redhat.

CREATE A FEDORA PACKAGE FROM YOUR RUST PROJECT

Introduction to Rust, RPM packaging and work of Fedora Rust SIG.

Martin Sehnoutka

April 2017

PURPOSE OF THIS PRESENTATION

- Introduce Rust programming language and its ecosystem
- Introduce RPM packages and Fedora ecosystem
- How to bring them together

RUST PROGRAMMING LANGUAGE

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

- Memory safe, threads without data races
- Compiled, statically typed
- Without garbage collector
- Mix of programming paradigms

RUST EXAMPLES

MEMORY SAFETY

- fat pointers, no undefined behavior
- C:

```c
int *a = (int *)malloc(10*sizeof(int));
int b = a[20];  // Run just fine :)
a[30] = 5;      // Until you use -fsanitize=address
```

- Rust:

```rust
let mut vec = vec![0u8; 10];
let a = vec[20];
```

SUM TYPES

- Say bye to tagged unions
- Heavily used in Rust
- No more integers as a return value
- You cannot unwrap the value without checking its type

```
pub enum Result<T, E> {
  Ok(T),
  Err(E),
}
```

GENERIC PROGRAMMING AND POLYMORPHISM

- No conventional OOP with classes and inheritance
- Only structs and traits (like typeclasses in Haskell)

```
trait Hash {
    fn hash(&self) -> u64;
}

impl Hash for bool {
    fn hash(&self) -> u64 {
        if *self { 0 } else { 1 }
    }
}
```

RUST'S ECOSYSTEM

CARGO

- Package manager
- Project defined in the file ”Cargo.toml”
- Can specify conditional compilation, dependencies, build targets, licenses etc.
- Automatically downloads dependencies to make working with Rust projects easier

```
$ cargo new hello-rust --bin
$ cd hello-rust
$ cargo run
$ cargo help
```

CARGO, CRATES, CRATES.IO

- Libraries distributed as a source code
- Public repository, anyone can publish

```
[package]
authors = ["Martin Sehnoutka <msehnout@redhat.com>"]
name = "netcat"
version = "0.1.0"

[dependencies]
clap = "2.20.5"
```

RPM PACKAGE MANAGER

HOW DOES IT WORK?

- Software is distributed in form of packages and stored in repositories
- We take: source code from the Internet
- We write: a spec file = the recipe for creating a package
- We produce: Source RPM (SRPM) and binary RPM
- We sign RPM packages with GPG key

```
%global crate hello-rust

Name:           rust-%{crate}
Version:        0.1.0
Release:        1%{?dist}
Summary:        # FIXME

License:        None
URL:            https://crates.io/crates/hello-rust
Source0:        https://crates.io/api/v1/crates/%{crate}/%{version}/download#/%{crate}-%{version}.crate

ExclusiveArch:  %{rust_arches}

BuildRequires:  rust
BuildRequires:  cargo

%prep
%autosetup -n %{crate}-%{version} -p1
%cargo_prep

%build
%cargo_build

%install
%cargo_install

%files       -n %{crate}
%{_bindir}/hello-rust

%changelog
* Tue Mar 21 2017 Martin Sehnoutka <msehnout@redhat.com> - 0.1.0-1
- Initial package
```

RULES FOR CREATING A PACKAGE (IN FEDORA)

- No bundling
- Build using SW already available in Fedora repositories (no internet connection during build)
- Quite complex topic: https://fedoraproject.org/wiki/Packaging:Guidelines

RUST SIG

RUST SIG

- Tool for automatic generation of spec files (rust2rpm)
- RPM macros for building Cargo projects

```
%prep
%autosetup -n %{crate}-%{version} -p1
%cargo_prep

%build
%cargo_build

%install
%cargo_install
```

- rustc/cargo packages in Fedora
- Packaging guidelines
- Improve support in upstream

BUILD YOUR OWN PACKAGE

INSTALL TOOLS AND DEPENDENCIES

```
# Install tools:
$ sudo dnf install rust cargo rust-gdb
$ sudo dnf install rpm-build rpmdevtools
$ sudo dnf copr enable @rpm-software-management/with-rich-dependency
$ sudo dnf update
$ sudo curl https://fedorapeople.org/groups/rust/repos/rust-sig.repo
-o /etc/yum.repos.d/rust-sig.repo
$ sudo dnf install rust2rpm rust-{,s}rpm-macros
# Prepare packaging environment:
$ rpmdev-setuptree
# rpmdev-wipetree
```

BUILD YOUR PACKAGE

```
# Configure git with your user name
# and email (~/.gitconfig)
$ cd ~/rpmbuild/SPECS
$ rust2rpm <your-project-name> # e.g. duplicate-kriller
$ spectool -g -R rust-duplicate-kriller.spec
$ sudo dnf builddep rust-duplicate-kriller.spec
$ rpmbuild -ba rust-duplicate-kriller.spec
# Result should be in ~/rpmbuild/RPMS/
# You can check it with mc
```

FUTURE WORK

WHAT CAN BE DONE?

In RPM world:

- Get RPM packages into RHEL/CentOS

In Rust world:

- Rewrite unsecure C libraries in Rust (gstreamer)
- Extend existing ones using FFI (GTK+, libsolv)
- Write completely new software (Trust-DNS)

QUESTIONS?

THANK YOU!