, úgy, hogy közben a teljes folyamatot elemezni tudjuk. Megpróbálom bemutatni a videotömörítés lépéseit. Kapcsolódó anyagok a 33 CD Magazin/MPlayer könyvtárban találhatóak.

Komáromi Zoltán
(komi_@freemail.hu)
21 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia.