Recording all Output from a Student Radio Station

John ffitch
Department of Computer Science
University of Bath
Bath BA2 7AY,
UK,
jpff@cs.bath.ac.uk

Tom Natt
Chief Engineer, URB
University of Bath
Bath BA2 7AY,
UK,
ma1twn@bath.ac.uk

Abstract

Legal requirements for small radio stations in the UK mean, *inter alia*, that the student station at Bath (University Radio Bath or URB) must retain 50 days of the station's output. In addition, as it has recently become easier to transfer data using disposable media, and general technical savvy amongst presenters has improved, there is now some interest in producing personal archives of radio shows. Because existing techniques, using audio videos, were inadequate for this task, a modern, reliable system which would allow the simple extraction of any audio was needed. Reality dictated that the solution had to be cheap. We describe the simple Linux solution implemented, including the design, sizing and some surprising aspects.

Keywords

Audio archive, Audio logging, Radio Station, Portaudio.

1 Introduction

The University of Bath Student's Union has been running a radio station(URB, 2004) for many years, and it has a respectable tradition of quality, regularly winning prizes for its programmes(SRA, 2004). Unfortunately the improved requirements for logging of output from the station coincided with the liquidation of a major sponsor and hence a significant reduction in the station's income, so purchasing a commercial logging device was not an option.

Following a chance conversation the authors decided that the task was not difficult, and a software solution should be possible. This paper describes the system we planned and how it turned out. It should be borne in mind that during development, cost was the overriding factor, in particular influencing hardware choices.

2 The Problem

The critical paragraph of the regulations on Radio Restricted Service Licences, which control such activities as student broadcasting in the UK read

> **You are required to make a recording of all broadcast output, including advertisements and sustaining services. You must retain these recordings ('logging tapes') for a period of 42 days after broadcast, and make them readily available to us or to any other body authorised to deal with complaints about broadcast programmes. Failure to provide logging tapes on request will be treated seriously, and may result in a sanction being imposed.**

where the bold is in the original (OffComm, 2003). In the previous state the logging was undertaken using a video player and a pile of video tapes. These tapes were cycled manually so there was a single continuous recording of all output. This system suffered from the following problems.

The quality was largely unknown. In at least the last 3 years no one has actually listening to anything recorded there; indeed it is not known if it actually works! The system required someone to physically change the tape. Hence, there were large gaps in logging where people simply forgot to do this, which would now expose the station to legal problems.

Assuming that the tapes actually worked, recovering audio would be a painstaking process requiring copying it from the tapes to somewhere else before it could be manipulated in any way. Hence this was only really useful for the legal purposes, rather than for people taking home copies of their shows. Also, as far as could be determined, whilst recovering audio, the logger itself had to be taken offline.

Put simply, the system was out of date. Over the last two years there has been a move to mod-

ernise URB by shifting to computers where possible, so it seemed logical to log audio in such a way that it would be easy to transmit onto the network, and be secure regarding the regulations.

3 Requirements

The basic requirement for the system is that it should run reliably for many days, or even years, with little or no manual intervention. It should log all audio output from the radio station, and maintain at least 50 days of material. Secondary requirements include the ability to recover any particular section of audio by time (and by a non-technical user). Any extracted audio is for archiving or rebroadcast so it must be of sufficient quality to serve this purpose. There is another, non functional, requirement, that it should cost as close to zero as possible!

Quick calculations show that if we were to record at CD quality (44.1KHz, 16bit stereo) then we would need $44100 \times 2 \times 2$ bytes a second, or $44100 \times 2 \times 2 \times 60 \times 24 = 14$Gb each day, which translates to over 700Gb in a 50 day period. While disks are much cheaper than in earlier times, this is significantly beyond our budget. Clearly the sound needs to be compressed, and lossy compression beckons. This reduces the audio quality but, depending on compression rates, not so much that it violates our requirements.

We sized the disk requirements on a conservative assumption of 1:8 compression, which suggests at least an 80Gb disk. Quick experiments suggested about a 400MHz Intel processor; the decision to use Linux was axiomatic. Given sufficient resource, a system to record DJ training sessions and demo tapes was suggested as a simple extension.

We assumed software would be custom-written C, supported by shell scripts, cron jobs and the like. A simple user recovery system from a web interface would be attractive to the non-technical user, and it seemed that PERL would be a natural base for this.

There are commercial logging computers, but the simple cost equation rules them out.

4 Hardware

A search for suitable hardware allowed the creation of a 550MHz Celeron machine with 128Mb of memory, ethernet, and two old SoundBlasters retrieved from a discard pile. SuSE9.1(Novell, 2004) was installed with borrowed screen, keyboard and mouse. The only cash expenditure was a new 120Gb disk; we decided that 80Gb was a little too close to the edge and the additional space would allow a little leeway for any extensions, such as the DJ training.

There were two unfortunate incidents with the hardware; the disk was faulty and had to be replaced, and following the detection of large amounts of smoke coming from the motherboard we had to replace the main system; the best we could find was a 433MHz Celeron system. Fortunately the disk, soundcards and other equipment were not damaged and in the process of acquiring a new motherboard and processor combination we were lucky enough to find another stick of RAM. Most important, what we lost was time for development and testing as we needed to hit the deadline for going live at the beginning of the 2004 academic year.

Hardware	Features
433MHz Celeron	slower than our design
120Gb disk	New!
2 × SoundBlaster 16	old but working
256Mb main memory	
10 Mbit ether	

Table 1: Summary of Hardware Base

5 Implementation

The main program is that the suite needs to perform two main tasks: read a continuous audio stream and write compressed audio to the disk. The reading of the audio feed must not be paused or otherwise lose a sample. The current design was derived from a number of alternative attempts. We use a threaded program, with a number of threads each performing a small task, coordinated by a main loop. A considerable simplification was achieved by using PortAudio(Por, 2004) to read the input, using a call-back mechanism. We shamelessly cannibalised the test program `patest_record` written by Phil Burk to transfer the audio into an array in large sections. The main program then writes the raw audio in 30 second sections onto the disk. It works on a basic 5 period cycle, with specific tasks started on periods 0, 3 and 4.

On 0 a new file is used to write the raw audio, and a message is written to the syslog to indicate the time at which the file starts. On period 3 a subthread is signalled to start the compres-

sion of a raw audio file, and on period 4 the next raw audio file is named and created. By sharing out the tasks we avoid bursts of activity which could lead to audio over-runs. This is shown in figure 1.

The compression is actually performed by a lower priority sub task which is spawned by a call to system. There is no time critical aspect of the compression as long as it averages to compressing faster than the realtime input. Any local load on the machine may lead to local variation but eventually it must catch up. There is a dilemma in the compression phase. The obvious format is OGG, for which free unencumbered software exists, but the student community is more used to MP3 format. We have experimented with `oggenc`(ogg, 2004), which takes 80% of elapsed time on our hardware and compresses in a ratio of 1:10, and `notlame`(Not, 2004), where compression is 1:11 and 74% of elapsed time. Our sources have both methods built in with conditional compilation.

We have varied the period, and have decided on a minute, so each audio file represents five minutes of the station's output; this is a good compromise between overheads and ease of recovery.

The result of this program is a collection of 5 minute compressed audio files. Every day, just after midnight, these files are moved to a directory named after the day, and renamed to include the time of the start of the interval of time when the recording started. This is achieved with a small C program which reads the syslog files to get the times. This program could have been written in PERL but one of us is very familiar with C. A snapshot of part of the logging directory is shown in figure 2, where compressed audio, raw PCM files, unused PCM files and a compression log can be seen.

The decision to rename actual files was taken to facilitate convenience during soak testing. We were running the system over this time as if it were live, and hence were using it to extract logs when they were requested by presenters. Cross-referencing files with the system log was a tedious task so an automatic renaming seemed the obvious choice. Using this opportunity to refile the logs in directories corresponding to the date of logging also assisted greatly in retrieval. A more long-term consideration was that renamed files would be easier to extract via a web interface and hence this work could probably be used in the final version also.

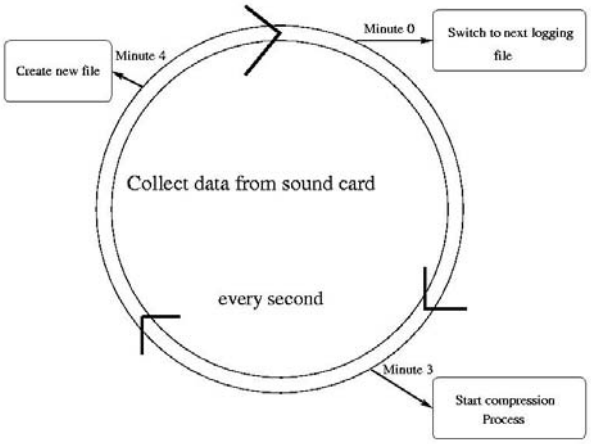

Figure 1: Overview of Software Cycle

6 Experience

The program has now been running for a significant time. Two problems with the design have emerged.

The first was the change to winter time which happened a few days after going live. Our logs, and hence times, where based on local time as that seemed to be closest to what the users would require. But with the clock being put backwards, times repeat. Clearly we need to maintain both times in the logs, and append the time zone to the ultimate file name, or some similar solution. But how did we manage this shift backwards without loss of data? The answer is in the second problem.

We are capturing the raw station output in 44.1KHz 16bit stereo. Every five minutes a new file is started. Actually we are not starting files by time but by sample count (13230000 frames). As was predicted, the sound card was not sampling at exactly CD rate, but faster, and as a result we are drifting by 19 seconds a day. In itself this is not a problem, and indeed rescued the possible data loss from the introduction of winter time, but it is less convenient for the student DJs who want a copy of their program. The suggestion is that the files should be aligned on five minute boundaries by the clock. This entails monitoring the clock to decide on a change of file, which would be a considerable departure from the simplicity of design. Exactness is not important, but we would like to be less than a minute adrift. Our revised code, not yet in service, makes the switch of files after reading the clock, and additional care is needed to avoid clock drift.

```
-rw-r--r--   1 root root   4801096 Mar  7 10:59 Arc0041022.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:04 Arc0041023.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:09 Arc0041024.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:14 Arc0041025.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:19 Arc0041026.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:24 Arc0041027.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:29 Arc0041028.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:34 Arc0041029.mp3
-rw-r--r--   1 root root   4801096 Mar  7 11:39 Arc0041030.mp3
-rw-r--r--   1 root root  52920000 Mar  7 11:44 Arc0041032
-rw-r--r--   1 root root   4801096 Mar  7 11:44 Arc0041031.mp3
-rw-r--r--   1 root root         0 Mar  7 11:47 log5Pwl4o
-rw-r--r--   1 root root  42332160 Mar  7 11:48 Arc0041033
-rw-r--r--   1 root root         0 Mar  7 11:48 Arc0041034
-rw-r--r--   1 root root   3145728 Mar  7 11:48 Arc0041032.mp3
```

Figure 2: Part of Directory for Running System

It was this clock drift, which can be seen in figure 3, that saved the situation when we changed from summer time to winter time. If we had implemented the time alignment method then the file names would have repeated for the one hour overlap (1am to 2am is repeated in the UK time scheme), but as the soundcard had read faster, the second hour was fortuitously aligned to a different second, and so we could rescue the system manually.

The zone change from winter to summer involves the non-existence of an hour and so raises no problems. Before next autumn we need to have deployed a revised system. It has been suggested that using the Linux linking mechanisms we could maintain all logging in universal time, and create separate directories for users to view.

There was one further problem. When the syslog file got large the usual logrotate mechanism started a new file. But as our renaming system reads the syslog, it subsequently missed transfer and rename of some output. This was fixed by hand intervention, but at present we do not have a good solution to this; nasty solutions do occur to us though!

Another minor problem encountered during initial testing was with the hardware: it seems under Linux older versions of the SoundBlaster chipset could not handle both recording from an input stream and outputting one simultaneously. The output stream took priority so unless we specifically muted the output channels on the card, no sound was captured. This is only mentioned here in case an attempt is made to duplicate this work, and so to avoid the hours of frustration endured during our initial tests. We expect that similar minor problems will appear later as we develop the system, but the main data collection cycle seems most satisfactorily robust. Most importantly, despite being forced to downgrade our hardware, the system performs within its new limitations without loss of data during compression phases — even during periods of additional load from users (*i.e.* when logs are being extracted). There is sufficient slack for us to consider adding additional services.

7 Conclusions

Tests have demonstrated that our original aim, of a cheap data logging system, has been easily achieved — the whole system cost only £60 in new purchased materials. What is also clear is that the whole structure of the Linux and Open Source movements made this much more satisfactory than we feared. The efficiency of Linux over, say, Windows meant that we could use what was in effect cast-off hardware. The ability to separate the data collection from the compression and filing allowed a great simplification in the design, and so we were able to start the logging process days before we had thought about the disposal process, but before the start of the university term. The `crontab` mechanism enables us to delete whole directories containing a single day after 60 days have passed. We still need to implement a web-interface to extracting of programs, but the availability of PERL, Apache, and all the related mechanisms suggests that this is not a major task.

```
-rw-r--r--   1 root root 4801096 Mar  4 22:55 22:45:08.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:00 22:50:08.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:05 22:55:08.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:10 23:00:08.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:15 23:05:08.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:20 23:10:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:25 23:15:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:30 23:20:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:35 23:25:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:40 23:30:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:45 23:35:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:50 23:40:07.mp3
-rw-r--r--   1 root root 4801096 Mar  4 23:55 23:45:07.mp3
-rw-r--r--   1 root root 4801096 Mar  5 00:00 23:50:07.mp3
-rw-r--r--   1 root root 4801096 Mar  5 00:05 23:55:07.mp3
-rw-r--r--   1 root root    6912 Mar  5 01:10 index
```

Figure 3: Part of an Archive Directory

Although it is not a major problem to write, the extraction web page for the system will be the only part most users see and hence design for ease of use must be key. Currently, the idea is to incorporate this into the current URB online presence(URB, 2004) which allows members of the station to log into a members area. We will add a logging page, which presents users with a very simple interface specifying only the beginning and end points of the period required. With their user-names tied to a download destination, presenters will always find their logs in the same place, limiting the possibility of confusion.

Being based on such reliable software packages, we are sure that if we ever have sufficient funds for an upgrade, for example to a digital input feed, this can easily be accommodated. We are aware that the current system lacks redundancy, and a secondary system is high on our wish-list. More importantly we have not yet completed a physically distributed back-up in case the next machine fire *does* destroy the disk.

We are confident that as the radio station continues to be the sound-track of the University of Bath, in the background we are listening to all the sounds, logging them and making them available for inspection. With this infrastructure in place we might even consider a "play it again" facility, if the legal obstacles can be overcome.

Naturally as the program is based on open source code we are willing to provide our system to anyone else who has this or a similar problem.

8 Acknowledgements

Our thanks go to Simon Lee, the instructor of Bath University Students' Union T'ai Chi Club, for tolerating our discussions before (and sometimes during) training sessions.

References

2004. Notlame mp3 encoder. http://users.rsise.anu.edu.au/~conrad/not_lame/.

Novell. 2004. http://www.novell.com/de-de/linux/suse.

OffComm. 2003. Office of Communications Document: Long-Term Restricted Service Licences. http://www.ofcom.org.uk/codes_guidelines/broadcasting/radio/guidance/lo%ng_term_rsl_notes.pdf, January.

2004. Ogg front end. http://freshmeat.net/projects/oggenc/.

2004. PortAudio — portable cross-platform Audio API. http://www.portaudio.com/.

2004. SRA: Student Radio Association. http://www.studentradio.org.uk/awards.

2004. URB: University Radio Bath. http://www.bath.ac.uk/~su9urb.