Wie bekomme ich mein Projekt in eine Linux-Distribution?

Chemnitzer Linux-Tage 2018, Chemnitz

Robert Scheck

fedora™

Robert Scheck

- Fedora Package Maintainer (etwa 120 Pakete)
- Fedora Provenpackager und Packager Sponsor
- Fedora Ambassador und Ambassador Mentor
- Open Source Contributor und Software-Entwickler

- Mail: robert@fedoraproject.org
- Web: https://fedoraproject.org/wiki/RobertScheck

Erste Gedanken…

- Möchte ich das wirklich? ☺
- Welche Relevanz hat mein Projekt für andere?
- Bin ich vielleicht der einzige Benutzer?
- Ist es ein Hindernis der einzige Benutzer zu sein?

Was bedeutet das eigentlich?

- Software wird bzw. ist paketiert
  - Meist Binärpakete: RPM, DEB, Pacman
  - Eventuell: Flatpak, Container, Appliance
- Leichte Installation über Paketverwaltung
- Spielregeln der Linux-Distribution(en) relevant
- (Zusammen-) Arbeit für und zwischen
  - Upstream
  - Downstream

Auch Software fließt abwärts!

- Upstream
  - Entwickler – mit allem was dazu gehört
- Downstream
  - Paketbetreuer bzw. Linux-Distributionen
  - Endbenutzer (je nach Sichtweise)
- Aber: Upstream kann auch Downstream sein

Wann ist man „drin“?

- Standard-Repository der Distribution
  - Entwicklungszweig
  - Reguläres Release
- Optionales Repository der Distribution
  - „non-free“ oder „freeworld“
- Dritt-Repository
  - z.B. Fedora EPEL (für RHEL oder CentOS)

Will man „drin“ sein?

- Kann Upstream eventuell nicht beeinflussen
  - Kernkomponenten sind „immer“ drin
  - Spielregeln der Linux-Distribution relevant
- Dritt-Repository
  - Ermöglicht gewisse Freiheiten
  - Schafft eventuell neue Probleme
  - Gewisse Hürde für Benutzer

Rechtliche Themen

- Für Linux-Distribution akzeptable FLOSS-Lizenz
  - https://fedoraproject.org/wiki/Licensing:Main
- Software-Patente können Probleme bereiten
  - MP3 unterlag bis 2017 Software-Patenten
  - Microsofts ActiveSync-Protokoll ist patentiert
- Name der Software ist ein Markenzeichen
  - Firefox → Veränderungen → Iceweasel
  - Markenzeichen-Verwässerung
- Rechtliche Heimat (EU vs. USA)

Umfang und Komplexität

- Kleines einfaches Software-Projekt
  - Schnell paketiert und nebenbei pflegbar
- Riesiges komplexes Software-Projekt
  - Benötigt Zeit und Geduld beim Paketbau
  - Modulare Paketierung (z.B. Unterpakete)
  - Verschiedene Backends (z.B. MariaDB/SQLite)
  - Eventuell exotische Programmiersprache
  - Umfangreiche Abhängigkeiten
  - Neue Architekturen (ARM64)

Welche Linux-Distributionen?

- Zielgruppe der eigenen Software?
  - Desktop-Benutzer, Server-Betrieb, Entwickler
- Zielgruppe der Linux-Distributionen?
  - Linux Mint oder Ubuntu eher auf Desktops
  - RHEL/CentOS oder SLES eher auf Servern
  - Fedora auf Desktops, eventuell Container
- Produktlebenszyklen mit Software vergleichen
- Ausnahmen bestätigen die Regel

Wie kommt man nun „rein“?

- Selbst „einfach“ Paketbetreuer werden
  - Wissen zu Paketbau aufbauen, z.B. RPM
  - Mehrere Pakete erforderlich; je nach Distribution
- Distributionen oder Paketbetreuer kontaktieren
  - Leute für die Software inhaltlich begeistern
  - Bier kann auch motivieren und begeistern ☺
- Perfekte Paketbetreuer benutzt die Software selbst
- Wunschliste der Distribution
- Bezahlter Paketbau (für Firmen)

Abhängigkeiten

- Eigene Software benötigt zwingend Python 2, Distribution hat aber nur Python 3
  - Entscheidung: Software anpassen? Aufgeben?
  - Wie lange pflegt man Kompatibilitäten?
- Benötigte Abhängigkeiten sind nicht paketiert
  - Paketieren? Alternativen unterstützen?
  - Bündelung von Drittsoftware vermeiden

Kompatibilität

- Möglichst große Kompatibilität zu Abhängigkeiten
  - Unterstützung von Bibliothek X nur in Version Y zu unterstützen hilft keinem Paketbetreuer
  - Wenn möglich API-Inkompatibilitäten bei den Abhängigkeiten abfedern, z.B. bei Bibliotheken
- Datenbank-Backends zur Laufzeit konfigurierbar machen, nicht nur zur Buildtime
- Verschiedene Datenbank-Backends sollten sich nicht gegenseitig ausschließen

Veränderungen

- Kompatibilität vs. Weiterentwicklung der Software
- Rewrite der eigenen Software, z.B. C → Ruby
  - Abhängigkeiten zu anderer Software prüfen
  - Paketbetreuer frühzeitig miteinbeziehen
- Änderung der Software-Lizenz
  - Meist nicht trivial und mit Auswirkungen
- Auch Distribution kann sich (drastisch) verändern
- Notfalls: Paket entfernen lassen

Up-/Downstream-Probleme

- Upstream mag SysVinit, Downstream hat systemd
  - Schlecht: Upstream distanziert sich
  - Gut: Upstream toleriert systemd-Anpassungen
  - Besser: Upstream öffnet sich für neues
- Upstream ist guter Entwickler & schlechter Admin
  - Paketbetreuer ist zwischen Entwickler & Admin
  - Upstream sollte FHS und LSB im Blick haben
- Up- und Downstream müssen sich nicht mögen
  - Hilft allerdings sehr ☺

Sicherheitslücken

- Up- und Downstream haben Verantwortung
  - Sicherheitslücken vertuschen hilft nicht
- Coordinated Disclosure
  - Immer empfehlenswert
  - Upstream informiert Downstream frühzeitig
- Ideal: Patch oder Update vorab vom Upstream
  - Downstream kann vorab in Ruhe testen
  - Perfekt: Patch *und* Update
- Kommunikation

Über den Tellerrand schauen

- Unterstützung für z.B.
  - IPv6 (zusätzlich zu „Legacy IP“)
  - Internationalisierte Domains (IDNA2008)
  - Server Name Indication (SNI)
  - UTF-8 bzw. Unicode
  - SELinux bzw. AppArmor
- Gute Patches an Drittsoftware-Upstreams senden anstatt geforkte Drittsoftware bündeln

Kommunikation

- Up- und Downstream sind Profis in ihrem Gebiet
  - Gegenseitiger Austausch extrem wichtig
  - Bedenken/Sorgen beidseitig berücksichtigen
  - Störrischkeit ablegen: Kompromisse treffen
  - Verantwortung gegenüber Benutzern
- Konferenzen ermöglichen persönlichen Dialog
  - Manches diskutiert sich bei einem Bier leichter als im IRC oder per E-Mail

Pflege einer Software

- Major-/Minor-Updates vs. Distributionslifecycle
- Inkompatibilitäten bei Minor-Updates
  - Bitte vermeiden!
- Umgang mit Datenbank-Änderungen (Schema)
  - Möglichst robust und automatisierbar
  - Endbenutzer können Versionen auslassen
- Konfigurationsänderungen
  - Neue Parameter bitte optional
- Changelog und Release Notes

Guter Upstream sein…

- E-Mails vom Downstream zeitnah beantworten
- Bugreports bearbeiten
  - Gemeldete Probleme wirklich beheben!
  - Ja, neue Features machen mehr Spaß…
- Upstream könnte Paketbetreuer benachrichtigen
  - Neue Version
  - Für Paketpflege relevante Änderungen
- Testsuite oder Regression Tests
- Features für nur Downstreams

…guter Upstream sein

- Nicht immer das Rad neu erfinden
  - `autoconf`/`automake` oder `cmake` statt eigenem unflexiblem und nicht portablem Wirrwar
  - Funktionierendes `./configure`
  - Übersteuerung von `$CFLAGS` oder `$LDFLAGS` durch Distribution ermöglichen
  - Distribution möchte `/usr/bin` statt hardcoded `/usr/local/bin` o.ä.
  - Unterstützung für `$DESTDIR`

Mehrwerte wenn man „drin“ ist

- Linux-Distributionen bauen gegen viel mehr unterschiedliche Versionen als man selbst testet
  - Neuerer Compiler wirft neue Warnungen/Fehler
  - Buildlogs normalerweise öffentlich einsehbar
- Software wird für „exotische“ Plattformen gebaut
  - Zugriffsmöglichkeiten auf andere Plattformen
- Anzahl der Benutzer vergrößert sich üblicherweise
- Automatisierte Rebuilds bei ABI-Veränderungen bei Abhängigkeiten, z.B. soname

Projekt oder Paket einstellen

- Verantwortung gegenüber Benutzern
- Benutzer haben es trotzdem weiterhin installiert
- Idealerweise schrittweise Abkündigung
  - Beschränkung z.B. auf Fehlerkorrekturen
  - Weitere Beschränkung auf Sicherheitsupdates
  - Paketbetreuer kann Einstellung vielleicht mit Releasezyklus der Distribution synchronisieren
- Benutzer nicht vergessen

Aus 15 Jahren Paketbau…

- Veränderung der Interessen der Leute
  - Up- oder Downstream „verschwindet“
- Persönlicher Lebenswandel
  - Student → Familie, Berufswechsel
- Todesfälle
  - Upstream-Entwickler stirbt – was nun?
- Inkompatibilitäten bei FLOSS-Lizenzen
  - OpenSSL vs. GPL (-Exceptions)
- Upstream forkt Drittsoftware

Angewendet auf Fedora/EPEL

- Account im Fedora Account System (FAS)
- RPM-Paket aus der Software bauen/versuchen
  - https://fedoraproject.org/wiki/Packaging:Guidelines
- Eintrag im Bugzilla für Package Review erstellen
- Fedora Packager Sponsor (Mentor) suchen
- Package Review erfolgreich abschließen
- Git-Repo für Paket beantragen (und einchecken)
- Buildsystem für Paket triggern
- Paket ins Repo pushen lassen

Fragen?

fedora™

Vielen Dank!