SUMAPACK: HERRAMIENTAS PARA LA COMPILACION, EMPAQUETADO E INSTALACION DE SOFTWARE LIBRE EN DIFERENTES ARQUITECTURAS, SISTEMAS OPERATIVOS Y DISTRIBUCIONES GNU/LINUX.

Diego Saravia
Facultad de Ciencias Exactas – Universidad Nacional de Salta (U.N.Sa.)
INENCO (U.N.Sa – CONICET)
Fax: 54-387-4255489, Av. Bolivia 5150. 4400 Salta, Argentina
mail: dsa@unsa.edu.ar, Diego.Saravia@gmail.com

RESUMEN: Se presenta un sistema de compilación, empaquetamiento e instalación de software apto para ser usado en una amplia variedad de distribuciones de GNU/Linux y potencialmente otros sistemas como Windows. Esto es particularmente útil en software que sirve a pequeñas comunidades en cuyo caso no tiene sentido pretender la distribución y empaquetamiento de software con pocos y muy específicos usuarios a través de los canales genéricos de cada sistema o distribución.

Palabras clave: Sumapack, empaquetado, distribuciones GNU/Linux.

INTRODUCCION

Sumapack (Saravia, D., 2010b) está siendo desarrollado para compilar, empaquetar e instalar paquetes de software en sistemas diferentes (cross-packaging) bajo un mismo esquema de comandos y funciones. Constituye un conjunto más o menos integrado de programas. Los objetivos de Sumapack y los programas vinculados a los mismos son:

a) Apoyar el análisis simbólico y formal del problema de las dependencias de paquetes binarios y fuente, y aportar a los estudios de calidad en sistemas GNU/Linux.

b) Contribuir a una mejor comprensión de las acciones y complejidades de cada sistema de administración de paquetes y las virtudes, potenciales y defectos de unos con respecto a otros.

c) Abstraer al usuario, y al desarrollador del problema de las diferentes distribuciones y sistemas de instalación y de sus diferentes comandos, esquemas y nombres de paquetes. En tal sentido los programas: **Rosetta y Babel**: constituyen un meta/summa/super instalador de paquetes que cumple estos objetivos. La Rosetta de aplicaciones y paquetería en sistemas GNU/Linux/BSD/Windows, base para construir la Babel del Software Libre (Free Software Foundation, 2010a).

d) Conseguir un mecanismo de instalación simple y robusto que pueda funcionar en todos los sistemas y que provea la funcionalidad básica tanto de instalar paquetes como apoyar el empaquetamiento y backup de datos de los usuarios. **Uinstall** cumple esos objetivos.

e) Facilitar la compilación, empaquetamiento, instalación, uso y distribución de software desde sus paquetes fuente originales, y diversos parches aportados desde distintos orígenes, sin interferir negativamente con la administración de cada sistema. **Autosuma**: es una extensión a las Autotools (Wikipedia, 2010b): autoconf, automake, libtoolize; para: generar paquetería binaria compatible con los manejadores de paquetes instalados en diferentes máquinas; instalar a través de los manejadores de paquetes cuando existan; instalar automáticamente dependencias y requisitos previos para la compilación.

f) Proveer sistemas de generación de documetnación que permitan con simplicidad y con editores de texto o vía web, generar todos los formatos pertinentes en forma automática. **Autodocs** es otra extensión a las Autotools para facilitar el uso de Muse (Olson, M., 2009) y Tex - LaTeX (Knuth, D., 1978) en la documentación de los paquetes.

g) Poder generar paquetes y distribuciones completas parametrizadas en forma automática. **Ubuild**: maneja las Autotools y otros programas para la compilación manual o en entornos diversos, como “cross-compiling”, generadores de repositorios, etc..

h) Facilitar la generación de interfaces de usuarios o de otros programas, que funcionen en muchos sistemas. Tanto Rosetta como Babel pueden constituir una capa intermedia para que cualquier interfaz de usuario pueda convocar a cualquier

instalador -según el sistema donde se ejecute- utilizando los mismos comandos, a través de estos programas. En particular estamos construyendo **XulPack,** que es una interfaz gráfica (GUI) para Sumapack (en diseño), basada en XUL (Mozilla Foundation, 2010), que puede servir de instrumentación de referencia de un instalador universal.

i) Summapack también provee programas auxiliares que ayudan a cumplir los objetivos ya citados a los otros programas: **Upack y Upackname** son programas auxiliares para empaquetar, desempaquetar e interpretar nombres de paquetes; **Bootstrap** es un programa para la auto-instalación de Sumapack.

Sumapack se escribe indistintamente con una o dos emes. El nombre de algunos programas inicia con la letra "u" por su posible integración con la aplicación Uget (Wikipedia, 2010a), usada en el Ututo (Ututo, El proyecto, 2000).

ANTECEDENTES

Cuando se desarrolla software para pequeñas comunidades no siempre es razonable pretender que los empaquetadores de cada una de las distribuciones y sistemas incluyan el software en sus repositorios, ni lo actualicen frecuentemente, ya que será usado por una porción realmente minoritaria de personas. De esta forma surge la necesidad de construir herramientas que faciliten el trabajo del desarrollador de sistemas y permitan extenderlo a la construcción de paquetes y los medios de instalación.

El paso de empaquetar el software para una gran variedad de distribuciones y sistemas no es simple y requiere estudiar a conciencia los diferentes esquemas en uso en cada uno de ellos. Los mecanismos de empaquetamiento y distribución del Software Libre, que son una de sus grandes fortalezas, como rpm, dpkg, etc. se han echo cada vez más incompatibles entre sí y es interesante plantear un camino de convergencia para resolver esta situación.

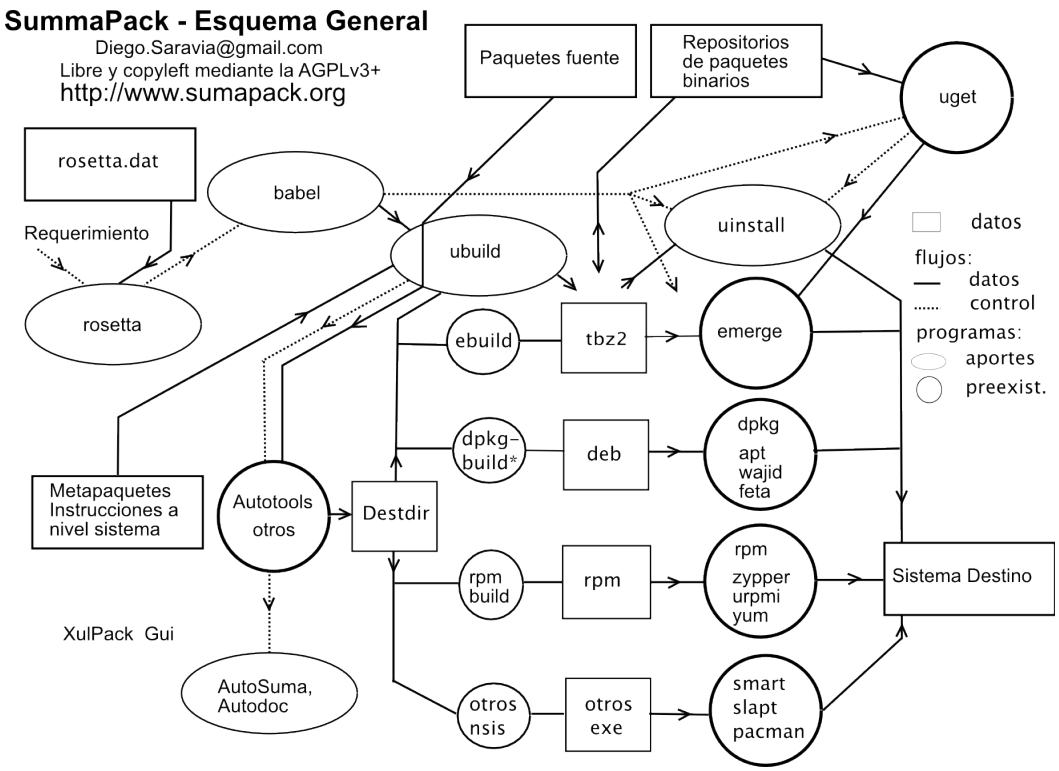

Figura 1: esquema general de Sumapack.

ANALISIS DEL PROBLEMA

El proceso de distribución del Software Libre.

Una de las contribuciones centrales del movimiento de SL al desarrollo de software en general, junto con la idea misma de la libertad del conocimiento, y del desarrollo colaborativo, es el conjunto de los sistemas de manejos de paquetes (Murdock, I., 2007).

Y uno de los principales problemas que se ha creado a si mismo, además de la distribución exclusiva de binarios, es la incompatibilidad de estándares entre los diferentes sistemas, violando una de las ideas centrales del movimiento de Software Libre, cual es que el software debe poder usarse en cualquier sistema (Free Software Foundation, 2009).

Ontología

Gran parte del Software Libre se adquiere e instala en cada computadora desde "Distribuciones" que combinan: paquetes binarios en ciertos formatos, distribuidos desde determinados conjuntos de repositorios, mediante herramientas específicas que mantienen

bases de datos, de diferentes formatos, con la información de los paquetes localmente instalados en cada computadora. Las distribuciones suelen tener, además, herramientas propias y diferenciadas de instalación, configuración, y administración del sistema. Los conjuntos de repositorios son accesibles vía Internet u otras vías: como cdroms, dvds, etc..

Los repositiorios dividen y unifican los proyectos originales en diferentes paquetes asignándoles nombres diferentes entre sí, diferentes a los nombres originales y diferentes de distro a distro (distro es la abreviación habitual de distribución). Algunos clasificados en desarrollo, documentos, binarios, librerías, etc, creando tantos binarios como arquitecturas y esquemas de configuración se desee.

Los repositorios también suelen proveer un paquete fuente para cada paquete binario provisto, cumpliendo los requisitos de la distribución del Software Libre.

Existen repositorios independientes que proveen paquetes extra a una o más distribuciones.

Algunos repositorios son versionados, en el sentido de que un subconjunto de paquetes son sindicados como integrantes de una versión que cambia periódicamente. Así se transforma un problema de tantas dimensiones como paquetes hay en un sistema, en un problema en cada momento unidimensional, simplificando las herramientas relacionadas. También se versionan variantes, más nuevas o más viejas, para permitir sistemas "estables" donde no existan novedades más allá de las estrictamente necesarias para asegurarlos.

Existen otros mecanismos, como Gentoo (Gentoo Foundation, 2010), o los descriptos en "Linux from Scratch" (Beekmans, Gerard 2010), basados generalmente en: wget, tar, configure, make y make install; para usar código fuente e instalarlo previa compilación. Estos obtienen el software directamente desde los proyectos originales sea mediante paquetes fuente o sistemas de control de cambios como svn, git, etc..

Estos otros métodos aportan instrucciones adicionales a los paquetes originales sea en forma de libro como “Linux From Scratch” o en forma de scripts: los ebuilds de Gentoo. Estos esquemas más que como distros, actúan como creadores de distros o como formas particulares para crear repositorios.

Es posible, si bien menos habitual, mezclar para un sistema o computadora todos los mecanismos preexistentes.

Es el interés comercial, o de liderazgo, de diferentes empresas y grupos el que ha llevado a desarrollar aplicaciones específicas y repositorios para cada una de estas distribuciones: Debian, OpenSuse, Fedora, Ubuntu, Mandriva, Ututo, etc..

El proceso de distribución y sus elementos:

- Desarrollo de software y almacenamiento compartido y dinámico mediante sistemas de *Control de Cambios*: svn, cvs, git, etc..
- Publicación periódica de las fuentes por los equipos de desarrollo: *Paquetes originales.*
- Creación de información adicional aportada por los empaquetadores: .ebuild, debian/control, .dsc, .spec, etc. que no se incorpora en cada paquete fuente y que presenta diferencias no siempre necesarias entre diferentes distros: *Información de empaquetado*.
- Publicación de repositorios, produciendo, compilando y empaquetando en forma coherente los paquetes fuente entre sí, con la información adicional y las decisiones globales de cada distro: *Repositorios* y *paquetes binarios*.
- Instalación de los paquetes en cada máquina, desde los repositorios, por internet o medios físicos, creando una base de datos local y usando herramientas específicas: *bases de datos locales*, *herramientas de gestión*, *software instalado en cada máquina*.

Crítica

Esta situación puede haberse construido en su momento para garantizar la venta períodica de CDROMS, medios, y subscripciones. La clave de la fragmentación del espacio GNU/Linux-ero se encuentra en el desarrollo de formatos, repositorios (para colmo versionados) y herramientas incompatibles entre sí con el fin de asegurar el control del usuario y su fidelización con la "distro", sea por intereses comerciales o poder atencional de diferentes "comunidades", empresas comerciales, estados, organizaciones o líderes (Saravia, D. 2005).

Así el “poder” de la meritocracia (Montoro, S. 2010), se desplaza desde los desarrolladores de los miles de paquetes de Software Libre, donde reside la creatividad que impulsa el emprendimiento, a los empaquetadores de las distros, que cobran protagonismo al auto-desarrollar herramientas incompatibles que refuerzan su necesidad de existencia, como cualquier burocracia conocida. Lo que paso con el proyecto GNU y Linux, pasa con miles de desarrolladores y unas pocas "distros". El poder atencional, que es la fuente del reconocimiento que obtienen los que aportan al Software Libre, se transfiere desde los

desarrolladores a los empaquetadores y a las Distros.

Nada impediría tener herramientas ortogonales en cuanto a los repositorios, bases de datos locales y formatos de empaquetamiento; de hecho algunos proyectos han avanzado en esta dirección.

Nada impediría automatizar y sistematizar el proceso de empaquetamiento de forma que pueda hacerlo en gran parte, sin esfuerzo ni gran atención, el desarrollador de software, quedando para los distribuidores aspectos de interoperatibidad con otros paquetes y sincronización de cada "release" de distro, que es lo que realmente aporta una distro: la capacidad de distribuir y la de hacer interoperar los paquetes.

La paquetería binaria presenta un conjunto de consideraciones a tener en cuenta: arquitectura, procesador, librerías, opciones de compilación, combinación de paquetes, etc.. Así que ciertamente hay necesidad de que en cada computadora todos los binarios compartan las librerías dinámicas y metodologías de configuración. Esto dificulta, o imposibilita en principio, usar diferentes conjuntos de repositorios binarios en un mismo sistema. Pero a esta cuestión central y objetiva no es necesario aplicarle confusión adicional en cuanto a formatos, programas y distros. En principio debiera ser posible para una persona que instalo Debian en un sistema, hacer un “dist-upgrade” (cambio de versión, o en este caso distribución) a un OpenSuse.

También es cierto que la consigna de la FSF de distribuir software fuente, y no binario y de no preparar herramientas que cubran esta última milla, para favorecer la distribución de fuentes, complica el panorama. Los vacíos se ocupan y no siempre de la mejor manera.

Existe una necesidad concreta de generar repositorios de binarios de terceros para facilitar el uso del Software Libre en cada computadora. No se puede pedir a cada proyecto de distribución de Software Libre que mantenga servidores propios centralizados desde los cuales se descarguen todos los binarios del mundo. Entre otras cosas, esto da demasiado poder a los pocos que controlan dichos servidores. Tampoco es razonable que todos compilen todo su software.

Es razonable que existan repositorios de binarios y que estos sean conocidos como "distros". Repositorios cuyos paquetes reflejen una gran diversidad en cuanto a opciones de compilación y uso de librerías y que cada uno de ellos tenga una selección de las mismas coherente entre sí. Y que cualquier sistema que tenga librerías compatibles con estos repositorios pueda bajar software de allí. En todo caso el versionado de los repositorios podría identificar globalmente las opciones claves de cada uno para que cada sistema

pueda saber si es compatible con él.

Lo que no es razonable es la integración vertical no interoperable entre los repositorios, los formatos de empaquetamiento, las herramientas y las bases locales. Esto va en contra de los criterios de diseño de un buen software que debiera operar con todos los formatos de datos conocidos relacionados con su función.

Propuesta

Los problemas mencionados son los que queremos ayudar a visualizar y superar con Sumapack.

Desde ya se incorporan los mecanismos preexistentes y se propone integrar sus contribuciones para conformar un método más racional de producir y distribuir software, paquetes y repositorios sin la fragmentación adicional de las distros tal como hoy se las conoce.

Entonces el objetivo central de este software es apoyar un proceso que vaya tendiendo a construir herramientas interoperables en la distribución del Software Libre

Cumplir el objetivo implica también limitar el poder de las distribuciones y sistemas, con sus empaquetadores, mantenedores y otros intermediarios entre usuarios y desarrolladores, aumentando la capacidad operativa de los desarrolladores y usuarios (prosumidores) del software. Eventualmente el uso y generalización de estas ideas llevaría a hacer totalmente ortogonal el uso de un software cualquiera con cualquier sistema de paquetería, y eliminaría en gran medida la necesidad de la existencia de empaquetadores específicos de cada distribución.

Sumapack es más un concepto que software. Podría verse reducido a una mínima expresión si sus ideas permeasen los mecanismos en uso hoy en día en las diferentes distribuciones y sistemas. Para ésto, cada proyecto de software podría distribuir algunos archivos con meta información básica, y todas los repositorios podrían poner los nombres originales a cada paquete, entre otras cosas.

Sumapack es un paso en la solución del problema planteado con el manifiesto Anti-distros (Saravia, D. 2005). Todavía hay mucho que avanzar y que pensar con relación a estas cuestiones. Representa un primer esquema de software para analizar estos problemas y

específicamente facilita la instalación del software desarrollado para pequeñas comunidades.

METODOLOGIAS Y HERRAMIENTAS

Se ha utilizado Bash para todos aquellas partes de Sumapack que deben funcionar en sistemas mínimos, por ejemplo cuando un sistema se comienza a instalar. Perl para las otras partes del mismo. También se utiliza entre otros make, las autotools de GNU, Muse, y LaTeX. Cygwin (Red Hat, 2008), Nsis (Nullsoft, 2009), para la instalación en Windows. Las interfaces gráficas se realizan en XUL (Mozilla Foundation, 2010) para Firefox, mediante el módulo de Perl XUL::Gui (Strom, 2009).

Se trabaja con un repositorio basado en “subversion” y se comparten resultados con comunidades de desarrollo de intereses similares.

CARACTERISTICAS DE CADA COMPONENTE

Rosetta y Babel

Ambos comandos trabajan juntos. Rosetta, para cada sistema, habitualmente instalado mediante una distribución, identifica el nombre solicitado, o función a instalar, con un proyecto en particular. El nombre puede estar asociado en rosetta.dat con varios proyectos de software que brindan esa función. Ante cada función puede instalar uno o más proyectos alternativos o complementarios, permitiendo definir en cada caso el comportamiento y proveyendo uno por defecto. Cada proyecto para cada distribución ofrece uno o más paquetes para su instalación. Hay tres niveles involucrados: función, proyecto y paquete. Para cada función y cada “distro” Rosetta propone diferentes paquetes para su instalación. Babel toma estos paquetes y los instala.

Babel aporta comandos tipo "wrappers" que traducen las diferencias entre las herramientas. Rosetta aporta una infraestructura para unificar y traducir los nombres de los paquetes en distintos repos.
Funcionan en muchos sistemas en forma similar. Presentan una interfaz común para algunas de las tareas de apt-get; zypper; emerge; ututo-get; urpmi; yum; installpkg, etc..

En cada sistema usa el comando apropiado para el administrador de paquetes mayoritario

del mismo. Si no encuentra un sistema o no encuentra un paquete apropiado, usa el sistema genérico desde las fuentes. Eventualmente provee de scripts específicos para instalar a nivel de cada sistema paquetes complejos. Si instala desde las fuentes, primero empaqueta, si puede.

Se usa un esquema común de nombres para los proyectos de Software Libre, independientemente de cada repositorio. Se provee un archivo "rosetta.dat" que acumula información sobre nombres de paquetes en diferentes repositorios. El esquema de función, proyecto y paquete provee y generaliza conceptos como paquetes virtuales, slots, etc..

Se puede poner un archivo rosetta.dat local, que sesuperpone sobre el genérico, para definir políticas particulares.

Babel clasifica los objetos según su tipo y los agrupa segun el packlet que deberá tramitarlos. Se puede pensar en un micro-lenguaje de administración de paquetes formado por frases donde la primera palabra es una acción y le siguen varios objetos. Segun la acción y el tipo de objeto se ejecuta el packlet específico para cada herramienta con el comando interno de packlet que corresponda.

Elementos involucrados:

- Acciones de Babel (su primer argumento): install, remove, clean, fix, search, info
- Diferentes tipos de argumentos del comando Babel: PACKAGES en PACKAGES-LIST (listas de argumentos); PACKAGES-SETS: WORLD, SYSTEM, ALL; REPO: identifican con precisión un repositorio; CACHE: para indicar que se opera con la base local de paquetes de cada sistema; NAMES: para buscar PACKAGES. Los PACKAGES pueden ser del tipo: EBUILD archivo de paquetería de gentoo; SUMMA archivo de paquetería Sumapack; PKG paquete cygnus; RPM paquete rpm; DEB paquete debian; TXZ nuevo paquete slackware; TGZ viejo paquete slackware; TBZ paquete ututo; SOURCE paquete fuente; SVN repositorio subversion; CPAN repositorio perl; SH archivo con instrucciones para instalar un software; LSH comando a ejecutar; COMMAND ejecutar el packlet particular que sigue; PAQ paquete estándar para el sistema en cuestion; REPO repositorio; NAME cadena con el que se buscará el nombre de un paquete,
- Packlets, (wrappers o backends): *PAQ-emerge.sh PAQ-pacman.sh PAQ-slapt.sh PAQ-urpmi.sh PAQ-yum.sh PAQ-apt.sh PAQ-feta.sh PAQ-smart.sh PAQ-uget.sh*

source.sh comain.sh, etc.

Interacción entre Acciones, Objetos y Packlets

La Tabla 1 muestra para cada acción de Babel y el tipo de objeto, el comando interno del packlet que se ejecuta. Se ejecutan distintos packlets según la distribución mayoritaria usada en el sistema en cuestión, y el tipo de objeto, Todos los packlets tienen similares comandos internos para hacer diferentes operaciones concretas: install, remove, clean, fix, search, info, addrepo, delrepo, update, deplclean, distupdate, refresh. Algunos de ellos admiten opciones y pueden operar sobre distintos argumentos.

.	NAMES	PACKAGES	PACKAGES	REPO	WORLD	SYSTEM	ALL	CACHE
install	.	.	install	addrepo	update	distupdate	.	refresh
remove	.	.	remove	delrepo	.	.	.	.
clean	.	.	clean	.	depclean	.	.	clean
fix	.	.	fix	.	fix	.	.	.
search	search	.	.	.	.	.	.	.
info	.	info	.	.	stats	stats	stats	stats

Tabla 1: Interacción entre acciones, objetos y packlets.

Para actualizar el sistema pero no los paquetes relativos a la versión, se indica "install WORLD"; para cambiar todo hasta la versión es "install SYSTEM"; al ejecutar "install ALL" debiera instalar todo el repositorio en la máquina, no se instrumentó todavía. Para actualizar o instalar desde cero el cache es "install CACHE". Install REPO es traducido como addrepo REPO. El comando de packlet update, en Debian suele conocerse como upgrade, el refresh es conocido en Debian como update. No siempre se logra que todas las herramientas hagan todas las funciones o exactamente lo mismo.

AutoSumma

Permite la instalación de paquetes desde las fuentes desde sistemas con autoconf/automake, en forma congruente con el sistema de paquetería de cada sistema. Así facilita la tarea del usuario y/o desarrollador. Genera en cualquier sistema paquetes deb, txz, rpm, ebuild, etc..

Los paquetes fuente, originales, son fáciles de usar con Sumapack, y la inexistencia de Sumapack en una computadora que instala un paquete preparado para Sumapack no bloquea las funciones esenciales.

Se proveen macros m4 genéricos y específicos para cada paquete. Con los mismos se puede instruir al configure.ac de un paquete fuente, para que averigüe si los paquetes requeridos para su compilación están instalados. Esta información es volcada a un archivo "lacks". De allí puede ser tomada por un script que los instala. Este script puede ser ejectuado desde un objetivo "prepare" del Makefile.am

En particular los archivos que portan información específica del paquete en cada tipo: .spec (de los paquetes rpm), .lsm (de la base de paquetes de GNU/Linux), control (de Debian), .ebuild (de Gentoo) se generan desde un archivo común.

Autogenera las listas de archivos y las dependencias con la información del proyecto fuente, hasta donde es posible. Se debe trabajar con un rosetta.dat específico para esto. Así se facilita la generación de binarios en cada sistema.

Puede funcionar con sistemas de cross-tools (compilación en una computadora, de binarios para otra, con diferente arquitectura) y con jaulas de compilación específicas.

Es útil para pruebas y desarrollo. Eventualmente para hacer las mismas distros con diferentes sistemas de paquetería, funcionando ortogonalmente. Así ayuda a los fabricantes de distros y repositorios. Que estos paquetes sean usables en otros sistemas ya no depende de estas herramientas, sino de las versiones de librerías y dependencias reales.

Genera los paquetes desde cada fuente; la idea es que el papel de un sistema externo que controle los make de cada proyecto fuente sea mínimo. En todo caso se generan los archivos rules, y partes del spec y del ebuild.

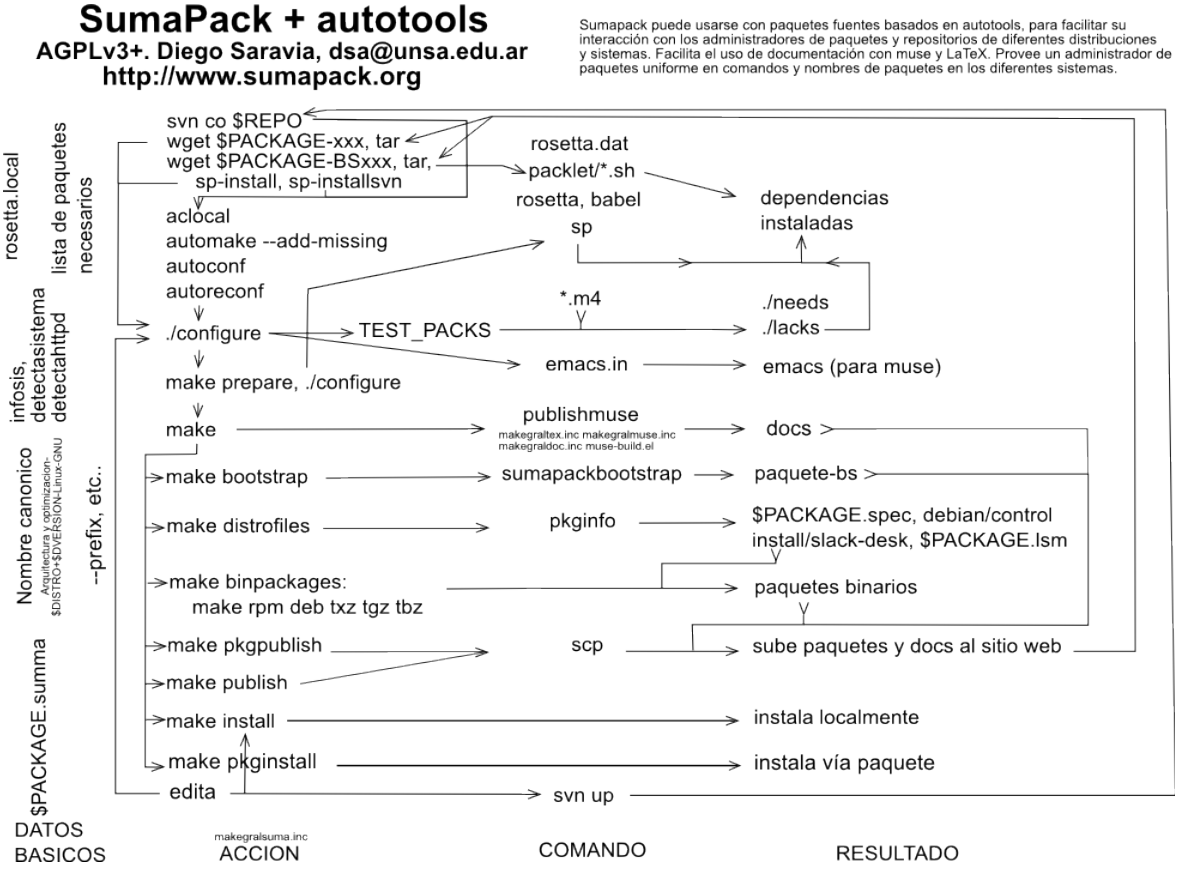

Figura 2: Esquema de autosuma, o el apoyo de Sumapack a las autotools.

Autodoc

Provee software para facilitar el uso de Muse y LaTeX en la realización y visualización de la documentación de los proyectos. Basado en el trabajo original realizado para Psicro (Saravia, D. 2010a).

Ubuild

Baja, desempaqueta (mediante upack), compila fuentes, y empaqueta binarios (mediante upack). Es una reescritura de ebuild.sh de Gentoo para que funcione en toda distro. Procesa paquetes fuente, como tar.gz, y tar.bz2, y metapaquetes como .ebuilds. Ubuild toma lo bueno del Gentoo incorporando la filosofia unix de herramientas simples interoperables, siendo ortogonal a la distro y pudiendo interoperar con otros empaquetadores como dpkg o rpm. Junto con Uinstall pueden reemplazar completamente a emerge además de ser instalable en cualquier distro.

Ubuild saca la lógica de control de pasos del shell, por lo tanto no se controla con emerge o

ebuild.py sino con un Makefile autogenerado denominado PACKAGE.rules. Este rules cumple las normas Debian y podría ser usado por dpkg. En vez de debhelper se podrían producir paquetes Debian desde ebuilds. Todo esto ayuda a desarrollar software en forma independiente de la distro, facilitando el meta manejo del esquema de Makefiles de las Autotools. Puede servir para generar paquetes o repositorios de una distro entera. Usa solo bash, aunque quedan todavía algunos programas en Python y en Perl que deberán ser eliminados.

Comandos: **rules**: configura operaciones; **fetch**: baja archivos fuente y parches, **unpack**: desempaqueta fuentes; **prepare**: prepara fuentes; **configure**: configura las fuentes; **compile**: compila las fuentes, **install**: instala el paquete en el directorio temporario "install"; **package**: crea un paquete tipo tarball; **installpkg:** lo instala**; clean**: limpia todas las fuentes y archivos temporarios.

Upack

Empaqueta y desempaqueta varios formatos de archivos, en particular .tbz2, o binarios de Gentoo/Ututo, recuperando la meta -información contenida.

Upackname

Dado el nombre de un paquete, sea de Gentoo, Debian, una distro basada en rpm u otros, lo separa en sus componentes como nombre, versión, "release", extensión, etc.. Dados dos nombres del mismo paquete compara cual es más nuevo.

Uinstall

Es un conjunto de scripts para administrar conjuntos de archivos, denominados paquetes, en computadoras.
Puede instalar, asignar, desinstalar, desasignar, y extraer archivos definiéndolos como miembros de un paquete.

Sus operaciones básicas son instalar+asignar, asignar, remover+desasignar, desasignar y extraer (back) Paquetes de una computadora. Además puede informar sobre los paquetes

asignados, buscarlos y generar estadísticas. También puede aumentar la versión de un paquete asignado (útil si se lo esta desarrollando).

Dado un paquete definido mediante a) un archivo tipo tar en sus diferentes compresiones; b) paquetes deb, rpm, o similares; c) un directorio conteniendo archivos; d) un archivo conteniendo un listado de archivos previamente existente en la computadora; Uinstall puede instalar los paquetes del tipo a, b y c en el sistema raíz de la computadora o en algún subdirectorio (por ej. una imagen de otro sistema), estableciendo además que esos archivos pertenecen a ese paquete.
O en el caso "d", puede asignar esos archivos a un paquete. A partir de tener un conjunto de archivos de la computadora instalados y asignados a un paquete se puede: desinstalarlos, desasignarlos, incrementar su versión, o extraerlos (back) a un directorio.

Al operar sobre cada archivo realiza numerosas comprobaciones sobre el mismo, verificando si cambió desde su instalación, salvándolo si se va a remplazar, etc..

El sistema convive con otros sistemas de paquetería específicos de cada distribución GNU/Linux.

Comandos: **Install**: -i CONTENT_PACKAGES; **Remove**: -r PACKAGE_NAMES; **Assign** -a FILE_LIST_PACKAGES; **Deassign**: -d PACKAGE_NAMES; **Back**: -b PACKAGE_NAMES; **Define_new_version**: -z; PACKAGE_NAME; **Info**: -n PACKAGE_NAME ; **Stats**: -t; **Search**: -s STRING

Es útil para:

- quien instala paquetes desde las fuentes, luego de hacer un "make DESTDIR=directorio install", se puede instalar y luego desinstalar o actualizar, en el sistema, estos archivos con uinstall;
- probar paquetes de otras distros en una computadora;
- definir archivos propios como pertenecientes a un paquete y con eso hacer backup y moverlos cómodamente a repositorios u otras computadoras

Usa una base de datos plana en ascii, que vincula archivos con paquetes, ordenada con look. Tiene por cada paquete un "log" historico de lo que se hizo con cada archivo. Se guarda mucha otra metainfo, según esté disponible.

Bootstrap

Sumpack genera un archivo en Bash para realizar un proceso de bootstrap que instala Sumapack. Baja las dependencias que Sumapack necesita y finalmente al propio Sumapack. Funciona incluso usando un directorio con los paquetes a usar en un medio portátil como un dvd, para poder instalar el software sin disponer de Internet. Este proceso puede generalizarse para hacer paquetes autoinstalables en cualquier distro.

RESULTADOS

Se ha completado la primera fase del desarrollo de Sumapack, obteniendo las primeras versiones operativas del sistema. Queda mucho trabajo por delante en cuanto a mejora, corrección de errores, integración, y ampliación de uso a diferentes sistemas.

Se ha desarrollado software en el marco de Sumapack para: instalar cualquier clase de paquete y manejar una base de datos propia; compilar y generar paquetes portando el sistema ebuild de gentoo a cualquier distribución, generar documentación con Muse, y facilitar el uso de las autotools para generar paquetes tipo tbz2, rpm y deb.

Se ha creado un sitio web para el proyecto Sumapack listado en las referencias.

Mediante Sumapack se ha mejorado los proyectos Psicro, Puyuspa, Sceptre-inenco y Simusol, estando en proceso la total incorporación de las capacidades de Sumapack a los mismos.

CONCLUSIONES

Del trabajo efectuado se desprende que es posible comenzar un camino que tienda a resolver los principales problemas de dispersión de los sistemas de paquetería comunmente usados en las distribuciones de Software Libre, facilitando en particular el desarrollo de software y su uso por diferentes interesados; particularmente en los entornos altamente especializados y de relativamente pocos usuarios como el de las energías alternativas.

REFERENCIAS

Alía de Saravia, D., Saravia, L. and Saravia, D. (2003) Simusol: Simulating thermal systems using Sceptre and Dia

http://www.frcu.utn.edu.ar/deptos/depto_3/32JAIIO/jsl/JSL_01.pdf

Alía de Saravia, D., Saravia, L. and Saravia, D. (2009). Simusol

http://www.simusol.org.ar

ArchLinux (2010). Pacman Rosetta.

http://wiki.archlinux.org/index.php/Pacman_Rosetta

Beekmans, Gerard (2010), Linux from Scratch.

http://www.linuxfromscratch.org/

Free Software Foundation (2007), Licencia AGPL. Ver también la GPL.

http://www.gnu.org/licenses/agpl.html

Free Software Foundation (2009), GNU Coding Standars

http://www.gnu.org/prep/standards/standards.html

Free Software Foundation (2010a). Software Libre

http://www.gnu.org/philosophy/free-sw.es.html

Free Software Foundation (2010b). GCC, the GNU Compiler Collection

http://gcc.gnu.org/

Gentoo Foundation, (2010). Gentoo Linux

http://www.gentoo.org/

Knuth, D. (1978). TeX

http://www.tug.org

http://es.wikipedia.org/wiki/TeX

LaTeX: http://es.wikipedia.org/wiki/LaTeX

Montoro, S. (2010). Mark Shuttleworth y la meritocracia

http://www.lapastillaroja.net/archives/002012.html

Mozilla Foundation, (2010). XUL. Mozilla Development Center
https://developer.mozilla.org/En/XUL

Murdock, I. (2007). How package management changed everything.
http://ianmurdock.com/solaris/how-package-management-changed-everything/

Novender, W. (1999). Sceptre: Simulation of Nonlinear Electrical Circuits Issue 63. July.
http://www.linuxjournal.com/article/3008

Nullsoft (2009). Nullsoft Scriptable Install System (NSIS)
http://nsis.sourceforge.net/Main_Page

Olson, M. (2009). Emacs Muse.
http://mwolson.org/projects/EmacsMuse.html

OpenSUSE (2009). Software Management Command Line Comparison.
http://old-en.opensuse.org/Software_Management_Command_Line_Comparison

Red Hat (2008). Cygwin.
http://www.cygwin.com

Salas, G. (2007). Instalar Simusol: gcc y g77 en Ubuntu 8.10
http://ubuntuforums.org/showthread.php?t=1147530

Saravia, D. (2005). Manifiesto Anti Distros.
http://docs.hipatia.info/distros/

Saravia D. (2008) Avances en Psicro, La calculadora Psicrométrica. Presentación de Puyuspa y Calcula. AVERMA,
Vol. 12, Sección 8, pp79, 20-08

http://www.cricyt.edu.ar/lahv/asades/averma/2008/20-08.pdf

Saravia, D. (2010a). Psicro
http://www.psicro.org

Saravia, D. (2010b). SumaPack
http://www.sumapack.org

Strom, E. (2009). XUL::Gui - render cross platform gui applications with firefox from Perl
http://search.cpan.org/~asg/XUL-Gui-0.36/lib/XUL/Gui.pm

Ututo, El proyecto, (2000). Ututo, Un GNU/Linux Simple,
http://softwarelibre.unsa.edu.ar/ututo/ututo.html
Actual: http://www.ututo.org

Wikipedia (2010a). Uget
http://en.wikipedia.org/wiki/Ututo

Wikipedia (2010b). GNU build system
http://es.wikipedia.org/wiki/GNU_build_system