I love deadlines.  
I like the whooshing sound they make as they fly by.  
(Douglas Adams)

Automatic binary building for \TeX{} Live  
(using buildbot)

Mojca Miklavec  
TUG@Bacho\TeX{}, 1\textsuperscript{st} May 2017
From: Karl Berry <karl@...>
To: tlbuild <tlbuild@...>

Hopefully the synctex-related bugs have been ameliorated now, so please rebuild, those of you who haven’t already.

--thanks, karl.
Dear Karl,

I know you are in the middle of finalizing the TeX Live 2017. I’m aware that I volunteered to build binaries for 8 platforms. But I’m super busy at the moment and will be gone in April & May for two weeks, most likely without internet connectivity.

Sorry,

Mojca
Binaries in TeX Live

~20 platforms

effort by a number of volunteers

built “once” per year – reasonable compromise between:

○ demand for new binaries
○ burden on volunteer builders and packagers
○ stability & amount of testing (LuaTeX)
Disadvantages

- No access to the latest features for more than a year.
- Long and “painful” bugfixing period around March.
ConTeXt Distribution (I)

- \(\sim 13 + 2\) platforms (5 more dropped for lack of interest)
- 100% compatible with T\(\text{E}\)X Live
- Windows binaries from W32T\(\text{E}\)X
- other binaries built by volunteers with a single command (or by cronjobs)
- distribution checks for updates every 15 minutes, binaries for every MetaPost, LuaT\(\text{E}\)X, X\(\text{E}\)T\(\text{E}\)X release
- (binaries found their way to TL users via TLContrib)
ConTeXt Distribution (II)

- many build problems discovered during the year (in particular on exotic platforms like Solaris), less debugging left for the tex Live freeze period

- little work to do, but it would be even better if computers would do all the work for us

- number of builders decreased, Hans started setting up VMs for binary builds at Pragma

- it would be nice to have nightly LuaTEX builds
Automating the builds
What is Buildbot?

- framework for automating builds, highly customisable (Python)
- **Build Master**: central server, delegates work, collects results
- **Build Slaves**: multiple “stupid” computers to compile binaries
Build Slaves

- a number of computers on a variety of platforms
- may live behind a firewall on a private network
- easy to set up

  - Debian: `sudo apt-get install buildbot-slave`
  - PIP: `pip install buildbot-slave`
  - `buildslave create-slave /some/path your.server:9989 name pass`

- install just compiler and any build dependencies, no need to worry what to build / how to build it
Build Master

- a single server, publicly accessible
- slaves connect to it with username & password
- contains all the “brains” to delegate work:
  - checks for software updates, schedules builds
  - sends build commands to build slaves
  - sends emails on build failures
  - collects results, may distributes binaries
Build Slaves

- (4) Solaris: sparc (sparc64), i386 (x86_64) @ opencsw.org
- (4) Mac OS X 10.6: (x86_64, i386, ppc) + macOS 10.12
- (5) Linux:
  - Raspbian (armhf) @ Raspberry PI
  - Debian 7 (oldstable) & 9 (testing), (i386, x86_64)
- (4) OpenBSD: 6.0, 6.1 (i386, amd64)

TODO:
- (2) Linux: CentOS 5.11 (i386, x86_64)
- (4) FreeBSD: 9.3 & 11.0 (i386, amd64)
- (2?) NetBSD
- (2) mingw64 on Linux for cross-compilation (win32, win64)
Demo
### Starting the build

<table>
<thead>
<tr>
<th>Platform</th>
<th>Status</th>
<th>Build 1</th>
<th>Build 2</th>
<th>Build 3</th>
<th>Build 4</th>
<th>Build 5</th>
<th>Build 6</th>
<th>Build 7</th>
<th>Build 8</th>
<th>Build 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>wget.darwin-x64</td>
<td>failed compile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.darwin-arm64</td>
<td>failed compile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.debian</td>
<td>failed test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.openbsd-amd64-6.0</td>
<td>successful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.openbsd-amd64-6.1</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.openbsd-ia64-6.0</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.openbsd-ia64-6.1</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.solaris-x86</td>
<td>failed compile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wget.solaris-x86-64</td>
<td>failed compile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Building:**
  - Building
  - Building ETA in ~2 mins at 08:11
- **Extract sources:**
  - Extract sources
- **Fetched sources:**
  - Fetched sources of wget-1.19.1 stdio

---

*Note: The table above illustrates the build status of various platforms and the corresponding builds.*
Force Selected Builds

**build-texlive**

To force a build on certain ** Builders, select the builders, fill out the following fields and push the 'Force Build' button.

- texlive.darwin-i386.prg
- texlive.darwin-powerpc.prg
- texlive.darwin-x86_64.prg
- texlive.linux-armhf.prg
- texlive.linux-i386-debian7.prg
- texlive.linux-i386-debian9.prg
- texlive.linux-x86_64-debian7.prg
- texlive.linux-x86_64-debian9.prg
- texlive.openbsd-amd64-6.0.prg
- texlive.openbsd-amd64-6.1.prg
- texlive.openbsd-i386-6.0.prg
- texlive.openbsd-i386-6.1.prg
- texlive.solaris-i386.csw
- texlive.solaris-sparc.csw
- texlive.solaris-x86_64.csw

**reason** force build

**Branch:**

**Revision:**

**Repository:**

**Project:**

**Name:** Value:

**Name:** Value:

**Name:** Value:

**Name:** Value:

**Force Build**
Advantages & Opportunities (I)

- build binaries for all platforms:
  after each commit / daily / manually triggered

- automatically send emails when something breaks,
  less problems left for \TeX{} Live pretest period

- get the latest binaries to (adventurous) \TeX{} Live users

- no need for Karl to send “please rebuild now” emails
  and wait until people have time
Advantages & Opportunities (II)

- also build wget, xz, asymptote, xindy, ...
- compile different programs with different compilers (C++11, bug in ICU & upmendex, …)
- one could build/test development versions of dependencies (icu, poppler, libpng, luajit, …), detect problems & get fixes before release
- adding some of the 150 VMs from Utah to the list to extend the OS coverage
- testing for reproducibility of builds
More work to do

- complete setups for different software & components
- add encryption, binary signing, signature checking of sources
- write more test cases to detect problems / consistency
- set up proper private \TeX{} Live repositories with updated binaries
... After Karl fixed a problem in TL sources, while no internet connectivity was available at BT ...

From: Karl Berry <karl@...>
To: Mojca Miklavec <mojca.miklavec@...>

I imported all the binaries.

Oh, and I just found the OpenBSD binaries on the server. Do you mind if I include also those in TL?

Karl
Do we still need manpower?

Definitely. We need skilled people to:

- figure out which OS version / compiler / flags to use
- setup working machines and keep them “up to date”
- **debug** problems, report problems and **fix bugs**
- **IMPROVE THE BUILD SYSTEM!**

But there should be no need for **repetitive** task of re-running the same script each night, wait for the build to finish, check for consistency, sync, commit to repository.

It would help to have a backup person, familiar with the sources and the full process.
"I spend a lot of time on this task. I should write a program automating it!"

**Theory:**

- **Writing Code:**
  - Work on original task
  - Automation takes over

- **Free Time:**

**Reality:**

- **Writing Code:**
  - Debugging
  - Rethinking

- **Ongoing Development:**
  - No time for original task anymore

**Time:**
Special thanks

- **Alan Braslau**
  ○ help setting up, debugging & maintaining lots of VMs
- **Hans Hagen @ PRAGMA ADE**
  ○ dedicated server with lots of VMs
- **Dagobert Michelsen @ OpenCSW project**
  ○ Solaris build farm, help with buildbot setup
- **Norbert Preining**
  ○ additional functionality in TL
- **Taco Hoekwater**
  ○ TLContrib
- **Johnny @ Jožef Stefan Institute**
  ○ administration and bandwidth for the main server