Cross-Compiling ALSA for Rust

Steps by Rahul Rajesh

This assumes we use a build script when building with Cargo. This file is usually the build.rs file located at the project root directory. The following example shows linking the asound library and assumes it lives in the same path as in the assignments.

Rust and C++ (with Rust's cxx library)

Inside build.rs and inside it's main function we add the following lines:

```
// inside the main function of build.rs

use directories::UserDirs;
let user_dirs = UserDirs::new().unwrap();
println!("cargo:rustc-link-search={}/cmpt433/public/asound_lib_BBB",
user_dirs.home_dir().to_str().unwrap());
```

We use the *directories* library/crate to avoid hardcoding the username to library path. `{} (in line 3 above)` should expand to something like `/home/rrajesh`. Since the *directories* library is a build dependency we can add it under `[build-dependencies]` inside of *Cargo.toml* instead of `[dependencies]`.

When we cross-compile Rust for BBG, we have a config file at .cargo/config. To add the dynamic library we add another line to config file.

```
// entirety of .cargo/config


[build]
target = "armv7-unknown-linux-gnueabihf"


[target.armv7-unknown-linux-gnueabihf]
linker = "arm-linux-gnueabihf-gcc"
rustflags = ["-ldylib=asound"]
```

The line of interest is the green line above from the entire config file posted above. Ideally, we would expect the following line to be also part of the build script and that's what our team understood from Cargo's docs. However, it didn't work.

Rust only (Maybe)

This depends because usually you don't need a build.rs file when cross compiling pure Rust code (you only need to specify the target in .cargo/config). If you do have build.rs file, I think the above config could work with or without

```
linker = "arm-linux-gnueabihf-gcc".
```

In conclusion, with Rust and C++ (with Rust's cxx library), inside *build.rs* we add the 3 lines seen above (yellow line most important) and inside *.cargo/config* we add the green line. I also wanted to mention that the cxx library has guides on using CMake or other build systems instead of Cargo which maybe easier for some people. With Rust (and has buildr.rs) only, this solution is worth trying but it has never been tested.