Hi everyone this is Tracy Salander. By making this video, I would like to share something about Substrate I learned.

What is Runtime

[runtime\_cargo图]

Runtime is mainly used to build the chain. It puts various pallets together into a WASM file. In addition to the similar content in cargo.toml corresponding to pallet, cargo.toml also add the buid script which is the dependency wasm-builder-runner of build.rs. Substrate build-in and developer-defined runtime module (also called pallet). ‘build.rs’ uses wasm-builder-runner to compile the current runtime project into Wasm.

[wasm图]

The compiled file is located in target/release/wbuiild/node-template-runtime/node\_template\_runtime.compact.wasm.

[runtime\_lib图]

‘src/lib.rs’ is the entry point to construct the runtime on our chain. Line 1 indicates that if the feature is not std when compiling, it must be no\_std. Line 3 is used to set the maximum threshold for infinite recursive operations such as macro expansion that may occur during compilation. When compiling with std feature, line 6 and line 7 will introduce the generated Wasm binary content into the current runtime code in a constant way. Runtimes imports dependent modules, and exposes some functions and data types of upstream modules for the covenience of downstream modules, such as line 36. Line 44 imports the template module. Line 47 aliases the basic type required by runtime and the principle is to be consistent with the associate type name in the module.

[opaque图]

The opaque module encapsulates some types used for CLI initialization, and specific information about these types and runtime. By the way, CLI is command-line interface, which refers to an interface where executable instructions can be entered at the user prompt. It usually does not support a mouse. The user inputs instructions through the keyboard and the computer executes the instructions after receiving the instructions. From the line 91 to line 105, this part specifies the runtime version information. When the runtime protocol is modified, ‘spec\_version’ needs to be increased by 1. ‘impl\_version’ is the implementation version of the protocol, which is used to indicate that the code running on the node is different, and this can only be modified when non-consensus-related optimization occurs. ‘RUNTIME\_API\_VERSTIONS’ contains all version information of the implemented runtime api, which is generated by the impl\_runtime\_apis macro. Line 113 defines constants related to block time, that is, each block is 6 seconds, and the configuration can be modified as needed.

[native图]

From line 126 to line 128, this part specifies the current NativeVersion, and compares the NativeVersion with the RuntimeVersion on the chain when executing the transaction. If it is inconsistent, usually, Wasm will execute the transaction. Next part uses the parameter\_types macro to generate some data types that meet the Get interface required by the later functional modules.

[construct\_runtime图]

In order to realize the interface of each functional module of the runtime, the runtime is generated by the construct\_runtime macro. The construct\_runtimemacro constructs the runtime according to the name such as TemplateModule and the components in the used module such as template::(Module, Call, Storage, Event), so that the information in the module is exposed through metadata, and the module is available in runtime. During the construction, the initial storage is loaded in order, so when module B depends on module A, module A should be placed before module B.

[runtime\_apis图]

Starting from line 328, it implements the interface.

[node\_cargo图]

Now, let’s see the node. For node part, right now. On the client side, those that are not on-chain operations are written here, the underlying ones, such as the network, consensus, and transaction pool. ‘cargo.toml’ uses [bin] to indicate that this package is executable. It imports compile-time dependencies through build-dependencies, and is used in ‘build.rs’. The other content has been introduced before.

[build图]

The content of ‘build.rs’ is the same as ‘build.rs’ in the workspace root directory.

[main图]

‘src/main.rs’ is the entry file of node-template compiled into executable program. By line 2, at compile time, warning will be printed when the module lacks documentation. This file imports other code modules in the current directory, such as mod chain\_spec. #[macro\_use] will load all the macros under the imported module. The main function is the entry point of the program, and it returns a custom Result type. Then it executes the run function provided by the command module.

[command图]

‘src/command.rs’ provides the run function required by main. It parses the execution parameters of the command line through from\_args and returns a Cli structure.

[general图]

‘chain\_spec.rs’ constructs ChainSpec, which defines the available configuration of the chain, used to construct the initial block.

‘src/cli.rs’ declares the client structure and subcommands.

‘src/service.rs’ provides tools and methods for constructing Substrate services.

‘src/rpc.rs’ provides a collection of RPC methods specified by the node. RPC refers to remote procedure calls, that is to say, two servers A and B, and one application on server A. If you want to call functions or methods provided by the application on the B server, it cannot be called directly because it is not in a memory space. Instead, it needs to express the semantics of the call and convey the data of the call through the network.

‘src/lib.rs’ is used to introduce library modules.

Thank you for watching.