Deepsolver: development status and suggestions

Michael Pozhidaev
michael.pozhidaev@gmail.com

Proceedings of the 10th conference of FOSS-developers (Kaluga, Russia, September 21, 2013)

This report includes a survey of Deepsolver developing status, covers some noticeable difficulties faced during the work and offers several suggestions for further implementation. In contrast with the statements made by Deepsolver developers earlier, at present Deepsolver got minisat-based solver. Any 2-SAT approaches, considered for implementation previously, cause a lack of flexibility and that makes them not suitable anymore. Deepsolver support is now integrated to girar utilities and successfully deployed on ALT Linux package building hosts.

Rather difficult problem faced amid development process is a unreasonably big number of the packages affected during user task processing. In most cases it can easily lead to removing from the system some set of packages, which presence does not make the same general solution impossible. For example, consider the following situation written as a part of SAT-equation: $(!r \lor a \lor i)$. Here r is some dependent package, a is a package in a repository and i is a installed package. If algorithm once selected a, i can be either "true" or "false" without an influence to a solution. And generally there is no priority between these branches.

To avoid any unnecessary consequences, Deepsolver now tries to do a couple of optimizations described below, but generally they are still insufficient and research should be continued. First attempt tries to predict which packages will be installed or removed anyway and exclude corresponding clauses from the equation, as we already know that the result does not depend on them anymore. After an exclusion some previously affected packages are not involved in equation and this yields desired result.

The second optimization idea revolves around packages already installed in the system. Once they appear in equation, there is a chance they will not appear again and if corresponding clause does not imply intentional package removing, we can consider such clause evaluated as "true", since installed package remains installed. This technique is called in Deepsolver as "postponed clauses". After first appearance of the installed package, the clause containing it is constructed but actually is added to equation only after the second appearance.

These attempts partially solve the problem of unreasonably affected packages, but that is not enough yet. The quality of described attempts is deeply depended on the order of the processed packages. The further evolution of these ideas is described below and their implementation is in progress now. All clauses can be divided onto groups describing the actions needed for installation or removing of a particular package. We can define a set of references between these groups, reflecting what group causes potential installation or removing of a package. The references should be counted and, if some of them are removed, we can launch a garbage collecting strategy to remove all groups, left without any references to them. Removing of references is possible before and after minisat algorithm execution. The first case implies removing of clauses containing the installed packages, which do not appear anywhere in a equation in negative state. After the minisat launch we can remove the references associated with the variables which state is not changed during SAT-solving. In both cases after the garbage collecting all variables used only in removed groups can be safely ignored, even if SAT-solution yields change of their value.

We would like suggest several things to be considered by the community for further Deepsolver evolution. All of them imply creation of a new level of package management, while current Deepsolver functions implement second level with using the package formats libraries as first level (librpm, etc.). All proposed functions do not change anything in current architecture, so, all users used to work with any APT-like utilities will not notice any changes.

The first idea proposes adding new entities for control and processing by Deepsolver utilities. In particular, such as fonts, jar-archives etc. Internal implementation should be done in general form allowing easy and flexible enhancing with new types of entities vendor wants. Any additional information required for association between packages and entities should be collected on repository metadata construction, either automatically or with package maintainers support.

The main reason why package manager is the most suitable place for a idea like that, can be easily explained since the package manager is nearly the single place in a system with complete awareness what exact software and data are accessible through attached repositories. It can intersect enhanced information about proposed entities with list of really available packages and provide to user the list of available entities for a given type. For example, the list of fonts. Even in case of any synchronization problems, provided information will be always actual in sense of availability but, probably, incomplete. We would like to urge be cautious considering various types of files to be processed as a entity with package manager. For example, it is unclear whether kernel modules should be handled such manner or not. There are some cases that must be installed and removed with traditional administrative tools.

The second offer proposes designation of some packages like packages changing system behaviour. For example, packaged event file for acpid service changes buttons handling (in particular, power button). A user may want to view complete lists of such options with easy way to change their status (leading to package installation and removing). To be a proper option as it is suggested above, a package should be available for installation or removing without affecting other packages for consequent installation or removing. Only the package manager is able to construct list of options satisfying all requirements, hence, the best quality could be obtained only if this technique is a package manager essential part. Entities and options features of a package manager evidently are well-suited for integration to high-level GUI-instruments for system configuration, making GNU/Linux desktop more user-friendly.