Anaconda 2/3

ToC

Diese OPSI-Paket für **Anaconda 2/3** wurde für die Verwendung im *OPSI-4-Institutes*-Repository sowie des *Max-Planck-Instituts für Mikrostrukturphysik* erstellt.
Es wird versucht auf die Besonderheiten der jeweiligen Repositories einzugehen; entsprechend werden durch ein einfaches *Makefile* aus den Quellen verschiedene Pakete erstellt.
Grundsätzlich werden separate OPSI-Pakete für **Anaconda2** und **Anaconda3** erstellt.

Was ist Anaconda?

Anaconda ist eine Freemium-Open-Source-Distribution für die Programmiersprachen **Python** und **R**, die unter anderem die Entwicklungsumgebung **Spyder**, den Kommandozeileninterpreter **IPython**, und ein webbasiertes Frontend für **Jupyter** enthält.
Der Fokus liegt vor allem auf der Verarbeitung von großen Datenmengen, Vorhersageanalyse und wissenschaftlichem Rechnen.
Das Ziel der Distribution ist die Vereinfachung von Paketmanagement und Softwareverteilung.
(Wikipedia)

Paket erstellen

Dieser Abschnitt beschäftigt sich mit der Erstellung des OPSI-Paketes aus dem Source-Paket und nicht mit dem OPSI-Paket selbst.

Makefile und spec.json

Da aus den Quellen verschiedene Versionen des Paketes mit entsprechenden Anpassungen generiert werden sollen (mpimsp/o4i/dfn; testing/release) wurde hierfür ein **`Makefile`** erstellt. Darüber hinaus steuert **`spec.json`** die Erstellung der Pakete.

Im Idealfall ist beim Erscheinen einer neuen Release von Anaconda lediglich die **`spec.json`** anzupassen.

Zur Vorbereitung der eigentlichen Paketerstellung sind zuvor die Softwarepakete mit

```
make download
```

herunterzuladen. Hierbei werden die später benötigten MD5sums erstellt.

Mustache

Als Template-Engine kommt **Mustache** zum Einsatz.
Im Detail wird hier eine Go-Implementierung verwendet. Die Software ist auf Github zu finden. Binaries für Linux und Windows liegen diesem Paket bei.

Verzeichnisstruktur

Die erstellten Pakete werden im Unterverzeichnis **`BUILD`** abgelegt.

Einige Files (derzeit `control, preinst, postinst`) werden bei der Erstellung erst aus *`.in`*-Files generiert, welche sich in den Verzeichnissen `SRC/OPSI` und `SRC/CLIENT_DATA` befinden. Die `SRC`-Verzeichnisse sind in den OPSI-Paketen nicht mehr enthalten.

Makefile-Parameter

Eine kurze Hilfe zu den verfügbaren Parametern liefert:

```
make help
```

Der vorliegende Code erlaubt die Erstellung von OPSI-Paketen für die Releases gemäss der Angaben in `spec.json`. Es kann jedoch bei der Paketerstellung ein alternatives Spec–File übergeben werden:

`SPEC=<spec_file>`

Aus den vorliegenden Skripten können OPSI-Pakete für Anaconda2 und/oder Anaconda3 erstellt werden:

`PYVER=(2|3|2,3|both)`

Das Paket kann mit *"batteries included"* erstellt werden. In dem Fall erfolgt der Download der Software beim Erstellen des OPSI-Paketes und nicht erst bei dessen Installation:

`ALLINC=[true|false]`

Standard ist hier die Erstellung des leichtgewichtigen Paketes (`ALLINC=false`).

OPSI erlaubt des Pakete im Format `cpio` und `tar` zu erstellen.
Als Standard ist `cpio` festgelegt.
Das Makefile erlaubt die Wahl des Formates über die Umgebungsvariable bzw. den Parameter:

`ARCHIVE_FORMAT=(cpio|tar)`

spec.json

Häufig beschränkt sich die Aktualisierung eines Paketes auf das Aendern der Versionsnummern und des Datums etc. In einigen Fällen ist jedoch auch das Anpassen weiterer Variablen erforderlich, die sich auf verschiedene Files verteilen.
Auch das soll durch das Makefile vereinfacht werden. Die relevanten Variablen sollen nur noch in `spec.json` angepasst werden. Den Rest übernimmt *`make`*

32-Bit-Umgebung

Die 32-Bit-Systeme kommen immer seltener zum Einsatz. Daher ist es nicht in jedem Fall erforderlich auch eine entsprechende Version von Anacoda zur Verfügung zu stellen. Beeinflussen lässt sich das über das spec-File mit:

```
"ifdef_64bit_only" :true
```

Standardmässig ist der Support für 32 Bit aktiviert..

Installation

Die Software selbst wird - sofern bei der Paketerstellung nicht anders vorgegeben - nicht mit diesem Paket vertrieben. Für die *"batteries included"*-Pakete entfällt dieser Abschnitt.

Je nach Art des erstellten Paketes erfolgt bei der Installation im Depot durch das `postinst`-Script der Download der Software vom Hersteller (möglich sind hier Umgebungen für Python 2 und Python 3, jeweils 32 und 64 Bit).
Ein manueller Download sollte dann nicht erforderlich sein. Auf dem Depot-Server ist **wget** erforderlich.

Das Gesamtvolumen der herunterzuladenden Dateien beträgt in der umfangreichsten Konfiguration ca. **1920 MByte** (Anaconada 2 und 3, 32/64 Bit)!

Properties

Zur Steuerung der Installation des Paketes auf den Clients ist eine Reihe von Properties vorgesehen. Einige hiervon sind eher generischer Natur, andere spezifisch für die vorliegende Software.

generische Properties

`local_installer_copy` - Hiermit kann fuer die Installation eine temporäre lokale Kopie des Installationspaketes auf dem Client erstellt werden, statt diesen direkt vom Netzlaufwerk aufzurufen. In langsamen Netzwerken kann das zu einer Beschleunigung der Installation führen.
(*default: false*)

`kill_running` - Wird bei der Installation eine laufende Instanz definierter Programme erkannt (`python.exe, pythonw.exe, Scripts\*.exe` im Zielverzeichnis), können diese durch das OPSI-Paket beendet werden.
Sollen ggf. laufende Programme nicht beendet werden, wird die Installation des Paketes zurückgestellt und später erneut versucht.
(*default: false*)

`install_architecture` - Diese Option ist nur verfügbar, wenn das Paket nicht ausschliesslich für 64-Bit-Umgebungen erstellt wurde (siehe 32-Bit-Umgebung) und legt fest, welche Architektur der Software installiert werden soll.
Zur Auswahl stehen: "32 bit", "64 bit", "sysnative". (*default: sysnative*)

`log_level` - Es kann hier ein abweichender Log-Level für das Paket definiert werden. Für Test-Pakete ist der Default-Wert 7, für Produktiv-Pakete wurde 5 festgelegt.

`custom_post_install` und **`custom_post_uninstall`** - Hier können Skripte hinterlegt werden, welche optional nach Abschluss der Installation bzw. Deinstallation ausgeführt werden sollen. Die Skripte müssen im Unterverzeichnis `custom` des Paketes auf dem Depot-Server liegen.
Die Skripte muessen Sektionen mit den Namen `SUB_POST_INSTALL` bzw. `SUB_POST_UNINSTALL` enthalten; beide koennen auch in einem gemeinsamen Skript untergebracht sein. Innerhalb dieser koennen weitere Sektionen aufgerufen werden.
(*default: none*)

`required_mimimum_space` - Aufgrund der Komplexität des vorliegenden Paketes und der daraus entstehenden Setups lässt sich der erforderliche Platzbedarf auf dem Zielsystem nicht im Voraus exakt bestimmen. Mit dieser Variablen lässt sich die Angabe präzisieren.

Der Default-Wert beträgt 9000 (Megabyte). Es können hier neben absoluten auch relative Werte angegeben werden. Hierfür der Wert mit einem Vorzeichen (+/-) zu versehen.
Bei Installation zusätzlicher Pakete (`additional_packages`) ist der Wert ggf. zu erhöhen.
Es ist zu berücksichtigen, dass für die Bemessung des Wertes nicht der Umfang des Setups nach der Installation relevant ist, sondern der Maximal-Bedarf während der Installation. Dieser Wert kann deutlich höher ausfallen.

spezifische Properties

`additional_packages` - Neben der Distribution können bei der Installation gleich weitere Pakete hinzugefügt werden (z.B. `tensorflow`, `django`, ...). Werden hier mehrere Pakete angegeben, erfolgt die Trennung über Leerzeichen.
Gegebenenfalls ist eine Anpassung von `required_mimimum_space` erforderlich.
Für diese Funktion ist eine Verbindung zum Internet erforderlich.

`additional_packages_install_mode` - Für die unter `additional_packages` angegebenen Pakete lässt sich festlegen, wann die Installation erfolgen soll: während des Setup-Skriptes, des Update-Skriptes ...oder gar nicht. (*default: setup*)

`clear_package_cache` - Hiermit kann festgelegt werden, ob nach Abschluss der Installation bzw. des Updates der Paketcache gelöscht werden soll um Plattenplatz freizugeben. (Empfohlen.)

`on_missing_uninstall_key` - Verhalten bei fehlendem Uninstall-Key in der Registry, aber existierendem Installationsverzeichnis.

- *defer*: Versuche es beim nächsten Durchlauf erneut
- *fail*: Abbruch mit Fehler (*default*)
- *ignore*: Versuche fortzufahren; kann zu einem Fehler bei der Installation führen
- *rough delete*: Löschen des Verzeichnisses ohne weitere Rückfrage

`register_as_system_python` - Legt fest, ob Anaconda als System-Python registriert werden soll.
Das erlaubt anderen Programmen wie z.B. VSCode oder PyCharm automatisch Anaconda als primären Python-Interpreter zu erkennen. (Empfohlen.)

`upgrade_release` - Das Update-Skript bietet die Möglichkeit für die Anaconda-Distribution inline ein Release-Upgrade vorzunehmen. Das erspart unter Umständen die Notwendigkeit einer kompletten Neuinstallation bzw. ermöglicht ein Distributions-Upgrade ohne Vorliegen einer neuen Version des OPSI-Paketes.
Hierbei werden jedoch die Versionsnummern in der Software-Verwaltung von Windows und auf dem Depot-Server aktualisiert. Das erfolgte Upgrade lässt sich hier nur anhand der Logs auf dem Depot-Server erkennen bzw. auf dem Client direkt (z.B. mit `conda list`). - Experimentell!
Für diese Funktion ist eine Verbindung zum Internet erforderlich.
(*default: false*)

`update_dry_run` - Mit dieser Einstellung lässt sich den Logs auf dem Depot-Server entnehmen, was bei einer Installation zusäztlicher Pakete bzw. einem Upgrade/Update installiert werden würde.
(*default: false*)

`update_verbose` - Hiermit liefert `conda` detailiertere Informationen.
(*default: false*)

`update_skip` - Standardmässig führt OPSI bei Vorliegen eines Update-Skriptes dieses im Anschluss an die Installation aus. Hiermit lässt sich die Ausführung unterbinden.
Unabhängig von der hier vorgenommenen Einstellung wird der Action-Request *Update* jedoch ausgeführt.
Für die Installation von Updates ist eine Verbindung zum Internet erforderlich.
(*default: false*)

`update_rights_skip` - Bei der Installation von zusätzlichen Paketen bzw. Updates kann es dazu kommen, dass Berechtigungen für Dateien falsch gesetzt werden. Diese Einstellung erlaubt das Ueberspringen der Reparatur von Zugriffsrechten während des Updates um Zeit zu sparen. - Nicht emp-

fohlen!
(*default: false*)

link_desktop_Anaconda_Navigator, **link_desktop_Spyder**, **link_desktop_Jupyter_Notebook**, **link_desktop_Anaconda_Prompt**, **link_desktop_Anaconda_PowerShell**, **link_desktop_Jupyter_QtConsole** - optional kann für die angegebenen Programme jeweils eine Verknüpfung auf dem Desktop angelegt werden. (*default: false*)

Allgemeines

Aufbau des Paketes

- **variables.opsiinc** - Da Variablen über die Scripte hinweg mehrfach verwendet werden, werden diese (bis auf wenige Ausnahmen) zusammengefasst hier deklariert.
- **product_variables.opsiinc** - die producktspezifischen Variablen werden hier definiert
- **setup.opsiscript** - Das Script für die Installation.
- **update.opsiscript** - Das Script für das Update bzw. Upgrade von Anaconda.
- **uninstall.opsiscript** - Das Uninstall-Script
- **delsub.opsiinc**- Wird von Setup und Uninstall gemeinsam verwendet. Vor jeder Installation/jedem Update wird eine alte Version entfernt. (Ein explizites Update-Script existiert derzeit nicht.)
- **checkinstance.opsiinc** - Prüfung, ob eine Instanz der Software läuft. Gegebenenfalls wird das Setup abgebrochen. Optional kann eine laufende Instanz zwangsweise beendet werden.
- **checkvars.sh** - Hilfsscript für die Entwicklung zur Ueberprüfung, ob alle verwendeten Variablen deklariert sind bzw. nicht verwendete Variablen aufzuspüren.
- **bin/** - Hilfprogramme; hier: **7zip**, **psdetail**
- **images/** - Programmbilder für OPSI

Nomenklatur

Präfixes in der Produkt-Id definieren die Art des Paketes:

- **0_** - Es handelt sich um ein Test-Paket. Beim Uebergang zur Produktions-Release wird der Präfix entfernt.
- **o4i_** - Das Paket ist zur Verwendung im opsi4institutes-Repository vorgesehen.
- **dfn_** - Das Paket ist zur Verwendung im opsi4institutes-Repository vorgesehen. (identisch mit o4i; abgekündigt!)

Die Reihenfolge der Präfixes ist relevant; die Markierung als Testpaket ist stets führend.

Unattended-Switches

Es handelt sich um ein *NSIS*-Paket mit den hier gebräuchlichen Parametern.

siehe auch: http://www.silentinstall.org/nsis

Lizenzen

Dieses Paket

Dieses OPSI-Paket steht unter der *GNU General Public License* **GPLv3**.

Ausgenommen von dieser Lizenz sind die unter **bin/** zu findenden Hilfsprogramme. Diese unterliegen ihren jeweiligen Lizenzen.

Anaconda

Anaconda steht unter **BSD-Lizenz**:

Anaconda End User License Agreement

Notice of Third Party Software Licenses

Anaconda Distribution contains open source software packages from third parties. These are available on an "as is" basis and subject to their individual license agreements. These licenses are available in Anaconda Distribution or at http://docs.anaconda.com/anaconda/pkg-docs. Any binary packages of these third party tools you obtain via Anaconda Distribution are subject to their individual licenses as well as the Anaconda license. Anaconda, Inc. reserves the right to change which third party tools are provided in Anaconda Distribution.

In particular, Anaconda Distribution contains re-distributable, run-time, shared-library files from the Intel(TM) Math Kernel Library ("MKL binaries"). You are specifically authorized to use the MKL binaries with your installation of Anaconda Distribution. You are also authorized to redistribute the MKL binaries with Anaconda Distribution or in the conda package that contains them. Use and redistribution of the MKL binaries are subject to the licensing terms located at https://software.intel.com/en-us/license/intel-simplified-software-license. If needed, instructions for removing the MKL binaries after installation of Anaconda Distribution are available at http://www.anaconda.com.

Anaconda Distribution also contains cuDNN software binaries from NVIDIA Corporation ("cuDNN binaries"). You are specifically authorized to use the cuDNN binaries with your installation of Anaconda Distribution. You are also authorized to redistribute the cuDNN binaries with an Anaconda Distribution package that contains them. If needed, instructions for removing the cuDNN binaries after installation of Anaconda Distribution are available at http://www.anaconda.com.

Anaconda Distribution also contains Visual Studio Code software binaries from Microsoft Corporation ("VS Code"). You are specifically authorized to use VS Code with your installation of Anaconda Distribution. Use of VS Code is subject to the licensing terms located at https://code.visualstudio.com/License.

Cryptography Notice

This distribution includes cryptographic software. The country in which you currently reside may have restrictions on the import, possession, use, and/or re-export to another country, of encryption software. BEFORE using any encryption software, please check your country's laws, regulations and policies concerning the import, possession, or use, and re-export of encryption software, to see if this is permitted. See the Wassenaar Arrangement http://www.wassenaar.org/ for more information.

Anaconda, Inc. has self-classified this software as Export Commodity Control Number (ECCN) 5D992b, which includes mass market information security software using or performing cryptographic functions with asymmetric algorithms. No license is required for export of this software to non-embargoed countries. In addition, the Intel(TM) Math Kernel Library contained in Anaconda, Inc.'s software is classified by Intel(TM) as ECCN 5D992b with no license required for export to non-embargoed countries.

The following packages are included in this distribution that relate to cryptography:

openssl The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols as well as a full-strength general purpose cryptography library.

pycrypto A collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).

pyopenssl A thin Python wrapper around (a subset of) the OpenSSL library.

kerberos (krb5, non-Windows platforms) A network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography.

cryptography A Python library which exposes cryptographic recipes and primitives.

Mustache

Die verwendete *Mustache Template Engine for Go* steht unter MIT-Lizenz.

psdetail

Autor der Software: Jens Böttge <boettge@mpi-halle.mpg.de>

Die Software **psdetail.exe** bzw. **psdetail4.exe** wird als Freeware kostenlos angeboten und darf für nichtkommerzielle sowie kommerzielle Zwecke genutzt werden. Die Software darf nicht verändert werden; es dürfen keine abgeleiteten Versionen daraus erstellt werden.

Es ist erlaubt Kopien der Software herzustellen und weiterzugeben, solange Vervielfältigung und Weitergabe nicht auf Gewinnerwirtschaftung oder Spendensammlung abzielt.

Haftungsausschluss:
Der Autor lehnt ausdrücklich jede Haftung für eventuell durch die Nutzung der Software entstandene Schäden ab.
Es werden keine ex- oder impliziten Zusagen gemacht oder Garantien bezüglich der Eigenschaften, des Funktionsumfanges oder Fehlerfreiheit gegeben.
Alle Risiken des Softwareeinsatzes liegen beim Nutzer.

Der Autor behält sich eine Anpassung bzw. weitere Ausformulierung der Lizenzbedingungen vor.

Für die Nutzung wird das *.NET Framework ab v3.5* benötigt.

Anmerkungen/ToDo

Unterschiede zwischen Neuinstallation und Upgrade

Mit *`upgrade_release`* lässt sich eine bestehende Installation inline via `conda` upgraden. Dabei sind u.U. kleinere Abweichungen zu einer vollständigen Installation der neueren Distribution zu beobachten.

Fehler, Requests for enhancement,...

Siehe hierzu: Issues

Jens Böttge <boettge@mpi-halle.mpg.de>, 2023-08-23 15:12:04 +0200