Speech Detection

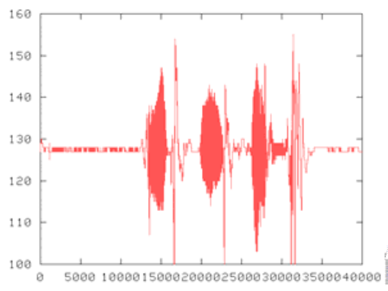
Project 1

Outline

- Motivation
- Problem Statement
- Details
- Hints

Motivation

- Word recognition needs to detect word boundaries in speech



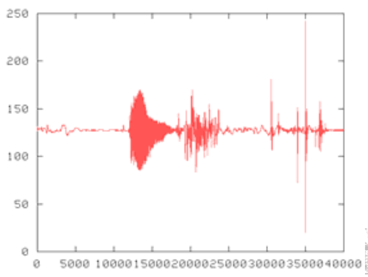
“Silence Is Golden”

Motivation

- Recognizing silence can reduce:
  – Network bandwidth
  – Processing load
- Easy in sound proof room, with digitized tape
  – Measure energy level in digitized voice

Research Problem

- Noisy computer room has loud background noise, making some edges difficult



“Five”

Research Problem

- Computer audio often for interactive applications
  – Voice commands
  – Teleconferencing
- →Needs to be done in ‘real-time’

Project Solution

- Implement end-point algorithm by Rabiner and Sambur [RS75]
  - (Paper for class, next)
- Implementation in Linux or Windows
- Basis for audioconference/Internet phone
  - (Project 2)

Details

- Voice-quality:
  - 8000 samples/second
  - 8 bits per sample
  - One channel
- Record sound, write files:
  - `sound.all` - audio plus silence
  - `sound.speech` - audio no silence
  - `sound.data` - text-based data: audio data, energy, zero crossings:

```
128 10 3
127 12 4
127 20 3
```

- Other features allowed

Sound in Windows

- Microsoft Visual C++
  - See Web page for basic tutorials
- Use sound device → WAVEFORMATEX:

  wFormatTag: set to WAVE_FORMAT_PCM

  nChannels, *nSamplesPerSec*, *wBitsPerSample*: set to voice quality audio settings

  nBlockAlign: set to number of channels times the number of bytes per sample

  nAvgBytesPerSec: set to the number of samples per second times the *nBlockAlign* value

  cbSize: set this to zero

Sound in Windows

- `waveInOpen()`
  - a device handle (HWAVEIN)
  - the device number (1 in the movie lab)
  - the WAVEFORMATEX variable
  - a callback function

    →gets invoked when the sound device has a sample of audio

Sound in Windows

- Sound device needs buffers to fill
- LPWAVEHDR
  - *lpData*: for raw data samples
  - *dwBufferLength*: set to nBlockAlign times the length (in bytes) of the sound chunk you want
- `waveInAddBuffer()` to give buffer to sound device
  - Give it device
  - Buffer (LPWAVEHDR)
  - Size of variable
- When callback invoiked, buffer (lpData) has raw data to analyze
  - Must give it another via waveInAddBuffer() again

Sound in Windows

- Useful header files:

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <mmsystem.h>
#include <winbase.h>
#include <memory.h>
#include <string.h>
#include <signal.h>

extern "C"
```

- Useful data types:

  HWAVEOUT
  - writing audio device

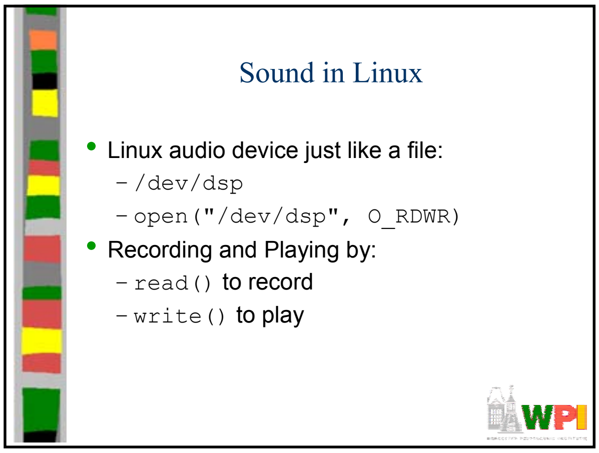
  HWAVEIN
  - reading audio device

  WAVEFORMATEX
  - sound format structure

  LPWAVEHDR
  - buffer

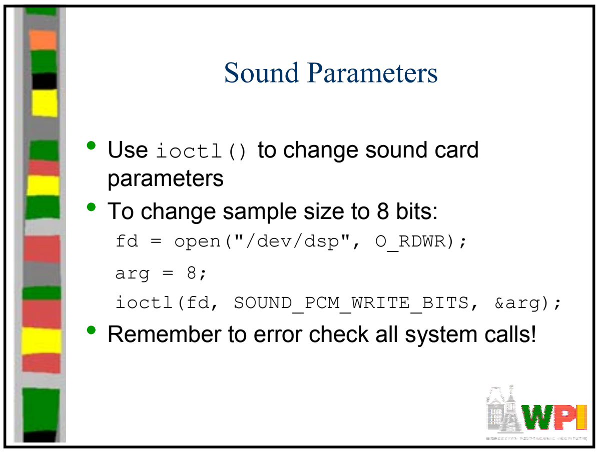
  MMRESULT
  - Return type from wave system calls

- See the online documentation from Visual C++ for more information



Sound in Linux

- Linux audio device just like a file:
  - `/dev/dsp`
  - `open("/dev/dsp", O_RDWR)`
- Recording and Playing by:
  - `read()` to record
  - `write()` to play

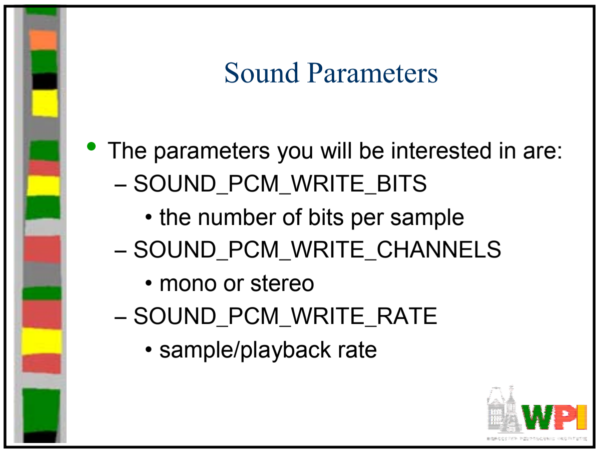


Sound Parameters

- Use `ioctl()` to change sound card parameters
- To change sample size to 8 bits:

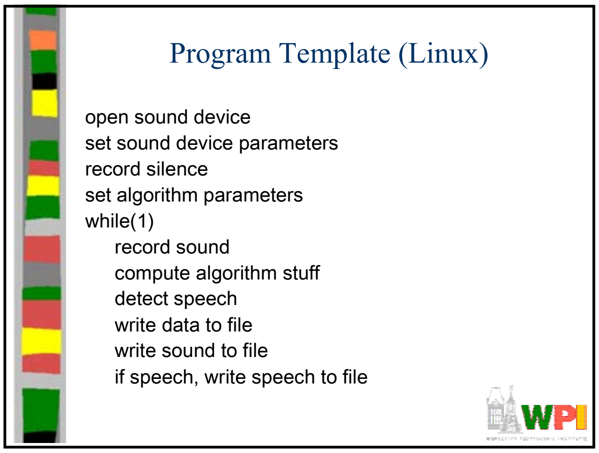
```
fd = open("/dev/dsp", O_RDWR);
arg = 8;
ioctl(fd, SOUND_PCM_WRITE_BITS, &arg);
```

- Remember to error check all system calls!



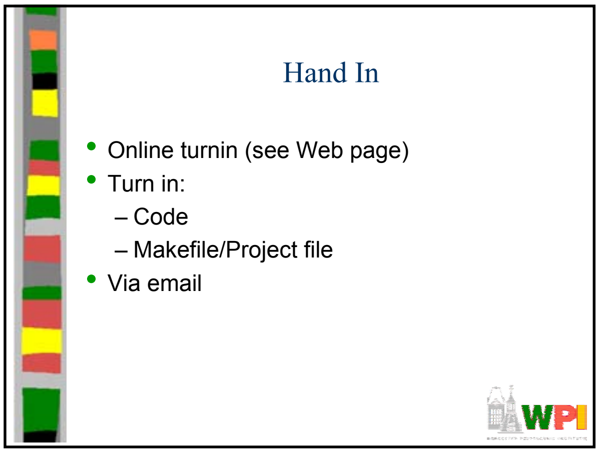
Sound Parameters

- The parameters you will be interested in are:
  - SOUND_PCM_WRITE_BITS
    - the number of bits per sample
  - SOUND_PCM_WRITE_CHANNELS
    - mono or stereo
  - SOUND_PCM_WRITE_RATE
    - sample/playback rate



Program Template (Linux)

```
open sound device
set sound device parameters
record silence
set algorithm parameters
while(1)
    record sound
    compute algorithm stuff
    detect speech
    write data to file
    write sound to file
    if speech, write speech to file
```



Hand In

- Online turnin (see Web page)
- Turn in:
  - Code
  - Makefile/Project file
- Via email