Cross-compile Guide for Flite

Contents

Introduction

CMU Flite (festival-lite) is a small, fast run-time open source text to speech synthesis engine developed at CMU and primarily designed for small embedded machines and/or large servers. Flite is designed as an alternative text to speech synthesis engine to Festival for voices built using the FestVox suite of voice building tools.
Since flite is a small and fast program, it is ideal to be embedded into beaglebone and provide library API for custom C programs. However, cross-compiling the library specially for beaglebone requires certain setups beforehand and it is not provided by the developers. Therefore, this is a quick setup guide that enable flite library functionalities on beaglebone and provide easy access to flite API for custom programs.

Install Flite application

Flite functionalities can be first experienced by installable package ‘flite’. This program uses flite library itself and is a good example of using API functions

1. Compile audio device tree and load audio cape
   (see audio guide for more details)

2. Make sure target have internet connection by
   `# ping google.ca`
   (see network guide for more details)

3. Install flite from apt-get
   `# apt-get update`
   `# apt-get install flite`

4. Run flite program with
   `# flite -t “hello”`
   It should play the sound for ‘hello’.
   `# echo ‘Hello, flite!’ > flite_test.txt`
   `# flite -f flite_test.txt`
   It should play the content of the specified file.

5. Trouble shooting

- Check network problems by
  `# ping google.ca`
  if apt-get cannot install package or other network problems. Go over network guide.
- When executing flite, if the error shows
  `oss_audio: failed to open audio device /dev/dsp`
  Go over audio guide and check if audio settings have completed

Cross-compile Flite Library

Flite header and library files with the correct format is required to write custom code that uses its API. Therefore, the source of flite should be downloaded and compiled on beaglebone with correct configurations to build the library files needed.

1. Install alsa dev package

Alsa sound engine is used by flite and is required as development environment. On target, install alsa dev tools:

```
# apt-get install libasound2-dev
```

This is needed by flite source compiling.

2. Download flite source to target from github and configure
   On target, type in the following commands:

```
# cd
# git clone http://github.com/festvox/flite
# cd flite
# ./configure --with-audio=alsa --enable-shared
```

   Wait for automatic configuration to be done. These specify the audio engine to be alsa and compile shared library for later cross-compiling.

3. Build flite sourse
   On target, in directory ~/flite:

```
# make
```

   The building process will take about 10 minutes to complete. This requires about 100 MB space in beaglebone. If there's not enough space, try remove some unused packages.

4. Copy the include directory and shared libraries (.so files) to public file
   On host, create directories in ~/cmpt433/public and make them accessible to target:

```
$ cd ~/cmpt433/public/
$ mkdir flite_inc_BBB/
$ mkdir flite_lib_BBB/
$ chmod a+rwx flite_inc_BBB
$ chmod a+rwx flite_lib_BBB
```

   After mounting NFS, on target:

```
# cd ~/flite
# cp include/* /mnt/remote/flite_inc_BBB/
# cd build/armv7l-linux-gnueabihf/lib/
```

```
# cp *.so /mnt/remote/flite_lib_BBB/
```
The shared library libasound.so is also needed for cross-compiling, and is covered in audio guide.

5. C code example
   The code and makefile example for cross-compiling is listed below. Copy each of them to a local file on host for testing:

   Code example:

```c
// flite_cross.c
#include "flite.h"
cst_voice* register_cmu_us_rms();
int main(int argc, char **argv)
{
    cst_voice *v;
    char* str = "Hello, world!";
    flite_init();
    v = register_cmu_us_rms(NULL);
    flite_text_to_speech(str,v,"play");
}
```

   Makefile example:

```
# Makefile
CROSS = arm-linux-gnueabihf-
CC = $(CROSS)gcc
CFLAG = -Wall -g -std=c99 -Werror
PUBDIR = $(HOME)/cmpt433/public/
LFLAGS = -L$(PUBDIR)asound_lib_BBB
-L$(PUBDIR)flite_lib_BBB \
         -lflite_usenglish -lflite_cmulex -lflite
-lflite_cmu_us_rms -lm -lasound
IFLAGS = -I$(PUBDIR)flite_inc_BBB
```

```
OUTDIR = $(HOME)/cmpt433/public/myApps/

all:
    $(CC) $(CFLAG) -o flite_cross flite_cross.c
$(IFLAGS) $(LFLAGS)
    cp flite_cross $(OUTDIR)

clean:
    rm flite_cross $(OUTDIR)flite_cross
```

There may be some extra line feed when copy-and-paste. Delete them if not parsed well.

6. Make and run
   In the directory containing the above code and makefile:
   `$ make`
   This should build the test to public folder. On target:
   `# cd /mnt/remote/myApps`
   `# ./flite_cross`
   Then the sound for ‘Hello, world!’ should be played out.

7. To change voices used for synthesis or acquire more controls over flite, go to http://www.festvox.org/flite/doc/flite.html for documentation. The documentation is not so updated, especially for beaglebone. If needed, refer to source code on https://github.com/festvox/flite. The source code for flite program locates in flite/src/flite_main.c

8. Trouble shooting

- When building flite source on target, if error shows
  `fatal error: alsa/asoundlib.h: No such file or directory`
  This is because alsa dev tools are not installed. check the start of the session and install the package.

- When cross compiling custom code, if `-lasound` cannot be found, check audio guide to copy the shared library `libasound.so` to `~/cmpt433/public/asound_lib_BBB` and recompile.
- When cross compiling custom code, if error shows
  `error adding symbols: File format not recognized`
  Check if shared library building is enabled in flite compilation with `--enable-shared` and the shared library files (.so files, not .a files) are correctly copied to public folder. Rebuild flite source if necessary.
- When cross compiling custom code, if error shows
  `./flite_cross: error while loading shared libraries: libflite.so.1: cannot open shared object file: No such file or directory`
  The shared library path should be further specified on target:
  `# export LD_LIBRARY_PATH=/root/flite/build/armv7l-linux-gnueabihf/lib`
  Rerun the program after that.
- When running custom program, if error shows
  `oss_audio: failed to open audio device /dev/dsp`
  Check if flite source is compiled with `--with-audio=alsa`
  Not using ALSA will default the sound engine to be OOS, on which no support is loaded through this guide.

Additional Resources

- Networking Guide, by Brian Fraser
  http://www.cs.sfu.ca/CourseCentral/433/bfraser/other/Networking.pdf

- Audio Guide, by Brian Fraser
  http://www.cs.sfu.ca/CourseCentral/433/bfraser/other/AudioGuide.pdf

- Flite Main Page, by CMU LTI
  http://www.festvox.org/flite/