Main Template B/maint

Purpose

The top-level logic of NI: the sequence of operations followed by NI up to the point where the output file is opened, resuming after it is closed again.

B/maint.§1 Startup; §2 Semantic Analysis; §3 Assertion Reading; §4 Model Construction; §5 Tables and Grammar; §6 Phrases and Rules; §7 Code Generation; §8 Metadata; §9 Indexing; §10 Shutdown

---

§**1. Startup.** This segment of the template is not like any other. It contains almost the complete logical sequence of operations followed by NI – indeed, NI essentially works by parsing some command-line arguments to set switches and then asking the template interpreter to run through "Main.i6t", the only template file which must always exist. (The other template files are only ever involved when called on by the `{-segment:...}` command from a template file already running.) Whereas most template segments contain I6 code to pass through into NI's output, this one runs both before and after NI's output file is being written, and contains only commands.

The Startup paragraph is not in fact modifiable in any easy way, because of course "Include..." sentences – used to tell the template interpreter to override the template files – will not be understood until long after this paragraph has been fully dealt with. But perhaps that's no bad thing.

```
{-callv:Memory::start}
{-callv:Text::start}
{-callv:Text::Vocabulary::stock}
{-callv:Config::Plugins::start}
{-callv:World::begin}
{-callv:Extensions::Files::handle_census_mode}
{-progress-stage:0}
{-log-phase:Lexical analysis}
{-callv:Text::Reader::read_primary_source_text}
{-callv:Parser::Sentences::create_standard_csps}
```

§**2. Semantic Analysis.** Similarly: Include... sentences are not read until this paragraph is long forgotten.

```
{-progress-stage:1}
{-log-phase:Semantic analysis Ia}
{-callv:Parser::Nodes::plant_parse_tree} ! Initialise the parse tree
{-callv:Parser::Sentences::break_source} ! Build first tranche of sentences
{-callv:Extensions::Inclusion::traverse} ! Expand extension inclusions and build sentences
{-callv:Parser::Sentences::Headings::satisfy_dependencies}
{-log-phase:Initialise language semantics}
{-plugin:load}
{-callv:Semantics::BPs::make_built_in}
{-callv:Semantics::VerbUsage::stock}
{-callv:Semantics::Quantifiers::make_built_in}
{-log-phase:Semantic analysis Ib}
{-callv:Parser::Sentences::VPs::traverse} ! Find verbs in the assertion sentences
{-callv:Text::traverse_for_plural_definitions} ! Build irregular plurals dictionary
{-callv:Parser::Sentences::tidy_up_ofs_and_froms} ! More "of" wrangling
{-callv:Parser::Sentences::register_recently_lexed_phrases} ! To make commands children of their ro
```

```
  ... utines
{-callv:Parser::Sentences::declare_source_loaded}
{-callv:Kinds::Interpreter::include_templates_for_kinds}
{-log-phase:Semantic analysis II}
{-callv:Parser::Nodes::verify} ! Purely to check that NI is running correctly
{-callv:Extensions::Files::check_versions} ! Do the extension version numbers meet needs?
{-callv:Parser::Sentences::Headings::make_tree} ! Stratify headings and subheadings
{-callv:Parser::Sentences::Headings::write_as_xml} ! Dump them to a file for the GUI to use
{-log-phase:Semantic analysis III}
{-callv:Code::Phrases::Adjectives::traverse}
{-callv:Data::Equations::traverse_to_create}
{-callv:Data::Tables::traverse_to_create}
{-callv:Code::Phrases::traverse_for_names}
```

§**3. Assertion Reading.** Since Include... sentences are read during pass 2, they might just be able to meddle by adding instructions to take place after this paragraph, but would be too late to work before or instead of it.

```
{-progress-stage:2}
{-log-phase:First pass through assertions}
{-read-assertions:1}
{-log-phase:Second pass through assertions}
{-read-assertions:2}
```

§4. Model Construction.

```
{-log-phase:Making the model world}
{-callv:Properties::Implementation::OfObjects::allocate_attributes}
{-callv:Plugins::Actions::name_all}
{-callv:Data::Nametags::name_all}
{-callv:World::complete}
{-callv:Properties::Measurement::validate_definitions}
{-callv:Semantics::BPs::make_built_in_further}
```

§5. Tables and Grammar.

```
{-log-phase:Tables and grammar}
{-callv:Data::Tables::check_tables_for_kind_clashes}
{-callv:Data::Tables::traverse_to_stock}
{-callv:Data::Equations::traverse_to_stock}
{-callv:Plugins::Parsing::traverse}
{-callv:World::complete_additions}
```

§**6. Phrases and Rules.**

```
{-progress-stage:3}
{-log-phase:Phrases and rules}
{-callv:Semantics::Nouns::LiteralPatterns::define_named_phrases}
{-callv:Code::Phrases::traverse}
{-callv:Code::Phrases::register_meanings}
{-callv:Code::Phrases::parse_rule_parameters}
{-callv:Code::Phrases::add_rules_to_rulebooks}
{-callv:Code::Phrases::parse_rule_placements}
{-callv:Problems::empty_all_headings}
```

§**7. Code Generation.** This is where we hand over to regular template files – containing code passed through as I6 source, as well as a few further commands – starting with "Output.i6t".

```
{-progress-stage:4}
{-log-phase:Code generation}
{-open-file}{-segment:Output.i6t}{-close-file}
{-log-phase:Compilation now complete}
```

§**8. Metadata.**

```
{-callv:Plugins::Bibliographic::write_ifiction_and_blurb}
```

§**9. Indexing.** This paragraph can be skipped without harming any of the rest of NI's work: the only effect is to suppress the generation of the index. (Indeed, if NI is called with the `-noindex` command-line switch, then the template interpreter ignores all commands in between `{-open-index}` and `{-close-index}`, thus skipping exactly this paragraph.)

```
{-open-index}
{-index:Phrasebook Index=A guide to the phrases allowed; [LEXICON]a lexicon of words; a [RELTABLE]t
  ...  able of relations, and [VERBTABLE]of verbs.|What are phrases?<PHRASES>; And descriptions?<D
  ...  ESCRIPTIONS>; Relations?<RELATIONS>; Verbs?<VERBS>=The Phrasebook Index}
    {-callv:Code::Phrases::index_page_Phrasebook}
{-index:Kinds Index=Table of all the kinds; [ARITHMETIC]how arithmetic affects them; and [KDETAILS]
  ...  details of each kind in turn.|What are kinds?<KINDS>; More about kinds of object<NEWKINDS>;
  ...    And kinds of value<KINDSVALUE>=The Kinds Index}
    {-callv:Data::Objects::page_Kinds}
{-index:World Index=A map of the geographical layout; [MDETAILS]contents of each room; and [LEXICON
  ...  ]an A to Z of rooms and things.|What's the map?<MAP>; Using regions<REGIONS>; If the map lo
  ...  oks wrong...<MAPHINTS>; Exporting to EPS<EPSMAP>=The World Index}
    {-callv:Data::Objects::page_World}
{-index:Actions Index=The actions (click magnifiers for details); [NAP]kinds of action; [COMMANDS]c
  ...  ommands A to Z; [LEXICON]actions A to Z; [ARULES]the applicable rules.|What are actions?<AC
  ...  TIONS>; Creating new actions<NEWACTIONS>; Out of world actions (in red)<OUTOFWORLD>=The Ac
  ...  tions Index}
    {-callv:Plugins::Actions::Index::page}
{-index:Rules Index=Rules not directly tied to actions (see the Actions Index) or scenes (the Scene
  ...  s Index).|What are rulebooks?<RULEBOOKS>; What are activities?<ACTIVITIES>; Moving or aboli
  ...  shing rules<RLISTING>=The Rules Index}
    {-callv:Code::Rulebooks::index_page}
```

```
{-index:Contents Index=Headings and [EXTLIST]extensions; [NAMES]named values, Tables and so on; the
  ...    [LCARD]Library Card; the [STORYFILE]story file.|How do headings work?<HEADINGS>; What's th
  ...  e Library Card?<LCARDS>=The Contents Index}
    {-callv:Parser::Sentences::Headings::index}
    {-callv:Extensions::Files::index}
    {-callv:Extensions::Files::update_census}
    {-callv:Data::NonlocalVariables::index_all}
    {-callv:Data::Tables::index}
    {-callv:Data::Equations::index}
    {-callv:Plugins::Figures::index_all}
    {-callv:Plugins::Sounds::index_all}
    {-callv:Plugins::Files::index_all}
    {-callv:Plugins::Bibliographic::index_library_card}
{-index:Scenes Index=A map of how the scenes begin and end; [SEVENTS]timed events, if any; [SRULES]
  ...  general rules about scenes; and [SDETAILS]details of each scene in turn.|What are scenes?<S
  ...  CENESINTRO>; How they link together<LINKINGSCENES>; About timed events<TIMEDEVENTS>=The Sc
  ...  enes Index}
    {-callv:Plugins::Scenes::index}
{-callv:Index::complete}
{-close-index}
```

§**10. Shutdown.** Closing the problems report, making any final reports to the debugging log, and freeing all allocated memory: and that's then it.

```
{-callv:Config::Template::report_unacted_upon_interventions}
{-callv:Problems::complete_problems_report}
{! -callv:Specifications::Taxa::report_pairs_observed}
{! -callv:Specifications::Taxa::report_pairs_allowed}
{! -callv:Memory::log_statistics}
{-callv:Index::log_statistics}
{! -callv:Parser::SParser::debug_parser_statistics}
{! -callv:Semantics::VerbUsage::log_all}
{-callv:Memory::free_memory}
```