Fuchsia

Fuchsia

Pink + Purple == Fuchsia (a new Operating System)

Welcome to Fuchsia! This document has everything you need to get started with Fuchsia.

NOTE: The Fuchsia source includes Zircon, the core platform that underpins Fuchsia. The Fuchsia build process will build Zircon as a side-effect; to work on Zircon only, read and follow Zircon's Getting Started doc.

Contents

Prerequisites

Prepare your build environment (Once per build environment)

Ubuntu

```
sudo apt-get install texinfo libglib2.0-dev liblz4-tool autoconf libtool libsdl-dev bu
```

macOS

1. Install the Xcode Command Line Tools:

```
xcode-select --install
```

1. In addition to the Xcode Command Line tools, you also need to install a recent version of the full Xcode. Download Xcode from https://developer.apple.com/xcode/.
2. Install the other pre-reqs:

- Using Homebrew:

```
# Install Homebrew
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/mast
# Install packages
brew install wget pkg-config glib autoconf automake libtool golang
```

- Using MacPorts:

```
# Install MacPorts
# See https://guide.macports.org/chunked/installing.macports.html
port install autoconf automake libtool libpixman pkgconfig glib2
```

[Googlers only] Goma

Ensure `goma` is installed on your machine for faster builds.

Build Fuchsia

Get the source

Follow the instructions to get the Fuchsia source and then return to this document.

[Googlers only] CIPD auth-login

Run

```
./buildtools/cipd auth-login
jiri run-hooks  # Re-run hooks now that you're logged in.
```

You should only need to do this once per host.

Build

If you added `.jiri_root/bin` to your path as part of getting the source code, the `fx` command should already be in your path. If not, the command is also available as `scripts/fx`.

```
fx set x64
fx full-build
```

The first command selects the build configuration you wish to build and generates the build system itself in an output directory (e.g., `out/debug-x64`).

The second command actually executes the build, transforming the source code in build products. If you modify the source tree, you can do an incremental build by re-running the `fx full-build` command alone.

Alternatively, you can use the underlying build system directly.

[optional] Customize Build Environment

By default you will get a x64 debug build. You can skip this section unless you want something else.

Run `fset-usage` to see a list of build options. Some examples:

```
fx set x64                # x64 debug build
fx set arm64              # arm64 debug build
fx set x64 --release      # x64 release build
```

[optional] Accelerate builds with `ccache` and `goma`

`ccache` accelerates builds by caching artifacts from previous builds. `ccache` is enabled automatically if the `CCACHE_DIR` environment variable is set and refers to a directory that exists.

[Googlers only: `goma` accelerates builds by distributing compilation across many machines. If you have `goma` installed in `~/goma`, it is used by default. It is also used by default in preference to `ccache`.]

To override the default behaviors, pass flags to `fx set`:

```
--ccache     # force use of ccache even if goma is available
--no-ccache  # disable use of ccache
--no-goma    # disable use of goma
```

Boot Fuchsia

Installing and booting from hardware

To get Fuchsia running on hardware requires using the paver, which these instructions will help you get up and running with.

Boot from QEMU

If you don't have the supported hardware, you can run Fuchsia under emulation using QEMU. Fuchsia includes prebuilt binaries for QEMU under `buildtools/qemu`.

The `fx run` command will launch Zircon within QEMU, using the locally built disk image:

```
fx run
```

There are various flags for `fx run` to control QEMU's configuration:

- `-m` sets QEMU's memory size in MB.
- `-g` enables graphics (see below).
- `-N` enables networking (see below).

Use `fx run -h` to see all available options.

QEMU tips

- `ctrl+a x` will exit QEMU in text mode.
- `ctrl+a ?` or `ctrl+a h` prints all supported commands.

Enabling Graphics

Note: Graphics under QEMU are extremely limited due to a lack of Vulkan support. Only the Zircon UI renders.

To enable graphics under QEMU, add the `-g` flag to `fx run`:

```
fx run -g
```

Enabling Network

First, configure a virtual interface for QEMU's use.

Once this is done you can add the `-N` and `-u` flags to `fx run`:

```
fx run -N -u $FUCHSIA_SCRIPTS_DIR/start-dhcp-server.sh
```

The `-u` flag runs a script that sets up a local DHCP server and NAT to configure the IPv4 interface and routing.

Explore Fuchsia

When Fuchsia has booted and displays the "$" shell prompt, you can run programs!

For example, to receive deep wisdom, run:

```
fortune
```

To shutdown or reboot Fuchsia, use the `dm` command:

```
dm help
dm shutdown
```

Select a tab

Fuchsia shows multiple tabs after booting with graphics enabled. The currently selected tab is highlighted in yellow at the top of the screen. You can switch to the next tab using Alt-Tab on the keyboard.

- Tab zero is the console and displays the boot and component log.
- Tabs 1, 2 and 3 contain shells.
- Tabs 4 and higher contain components you've launched.

Note: to select tabs, you may need to enter "console mode". See the next section for details.

Launch a graphical component

QEMU does not support Vulkan and therefore cannot run our graphics stack.

Most graphical components in Fuchsia use the Mozart system compositor. You can launch such components, commonly found in `/system/apps`, like this:

```
launch spinning_square_view
```

Source code for Mozart example apps is here.

When you launch something that uses Mozart, uses hardware-accelerated graphics, or if you build the default package (which will boot into the Fuchsia System UI), Fuchsia will enter "graphics mode", which will not display any of the text shells. In order to use the text shell, you will need to enter "console mode" by pressing Alt-Escape. In console mode, Alt-Tab will have the behavior described in the previous section, and pressing Alt-Escape again will take you back to the graphical shell.

If you would like to use a text shell inside a terminal emulator from within the graphical shell you can launch the term by selecting the "Ask Anything" box and typing `moterm`.

Running tests

Compiled test binaries are installed in `/system/test/`. You can run a test by invoking it in the terminal. E.g.

```
/system/test/ledger_unittests
```

If you want to leave Fuchsia running and recompile and re-run a test, run Fuchsia with networking enabled in one terminal, then in another terminal, run:

```
fx run-test <test name> [<test args>]
```

You may wish to peruse the testing FAQ.

Contribute changes

- See CONTRIBUTING.md.

Additional helpful documents

- Fuchsia documentation hub
- Working with Zircon - copying files, network booting, log viewing, and more
- Information on the system bootstrap component.