Universal System for Creation and Installation Linux Packages

Chernov Evgeny
Institute for System Programming of RAS
Moscow, Russia
e-mail: ches@ispras.ru

***Abstract*—This paper discuss about the problem with distribution of software for Linux operation system. The problem is that Linux systems are different: different package formats, different names of packages for one program. It leads to creation of several packages for every Linux system for only one program you want to distribute. For resolving the problem mathematical model was developed. Based on the model database with information about dependences and packages of some Linux systems and web service for searching equivalent dependences are developed. Besides, some special tools that can help developers and users to work with different distribution and alien packages are developed.**

Keywords - Software packages; Software portability; Operating systems

I. INTRODUCTION

There are several ways for distribution programs from developer (where it's presented as source code) to users (who use it in compiled form). The first one is a distribution in source code form (Fig. 1).

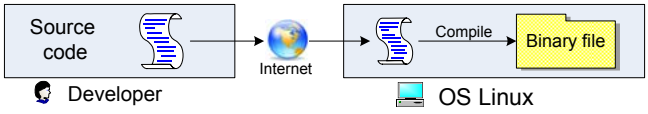
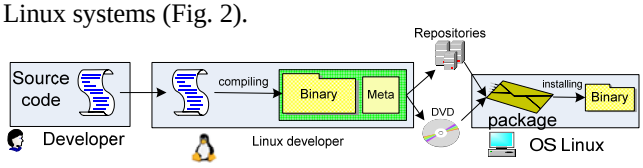


Figure 1. Distribution in source code form.

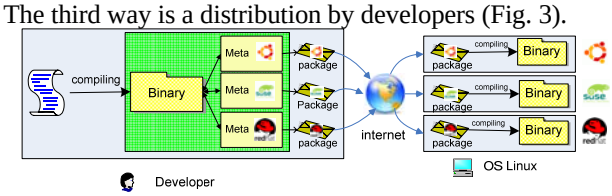
In this case user should compile and install program. But not all users are able to do it. So this way is more convenient for advanced users who know what compilation means and can fix some problems than can appear during compilation.

The second way is a distribution through repositories of Linux systems (Fig. 2).



In this case Linux developers compile the program by yourself in their system and build a package – a special format file that contains archived program and meta-data for it (name, version, architecture, description, checksum and dependency list). Such packages are placed into repositories that are different for every Linux system. Users can install these packages on their systems via special tools from corresponding Linux system. However not all program developers can use this way because Linux developers can't serve all programs.

The third way is a distribution by developers (Fig. 3).



In this case developer build program and create package by himself. However meta-data of package depends on distribution on which it's targeted to install. Names of dependencies are different on different Linux distributions. So developers have to create several packages for every Linux system. This makes it difficult to distribute programs by developers and to find proper packages for users.

So in common developers have to distribute programs by themselves. In this case they are called *Independent Software Vendor (ISV)*. The way is hard because of package incompatibility of Linux system. Different package format can be used in different Linux system and the name of packages of one program can differ. And it leads to the different dependency names.

This paper deals with the system which can determine corresponding names of dependency on different Linux systems automatically and about tools that using the system and help developers to create packages for different distributions and users to install alien packages (created for other distributions). The system is developed as web service. The tools can connect to the service and receive the names of dependencies for particular Linux system.

II. WEB SERVICE AND ALGORYTHM OF DEPENDENCEC SEARCHING

The web service for definition of equivalent dependences on different Linux distributions contains two interfaces:

1) For determination of the status of dependence on the particular Linux. It takes the name of dependency and returns one of the following values:
   - OK – the dependence exists on the distribution (there is a package that provides such entity).
   - NO – the dependence is not present on the distribution (no package provides it).
   - FILE – the dependence is presented in the distribution as a file.
   - MULTIPLE – the dependence is provided by several packages on the distribution.

2) For determination of names of dependences on some distribution. This interface takes list of dependences on any distribution and the information about the distribution (the name, the version, architecture) and returns the list of equivalent dependences on other distributions.

Based on this service the web interface has been developed. It can help user to know the names of some dependences on all supported distributions. Program interface of web service can be used by some systems like system for automatic package building (openSUSE Build Service) and by tools for converting package format – alien.

The web service works on the basis of a database. The main tables in this base are tables of distributions, packages, files and elements of dependences. Besides, there are some tables for links: the first defines, what files are provided by each package, the next two define what elements of dependences packages require and provide. Besides, there is a table for storage of results - equivalent sets of dependency names for different distributions. In case of repeated search of equivalent dependences results takes from this table and new search is not made.

The mathematical model lies in the basis of database structure and algorithm of search of equivalent sets. It operates with the following sets: set of distributions, packages, files and elements of dependences. Links between them (mapping from one set to another) and their properties are defined. Besides, the task of search of equivalent set is formally given.

Informally the task can be formulated by the following ways:

Having set of dependency names from one distribution P and information about the distribution D, determine the set of equivalent dependences on other distributions which satisfy to following conditions:

1) ***The Condition of existence of the decision****: if any distribution does not provide files, which P requires (i.e. the set of files of all packages of this distribution does not include set of files of packages, which package P requires), then the decision for this distribution doesn't exist (i.e. there is no equivalent set for this distribution). Otherwise it should satisfy to following conditions:*
2) ***The Condition of sufficiency of the decision****: the received set for any distribution should include all packages which initial package P on this distribution requires.*
3) ***The Condition of necessity of the decision****: the received set should not contain superfluous elements, i.e. at an exception of any element of it the sufficiency condition should be violated.*

The algorithm of the decision of this task is based on the following assumptions:

1) If any application depends on a certain package on the distribution then this application depends on the files provided by the given package on this distribution.
2) Names of files of the same program on different distributions are identical (thus a path can vary).

Thus, having the information on packages and files which the packages provide, it is possible to find what files the initial package requires on one distribution. Then to find these files on other distribution and to find what packages provide them. Thus we have the list of dependences for other distribution.

The described way of search allows finding the decision, satisfying to a sufficiency condition, but thus it can not satisfy to a necessity condition. Really, many "superfluous" elements can appear in search result. For example, in case of a file on some distribution is provided by several packages all of them will be presented in the result set, however presence only one of them is necessary. Therefore in the course of search it is necessary to delete such "duplicating" packages. Formally, it is possible to leave any of these packages since in any case necessity and sufficiency conditions will be satisfied. However for the best visual representation of the result set some heuristics are used for removing such duplicating packages: to remove the package that provides smaller number of files or not to remove package which name coincides with the name of one of dependences on the initial distribution.

There is one more reason of growth of number of dependences in the result set. The matter is that packages on which depends initial one provides set of files which aren't program and libraries. It can include documentation files, video, the pictures, etc. Such files on other distribution can be provided by set of packages (for example, a file "readme"), however in reality the initial package doesn't require it. So the packages found this way will be superfluous in the result set. To avoid such case it's necessary to specify the first assumption:

If any application depends on a certain package on the distribution then this application depends on <u>some</u> files provided by the given package on this distribution.

Determination of this set of "some" files occurs on the basis of classification of all files on the distribution. There are 7 classes of files: *programs*, *libraries*, modules, *links*, the empty files (*gags*), *not existing* in a package files (they can be created after package installation) and *others*. Each class has some more subclasses (for example, "the program on Perl" or "module Python"). Text files, archives, video are placed in "others" class and in a corresponding subclass ("a text file", "archive", "media").

For each class with a subclass some priority is attributed. For example, "programs" have the highest priority – 1, "text files" – the lowest – 7, "links to modules" – 4 priority. By means of these priorities, in each package it is possible to allocate the *main files* – files with the highest priority. Thus, search of packages on other distributions can be made on the basis of search not everything, but only main files of the package. For example, if the initial package requires on a package that provides 1 program, 5 pictures and 10 text files, then search only one program will be made on other distribution.

The result of work of the web service, in which the described algorithm is implemented, is presented in table 1. The first line contains names of distributions. Further there

are lists of equivalent dependences. The bold type in each line specifies dependence for which search was made.

OpenSUSE 11.1	Fedora 11	Debian 5	Mandriva 2009
kdebase4-runtime	kdebase-runtime	kdebase-runtime	kdebase4-runtime
	kdebase-runtime-libs	kdebase-bin-kde3	libkaudiodevicelist4
		kdebase-runtime-data	
libqt4-x11	qt-x11	**libqtgui4**	qtguilib
perl	**perl(Test::More)**	perl-modules	perl
xorg-x11-libXext	libXext	**libxext6**	libxext

Table 1. Equivalent dependences search result

As it's apparent from the table, for one dependence on one distribution can exist a little ones on others. Besides, names of dependences can essentially differ.

The described functionality of the web service can be used by some special tools that simplify creation and installing packages for different Linux distributions. There are 4 different tools: for creating and installing universal packages, for installing alien (created for other distribution) packages and for converting packages from one distribution to the package for other distribution.

III. Tools for Management of Universal Packages

The universal package – is a package of a standard format (rpm [1] or deb), storing in itself lists of dependences for all distribution kits. Thus, in the course of installation for any distribution it will be possible to check up, whether dependences of the given package on concrete system are resolved.

Author has developed a tool called '*UPackageBuild*'. The tool works on a basis of the web service and is used for creation of universal packages. This tool takes a package of a standard format (rpm or deb) and Linux distribution name on which it has been created. The list of dependences is taken from the package and is sent to the web service. The web service returns the lists of equivalent dependences for other distributions. The received information is placed in a file "/etc/upackage/package_name.requires" which is added to file-list of an initial package. Initial dependences at a package are cleaned, and to the version the prefix "Universal" is added.

The universal package received by this way is a usual rpm - or deb - package which does not have dependences, but have "/etc/upackage/package_name.requires" file that contains the description of dependences for each distributions. The given package has not own format. That makes it possible to install it by standard utilities (rpm – for rpm-packages and dpkg – for deb-packages). However in this case there is no check of dependences and that can lead to incorrect operation of the program or it can not work at all.

For correct installation of universal packages it is necessary to take advantage of the other special tool – '*Upackage*'. The given tool takes from the installing package the file with dependences, finds in it the list of dependences for the concrete distribution and checks, whether the given dependences on the concrete distribution are resolved or not. In success the package is installed, otherwise the list of packages which are necessary for install before installation of the given package is displayed.

In case of package installation by standard tools, it is checked, if 'Upackage' is installed on the system, then installation interrupts with the requirement to make installation by the given tool. If such tool is not installed, the message is displayed that the given package is intended for installation by Upackage tool, where it is possible to download it and the prevention that in case of package installation by standard tools, the user is responsible that all necessary dependences are installed on the system.

IV. Conclusion

The given paper describes the system for determination of equivalent dependences of packages from different Linux distribution and some special tools that using this system help developers to create packages for different distributions and help users to install alien packages on their system. The offered decision is based on mathematical model where the formal task is set. Then the database of dependences of distributions is developed according to this model. Using the database the web service provides interfaces for the analysis and search of dependences on different Linux distributions. On a basis of web service the author had developed tools for creation and installation of universal packages which can be installed on different Linux distributions.

V. References

[1] Eric Foster-Johnson. RPM Guide. — Indianapolis, Indiana: Wiley Publishing., Inc. 2003 — C. 3-6