Introduction

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 about 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.

This presentation describes our solution.
The Regulation

- 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.
The Old System

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. The quality was largely unknown — not even sure they work! The system required someone to physically change the tape, and this was often forgotten. Even if the tapes worked there was no mechanism for copying the tapes, certainly without removing logging.
The Old System

- Logging was undertaken using a video player and a pile of video tapes.
The Old System

- 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.
The Old System

- 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.

- The quality was largely unknown — not even sure they work!
The Old System

- 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.
- The quality was largely unknown — not even sure they work!
- The system required someone to physically change the tape, and this was often forgotten.
The Old System

- 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.
- The quality was largely unknown — not even sure they work!
- The system required someone to physically change the tape, and this was often forgotten.
- Even if the tapes worked there was no mechanism for copying the tapes, certainly without removing logging.
A Shinning New Solution

has been moving to a computer-based station, so naturally the question was how easy would it be to incorporate logging.

But....
A Shinning New Solution

has been moving to a computer-based station, so naturally the question was how easy would it be to incorporate logging.

But....

- A financial crisis meant that there was no budget.
A Shinning New Solution

has been moving to a computer-based station, so naturally the question was how easy would it be to incorporate logging. But....

- A financial crisis meant that there was no budget.
- Chance questions led to the design below.

end
Requirements

Should run reliably for many days, or even years, with little or no manual intervention.

Should log all audio output from the radio station.

Maintain at least 50 days of material.

Able to recover any particular section of audio by time (and by a non-technical user).

Cost as close to zero as possible!
Requirements

- Should run reliably for many days, or even years, with little or no manual intervention.
Requirements

- Should run reliably for many days, or even years, with little or no manual intervention.

- Should log all audio output from the radio station.
Requirements

- Should run reliably for many days, or even years, with little or no manual intervention.
- Should log all audio output from the radio station.
- Maintain at least 50 days of material.
Requirements

- Should run reliably for many days, or even years, with little or no manual intervention.
- Should log all audio output from the radio station.
- Maintain at least 50 days of material.
- Able to recover any particular section of audio by time (and by a non-technical user).
Requirements

- Should run reliably for many days, or even years, with little or no manual intervention.

- Should log all audio output from the radio station.

- Maintain at least 50 days of material.

- Able to recover any particular section of audio by time (and by a non-technical user).

- Cost as close to zero as possible!
Calculations

Assuming input is at CD quality we need 44100 bytes a day. That is approximately 14Gb/day or 700Gb in 50 days. That is beyond our budget so compression is necessary. Assuming 1:8 compression we need at least an 80Gb disk, and experiments suggested a 400MHz Intel processor; I assumed Linux without thought.

Software base – C, shell scripts, cronjobs and PERL
Calculations

- Assuming input is at CD quality we need
  \[44100 \times 2 \times 2 \times 60 \times 60\] bytes a day.

That is approximately 14Gb/day or 700Gb in 50 days. That is beyond our budget so compression is necessary. Assuming 1:8 compression we need at least an 80Gb disk, and experiments suggested a 400MHz Intel processor; I assumed Linux without thought.
Calculations

- Assuming input is at CD quality we need $44100 \times 2 \times 2 \times 60 \times 60$ bytes a day.

- That is approximately 14Gb/day or 700Gb in 50 days.
Calculations

- Assuming input is at CD quality we need $44100 \times 2 \times 2 \times 60 \times 60$ bytes a day.

- That is approximately 14Gb/day or 700Gb in 50 days.

- That is beyond our budget so compression is necessary.
Calculations

- Assuming input is at CD quality we need
  \[ 44100 \times 2 \times 2 \times 60 \times 60 \text{ bytes a day}. \]

- That is approximately 14Gb/day or 700Gb in 50 days.

- That is beyond our budget so compression is necessary.

- Assuming 1:8 compression we need at least an 80Gb disk, and experiments suggested a 400MHz Intel processor; I assumed Linux without thought.
Calculations

- Assuming input is at CD quality we need $44100 \times 2 \times 2 \times 60 \times 60$ bytes a day.

- That is approximately 14Gb/day or 700Gb in 50 days.

- That is beyond our budget so compression is necessary.

- Assuming 1:8 compression we need at least an 80Gb disk, and experiments suggested a 400MHz Intel processor; I assumed Linux without thought.

- Software base – C, shell scripts, cronjobs and PERL
Hardware

We scavenged a 550MHz Celeron with 128Mb, also ethernet and two old Soundblasters. Installed SuSE9.1 with borrowed keyboard, screen and mouse. Spent money on a new 120Gb disk.
Hardware

- We scavenged a 550MHz Celeron with 128Mb, also ethernet and two old Soundblasters.
Hardware

- We scavenged a 550MHz Celeron with 128Mb, also ethernet and two old Soundblasters.
- Installed SuSE9.1 with borrowed keyboard, screen and mouse.
Hardware

- We scavenged a 550MHz Celeron with 128Mb, also ethernet and two old Soundblasters.

- Installed SuSE9.1 with borrowed keyboard, screen and mouse.

- Spent money on a new 120Gb disk.

end
Problems
Problems

- The disk delivered was faulty and had to be returned.
Problems

- The disk delivered was faulty and had to be returned.
- A small fire in the computer destroyed the processor (but luckily not the disk).
Problems

- The disk delivered was faulty and had to be returned.
- A small fire in the computer destroyed the processor (but luckily not the disk).
- Second scavenged machine was a 433MHz Celeron, but also more memory.
Problems

- The disk delivered was faulty and had to be returned.
- A small fire in the computer destroyed the processor (but luckily not the disk).
- Second scavenged machine was a 433MHz Celeron, but also more memory.
- The dead-line for the 2004 academic year was getting close.....

end
The Software Design

Need a continuous reading of the audio, and periodic compression. After experimentation we created a threaded program, with threads to read audio, file audio, compress data, and other tasks. Largely based on the PortAudio test program `patest`, writing 30sec audio buffers to the disk. File creation, starting compression, and changing output are distributed.
The Software Design

- Need a continuous reading of the audio, and periodic compression.
The Software Design

- Need a continuous reading of the audio, and periodic compression.

- After experimentation we created a threaded program, with threads to read audio, file audio, compress data, and other tasks.
The Software Design

- Need a continuous reading of the audio, and periodic compression.

- After experimentation we created a threaded program, with threads to read audio, file audio, compress data, and other tasks.

- Largely based on the PortAudio test program patest_record, writing 30sec audio buffers to the disk.
The Software Design

- Need a continuous reading of the audio, and periodic compression.

- After experimentation we created a threaded program, with threads to read audio, file audio, compress data, and other tasks.

- Largely based on the PortAudio test program `patest_record`, writing 30sec audio buffers to the disk.

- File creation, starting compression, and changing output are distributed.
Audio Cycle

Collect data from sound card

every second

Create new file

Minute 0

Switch to next logging file

Minute 3

Start compression Process

Minute 4
Compression

- I wanted to use OGG with `oggenc` – takes 80% of elapsed time to compress and gives 1:10 compression

- Alternative is MP3 with `notlame` – takes 74% of elapsed time to compress and gives 1:11 compression

- The program can be built for either – but student DJs think MP3 so I lost for the present....

end
The Other Software

Small C program renames compressed files by time and moves to daily directory.

Cronjob deletes directories more than 60 days old.

(unfinished) Web-based PERL script to retrieve and deliver timed segments to DJs.

end
The Other Software

- Small C program renames compressed files by time and moves to daily directory
The Other Software

- Small C program renames compressed files by time and moves to daily directory
- Cronjob deletes directories more than 60 days old
The Other Software

- Small C program renames compressed files by time and moves to daily directory
- Cronjob deletes directories more than 60 days old
- (unfinished) Web-based PERL script to retrieve and deliver timed segments to DJs.

end
## Directory for Running System

<table>
<thead>
<tr>
<th>Mode</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>10:59</td>
<td>Arc0041022.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:04</td>
<td>Arc0041023.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:09</td>
<td>Arc0041024.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:14</td>
<td>Arc0041025.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:19</td>
<td>Arc0041026.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:24</td>
<td>Arc0041027.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:29</td>
<td>Arc0041028.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:34</td>
<td>Arc0041029.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:39</td>
<td>Arc0041030.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>52920000</td>
<td>Mar 7</td>
<td>11:44</td>
<td>Arc0041032</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>4801096</td>
<td>Mar 7</td>
<td>11:44</td>
<td>Arc0041031.mp3</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>0</td>
<td>Mar</td>
<td>11:47</td>
<td>log5Pwl4o</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>42332160</td>
<td>Mar 7</td>
<td>11:48</td>
<td>Arc0041033</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>0</td>
<td>Mar</td>
<td>11:48</td>
<td>Arc0041034</td>
</tr>
<tr>
<td>-rw-r--r-</td>
<td>root</td>
<td>root</td>
<td>3145728</td>
<td>Mar 7</td>
<td>11:48</td>
<td>Arc0041032.mp3</td>
</tr>
</tbody>
</table>
Experience

Been running since October ... with no crash.
I forgot about summer time/winter time ... but we were saved by clock drift of 19 sec/day.
Still have problems when the system log is cycled as we lose times for files.
Experience

● Been running since October
Experience

- Been running since October
- I forgot about summer time/winter time
Experience

- Been running since October ....with no crash.
- I forgot about summer time/winter time
Experience

- Been running since October ....with no crash.

- I forgot about summer time/winter time ....but we were saved by clock drift of 19sec/day.
Experience

- Been running since October ....with no crash.
- I forgot about summer time/winter time ....but we were saved by clock drift of 19sec/day.
- Still have problems when the system log is cycled as we lose times for files.

end
Conclusions

Original aims were achieved — a robust logging system for a cost of £60.
Still rough edges but usable.
The Linux tool-chain and libraries made it much simpler than I feared.
Still need backups and physical distribution.
Conclusions

- Original aims were achieved — a robust logging system for a cost of £60.
Conclusions

- Original aims were achieved — a robust logging system for a cost of £60.
- Still rough edges but usable.
Conclusions

- Original aims were achieved — a robust logging system for a cost of £60.

- Still rough edges but usable.

- The Linux tool-chain and libraries made it much simpler than I feared.
Conclusions

• Original aims were achieved — a robust logging system for a cost of £60.

• Still rough edges but usable.

• The Linux tool-chain and libraries made it much simpler than I feared.

• Still need backups and physical distribution.

day
Conclusions

A fun project, fitting the *what I did in my summer holidays* model. Despite problems a resounding success.

The basic design was dreamt up before and during the warm-up of The Bath University Students’ Union T’ai Chi Club. Tom, an official of the club, and I wish to thank our instructor Simon Lee for tolerating our improper talking.