LibreOffice Debugging Tips

How to find your way in a large codebase

Jan Holesovsky <kendy@suse.cz>

Basic techniques

- Asking around ;-)
  - You'll soon run out of people willing to fix things for you :-)
- Reading the code
  - That way you can fix only basic bugs, 99% of the more challenging ones resist that
- fprintf(stderr, “I'm here”);
  - Time consuming, but reliable

Using tools

- Gdb
  - It might take you some time to learn live with it; you'll save time when you start with TUI – text user interface, especially the 'single key mode' – single press does the function
    - 'gdb –tui' to start it, some distros do that by default
    - Arrows move in the code, ctrl+p / ctrl+n to navigate history
  - Areas of problems to debug
    - Crashes – via getting bactrace ('bt' / 'thread apply all bt')
    - Misbehaviors – by going through the program flow carefully [using 'n' (next), 's' (step)]
- Valgrind
  - Memory corruptions, when the backtraces are unusable

Using tools

- Strace
  - Problems like early startup crashes
- Git bisect
  - For searching for regressions
  - Problematic with LibreOffice (LO too large, and changing aggressively)

Advanced tips

- When the problem looks complex, write notes, read them, understand them
- From time to time, you may hit a compiler problem
  - Does not happen often, but can; compiler can be suspect when code compiled with -O2 crashes, but -O0 is fine – but even that can be a mistake in the program, not toolchain
  - In that case – find the offending file, and bisect it (cut into 2, compile one with the OK options, the other half with not-OK options, etc.)
- **And don't give up!**
  - The more you debug, the better you get – you'll quickly decide which tool to use, etc.


suse®