

Summapack 2026

El `.summa` como fuente única de los paquetes:
`tar.gz`, `tgz`, `deb` y `rpm` desde el proyecto original

Diego Saravia

Facultad de Ciencias Exactas – Universidad Nacional de Salta

INENCO (UNSa – CONICET)

`dsa@unsa.edu.ar`

7 de octubre de 2026

Resumen

Summapack es el conjunto de herramientas con que se compilan, empaquetan, publican e instalan los programas de Siru y Simusol: calcula, puyuspa, siru, slex, tabular2, unidades, exprtk. Este documento describe su estado en octubre de 2026. Cada proyecto lleva un archivo `.summa` con toda la información necesaria para armar sus paquetes; de él se generan los archivos de Debian (`debian/control`, `rules`, ...) y el `.spec` de rpm, y de cada proyecto salen las fuentes en `tar.gz` y, por cada paquete, un `.deb`, un `.tgz` y un `.rpm`, con sus sumas md5. Summapack es además la base común de los proyectos: incluye la biblioteca C `mem_util`. Se retoman las ideas del Manifiesto Anti-distros (2005) y del artículo de Sumapack presentado en las JSL 2011, y se indica qué parte de aquellas propuestas está hoy en funcionamiento y qué falta.

Índice

1. Guía rápida	2
1.1. Publicar una versión nueva	2
1.2. Instalar en otra máquina	2
1.3. El acceso al servidor	2
2. Antecedentes	2
3. Qué es Summapack hoy	3
3.1. La biblioteca de base: <code>mem_util</code>	3
4. El archivo <code>.summa</code>	3
4.1. Campos	4
4.2. Un <code>.summa</code> , varios paquetes	4
4.3. <code>summa.py</code>	5
5. Armar los paquetes: <code>hacer-deb</code>	5
5.1. El rpm	6
6. Publicar: <code>subir-paquetes</code>	6
7. Bajar e instalar: <code>bajar-paquetes</code> e <code>instalar-tgz</code>	6
8. <code>makefile.lib</code> y lo demás	7

9. Proyectos	8
10. Flujo de trabajo	8
11. Otras distribuciones	8
12. Lo que falta: hoja de ruta	9

1. Guía rápida

1.1. Publicar una versión nueva

1. Cambiar el código y, si cambia algo del paquete, el `.summa`.
2. Anotar el cambio en `debian/changelog` (`dch -i`): sube la revisión.
3. Armar con `PROYECTO/hacer-deb`. Deja `tar.gz`, `deb`, `tgz`, `rpm`, `.summa` y `.md5`, y los anota en `subir.lista` de la raíz del repositorio.
4. Subir, como inestable: `subir-paquetes $(cat ~/svn/siru/subir.lista)` (y lo mismo con `~/svn/calcula/subir.lista`).
5. Probar: `bajar-paquetes -inestable PROYECTO`.
6. Marcar estable: `subir-paquetes -estable PROYECTO`.

1.2. Instalar en otra máquina

Con `summapack`: `bajar-paquetes -instalar calcula`. Sin `summapack`, una vez:

```
curl -fsSL0 https://www.simusol.org/downloads/installsumma.sh
sh installsumma.sh tabular2      # summapack y tabular2, con lo que necesita
```

Para eso, `make subir-bootstrap` en `summapack` pone en el servidor `installsumma.sh`, `bajar-paquetes` e `instalar-tgz`.

1.3. El acceso al servidor

En `simusol.org`, `www-data` no entra por `ssh` (intentarlo hace que `fail2ban` bloquee la IP). Se publica con un usuario propio del grupo `www-data` que solo escribe en `downloads/`, configurado en `~/config/simusol/subir` (`SUBIR_SERVIDOR`, `SUBIR_USUARIO`, `SUBIR_DIR`). Cómo se creó está en `PAQUETES.md`.

2. Antecedentes

En el *Manifiesto Anti-distros* [1] se planteaba que el empaquetado debía ser un trabajo vinculado estrechamente al desarrollo de cada programa y no a cada distribución: que los `Makefile` de cada proyecto pudieran producir los paquetes necesarios —“`deb`, `rpm`, `tgz`”— y que la diferencia entre un `deb` y un `rpm` fuera “solo de formato, como puede ser la diferencia entre un `gz` y un `bz2`”.

El artículo de `Sumapack` [2] describió el primer sistema construido con esas ideas. Entre sus componentes: `Rosetta` y `Babel` (traducir el nombre de un paquete al de cada distribución e instalarlo con la herramienta de cada una), `AutoSumma` (extensión de las `autotools` para generar paquetes `deb`, `rpm`, `txz` o `ebuild`, donde “los archivos que portan información específica del paquete en cada tipo: `.spec`, `.lsm`, `control`, `.ebuild`, se generan desde un archivo común”), `Uinstall`, `Upack`, `Upackname`, `Ubuild` y un `Bootstrap`. Ambos textos están en `doc/`.

Con los años la parte de empaquetado quedó repartida: una biblioteca de make (`makefile.lib`) con muchos objetivos de los que no se sabía cuáles andaban, y en cada proyecto un `debian/` escrito a mano y un script `hacer-deb` que solo producía el `.deb`. No había herramientas para generar el `tar.gz`, el `tgz` ni el `rpm`, ni para subir los paquetes al servidor, ni para mantener los enlaces a la última versión y las sumas md5. Este trabajo vuelve al “archivo común” del artículo: el `.summa`.

3. Qué es Summapack hoy

Un solo proyecto (`siru/summapack`), un solo `summapack.summa`, y tres paquetes:

Paquete	Contenido	Arquitectura
<code>summapack</code>	<code>makefile.lib</code> , <code>diccionariodepaquetes</code> , <code>hacer-deb.inc</code> , <code>summa.py</code> , <code>subir-paquetes</code> y los scripts de <code>bin/</code>	<code>all</code>
<code>libmem-util0</code>	la biblioteca C <code>mem_util</code>	<code>any</code>
<code>libmem-util-dev</code>	sus encabezados, el <code>.a</code> , el enlace de desarrollo y <code>mem_util.pc</code>	<code>any</code>

`summapack` depende de `libmem-util-dev`: alcanza con pedir `summapack` para tener las dos cosas.

3.1. La biblioteca de base: `mem_util`

`mem_util` reúne lo que todos los programas de Siru y Simusol necesitaban y cada uno reimplementaba: crecimiento de vectores en memoria, apertura y búsqueda de archivos, conversión entre rutas relativas y absolutas, y fechas. Desde octubre de 2026 `datetime()` devuelve segundos desde 1970 (epoch), respetando la zona local o la indicada en el dato; `tabular2 2.0-14` lo requiere.

Se copió de `siru/mem_util` a `summapack/mem_util`, con su historia de subversion. Los paquetes conservan sus nombres (`libmem-util0`, `libmem-util-dev`), de modo que `slex`, `tabular2` y `calcula` no cambian, y salen con una versión mayor (2017.2.1.R1008-2 frente a 1.0-2), que reemplaza a la anterior. Más adelante `siru/mem_util` se borra y los proyectos que la usan pasan a pedir `summapack`.

4. El archivo `.summa`

Cada proyecto tiene en su directorio un `NOMBRE.summa`. Es un archivo de shell (lo leen también las herramientas viejas con `. NOMBRE.summa`) y es *la* fuente: si alguien cambia el `.summa`, cambian todos los paquetes. En `debian/` queda únicamente `changelog`, que es la historia y lleva la revisión (el -38 de 6.0-38); su versión de arriba tiene que coincidir con `BASEVERSION`.

4.1. Campos

Campo	Qué es
BASEVERSION	la versión de arriba (6.0)
SUMMARY, DESCRIPTION	resumen y descripción del paquete principal
DEPENDS	sus dependencias (admite <code>\${shlibs:Depends}</code>)
MAINTAINER, HOMEPAGE, SECTION, PRIORITY, LICENSE, STANDARDS	los datos generales
BUILDDEPENDS	lo necesario para compilar
CONFIGURE	las opciones de <code>./configure</code> (<code>-with-klu, ...</code>)
DH	las opciones de <code>dh</code>
RULES	el resto de <code>debian/rules</code> , lo propio del proyecto
SOURCE_EXTRA	otros campos del paquete fuente
PACKAGES	los paquetes que salen; el primero es el principal
<i>P</i> _SUMMARY, <i>P</i> _DESCRIPTION, <i>P</i> _DEPENDS	los de cada paquete <i>P</i> que no es el principal
<i>P</i> _ARCH, <i>P</i> _EXTRA	arquitectura y otros campos de control (Section, Multi-Arch, Suggests, ...) de cada paquete
ARCHIVOS, ARCHIVO_ <i>ruta</i>	los demás archivos de <code>debian/</code> (<code>copyright</code> , <code>P.install</code> , <code>P.docs</code> , <code>postinst</code> , ...), cada uno entero
EJECUTABLES	cuáles de ellos van con permiso de ejecución

En los nombres de variable, *P* es el nombre del paquete con `_` en lugar de `-`, `.` y `+` (por ejemplo, `libmem_util_dev_SUMMARY`). Los valores de varias líneas se escriben como documento en línea:

```
BASEVERSION="2025.01.01"
SUMMARY="C++ Mathematical Expression Toolkit Library (cabecera)"
DESCRIPTION=$(cat << 'FIN_SUMMA'
exprtk evalua expresiones matematicas escritas como texto: variables,
funciones, condicionales y las operaciones habituales, ...
FIN_SUMMA
)
BUILDDEPENDS="debhelper-compat (= 13)"
DH="--makefilename=Makefile.simple"
PACKAGES="libexprtk-dev"
libexprtk_dev_ARCH="all"
```

4.2. Un .summa, varios paquetes

Un proyecto puede dar varios paquetes. Summapack da tres; `tabular2` da `libtabular2-0`, `libtabular2-dev` y `tabular`; `slex` da `libslex0`, `libslex-dev` y `d2datdinc`. Todo se instala en `debian/tmp` y cada `ARCHIVO_P_install` elige lo suyo:

```
PACKAGES="summapack libmem-util0 libmem-util-dev"
summapack_ARCH="all"
libmem_util0_ARCH="any"
libmem_util_dev_DEPENDS="libmem-util0 (= ${binary:Version}), ${misc:Depends}"
ARCHIVO_summapack_install=$(cat << 'FIN_SUMMA'
debian/tmp/etc
usr/bin/*
usr/share/summapack
FIN_SUMMA
)
```

Si una ruta existe también en el árbol fuente (como `etc` en `summapack`), hay que escribirla `debian/tmp/etc: dh_install` busca primero en el árbol fuente.

4.3. `summa.py`

Orden	Qué hace
<code>summa.py crear DIR</code>	escribe un <code>.summa</code> nuevo desde un <code>debian/</code> completo (una vez por proyecto)
<code>summa.py actualizar DIR</code>	pasa un <code>.summa</code> viejo de <code>makefile.lib</code> al formato nuevo, conservando sus otros campos (<code>KEYWORDS</code> , <code>AUTHOR</code> , <code>VENDOR</code> , ...), y dice qué cambió
<code>summa.py debian DIR</code>	genera <code>DIR/debian/</code> completo (menos el changelog)
<code>summa.py probar DIR</code>	crear y luego <code>debian</code> en un temporal, y comparar con el <code>debian/</code> que hay
<code>summa.py campo DIR CAMPO</code>	el valor de un campo

Antes de dejar `debian/` solo con el changelog, se comprobó con los nueve proyectos que lo generado desde el `.summa` es equivalente, campo por campo, a lo que estaba escrito a mano, y que el `.deb` de `exptk` armado desde el `.summa` es idéntico (archivos, permisos y control) al anterior.

5. Armar los paquetes: `hacer-deb`

Cada proyecto tiene un script `hacer-deb` que incluye `hacer-deb.inc` de `summapack`. Los pasos:

1. Comprueba que estén las órdenes necesarias y, si falta alguna, dice qué paquete instalar.
2. Hace una copia limpia con `svn export`. Lo que no está agregado a subversion no entra: un `.summa` nuevo hay que agregarlo con `svn add`.
3. Genera `debian/` desde el `.summa` en esa copia.
4. Arma el `tar.gz` de las fuentes, reproducible: mismos archivos, mismo orden, dueño 0, fecha del changelog y `gzip -n`, de modo que dos corridas dan los mismos bytes.
5. Compila y corre las pruebas con `dpkg-buildpackage`.
6. Por cada `.deb`, arma el `.tgz` (el mismo árbol, para desempaquetar en / en cualquier distribución) y el `.rpm`.
7. Deja todo junto al proyecto con un `.md5` por archivo.

Cuando algo falla, dice en qué paso, en qué línea y qué orden, y en los casos conocidos qué hacer. Los archivos que quedan, con calcula como ejemplo:

Archivo	Qué es
<code>calcula_6.0-38.tar.gz</code>	las fuentes
<code>calcula_6.0-38_amd64.deb</code>	el paquete Debian/Ubuntu
<code>calcula_6.0-38_amd64.tgz</code>	el mismo contenido, en tar
<code>calcula-6.0-38.x86_64.rpm</code>	el mismo contenido, en rpm
<code>calcula_6.0-38.summa</code>	el <code>.summa</code> con que se armó
<code>*.md5</code>	uno por cada uno de los anteriores

5.1. El rpm

El `.rpm` se arma desde el árbol del `.deb` con un `.spec` generado: nombre, versión, resumen, licencia, arquitectura (`all` pasa a `noarch`, `amd64` a `x86_64`) y la lista de archivos. Las dependencias se traducen con `diccionariodepaquetes`, que es lo que el artículo de 2011 llamaba Rosetta: el nombre de cada paquete en cada distribución. Hace falta `rpmbuild` (paquete `rpm`); si no está, se avisa y se arma el resto.

Es, como proponía el manifiesto, “el mismo paquete en otro formato”, con dos límites que hoy lo atan a Debian: las dependencias que calcula Debian (`libc6`) no tienen todavía su nombre en Fedora o SUSE (`glibc`), y las bibliotecas quedan en `/usr/lib/x86_64-linux-gnu` en lugar de `/usr/lib64`.

6. Publicar: subir-paquetes

El directorio de descargas está organizado por proyecto (pedido del autor, 7 de octubre de 2026):

```
downloads/
  viejos/          todo lo que habia antes
  installsumma.sh  instala summapack: lo que hace falta para
                   instalar cualquier paquete
  calcula.summa    el .summa estable de cada proyecto, con
                   VERSION="6.0-38"
  calcula/         los paquetes estables (.deb .rpm .tgz .tar.gz)
  otros/           .md5, inestables, versiones anteriores, .summa
                   con version, y PAQ_UNSTABLE mientras haya algo
                   inestable
  tabular2/        libtabular2-0, libtabular2-dev y tabular
  otros/
```

```
subir-paquetes ARCHIVOS...      a PROYECTO/otros/, con su .md5,
                                y PAQ_UNSTABLE apunta a cada uno
subir-paquetes --estable PROYECTO... lo UNSTABLE pasa a PROYECTO/, el
                                estable anterior vuelve a otros/,
                                el .summa se copia arriba con
                                VERSION=... y el UNSTABLE
                                desaparece
subir-paquetes --lista [PROYECTO...]
subir-paquetes --reorganizar     una vez: lo anterior a viejos/
```

El proyecto de cada paquete sale de su `.summa`. La versión estable la dice el `PROYECTO.summa` de arriba (`VERSION="6.0-38"`); lo inestable tiene un nombre fijo con `UNSTABLE` en lugar de la versión (`calcula_UNSTABLE_amd64.deb`) mientras exista. Los `_last` de antes quedaron en `viejos/`. Lo que ya está con el mismo md5 no se vuelve a subir; si está distinto (una subida cortada), sí. La configuración del servidor va en `~/config/simusal/subir`, y lo que venga en el entorno manda. En `simusal.org` se publica con un usuario del grupo `www-data` (`www-data` no entra por `ssh`).

7. Bajar e instalar: bajar-paquetes e instalar-tgz

`bajar-paquetes` es la contraparte de `subir-paquetes`: dado un proyecto, baja primero su `PROYECTO.summa` y, con lo que dice (paquetes y versión), los binarios; también parte de un `.summa` que ya se tenga, o baja uno solo con `PROYECTO/PAQUETE`; la versión estable, la última subida (`-inestable`) o una dada, en el formato del sistema (`deb` en Debian y Ubuntu, `rpm` en Fedora, Red Hat y SUSE, `tgz` en los demás), con su `.md5`, por `https` y comprobando el certificado. Si

el md5 no coincide, el archivo se descarta. Con `-instalar` lo instala con la herramienta del sistema. Baja también las dependencias que son de nuestros proyectos y no están instaladas (o están en una versión menor que la pedida): pedir `tabular2/tabular` trae `libtabular2-0`, `libslex0` y `libunidades0`; las del sistema las resuelve `apt` o `dnf`.

`instalar-tgz` es el `Uinstall` del artículo de 2011, para las máquinas sin `dpkg` ni `rpm`: instala un `.tgz` llevando registro en `/var/lib/summapack/tgz`, de modo que después se puede listar, ver sus archivos y quitar. No pisa archivos de paquetes de `dpkg` o `rpm`; lo que pisa lo respalda y lo devuelve al quitar; al actualizar saca lo que la versión nueva ya no trae.

El Bootstrap del artículo es ahora `installsumma.sh`: en una máquina sin `summapack`, baja `bajar-paquetes` e `instalar-tgz` del servidor e instala `summapack` y `mem_util`:

```
curl -fsSLO https://www.simusol.org/downloads/installsumma.sh
sh installsumma.sh tabular2      # summapack y tabular2, con lo que necesita
```

Lo necesario para compilar. `summa.py` dependencias da los nombres de `BUILDDEPENDS`; `hacer-deb` comprueba antes de compilar que estén y, si falta algo, da la orden para instalarlo; `make dependencias` lo instala con `packinstall`, que traduce los nombres en cada distribución. Es el `lacks` y `make prepare` del artículo, conectado ahora al `.summa`.

8. makefile.lib y lo demás

makefile.lib. La biblioteca de `make` que incluyen los `Makefile` de los proyectos: unos treinta objetivos documentados (`install`, `dist`, `tgzbuild`, `ayuda`, ...). `make ayuda` lista los de cualquier proyecto que la incluya; `make prueba` la prueba con un proyecto de ejemplo y distingue lo que anda, lo que falla y los problemas conocidos. Su estado está en `MAKEFILE-LIB.md`.

diccionariodepaquetes. Los nombres de los paquetes en cada distribución.

bin/. `distro` dice el sistema (`UBUNTU`, `DEBIAN`, `FEDORA`, `SUSE`, ..., y con `+` nombre y versión de `/etc/os-release`); `packinstall` es el Babel del artículo: traduce nombres con `diccionariodepaquetes` e instala con la herramienta de cada sistema, con `-prueba` para ver las órdenes sin correrlas; `summapack` da la versión instalada; `icon_install` pone lanzadores en el menú y el escritorio; `findbinlib` encuentra instalaciones viejas que taparían a las de un paquete (lo usa `instalar-tgz`). Los que nadie usaba o no andaban se pasaron a `viejo/bin`. `BIN.md` tiene el relevamiento de cada uno y `pruebas/prueba-bin.sh` los prueba sin cambiar la máquina.

Dónde están. Una sola copia, en `summapack`:

Archivo	En el árbol	Instalado
<code>hacer-deb.inc</code> , <code>summa.py</code> , <code>diccionariodepaquetes</code>	<code>share/summapack/data/</code>	<code>/usr/share/summapack/d</code>
<code>subir-paquetes</code> , <code>bajar-paquetes</code> , <code>instalar-tgz</code>	<code>bin/</code>	<code>/usr/bin/</code>

Los `hacer-deb` de siru la toman del mismo repositorio. Los de `calcula`, que es otro repositorio, la buscan en `SUMMAPACK_DATA`, en la copia de trabajo de siru que esté al lado, o en `summapack` instalado. Como decía el artículo, “la inexistencia de `Sumapack` en una computadora que instala un paquete preparado para `Sumapack` no bloquea las funciones esenciales”: los paquetes generados no dependen de `summapack` para instalarse.

9. Proyectos

Proyecto	Paquetes	Versión
siru/summapack	summapack, libmem-util0, libmem-util-dev	2017.2.1.R1008-7
siru/expvtk	libexpvtk-dev	2025.01.01-1
siru/unidades	libunidades0, libunidades-dev	1.0-1
siru/slex	libslex0, libslex-dev, d2datdinc	1.0-24
siru/tabular2	libtabular2-0, libtabular2-dev, tabular	2.0-14
siru/siru	siru	1.0-3
calcula/calcula-6.0	calcula	6.0-38
calcula/puyuspa-6.0	puyuspa	6.0-22
siru/mem_util	libmem-util0, libmem-util-dev (se va a borrar)	1.0-2

Los nueve se arman desde su `.summa`; de los ocho que se publican salieron, en octubre de 2026, 61 archivos para el servidor: 17 `.deb`, 17 `.tgz`, 17 `.rpm`, 8 `tar.gz` de fuentes y 8 `.summa`, cada uno con su `.md5`.

10. Flujo de trabajo

```
# cambiar algo del paquete: editar el .summa y anotar el cambio
vi tabular2/tabular2.summa
dch -i -c tabular2/debian/changelog
# armar: tar.gz, deb, tgz, rpm, summa y md5
tabular2/hacer-deb
# publicar (hacer-deb anoto lo armado en subir.lista)
subir-paquetes --prueba $(cat ~/svn/siru/subir.lista)
subir-paquetes $(cat ~/svn/siru/subir.lista)
bajar-paquetes --inestable tabular2      # probarlo
subir-paquetes --estable tabular2
```

Para un proyecto nuevo: escribir `debian/` una vez (o copiar el `.summa` de uno parecido), correr `summa.py crear`, agregar el `.summa` a subversion, dejar en `debian/` solo el changelog y copiar un `hacer-deb`.

11. Otras distribuciones

Los paquetes se compilan en Ubuntu 24.04, y un binario compilado ahí pide las bibliotecas de ese sistema, con su versión. No siempre anda en otro:

- **libc:** lo compilado pide `libc6 ≥ 2.38`; Debian 12 trae 2.36 y no instala.
- **Versiones de bibliotecas:** calcula usa `gsl 2.7 (libgsl.so.27)`; Fedora 44 y openSUSE Tumbleweed traen `gsl 2.8 (libgsl.so.28)`, y no corre aunque se instale. Lo mismo puede pasar con `suitesparse`.
- **Nombres de paquetes:** `libc6` en Debian es `glibc` en Fedora y SUSE. Se resuelve con el `diccionariodepaquetes`: los `.rpm` piden `glibc` y, para las bibliotecas, la biblioteca misma (`libreadline.so.8()(64bit)`).
- **Lugar de las bibliotecas:** van en `/usr/lib/x86_64-linux-gnu`, que es de Debian; Arch no la mira, e `instalar-tgz` la agrega a `/etc/ld.so.conf.d`.

Probado el 7 de octubre de 2026 en contenedores, con `sh installsumma.sh tabular2`: en Ubuntu 24.04 y Debian 13 instala y anda; en Debian 12 no instala (`libc`); en Fedora 44, openSUSE Tumbleweed y Arch fallaba por los nombres y el lugar de las bibliotecas, y se corrigió.

La solución: compilar en cada distribución. En `siru/distros` van los entornos (contenedores con podman) para compilar los paquetes en la versión actual de openSUSE, Fedora y Arch, y en Debian 12 si hace falta, cada vez que aparece uno de estos problemas. Así se proveen paquetes propios de cada una, que `bajar-paquetes` elige según lo que dice `distro`.

12. Lo que falta: hoja de ruta

Del artículo de 2011	Hoy
Rosetta: el nombre de cada paquete en cada distribución	<code>diccionariodepaquetes</code> ; lo usa el rpm, pero faltan los nombres de lo que calcula Debian (<code>libc6</code> $\rightarrow$ <code>glibc</code>)
<code>.spec</code> , <code>.lsm</code> , <code>.ebuild</code> desde el archivo común rpm propio de cada distribución	<code>debian/</code> y <code>.spec</code> ; faltan <code>.lsm</code> y <code>.ebuild</code> es el árbol del deb: faltan <code>/usr/lib64</code> y las dependencias por biblioteca
<code>lacks</code> y <code>make prepare</code> : instalar lo que falta para compilar	hecho: <code>make dependencias</code> y el aviso de <code>hacer-deb</code>
<code>Uinstall</code> : instalar un tgz llevando registro para desinstalarlo	hecho: <code>instalar-tgz</code>
<code>Bootstrap</code> : un script que instala <code>summapack</code> y sus dependencias	hecho: <code>installsumma.sh</code> con <code>bajar-paquetes</code>
Dependencias por lo que se provee y requiere (bibliotecas) y no por nombres	<code>\${shlibs:Depends}</code> en el deb; en el rpm, <code>AutoReqProv</code> apagado

Y de la organización de los proyectos: borrar `siru/mem_util` y que `slex`, `tabular2` y `calcula` pidan `summapack`.

Referencias

- [1] D. Saravia. *Manifiesto Anti Distros*, versión 0.60, 2005. <http://docs.hipatia.info/distros/>. Copia en `doc/UTUTO-ANTIDISTROS.html`.
- [2] D. Saravia. *Sumapack: herramientas para la compilación, empaquetado e instalación de software libre en diferentes arquitecturas, sistemas operativos y distribuciones GNU/Linux*. 40 JAIIO, Jornadas de Software Libre, 2011, pp. 41–59. Copia en `doc/738.pdf`.
- [3] `siru/summapack/PAQUETES.md`: la referencia de uso, que se mantiene junto al código.