Arduino Build Process

Overview

A number of things have to happen for your Arduino code to get onto the Arduino board. First, the Arduino environment performs some small transformations to make sure that the code is correct C or C++ (two common programming languages). It then gets passed to a compiler (avr-gcc), which turns the human readable code into machine readable instructions (or object files). Then, your code gets combined with (linked against), the standard Arduino libraries that provide basic functions like digitalWrite() or Serial.print(). The result is a single Intel hex file, which contains the specific bytes that need to be written to the program memory of the chip on the Arduino board. This file is then uploaded to the board: transmitted over the USB or serial connection via the bootloader already on the chip or with external programming hardware.

Multi-file sketches

A sketch can contain multiple files (tabs). To manage them, click on the right-facing arrow just above the scroll bar near the top of the environment. Tabs have one of four extensions: no extension, .c, .cpp, or .h (if you provide any other extension, the period will be converted to an underscore). When your sketch is compiled, all tabs with no extension are concatenated together to form the "main sketch file". Tabs with .c or .cpp extensions are compiled separately. To use tabs with a .h extension, you need to #include it (using "double quotes" not <angle brackets>).

Transformations to the main sketch file

The Arduino environment performs a few transformations to your main sketch file (the concatenation of all the tabs in the sketch without extensions) before passing it to the avr-gcc compiler.
First, #include "Arduino.h", or for versions less than 1.0, #include "WProgram.h" is added to the top of your sketch. This header file (found in <ARDUINO>/hardware/cores/<CORE>/) includes all the defintions needed for the standard Arduino core.
Next, the environment searches for function definitions within your main sketch file and creates declarations (prototypes) for them. These are inserted after any comments or pre-processor statements (#includes or #defines), but before any other statements (including type declarations). This means that if you want to use a custom type as a function argument, you should declare it within a separate header file. Also, this generation isn't perfect: it won't create prototypes for functions that have default argument values, or which are declared within a namespace or class.
Finally, the contents of the current target's main.cxx file are appended to the bottom of your sketch.

Targets

The Arduino environment supports multiple target boards with different chips (currently, only AVRs), CPU speeds, or bootloaders. These are defined in a board preferences file (http://arduino.cc/en/Hacking/Preferences). Relevant variables include:
<BOARD>.name: the name to display in the Boards menu
<BOARD>.build.mcu: the microcontroller on the board (normally "atmega8" or "atmega168").
<BOARD>.f_cpu: the clock speed at which the microcontroller operates (normally "16000000L", or, for an ATmega168 running on its internal clock, "8000000L").
<BOARD>.core: which sub-directory of the hardware/cores/ directory to link sketches against (normally "arduino").
Also useful is this setting in the main preferences.txt file:
build.verbose: whether or not to print debugging messages while building a sketch (e.g. "false"). If true, will print the complete command line of each external command executed as part of the build process.
Note: that in Arduino 0004 and later, build.extension is **unused** - the main sketch file is always treated as a .cpp file.

Build process

Sketches are compiled by avr-gcc.
The include path includes the sketch's directory, the target directory (<ARDUINO>/hardware/core/<CORE>/) and the avr include directory (<ARDUINO>/hardware/tools/avr/avr/include/), as well as any library directories (in <ARDUINO>/hardware/libraries/) which contain a header file which is included by the main sketch file.
When you verify a sketch, it is built in a temporary directory in the system temp directory (e.g. /tmp on the Mac). When you upload it, it is built in the applet/ subdirectory of the sketch's directory (which you can access with the "Show Sketch Folder" item in the "Sketch" menu).
The .c and .cpp files of the target are compiled and output with .o extensions to this directory, as is the main sketch file and any other .c or .cpp files in the sketch and any .c or .cpp files in any libraries which are #included in the sketch.

These .o files are then linked together into a static library and the main sketch file is linked against this library. Only the parts of the library needed for your sketch are included in the final .hex file, reducing the size of most sketches.
The .hex file is the final output of the compilation which is then uploaded to the board. During a "Verify" the .hex file is written to /tmp (on Mac and Linux) or \Documents and Settings\<USER>\Local Settings\Temp (on Windows). During upload, it's written to the applet sub-directory of the sketch directory (which you can open with the "Show Sketch Folder" item in the Sketch menu).

Upload process

Sketches are uploaded by avrdude.
The upload process is also controlled by variables in the boards and main preferences files. Those in the boards file include:
<BOARD>.upload.protocol: the protocol that avrdude should use to talk to the board (typically "stk500").
<BOARD>.upload.speed: the speed (baud rate) avrdude should use when uploading sketches (typically "19200").
<BOARD>.upload.maximum_size: the maximum size for a sketch on the board (dependent on the chip and the size of the bootloader).
And in the main preferences file:
upload.verbose: whether or not to dump debugging messages while upload a sketch to a board (defaults to "false").

Share

(https://twitter.com/arduino) (http://www.facebook.com/official.arduino)

(https://plus.google.com/+Arduino) (http://www.flickr.com/photos/arduino_cc)

(http://youtube.com/arduinoteam)