

Mac Ports

Packaging and Installation Management

Seminarvortrag in der
Arbeitsgemeinschaft Rechnerbetrieb
Universität Bielefeld



Worüber möchte ich hier erzählen?

- Basiskonzepte
 - Warum MacPorts
 - Look & Feel
- MacPorts als Paketmanager
 - Die Grundsätze
 - MacPorts Sourcearchiv
- Einen Port erstellen
 - MacPorts Prozeß
 - Portfile
- Konzeptioneller Ausblick
 - Nicht nur für Mac(?)

Was ist MacPorts?

- Aus Benutzersicht
 - Ein Packetmanager wie apt-get, aptitude, yast, etc.
- Aus Entwicklersicht
 - unsichtbar (!)
- Aus Systemadministratorsicht
 - Eine weitreichende Ablaufautomatisierung bez.
 - Installationsverwaltung
 - Abhängigkeitsverwaltung
 - Softwareverteilung
 - Lokalem Patchmanagement

- Ersteinrichtung
 - Gut und vollständig in der Dokumentation beschreiben
 - XQuartz und XCode installieren, DMG herunterladen PKG ausführen
 - ggf. Basiskonfiguration anpassen (Architektur 64bit/32bit, Std'parameter)
 - `sudo port selfupdate ... done`
- Reguläres Update
 - `sudo port selfupdate; port outdated` -> neue Versionen werden angezeigt
 - `sudo nice port -v upgrade outdated` -> alle neuen Vers. werden installiert
- Standard Wartung
 - `port installed inactive` -> zeigt installierte aber inaktive Versionen
 - `sudo port uninstall inactive` -> löscht alle inaktiven Versionen
- Standard Installation (mit Varianten)
 - `sudo port -v install curl +ssl` -> Installiert curl mit SSL-Unterstützung
 - Typischerweise werden die zu benutzenden Architekturen (intel/ppc) und Wortlängen (32/64) zentral im MacPorts conf'file hinterlegt

Die Grundsätze

- MacPorts ist eine opt-ware
 - (fast) alles ist unter /opt/local/ gespeichert
 - also /opt/local/bin; /opt/local/etc/
 - Es existieren Patches, die /opt/local in die einschlägigen Pfade bringen
- MacPorts benutzt eine hard-link Versionierung
 - Um verschiedene Versionen einer Software gleichzeitig vorzuhalten, werden alle Versionen prinzipiell in eigenen Verzeichnissen gehalten.
 - Die aktive Version wird in die aktiven Verzeichnisse (bin, lib, etc) hart gelinkt.
 - Versionierung im Dateinamen ist nur nötig, wenn verschiedene Versionen gleichzeitig benutzt werden sollen. (Libraries)
- MacPorts benutzt „compile on demand“
 - Es wird zuerst versucht ein Binärarchiv zu laden und zu installieren
 - Sonst wird das Sourcearchiv geladen und der MacPort durchgeführt
 - dann wird ein lokales Binärarchiv erzeugt und installiert
 - das Binärarchiv kann dann zurück auf den Server geladen werden
 - es hat alle Optionen/Versionen im Namen codiert

MacPort Sourcearchiv

- Ein Sourcearchiv besteht im wesentlichen aus drei Teilen
 - dem Portfile
 - ist eine deklarative Beschreibung der SW und der Source
 - enthält Versions- und Abhängigkeitsinformation
 - enthält Basis Konfigurations- und Compilationsinformationen
 - enthält Informationen über die Sourcecodequelle
 - dem Patchset
 - eine Anzahl von Patchdateien, die die Änderungen im Code darstellen, die nötig sind, um die SW auf dem Zielsystem (hier Darwin) zu compilieren und laufen zu lassen
 - dem upstream Source
 - Dies ist typischerweise ein vom Entwicklerteam bereitgestelltes Sourcearchiv (tarball)
 - Wenn die Information im Portfile auf eine öffentliche VCS-verwaltete Revision verweist, kann dies auch leer sein. Der Source wird dann „on demand“ aus dem VCS (git, cvs, svn, hg usw.) ausgecheckt.
 - ggf. wird dabei dann ein tarball erstellt, um eine lokale Kopie unabhängig vom öffentlichen VCS zu haben.

MacPort Prozeß

- **fetch**
 - Fetch the `${distfiles}` from `${master_sites}` and place it in `${prefix}/var/macports/distfiles/${name}`.
- **checksum**
 - Compare `${checksums}` from Portfile to the ones of the fetched `${distfiles}`.
- **extract**
 - Unzip and untar the `${distfiles}` into the path `${prefix}/var/macports/build/<package>/work`
- **patch**
 - Apply optional patch files from `${patchfiles}` to modify a port's source code.
- **configure**
 - Execute the command `configure` in `${worksrcpath}`.
- **build**
 - Execute `${build.cmd}` in `${worksrcpath}`.
- **test**
 - Execute commands to run test suites bundled with a port.
- **destroot**
 - Execute the command `make install DESTDIR=${destroot}` in `${worksrcpath}`.

Portfile

■ PortSystem	1.0	BOILERPLATE
■ PortGroup	github 1.0	USE GITHUP MACRO
■ github.setup	zeromq jzmq 1.0.0 v	PARAMS FOR GITHUP MACRO
■ version	2.1.0	SOFTWARE VERSION
■ revision	1	PORT VERSION/REVISION
■ categories	sysutils devel net	EINSORTIERUNG
■ license	LGPL-3+	
■ platforms	darwin	
■ maintainers	openmaintainer email.de:Dr.Martin.Froehlich	
■ description	This package contains the Java Bindings for ZMQ	
■ long_description	\${description} \ The ZMQ ...	
■ homepage	http://www.zeromq.org/	
■ depends_lib	port:zmq22	ABHÄNGIGKEITEN
■ checksums	rmd160 24ddd549560b7d485e51b273345c4a4f08bef9ca	
■ use_parallel_build	no	SERIELLES MAKE
■ use_configure	yes	RUFE GNU/CONFIGURE AUF
■ universal_variant	yes	VERSUCHE FÜR MEHRERE ARCHITEKTUREN ZU BAUEN

Nicht nur für Mac

- Theoretisch auch für andere Plattformen geeignet
 - Es gibt keinen Port für Port
 - selber machen ;-)
- Hilfreiche Eigenschaften
 - Server URL frei wählbar, auch „file://“-URL möglich
 - Funktionierende Abhängigkeitsverwaltung
 - Versionierung problemarm
 - Install/uninstall funktioniert
 - Jederzeit Überblick über installierte und aktive Pakete und Versionen
 - einfache Paketierung
 - Sehr einfach bei Autotool-Code, da install etc. „von Haus aus“ vorhanden
 - Ansonsten moderater Aufwand für eigene Prozeduren im Portfile
- Weniger hilfreiche Eigenschaften
 - Ist in TCL geschrieben ;-)
 - Muß portiert werden, da einige Mac-ischkeiten eingebaut
 - Nicht gemacht um zentral viele Klienten zu verwalten (workaround einfach)
Portierungsversuch scheint mir der Mühen Wert zu sein!

Mac Ports: FARTHER THAN MAC!



FRAGEN?