Node.js code cache builder

This is the V8 code cache builder of Node.js. It pre-compiles all the JavaScript native modules of Node.js and serializes the code cache (including the bytecodes) that will be embedded into the Node.js executable. When a Node.js JavaScript native module is `require`d at runtime, Node.js can deserialize from the code cache instead of parsing the source code and generating the bytecode for it before execution, which should reduce the load time of these JavaScript native modules.

How it's built and used

The code cache builder is built with the `mkcodecache` target in `node.gyp` when `node_use_node_code_cache` is set to true, which is currently done by default.

In the default build of the Node.js executable, to embed the V8 code cache of the native modules into the Node.js executable, `libnode` is first built with these unresolved symbols:

- `node::native_module::has_code_cache`
- `node::native_module::NativeModuleEnv::InitializeCodeCache`

Then the `mkcodecache` executable is built with C++ files in this directory, as well as `src/node_code_cache_stub.cc` which defines the unresolved symbols.

`mkcodecache` is run to generate a C++ file `<(SHARED_INTERMEDIATE_DIR)/node_code_cache.cc` that is similar to `src/node_code_cache_stub.cc` in structure, but contains the code cache data written as static char array literals. Then `libnode` is built with `node_code_cache.cc` to produce the final Node.js executable with the code cache data embedded.

For debugging, Node.js can be built without code cache if `--without-node-code-cache` is passed to `configure`. Note that even if the code cache is not pre-compiled and embedded into the Node.js executable, the internal infrastructure is still used to share code cache between the main thread and worker threads (if there is any).