Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Copias de seguridad en cintas de video MiniDV (25523 lectures)

Per **Guillem Cantallops Ramis**, ***Beowulf*** (http://bulma.net/beowulf/)

Creado el 03/03/2003 02:56 modificado el 03/03/2003 16:38

Lo primero que se te ocurre cuando usas Linux o algún UNIX y sabes algo del formato MiniDV es *"vaya, en esa cinta tan pequeña caben casi 13GB de datos crudos, y se graban a razón de 3.6 MB/segundo, mejor que muchas unidades de cinta para backup... estaria bién poder usar la cámara para eso!"*

Y la verdad es que tiene sus problemillas, porque se trata de usar para datos generales algo que ha sido diseñado desde el principio para almacenar video, pero también tiene algunas ventajas: es divertido, si lo haces bién funciona, y sale barato si ya tienes la cámara. Veamos con qué herramientas podemos hacerlo.

Herramientas disponibles

El "pack" básico de herramientas consiste en el *dvbackup*[1] y el *rsbep*[2]. Que yo sepa no están en Debian (en otras distribuciones ni idea) así que habrá que bajarlos, compilarlos y opcionalmente empaquetarlos a mano. Son programas bastante pequeños, con un *Makefile* muy simple, y se compilan en un momento. Yo para las pruebas preliminares simplemente he puesto los binarios en mi ~/bin. Un detalle importante: para IA-32 el rsbep tiene la parte "dura" del código optimizada en ensamblador, y corre bastante, pero para otras plataformas (por ejemplo yo lo he probado sobre PowerPC) hay que compilarlo en C puro y funcionar, funciona, pero... ya me entendeis... no vuela precisamente.

Bién, antes de empezar a jugar con los programas, veamos cómo lo hacen (aunque la explicación "buena" está en los enlaces que os he pasado ;-)

Un poco de teoria

Básicamente el MiniDV usa una compresión *lossy*, es decir, con pérdida de calidad, que reduce la información de video a 1/5 de su tamaño "crudo" (este *ratio* es fijo), lo cual no suele afectar gravemente a la calidad de las imagenes (salvo en cambios muy bruscos) debido a la naturaleza altamente redundante del video. Esta compresión *lossy* en la práctica parece *lossless*, porque es mucho menos agresiva que cualquier tipo de MPEG, por poner un ejemplo: se trataba de llegar a un buen compromiso entre ratio de compresión y calidad de imagen, y lo hicieron bastante bién.

Pero es una compresión claramente *lossy*, incluso sin entrar en detalles sobre los algoritmos utilizados yo lo veo intuitivo en terminos de de entropia: en el peor de los casos (altamente improbable si capturamos imágenes reales) podria llegar una secuencia de imágenes virtualmente imposible de comprimir, con una entropia cercana 8 bits por byte. En un caso así la única forma de reducir esa información a una quinta parte de su tamaño inicial pasaria por tirar a la basura cuatro quintas partes de esa información. Y como MiniDV **siempre** mantiene el ratio de 1:5... queda claro que es *lossy*. Esto bién escrito podria ser una demostración por reducción al absurdo, pero bueno, si así se entiende... ;-)

Ya está bién de filosofia. La cuestión es que el video se queda en unos "tristes" (?) 25Mbps gracias a esta compresión. Sin contar el audio. Y la compresión utiliza en alguna parte fundamental la transformada discreta del coseno (DCT) sobre bloques de 8x8 pixels. Sí, recuerda al JPEG, pero ahora hablamos de video y además trabajamos con mucha calidad. Cada bloque tiene asociados unos coeficientes AC y DC, y el estándar permite terminar ("truncar") los AC con un código especial, así que lo que han hecho los autores de dvbackup ha sido simplemente usar los DC para pintar un bonito pingüino que se visualizará en todos los *frames* de los videos de backup, y truncar los coeficientes AC inmediatamente para usar todo el espacio restante para los datos del backup. En cuanto al audio, se guarda sin comprimir en la cinta. Simplemente el dvbackup indica en cada frame que no se usará audio, y rellena el espacio

reservado con más datos. Genialmente sencillo, y funciona con muchos modelos de cámaras MiniDV, incluida mi Sony DCR-PC101E (por cierto, muy recomendable porque tanto el video por FireWire como las fotos por USB se transmiten sin problemas a cualquier Linux con un kernel reciente... hay muchos otros modelos, pero este es el que yo he probado ;-)

Backups genéricos y restricciones de las cintas MiniDV

Como las herramientas que veremos a continuación toman los datos de la entrada estándar y de alguna manera acaban metiéndolos en la cinta de video, y lo que queremos nosotros es crear y restaurar backups de uno o varios sistemas de ficheros, o partes de ellos, tendremos que procesar los datos de alguna manera para convertirlos en un stream (a.k.a. "churro" :-) de bytes que sirva para alimentar a esos programas.

En pocas palabras, habrá que usar los típicos `find|cpio`, o alguna cosa similar, para seleccionar y empaquetar los ficheros en uno solo como se ha hecho toda la vida. Y en principio ese fichero resultante será la salida estándar, pero hay un pequeño problema: la cinta MiniDV se graba y se reproduce a sus 25 frames por segundo (PAL) y de ahí no la mueves, por lo menos no en una cámara normal. Si tu máquina es demasiado lenta puedes tener problemas escribiendo en (e incluso leyendo de) la cinta, lo que los programas que usaremos más adelante llaman *underrun* u *overrun* según el caso.

- Si no eres capaz de alimentar suficientemente rápido la cinta cuando estás grabando, en principio el programa es capaz de introducir unos frames especiales sin información para señalizar el underrun, y lo único que pasa a priori es que desperdicias espacio en la cinta... pero si sucede muy a menudo (y seguramente sucederá, si tu máquina no llega es que no llega y punto) puedes perder **demasiado** espacio para tu gusto.
- Si en cambio no eres capaz de procesar en "tiempo real" lo que te llega de la cinta, pierdes partes relativamente grandes de información (overrun) y puedes quedarte sin restaurar el backup, lo cual es obviamente malo.

En realidad si tu disco duro es demasiado lento y/o está demasiado lleno no hay nada que hacer. Pero es que también te puede limitar la CPU, como es el caso de mi *iBook*, con un PowerPC G3 a 500MHz y un disco duro UDMA33. Veremos más adelante que muchas veces es conveniente utilizar compresión (gzip o bzip2), y sobre todo detección y corrección de errores (con rsbep, el otro plato fuerte del artículo). Todo eso, unido al trabajo anterior del `find|cpio` (recorrer el sistema de ficheros y copiar cosas no es "gratis") y al trabajo posterior del empaquetamiento de los datos en frames de video y la transmisión por el FireWire, puede ser demasiado para muchas máquinas. Todas pueden hacerlo, pero el problema es hacerlo produciendo una salida de unos 3.6MB/s **sostenidos**.

Por esa razón a menudo es recomendable utilizar un fichero como almacenamiento intermedio. Por lo menos yo con esa técnica conseguí salvar las limitaciones de potencia de mi iBook. La idea es "partir" los *pipes* que veremos más adelante por la parte que conecta dvbackup|dvconnect con todo lo demás. Así la parte crítica de lectura/escritura de la cinta se hace directamente desde/hacia un fichero secuencial, sin apenas cálculo adicional. Claro que lo mejor es tener una máquina grande y dejarse de tonterias ;-)

Empezamos a formar los comandos...

Para ver unos ejemplos de uso de `find|cpio`, ejemplos sencillos porque estos programas no son el tema del artículo, supondremos que queremos hacer un backup de todo nuestro `/home`, que está en una sola partición. En ese caso, una forma de obtener el stream del backup por la salida estándar (que es lo que nos interesa para empalmarlo luego con comandos posteriores, aunque podriamos redirigirlo a un fichero si quisiéramos) es esta secuencia, bastante clásica:

```
find /home | cpio -o -H crc
```

La parte del `find` lista por su salida estándar los ficheros que queremos guardar, y la parte del `cpio` toma esta lista por su entrada estándar, empaqueta los ficheros añadiendo un código detector de errores CRC, y saca el resultado por su salida estándar. Así que no se os ocurra ejecutarlo tal cual porque os mostrará el backup por la terminal, cosa que puede ser interesante en algunos momentos para entrar en trance o algo así, pero no ahora. Además tardará mucho y teneis que terminar de leer este artículo :-P

Lo siguiente que deberiamos añadir es: compresión, de forma opcional; y deteccion y corrección de errores, de forma obligada. Sobre la compresión, nada que decir: cada cual sabrá cuando la necesita, y como es un proceso relativamente

lento a menudo lo mejor es no utilizarla si hay espacio suficiente en la cinta. Es casi lo mismo que cuando transmites datos por red, sobre todo si la red es una LAN rápida: si utilizas compresión puedes encontrarte las dos máquinas que se están comunicando con la CPU a tope comprimiendo y descomprimiendo, y la red muriéndose de risa, cuando podrias haber acabado mucho antes desactivando la compresión, dejando en paz a las CPUs y aprovechando a tope la red ;-)

Haz backups en MiniDV, pero seguro...

Sin embargo sobre la detección y corrección de errores hay que remarcar unos detalles importantes. A no ser que useis cintas (y supongo que cámaras) de mucha calidad podeis estar seguros de que se produciran errores que os impedirán recuperar solamente con los comandos dvbackup y dvconnect un stream idéntico al que en su dia enviasteis hacia la cámara. Y el CRC y técnicas similares de **detección** de errores (que ya hemos usado en el `cpio`, y que probablemente añadirán otra vez más tarde los compresores si los usamos) pueden estar bién para saber que el backup está mal, pero... la idea es RECUPERARLO, con lo cual necesitamos técnicas de **corrección** de errores, o un buen plan para suicidarse al primer error. Así que usaremos técnicas de **corrección** de errores, concretamente una de las más conocidas y resistentes es la *Reed Solomon* con interleaving, que es más o menos la que se usa en los CDs, por ejemplo.

Esto lo haremos con la pequeña herramienta `rsbep`, que concretamente utiliza RS(255,223) con interleaving de 765 bytes (por defecto; estos valores pueden cambiarse pero así suele ir bién). Esto significa que por cada 223 bytes de entrada, el código detector y corrector de errores genera 255 bytes de salida. Esto añade bastante información redundante que permite detectar muchos errores y corregir una parte importante de ellos. El 765 significa que los bytes de un bloque de Reed Solomon están entrelazados, repartidos, dispersos, de manera que entre dos bytes de un mismo bloque hay 765 bytes. No es cuestión de explicarlo aquí con detalle, pero si lo pensais con calma esto hace el stream resultante muy resistente a las ráfagas de errores porque si por ejemplo se dañan 100 bytes seguidos (que es lo más común, que los errores vengan en ráfagas) se verán afectados 100 bloques Reed Solomon pero cada uno de ellos tendrá solo un byte dañado y podrá recuperarse sin problemas. Digamos que el interleaving reduce drásticamente la probabilidad de que un bloque quede tan dañado que no se pueda recuperar, o incluso que no se pueda detectar el error. Este interleaving unido a las propiedades del código Reed Solomon hace que el stream resultante soporte sin problemas (datos originales 100% recuperables) ráfagas de hasta 12240 bytes consecutivos dañados.

Solo queda un detalle acerca del `rsbep`: al crear streams a partir de un código de bloques con interleaving, a veces tiene que hacer "padding" y añadir ceros al final, siempre que el tamaño de la entrada no sea múltiplo de 223 en este caso. Esto no es importante en el caso de las herramientas que utilizaremos aquí (`cpio`, etc.) y en general es inofensivo a todos los efectos, pero si decidimos proteger otro tipo de información con `rsbep`, es algo a tener en cuenta. Bién, una vez visto todo esto y continuando con nuestro comando, si decidieramos añadir compresión con `bzip2` quedaria algo así:

```
find /home | cpio -o -H crc | bzip2 | rsbep
```

Un añadido un poco soso, pero es que tanto `bzip2` como `rsbep` toman por defecto unas opciones bastante buenas para hacer estos backups... Por cierto, todavia NO os recomiendo ejecutar el comando, porque no está terminado! Simplemente tarda más que el de antes `X'-DDD`

Lo prometido: backups en MiniDV ;-)

Ahora solo queda el último paso: meter los datos en algo que parezca que es video digital en el formato adecuado (tarea de `dvbackup`) y mandarlos a la cámara MiniDV por el FireWire (tarea de `dvconnect`). El comando completo para hacer los backups (dada una máquina suficientemente rápida) seria el siguiente:

```
find /home | cpio -o -H crc | bzip2 | rsbep | dvbackup --prefix=125 | dvconnect -s
```

Sencillamente el `--prefix=125` añade 125 frames sin datos al principio, y el `-s` le dice al dvconnect que va a hacer un "send". Curiosamente los autores de dvbackup/dvconnect y rsbep no coincidieron en el tema de los argumentos por defecto, ya vereis que a la hora de recuperar hay que indicarle `-d` al `rsbep` para decirle que está descodificando, y también al `dvbackup`, y sin embargo el `dvconnect` cuando está recibiendo datos de la cámara justamente es cuando no necesita argumentos :-)

Ups... me dejo unos cuantos detalles obvios que ya habreis notado: si hay que romper el comando por alguna parte para utilizar un fichero intermedio, la mejor parte es entre el `rsbep` y el `dvbackup`:

```
find /home | cpio -o -H crc | bzip2 | rsbep > FICHERO.TMP
cat FICHERO.TMP | dvbackup --prefix=125 | dvconnect -s
```

Además estos comandos ya **sí** que se pueden ejecutar tal cual; pero hay que tener la cámara conectada y grabar en el momento adecuado `X'-DDD`

Eso es muy simple. Hay que tener el FireWire conectado entre la cámara y el ordenador (normalmente con un cable de 4 a 6 pins), el dispositivo /dev/video1394 bién creado (de carácteres, major 171, minor 16: `mknod /dev/video1394 c 171 16`) y con los permisos correctos (lectura/escritura para el usuario/grupo que vaya a hacer los backups), y el módulo del kernel video1394 cargado (con `modprobe video1394`). Luego hay que poner la cámara en modo VCR, colocar la cinta (preferiblemente virgen) en el punto donde deseemos grabar el backup (tiene que haber espacio suficiente a continuación!) y finalmente empezar a grabar instantes antes de ejecutar el comando. No es recomendable hacer esto con baterias, mejor con alimentación directa de la red, porque por ejemplo para llenar una cinta de una hora tardaremos... una hora. Hablaremos de capacidades aproximadas al final.

Ejem... y la recuperación?

No me seais vagos... es lo mismo pero al revés, leyendo unos cuantos manuales se arregla. En fin... para el comando de backup del ejemplo, el "restore" seria algo así:

```
dvconnect | dvbackup -d | rsbep -d | bunzip2 | cpio -imV
```

Por supuesto, teniendo la cámara conectada como antes, con la cinta al principio del backup que queramos restaurar, e iniciando la reproducción instantes después de ejecutar el comando. Una vez más, la mejor parte para separar el comando utilizando un fichero intermedio es entre el `dvbackup` y el `rsbep`:

```
dvconnect | dvbackup -d > FICHERO.TMP
cat FICHERO.TMP | rsbep -d | bunzip2 | cpio -imV
```

Y... tachán! Se supone que funciona. Pero haced unas cuantas pruebas antes de usarlo como vuestro único medio de backup, y por supuesto mirad la ayuda de todos los programas para encontrar variantes que se adapten mejor a vuestras necesidades o gustos. En cualquier caso ni yo ni los autores de los programas ni los responsables de esta web nos hacemos cargo de ninguna de **vuestras** acciones, se adapten o no a las instrucciones. Esto es un truco para hacer de forma divertida algo muy serio, y es solo apto para linuxer@s valientes `O:-)`

Conclusiones

Para empezar, es tan fácil que casi casi da asco. De hecho no queria escribir este artículo... pero bueno, tenerlo todo junto siempre viene bién. La cuestión es que es posible hacer cópias de seguridad en cintas MiniDV, y como decia al principio: es divertido, si lo haces bién funciona, y sale barato si ya tienes la cámara.

La capacidad de las cintas es un detalle importante. Ahí va un párrafo que escribí en nuestra lista de correo, creo que lo condensa bastante bién:

```
En una cinta de 60 minutos caben 13GB de video en formato MiniDV en el que
puedes entremezclar hasta 10GB de tus datos con la imagen fija y silenciosa de
un pingüino, y en LP la capacidad de la cinta aumenta un 50% de forma que
consigues meter hasta 15GB, pero también aumenta el peligro de errores así que
no queda más remedio que perder alrededor de un 14% por el overhead del código
corrector de errores RS(255,223) con un estupendo interleaving de 765 posiciones
que permite recuperar al 100% sin error ráfagas de hasta 12240 bytes dañados.
Una forma curiosa de partir de 13GB para llegar a 13GB, no?  X'-DDD
```

Una última recomendación sobre las cintas: las que llevan el logotipo "CM4K" o "CM16K" o algo así son más caras pero también mucho más recomendables. Son la que incluyen la tecnologia denominada "Chip Memory", que no es más que una pequeña memoria integrada en la cinta (se ven los contactos metálicos junto a la lengüeta de protección contra escritura) que guarda datos como el título de la cinta, y los inicios y los títulos de las escenas. Por una parte,

estas cintas suelen ser de mayor calidad. Y por otra parte, si hay varios backups en una cinta resulta MUY práctico tenerlos indexados y poder colocar la cinta al principio de cualquiera de ellos con total precisión simplemente seleccionando unas cuantas opciones del menú de la cámara (la mia tiene pantalla táctil, no es que sea relevante aquí pero me encanta decirlo];-)

Lista de enlaces de este artículo:

1. http://dvbackup.sourceforge.net/
2. http://www.s.netic.de/gfiala/dvbackup.html

E-mail del autor: beowulf _ARROBA_ bulma.net