RISC-V support

Goal: install lix on a riscv64-linux system

The target is a DevTerm R-01, so it's an AllWinner D1 RISC-V processor @ 1GHz, with 1GB of memory and 32GB of microSD.

We can't run the Lix installer without building it, because there's no canned build for it. So let's try building it natively:

```
$ rustup
$ git clone https://git.lix.systems/lix-project/lix-installer
$ cd lix-installer
$ RUSTFLAGS="--cfg tokio_unstable" cargo install --path .
```

This doesn't work because there's some conditional complication^Wcompilation that doesn't cover riscv64. So we need to open `self_test.rs` and add an entry:

```
    #[cfg(all(target_os = "linux", target_arch = "riscv64"))]
    const SYSTEM: &str = "riscv64-linux";
```

At this point, it will, in principle, build. In practice, however, 1GB is just not enough RAM. If you add some swap it'll make it to the last step, but then it wants 1.5GB+ for that. I wouldn't try it on a system with less than 2GB, and ideally more.

Ok, native build is a bust unless I want to let it thrash all night. So let's cross-compile it on `ancilla`, which is, conveniently, already running nixos.

The nix-installer flake doesn't come with riscv64 cross support, and rather than try to figure it out I just winged it with nix-shell. I am skipping over a lot of false starts and blind alleys here as I ran into things like dependency crates needing a cross-compiling gcc, or rust not having a stdlib on riscv64-musl.

```
$ git clone https://git.lix.systems/lix-project/lix-installer
$ cd lix-installer
$ $EDITOR shell.nix
with import <nixpkgs> {
  crossSystem.config = "riscv64-unknown-linux-gnu";
};
mkShell {
  nativeBuildInputs = with import <unstable> {}; [ cargo rustup ];
}
```

```
$ nix-shell
[long wait for gcc to compile]

$ export RUSTUP_HOME=$PWD/.rustup-home
$ export CARGO_HOME=$PWD/.cargo-home
$ rustup default stable
$ rustup target add riscv64gc-unknown-linux-gnu
$ edit src/self_test.rs
[apply that same patch to SYSTEM]
```

The build invocation is a bit more complicated here, because we need to tell it where to find the linker:

```
$ RUSTFLAGS="--cfg tokio_unstable" cargo build \
    --target riscv64gc-unknown-linux-gnu \
    --config target.riscv64gc-unknown-linux-gnu.linker='"riscv64-unknown-linux-gnu-gcc"'
[another long wait]

$ file target/riscv64gc-unknown-linux-gnu/debug/lix-installer
target/riscv64gc-unknown-linux-gnu/debug/lix-installer:
  ELF 64-bit LSB pie executable, UCB RISC-V, RVC, double-float ABI,
  version 1 (SYSV), dynamically linked,
  interpreter /nix/store/g4xam7gr35sziib1zc033xvn1vy9gg8m-glibc-riscv64-unknown-linux-gnu-
2.38-44/lib/ld-linux-riscv64-lp64d.so.1,
  for GNU/Linux 4.15.0, with debug_info, not stripped
```

Since we couldn't do a static musl build it needs the nix ld.so, but we can get around that!

```
$ scp target/riscv64gc-unknown-linux-gnu/debug/lix-installer root@riscv:.
$ ssh root@riscv
# ./lix-installer
-bash: ./lix-installer: no such file or directory

# ldd ./lix-installer
/nix/store/.../ld-linux-riscv64-lp64d.so.1 => /lib/ld-linux-riscv64-lp64d.so.1
[other output elided]

# /lib/ld-linux-riscv64-lp64d.so.1 ./lix-installer
The Determinate Nix installer (lix variant)
```

```
[...]
```

Sadly we can't actually use it to install, because `nix_package_url` needs a default value, and on RISC-V, it doesn't have one! It's `self_test.rs` all over again except it doesn't manifest until runtime.

So, off to `src/settings.rs` we go. It doesn't need to be a valid URL, just something URL-shaped.

```
/// Default [`nix_package_url`](CommonSettings::nix_package_url) for unknown platforms
pub const NIX_UNKNOWN_PLATFORM_URL: &str =
    "https://releases.lix.systems/unknown-platform";
```

```
    #[cfg_attr(
        all(target_os = "linux", target_arch = "riscv64", feature = "cli"),
        clap(
            default_value = NIX_UNKNOWN_PLATFORM_URL,
        )
    )]
```

Rebuild, re-push, re-run:

```
# /lib/ld-linux-riscv64-lp64d.so.1 /opt/lix-installer install linux
Error:
   0: Planner error
   1: `nix-installer` does not support the `riscv64gc-unknown-linux-gnu` architecture right
now
```

Ok, missed a few places in settings.rs, let's put a quick and dirty hack in there:

```
            #[cfg(target_os = "linux")]
            (_, OperatingSystem::Linux) => {
                url = NIX_UNKNOWN_PLATFORM_URL;
                nix_build_user_prefix = "nixbld";
                nix_build_user_id_base = 30000;
                nix_build_user_count = 32;
            },
```

```
            #[cfg(target_os = "linux")]
            (_, OperatingSystem::Linux) => {
                (InitSystem::Systemd, linux_detect_systemd_started().await)
            },
```

It also needs a tarball to install; jade_ kindly updated the flake for it to support riscv64, so we just check it out (or, well, check out review branch 1444) and then `nix build -L .#nix-riscv64-linux.binaryTarball` and away we go.

This, it turns out, also doesn't work, because the installer is hardcoded to expect the directory the tarball contains to start with `nix-*`. You can either unpack and repack the tarball to meet that requirement, or find all the places in lix-installer that assume that and edit them -- they're in `src/action/base/move_unpacked_nix.rs` and `src/action/base/setup_default_profile.rs`.

Finally, this particular kernel lacks `seccomp` support -- in order to get it working, I had to edit the lix (not lix-installer) `package.nix` and add `(lib.mesonEnable "seccomp-sandboxing" false)` to the meson flags.

And with that done, it works!

```
root@devterm-R01:~# uname -a && nix --version
Linux devterm-R01 5.4.61 #12 PREEMPT Wed Mar 30 14:44:22 CST 2022 riscv64 riscv64 riscv64
GNU/Linux
nix (Lix, like Nix) 2.90.0pre20240613_dirty
```

Revision #2
Created 2024-06-14 08:26:53 UTC by Becca
Updated 2024-06-15 04:30:29 UTC by Becca