Small project: Part 1
Environment and cross-compiler set up

Christine Chan
Motivation

- Embedded devices often have smaller (if any) operating system installed than host computers
- A development **host** compiles applications into executables, and the **target** embedded device runs them
- Cross-compilation is when the host platform != target platform
Project stages

- Part 1
  - Own machine required, with wired/wireless internet

- Part 2
  - Android development phone provided, with WiFi
  - Connects to host machine by USB
Part 1 deliverables

- Install Linux environment and development tools
- Cross-compile a C program on an x86 host machine to run on embedded ARM processors
Host requirements

- Intel x86 64-bit processor
  - Windows or Mac OK
- Ubuntu 14.04 LTS
  - Root access required
  - Recommend running on a virtual machine
  - Probably OK on bare metal or Ubuntu 12.04 if you already have it
Tools for Android development

- **cURL**
  - Transfer data to/from server (to download project files)

- **ARM cross-compiler (arm-linux-gnueabi-cc)**
  - **gcc** compiles C programs
  - for **arm** architecture
  - on **linux** target system (your laptop or desktop)
  - Using **eabi** embedded application binary interface convention

- **Android Debug Bridge (adb)**
  - Command-line tool that communicates with an Android device
  - Rest of the Android SDK is not required
Cross-compilation workflow

- On the host
  - arm-linux-gnueabi-gcc -o test.out test.c # compile
  - file test.out # sanity check
  - adb push test.out /data/local

- On the target (using adb shell)
  - ./data/local/test.out # run

- For part 1, just submit test.out on ted.ucsd.edu