DOING NIX FOR SEL4

Towards more Infrastructure-as-Code

Developing for/with seL4 by the book

To tinker with seL4, one might (1/2)

Commands

```
apt-get update
apt-get install build-essential

apt-get install cmake ccache ninja-build cmake-curses-gui
apt-get install libxml2-utils ncurses-dev
apt-get install curl git doxygen device-tree-compiler
apt-get install u-boot-tools
apt-get install python3-dev python3-pip python-is-python3
apt-get install protobuf-compiler python3-protobuf

apt-get install qemu-system-arm qemu-system-x86 qemu-system-misc

apt-get install gcc-arm-linux-gnueabi g++-arm-linux-gnueabi
apt-get install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu

apt-get install gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf
```

Comments

- Instructions are for Ubuntu 20.04 and 22.04
- *"As dependencies and packages may be frequently changed, deprecated or updated these instructions may become out of date."*

Taken from `https://docs.sel4.systems/projects/buildsystem/host-dependencies.html` on (2024-10-10)

To tinker with seL4, one might (2/2)

Commands

```
apt-get updategit clone \
    https://github.com/riscv/riscv-gnu-toolchain.git
cd riscv-gnu-toolchain
git submodule update --init --recursive
export RISCV=/opt/riscv
./configure --prefix="${RISCV}" --enable-multilib
make linux

apt-get install texlive texlive-latex-extra texlive-fonts-extra

pip3 install --user setuptools
pip3 install --user sel4-deps
pip3 install --user camkes-deps

curl -sSL https://get.haskellstack.org/ | sh # OR
apt-get install haskell-stack
```

Comments

- Also install Google's repo tool `https://gerrit.googlesource.com/git-repo#install`
- For the proofs, there are also instructions for Debian Bullseye

Taken from `https://docs.sel4.systems/projects/buildsystem/host-dependencies.html` on (2024-10-10)

Observations

- Specific OS releases are recommended (Ubuntu 20.04 or 22.04)
- Manual asks to report missing dependencies
  - Hints at only informal tracking of dependencies
- Involves various package managers:
  - Ubuntu's package manager `apt-get`
  - Python's package manager `pip`
  - Haskell's package manager `stack`
  - Git + make + `...` to compile RISC-V cross-compiler yourself
  - `curl ... | sh` to install Haskell toolchain

What about Docker/OCI-Container?

- Docker helps!
- But:
  - Not easily reproducible
    - `RUN curl https://example.org/latest > /bin/my-app`
  - Heavy (often with an entire distro's user-land)
  - More of a deployment, less of a package manager tool
  - Only hides the mechanisms previously mentioned
    - Internally `apt-get`, `pip`, `stack`, `curl ... | sh` still have to play together

Question: Can we do better?

What could *better* mean?

- A few possible metrics:
  - Works on any Linux distro
  - Single command development environment
  - Single command builds
  - Multiple toolchains, collision free
  - Reproducible
  - Avoid hidden dependencies by design
  - No 2+ package managers installing `$stuff` to the same system
  - Overridable & Composable
  - Versioned in a repo

Alternatives

- Our suggestion: **Nix**
- Honorable mention: **Microkit**
  - Significantly improves upon this (when developing for the pre-defined platforms)

A Brief introduction to Nix

Key Points

- Functional programming applied to package-management
- ~20 years of history
- Decent, extensive cross compilation infrastructure
- Sound caching mechanism
- Extensively documented
- Severely under-documented
- One-stop solution even for complex builds
- Main activity in the `github:NixOS/nixpkgs` repo:
  - ~3.8M LoC, ~3.1 M LoC in **Nix**
  - ~7.1k contributors
  - ~660k commits
  - ~98k packages

What does Nix actually do?

- Fetch sources
- Fetch dependencies (toolchain, runtime libs, . . .)
- Apply patches (if any)
- Run configure/build/install script, for example using `Make`
- Fix outputs (i.e. repair rpath, shebangs, . . .)
- All that in a hermetically sealed build environment
  - ✗ `curl https://xyz.com -o assets.tar`
  - ✓ Downloaded resources are pinpointed via hash
  - ✗ Dependency satisfied by `apt install x` two years ago
  - ✓ Only specified dependencies are exposed
- Results stored in a content addressable, shareable cache

Terms

- **Nix**: Dynamically typed, functional, interpreted PL
- **Derivation**: A build recipe, generated from evaluating a **Nix** expression
- **Realisation**: Result of an executed **Derivation**
- **Nix Store**: Directory with all **Realisations**, usually `/nix/store`
- **nixpkgs**: Large set of **Nix** expressions to compile various FLOSS
- **NixOS**: Linux distribution built on top of **nixpkgs**
- **Flake**: Bundle of **Nix** expressions, in standardized form

Workflow

Example Nix Expression

```
{ lib, buildPythonPackage, fetchFromGitHub }:

buildPythonPackage rec {
  pname = "pyfdt";
  version = "0.3";
  src = fetchFromGitHub {
    owner = "superna9999";
    repo = pname;
    rev = "${pname}-${version}";
    hash = "sha256-lt/Mcw3j1aTBVOVhDBSYtriDyzeJHcSli69EXLfsgDM=";
  };

  meta = with lib; {
    description = "Python Flattened Device Tree Library";
    homepage = "https://github.com/superna9999/pyfdt";
    license = with licenses; [ asl20 ];
    maintainers = with maintainers; [ wucke13 ];
  };
}
```

Corresponding Example Derivation

```
nix derivation show github:DLR-FT/seL4-nix-utils#pyfdt
```


```
{
  "/nix/store/hh95y9dkwy08impvql532cgff6zdjrc5-python3.11-pyfdt-0.3.drv": {
    "args": [
      "-e",
      "/nix/store/v6x3cs394jgqfbi0a42pam708flxaphh-default-builder.sh"
    ],
    "builder": "/nix/store/r9h133c9m8f6jnlsqzwf89zg9w0w78s8-bash-5.2-p15/bin/bash",
    "env": {
      "LANG": "C.UTF-8",
      "builder": "/nix/store/r9h133c9m8f6jnlsqzwf89zg9w0w78s8-bash-5.2-p15/bin/bash",
      "dist": "/nix/store/k8yhv923zaqxvkb5x4q4s7nfwydi7wd1-python3.11-pyfdt-0.3-dist",
      "doInstallCheck": "1",
      "name": "python3.11-pyfdt-0.3",
      "nativeBuildInputs": "/nix/store/yvhwsfbh4bc99vfvwpaa70m4yng4pvpz-python3-3.11.8 /nix/store/
      "out": "/nix/store/54l257flzxniqd7vmn49c6rzansshy2k-python3.11-pyfdt-0.3",
      "pname": "pyfdt",
      "postFixup": "wrapPythonPrograms\n",
      "propagatedBuildInputs": "/nix/store/yvhwsfbh4bc99vfvwpaa70m4yng4pvpz-python3-3.11.8",
      "propagatedNativeBuildInputs": "",
      "src": "/nix/store/gzyi76bxfqg7qfr9m57imqk4nz5v52lx-source",
      "stdenv": "/nix/store/10i1kiic5azin1gp6418r8bc1bqvgc90-stdenv-linux"
```

Applying Nix to seL4's Ecosystem

The `seL4-nix-utils` contain

Nix expressions covering

- microkit-sdk
- seL4-camkes-vm-examples-{aarch64,armv7l}
- seL4-kernel-{arm,riscv64,x64}
- seL4-test-{aarch64,armv7l,i686,x86_64}
- arm-trusted-firmware-zynqmp
- capDL-tool
- pmufw-mblaze-zcu102
- U-Boot

Soon™ to come:

- Linux kernel + userland for Microkit/seL4 VMs
- CAmkES VMM examples on RISCV

Which enables

- One-command to
  - Realize artifacts (see left box)
  - Enter dev environment for interactive work
- Declarative description of
  - Dependencies
  - Build steps to generate the artifacts
- Reproducible build environments
- Composable, hackable, overridable
  - Build your Nix expressions on top of ours!

Use Case: Build and Develop (for) seL4

Play the Microkit tutorial

```
curl -L trustworthy.systems/Downloads/microkit_tutorial/tutorial.tar.gz -o tutorial.tar.gz
tar xf tutorial.tar.gz
nix develop github:DLR-FT/seL4-nix-utils#microkit
```

Build & run seL4-test for x86_64-linux in QEMU

```
nix build github:DLR-FT/seL4-nix-utils#seL4-test-x86_64-x86_64-simulate
cd result && ./simulate
```

Build U-Boot for the zcu102 platform, patched to enable EL2

```
nix build github:DLR-FT/seL4-nix-utils#uboot-aarch64-zcu102
```

Get interactive DevShell for zynq7000, build it

```
nix develop github:DLR-FT/seL4-nix-utils#seL4-test-armv7l-zynq7000-simulate
unpackPhase && cd source # extract source code to new dir, cd there
configurePhase # configure the build dir, target etc.
ninja # compile
```

Use-Case: Retrieve full SBOM

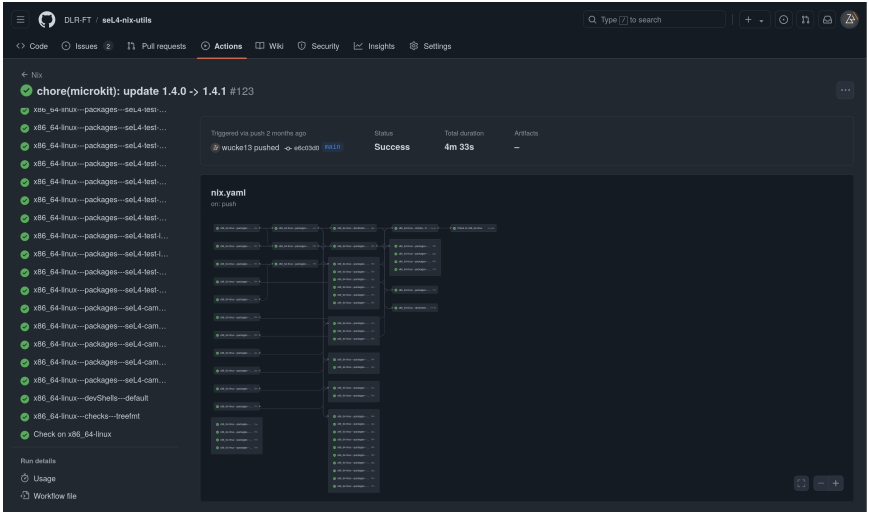
True Software Bill Of Materials (SBOM) becomes visible

```
# show run-time dependencies
nix-store --query --requisites $(nix eval --raw .\#microkit-sdk)

# show build-time dependencies, also supports graphml or dot output
nix-store --query --requisites $(nix eval --raw .\#microkit-sdk.drvPath)

# interactive tree view
nix-tree $(nix eval --raw .\#microkit-sdk.drvPath)
```

Example output

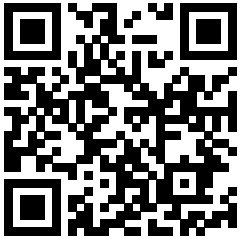
```
/nix/store/9jwix1rc0nggrv2w2pcaiv9sfvr3wj9q-microkit-sdk-1.4.1.drv
├───/nix/store/v6x3cs394jgqfbi0a42pam708flxaphh-default-builder.sh
├───/nix/store/hpkl2vyxiwf7rwvjh9lpij7swp7igilx-bash-5.2-p15.drv
│   ├───/nix/store/ks6kir3vky8mb8zqpfhchwasn0rv1ix6-bootstrap-tools.drv
│   │   ├───/nix/store/b7irlwi2wjlx5aj1dghx4c8k3ax6m56q-busybox.drv
│   │   ├───/nix/store/bzq60ip2z5xgi7jk6jgdw8cngfiwjrcm-bootstrap-tools.tar.xz.drv
│   │   └───/nix/store/i9nx0dp1khrgikqr95ryy2jkigr4c5yv-unpack-bootstrap-tools.sh
│   ├───/nix/store/v6x3cs394jgqfbi0a42pam708flxaphh-default-builder.sh [...]
```

Use-Case: Easy CI



Thank you!

- Thank you for listening!
- Questions?
  - Ask now, or find me later today & tomorrow!
- Problems?
  - Please open up an issue on GitHub!
- Find our work over at `https://github.com/DLR-FT/seL4-nix-utils`



Imprint

Topic:	**Doing Nix for seL4** Towards more Infrastructure-as-Code
Date:	2024-10-16. Copyright © 2024 by Deutsches Zentrum für Luft- und Raumfahrt e. V.
Author:	Wanja Zaeske
Institute:	Institute of Flight Systems
Credits:	All images „DLR (CC BY-NC-ND 3.0)“