Ruby - Feature #18360

PrettyPrint enhancements

11/24/2021 12:15 AM - kddnewton (Kevin Newton)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

The message below is a duplicate of the commit message and the PR that I just opened on GitHub (https://github.com/ruby/ruby/pull/5163). I figured I'd open this issue here as there would probably be more discussion than just a GitHub PR warranted.

Overall

This commit adds a bunch of new functionality to the PrettyPrint class. It is part of my continued work on the Ruby formatter sponsored by the Ruby association (https://www.ruby.or.jp/en/news/20211025). The goal is to enhance the PrettyPrint class enough to support all of the necessary print functionality that the formatter will require in order to print out Ruby source correctly.

Current algorithm

First, let's take a look at how PrettyPrint works today. PrettyPrint has 3 primitives:

- Group - an object that represents a set of content and its associated breakpoints. These objects are usually nested. When one of them no longer fits on a single line, it is broken at all of its associated breakpoints. For example, if you have something like [text("abc"), breakable, text("def")] and that group no longer fits on one line, then you will have two lines of output.
- Breakable - an object that represents a location where a line of content can be split. When it is created, if the current group is broken already, then it inserts a newline and carries on. If it is not, then it adds itself to the output buffer.
- Text - this is a set of objects that are appended to a buffer. If the buffer overflows the maximum print width for the line, then the surrounding group is broken and the buffer is flushed to the output.

With those primitives in place, the printer maintains a buffer of content that is being added. If the buffer overflows the line, the content is flushed until another group is hit or there are no more breakpoints.

Limitations

There are a couple of limitations with the current approach.

- PrettyPrint assumes that content in Text will not change its representation if it is contained within a broken group versus contained within a flat group. This isn't a problem for the existing uses of PrettyPrint, but for my purposes of building a formatter, it definitely is. Consider something like trailing commas (where you want a comma if it is broken but nothing if it's not) or block operators (where you would use a do and end for multi-line (broken group) or braces for single line (flat group)).

- The Breakable class assumes that you always want to indent the subsequent line up to the current indentation level. This is true in most cases, and certainly for all the existing use cases. But there are times when you don't want that behavior (for example if you're in the middle of a nested indentation structure but have to force content to start at the beginning of the next line as in a comment with =begin..=end bounds).

- There's no way to force a group to break. You can access the current group in the printer with current_group, but that won't force the parent groups to break. Without hacking around a lot of stuff, it's difficult to get this behavior. This is necessary if you want to ensure a newline is added and respected, like after a keyword where it would be necessary.

Enhancements

This commit adds a couple new nodes to the printing tree, as well as enhancing the Breakable class. First, the new nodes:

- Align - this node effectively wraps the old @indent variable but allows you to align content within any of the other containers. You can also align to a string now instead of just an integer, which will print that string before each line.
- BreakParent - enforces that the surrounding group and all ancestral groups are broken.
- IfBreak - contains two sets of nodes, one for if the surrounding group is broken and one for if the surrounding group is flat.

- Indent - similar to the align node, but you don't have to specify anything and it just indents by one level.
- LineSuffix - this is a big enhancement to the printing algorithm that maintains a separate buffer for content that should be flushed before the next newline. This is convenient for implementing things like heredocs and trailing comments.
- Trim - a rarely used but important node that trims off whitespace that has already been added to the buffer in the case that you need to force something to begin at the start of the next line.

As for the enhancements to Breakable:

- It now accepts a force parameter (default to false), which will insert a BreakParent and slightly change semantics so that a newline is *always* added.
- It now accepts an indent parameter (default to true), which allows you to specify if you want the subsequent line to indent automatically.

Compatibility

For the most part, the code is completely compatible with the previous version.

There are a couple of things that were removed that appeared to be all internally-facing functions. When they were removed all of the tests still passed, so I'm assuming they were only called internally. I can certainly add them back if it's deemed too risky but I very much doubt this is a problem.

- indent attr_reader which is now encapsulated in the printing algorithm
- group_queue attr_reader which had a reference to a queue that is no longer necessary
- break_outmost_groups method, which is now encapsulated in the printing algorithm
- group_sub method, which was only called by the group method anyway and is no longer necessary

There were a bunch of things that were added, including:

- force and indent parameters to the breakable method (they both have defaults so this shouldn't be an issue)
- Align, BreakParent, IfBreak, Indent, LineSuffix, and Trim nodes
- Buffer::DefaultBuffer, Buffer::StringBuffer, and Buffer::ArrayBuffer, which is just there to provide the ability to trim trailing whitespace
- PrettyPrint.visit(doc, visitor), which is useful for debugging and also necessary for propagating break parent nodes up the tree

All in all, none of the tests had to change, which is a good sign.

Summary

From the user of this class's perspective, nothing is different. Internally however, there's a bunch of additional functionality and a lot more control over the printing algorithm! Also the ability to debug has been greatly enhanced with pretty_print methods on each of the nodes and the ability to walk the print tree nodes before they're printed.

I'd still like to add some more tests and work on the documentation a bit more before it's merged. Also, the RBS tests are failing because some of the things have changed type signatures. I'm not sure if I should open a PR over there first or what the policy is for that. Please let me know what I should do with regard to that gem.

History

#1 - 08/09/2023 01:20 PM - kddnewton (Kevin Newton)

- Status changed from Open to Closed

Files

prettyprint.patch	47.8 KB	11/24/2021	kddnewton (Kevin Newton)