



PACMAN Evaluation

Natalia Ratnikova
CMS Production Meeting
October 31, 2002

Introduction

- The purpose of this evaluation is to compare the functionality of PACMAN and other distribution tools used in CMS, to evaluate PACMAN flexibility, identify possible restrictions, specify a list of desired enhancements and give some recommendations for using PACMAN in CMS.
- The scope of evaluation is not limited to the CMS production environment. We also consider using PACMAN to support individual analyses, development environment and its interaction with the GRID tools.

PACMAN functionality

- The PACMAN documentation has a descriptive character, it mostly describes interfaces and instructions intended for users and cache managers. The fact that it does not contain a separate list of current functionalities makes the evaluation somewhat challenging. I considered the list of commands described in **pacman -help** output, online PACMAN documentation, as well as the details learned from tests, communications with the developers and from other of projects using PACMAN (see list of references at the end).

PACMAN functionality (cont)

Pacman is a package manager. With Pacman, you can transparently fetch, install and manage software packages. Typically, these are tarballs or rpm files. The advantage of using pacman is that all the annoying details of doing this, such as:

- Where to get the software?

- Which version of the software is right for your system?

- Whether there are dependent packages that you have to install first?

- Whether you have to be root or not?

- What are the exact instructions for installing the software?

PACMAN functionality (cont)

How to setup environment variables and paths for the packages once they are installed?

How to conveniently setup the same environment on multiple machines?

When a new version of the package is available and when you should upgrade?

are taken care of for you.

Ideally, if something in this procedure doesn't work, you get to complain to the person who setup the "software cache" rather than fixing it yourself.

That way, if a problem comes up, it can get solved once rather than forcing everyone to fix it individually.

PACMAN functionality (cont)

Use: `pacman {Options} <package1> <package2> ...`

Options:

- help To get this message.
- version Print version number.
- info Launch a browser showing the local installation.
- doc Launch a browser pointing to the local Pacman documentation (use -mozilla, -netscape or -lynx to force a browser choice).
- infomirror Sets up and maintains a mirror of the Installation web page. documentation at directory dir.
- get Fetch and install packages from the caches.
- fetch Fetch packages from the caches.
- install Install the named packages.
- remove Remove the named packages.
- update Update installed or fetched package(s).
- updateall Update all packages which have an update.
- removeall Remove all packages in the installation.

Oct. 31,
2002

PACMAN functionality (cont)

- cache:cachename Use cache "cachename" to search for packages.
- rel:dir Installs relative to an installation in directory dir.

- recursive (-r) Recursively removes or uninstalls packages.
- no-rpmcheck To skip rpm verification.
- force-rpm To automatically replace rpms without asking.
- no-native To never accept non-Pacman local installations.
- savedownload To save downloaded file even after installation.

PACMAN functionality (cont)

- http_proxy:URL To set and remember an http_proxy.
- system-setenv:EV To set system wide environment variable "EV" pointing to the current installation.
- system-unsetenv:EV To unset a system wide environment variable "EV".

- ask Ask before executing build commands.
- v Verbose messages.
- registry Display the available symbolic cache names.
- trust-registered-caches Automatically trust all registered caches (use at your own risk).
- trust-all-caches Automatically trust *all* caches (use at your own risk).
- convert To convert an old Pacman 1 installation.

PACMAN vs SCRAM

Where to get the software?

~~SCRAM: [scram download pages](#)~~

Which version of the software is right for your system?

SCRAM: `scram -arch <system >list <project>`

Whether there are dependent packages that you have to install first?

SCRAM: `scram tool list`

Whether you have to be root or not?

SCRAM: you do not have to be root

What are the exact instructions for installing the software?

SCRAM: may depend on project, basically it is
`scram b`
`scram install`

PACMAN vs SCRAM

How to setup environment variables and paths for the packages once they are installed?

SCRAM: `eval `scram runtime -csh``

How to conveniently setup the same environment on multiple machines?

???

When a new version of the package is available and when you should upgrade?

SCRAM: `watch_projects` notification system, `scram list`

Ideally, if something in this procedure doesn't work, you get to complain to the person who setup the "software cache" rather than fixing it yourself.

SCRAM: CMS Bug Reporting System (BugsRS)

PACMAN vs DAR

see presentation on previous Production meeting for more details

A supplemental feature compare to DAR is that PACMAN automatically fetches the distribution tar file from the specified cache.

However this does not work with any secure methods of downloading software, which makes PACMAN inapplicable in case of restricted access to distributions.

Currently CMS uses https server at CERN, and AFS authorization at FNAL to control access to ORCA

distributions.

Oct 11
2002

PACMAN vs RPM

(from PACMAN documentation):

Why not do everything with rpms?

Although rpm is multi-platform these days, most software is still configured as tarballs.

Although Pacman is much simpler than rpm, it handles dependent packages correctly and provides setup scripts to use the installed software.

Evaluating flexibility and interoperability of PACMAN

Pacman is designed to be flexible. It allows to install practically any package that could be installed with a shell script. Some limitations will be discussed below. A product distribution includes a pacman file `<productname>.pacman` of a special pacman format and contains product attributes and installation instructions and an optional distribution kit. Currently pacman is compatible with the tar or rpm format. It has shown to work fairly well both with binary and source distributions (e.g. BOSS).

Evaluating flexibility and interoperability of PACMAN

Pacman implies that the distribution is contained in a single file, that contradicts general distribution practices of many multi-platform products, that prefer a separate packaging of sources and binaries. This limitation however can be addressed by using pacman dependencies mechanisms or simply by packing multiple files in one tar file.

For products distributed via rpms pacman files can be produced automatically using `rpm2pacman.pl` script.

Evaluating flexibility and interoperability of PACMAN

Pacman allows easily install/uninstall products. However it does not guarantee any security or even that product will be really installed/uninstalled: it only executes scripts specified in a pacman file. Either Pacman does not guarantee that installation of new packages does not overwrite the existing installations. Pacman does not have any protection against using tarfiles or products with duplicated names. Pacman approach implies serious responsibility of the cache managers, who maintain pacman files for products in their cache, and trust from the end-user's side.

Oct. 31,
2002

Evaluating flexibility and interoperability of PACMAN

Pacman requires that the distribution kits (tarfiles or rpms) are available in the web area. There is a feeling that this solution may not scale very well.

Evaluating flexibility and interoperability of PACMAN

Pacman installations are location-oriented. Whenever pacman is called from a new directory, it creates a new installation with independent database, that traces only products installed in this particular directory or relative to it.

This creates some confusion for the collections of dependent software products. By default pacman installs all products in one directory, which is often unpractical in reality. In particular, if software installations are large, or we need several version of products simultaneously, we should be able to locate them in place where we have enough disk space.

Oct. 31,
2002

Evaluating flexibility and interoperability of PACMAN

Two approaches attempt to address this problem: using environment variables to specify alternative locations and using relative installations. However both have limitations and result in partial loss of desired functionality. This question needs additional elaboration.

Pacman does not support any global database. Storage and bookkeeping of distributions is currently out of scope of pacman.

Evaluating flexibility and interoperability of PACMAN

Pacman is distributed and supposed to work in multi-platform environment. However there are certain complications in using pacman on distributed file systems or clusters of computers. Whenever a product is installed, pacman notes in the local pacman database a node on which the installation was performed. Although this installation may be available to the whole cluster, pacman information provider would still list products as installed on one single node. If this issue goes beyond pacman's functionality, we need to look for some external solution, as pacman's node-based approach makes the bookkeeping of installed products on the computer clusters quite problematic.

Enhancement wish list

- To specify the arbitrary locations where the product will be installed
- To get list of already installed products, and locations where they have been installed.
- To be able to pass this information (e.g. location) to another (dependent) project during the installation.
- Security issues
- we do not want to overwrite any existing installation (same product, same version, same location), because it may be in use.
- We want to be able to install another version of product in the same location, and to be sure that they do not overlap.

Summary

Pacman is flexible and convenient in use. It is important to realize however that Pacman's main mission is to provide convenient and uniform interface. Pacman itself does not solve many problems related to distribution, such as packaging, verification, security, monitoring, configuration, storage, access etc. All these issues still have to be addressed before (or after) pacman comes into play. In other words, pacman would not create an order from chaos.

Summary (cont)

Pacman is successfully used to distribute and Install VDT and Atlas software. These caches are Maintained with the direct assistance from the Pacman developers, and are quite reliable.

In general Pacman works well in cases when product developers themselves support pacman caches, or when caches are created using automated packaging tools. Such caches certainly can be trusted.