A regression test suite for LaTeX 2_ε

Frank Mittelbach

Abstract

This paper describes the history of the development of a regression test suite for LaTeX and its importance for the release of stable and reliable future distributions of that software. A more detailed description of the concepts and the implementation of the test suite will be given in [1].

As experience shows that there can't be enough test files in such a suite, we make a plea to the TeX community to help us in making LaTeX distributions even more reliable by joining a new volunteer group working on the task of updating and adding to this suite.

1 Introduction

Back in 1992 when the LaTeX3 team took over maintenance of LaTeX and started to work on the current LaTeX version [3], also known as LaTeX 2_ε, I had the idea of producing a test environment for LaTeX that would help us in providing stable and reliable distributions. My idea originated in the `trip` test for the TeX program [2], a fixed test file which is run through TeX, containing code that tries to exercise as many boundary cases as Don Knuth could think of. The output of this run is then compared with a set of files certified by Don to contain the correct information. Only if a new implementation of the TeX program produces the same output (with well defined minor deviations in certain places) is it allowed[1] to be called TeX. The idea behind this is to ensure that TeX behaves identically on all implementations and the `trip` test was the measure proving this.

With LaTeX the idea was not to ensure that it is identical on all platforms — this is automatically the case if the standard installation is obtained and the installation procedures are applied — but to ensure that that new releases of LaTeX do not inadvertently modify the behavior of commands. Since LaTeX is a large and complex system, this is definitely a non-trivial task: in 'fixing' one bug, it is often necessary to modify the definitions of several 'internal' commands, and these may in turn affect many other commands which have no obvious connection with the original problem.

We have had some pretty disastrous experiences of this type, often finding that harmless looking corrections had effects on what seemed, at first glance, completely unrelated areas. This is in part due to the fact that LaTeX is based on the macro language of TeX, which allows reuse and redefinition of arbitrary code fragments.

For that reason we started working on a concept for automated tests to detect such problems. When that system was available, we asked for volunteers to help us in building up a suitable test suite for LaTeX (which at that time was LaTeX 2.09). Part of the rationale behind this work was to ensure that a future transition from LaTeX 2.09 to LaTeX 2_ε (for which development was under way) would become as painless as possible, i.e., these tests were also supposed to ensure that the new code for LaTeX 2_ε would not change the user interface behavior without detection.

This approach seems to us to have been very successful; this is in large parts due to the quality and quantity of the work of the volunteers helping us at that time, in particular Daniel Flipo and Chris Martin. Figure 1 shows an excerpt from the volunteer task list from 1993 describing this task (and my rather optimistic time requirements for it).

When LaTeX 2_ε was released for the first time in 1994 we updated the regression test support macros and tried to improve the test suite by adding new test files when we fixed bugs or when we added new functionality to LaTeX. However, being human, we have not followed this practice as rigorously as we should have: especially since the first releases it has become more and more common for us to fix a small bug without spending the additional time necessary to also write a test file that exhibits the correct behavior.

Today our test suite has about 300 test files which are automatically executed and compared before a new release hits the streets. And indeed, these test files have saved us from embarrassment many times already.

2 This year's boo-boos!

However, results show that such a suite can never be large enough to avoid the need for a patch release once in a while. It is particularly important that new features, such as the release of additional files or the correction of recently found bugs, get tested and frozen within this suite so that there is no unexpected change later on. For example, with the December 1997 release we added the packages

[1] An additional requirement according to the `trip` test documentation is that the author of the TeX implementation has to be satisfied with the product. In other words, a simple program that throws away all its input and always output the files needed to satisfy the `trip` test would be allowed to call itself TeX as long as the author of that program is happy with it.

Validating LaTeX2.09

Writing test files for regression testing: checking bug fixes and improvements to verify that they don't have undesirable side effects; making sure that bug fixes really correct the problem they were intended to correct; testing interaction with various document styles, style options, and environments.
We would like three kinds of validation files:

1. General documents.
2. Exhaustive tests of special environments/modules such as tables, displayed equations, theorems, floating figures, pictures, etc.
3. Bug files containing tests of all bugs that are supposed to be fixed (as well as those that are not fixed, with comments about their status).

A procedure for processing validation files has been devised; details will be furnished to anyone interested in this task.

Estimated time required: 2 to 3 weeks, could be divided up.

Coordinator [25 August 1992]:
Daniel Flipo `flipo@citil.citilille.fr`
Other volunteers:
Chris Martin `cs1cwm@sunc.sheffield.ac.uk`

Figure 1: An excerpt from the volunteer task list 1993

`calc` and `textcomp` to the distribution but, due to time constraints, did not add to the suite additional test files designed to exercise these packages; and, by Murphy's law, `textcomp` did not contain a necessary `\ProvidesPackage` command, with the result that it claimed to be written for a future release[2] — something that would have been caught by any test file exercising the package.

Another embarrassing example of a missing test file in that release was the `\t` error. To better support language files from the Babel suite, some of which make the `"` character active, we changed all internal definitions of characters and accents from hexadecimal notation, such as `"7F`, to decimal, i.e., to `127` in that case. Unfortunately in the definition for `\t` we did this wrong and `"7F` became `79`, giving very strange effects when the accent was used.[3] An error like this would have been automatically caught if we had, for each output encoding, a test file to check that each definition in the encoding results in the 'right' glyph or glyphs.

[2] The technical reason for this behavior, for those who wonder, was that the release date of the package, which is an optional argument to the (missing) `\ProvidesPackage` command, was there but was mistakenly picked up the `\NeedsTeXFormat` which then produced a warning as the release date of `textcomp` was later than the nominal release date (of 1997/12/01) for the format of the distribution.

[3] Both errors got found and reported several times within two days after the release, so the patch release came out quite quickly this time.

3 Call for volunteers

Thus to make the LaTeX system even more reliable we call on you for help! What we hope to find is a new group of volunteers that is interested in working on an extension of the LaTeX test suite system. There is no need to be an expert TeX or LaTeX programmer for this task though some experience with LaTeX and its inner workings will be necessary.

If you are interested in joining this effort, please contact Daniel Flipo at

`Daniel.Flipo@univ-lille1.fr`

who kindly agreed to act as a coordinator between the individual volunteers.

There are a number of areas in which further test files would improve the system enormously. They are outlined in the following sections.

3.1 Testing existing interfaces

Testing existing interfaces is a very important task, one not so far, for several reasons, adequately covered by the test suite. This will not only help us to detect problems when fixing errors in LaTeX but, more importantly, it will help one day in the transition to a new system since these test files will then clearly identify which interfaces are compromised (deliberately or by mistake) by the new system. This in turn will then help to produce, if necessary, procedures to automatically translate source documents from LaTeX to its successor.

What we are looking for are test files that describe and test the current interfaces on all levels. This is certainly an ambitious task, but perhaps also one of the most interesting and rewarding ones within this list.

3.2 Testing corrected bugs

As described above, several of the bugs reported to us have been fixed and a test file showing the correct behavior has been added. But for many this is not the case.

What we are looking for is the provision of test files for all bugs reported and fixed, so that future releases will not by mistake revert any of these fixes without alerting the maintainers. This means working through our bug database and devising test files showing the correct behavior. As we ask submitters of bug reports to send in a test file that shows the incorrect behavior, and they usually do so, it is often possible to start from the submitted file and modify it slightly so that it fits into the regression suite concepts.

3.3 Testing new extensions

What is important for the kernel interfaces is also important for the core packages and extensions: these interfaces should be exercised in such a way that any future changes will be automatically detected. Again this provides interesting mental exercise since it isn't always easy to decide what is pertinent for the interface and how to exercise it so that enough (but not too much) information ends up in the `.log` file.

3.4 Testing contributed packages

A final area which is important is the testing of packages which lie outside the control of the LaTeX maintainers. Although we cannot in all cases guarantee that corrections to the kernel software will not harm any such package, we are, of course, very much concerned to avoid making any change that makes third party packages invalid. In the past, whenever we noticed (or even suspected) such a problem we tried either to avoid it, by choosing a different solution, or, if that was not possible for some reason, to find the maintainers of the package and give them notice of a possible clash so that such problems could be avoided.

There is a problem with testing the interfaces of third party packages: changes by the package author, to either the interface or the implementation of the package, can upset the test suite as easily as can changes to the LaTeX kernel by the LaTeX3 project team. Thus, to avoid our limited time resources being used up in chasing after errors introduced in this way (being neither our fault nor being correctable by us), it would be necessary to develop clear protocols for how this part of the test suite should be maintained, e.g., what requirements a package must fulfill to be included into it, what obligations an author of such a package agrees to, etc. This is not yet done and so it is part of the volunteer effort.

We close our plea for help with a quote taken from [2] which shows how the Grand Wizard sees the task of writing such test files (which does not mean you have to follow his advice):

> To write such a fiendish test routine, one simply gets into a nasty frame of mind and tries to do everything in the unexpected way. Parameters that are normally positive are set negative or zero; borderline cases are pushed to the limit; deliberate errors are made in hopes that the compiler will not be able to recover properly from them.
>
> *Donald Knuth 1984*

References

[1] David Carlisle and Frank Mittelbach. *The LaTeX regression test suite: concepts and implementation. TUGboat*; to appear.

[2] Donald E. Knuth. A torture test for TeX. Report STAN-CS-84-1027, Stanford University, Department of Computer Science, Stanford, CA, USA, 1984.

[3] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994.

◇ Frank Mittelbach
LaTeX3 project
`Frank.Mittelbach@eds.com`