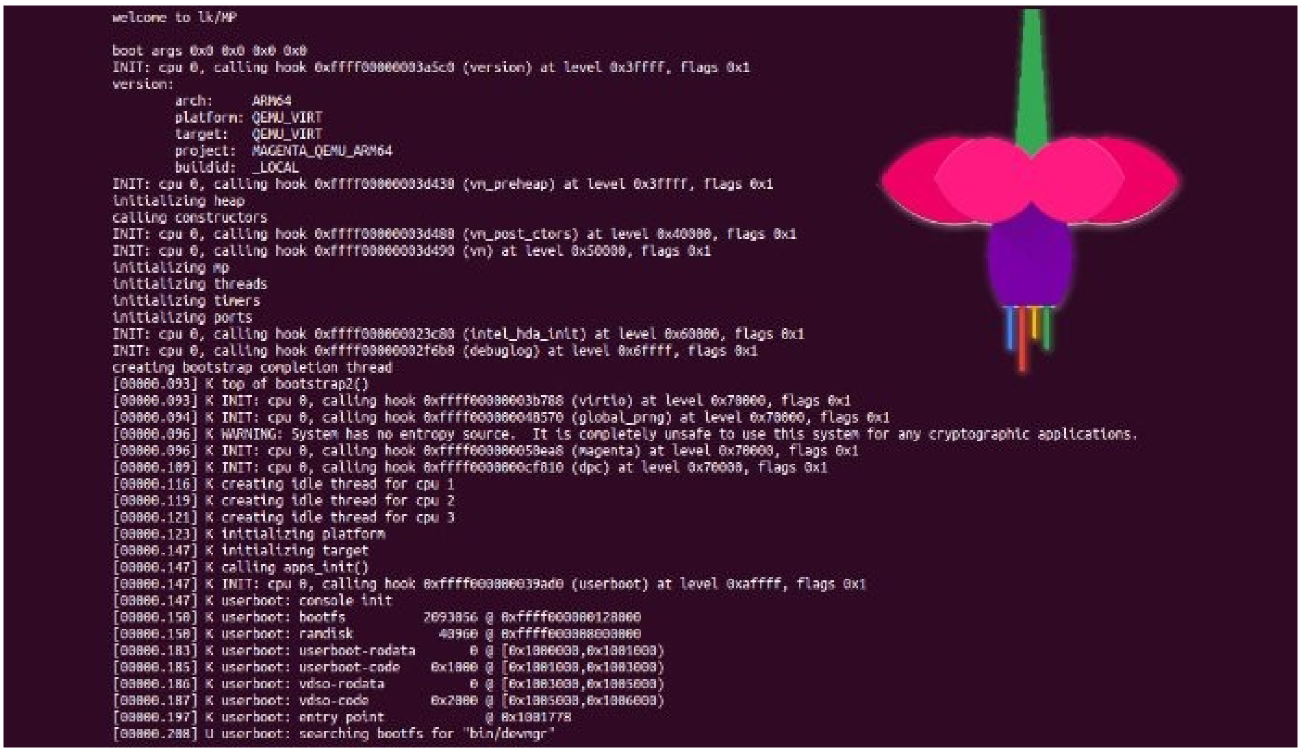
Google Fuchsia Embedded OS

linkedin.com/pulse/google-fuchsia-embedded-os-roberto-a-foglietta



```
welcome to lk/MP

boot args 0x0 0x0 0x0 0x0
INIT: cpu 0, calling hook 0xffff00000003a5c0 (version) at level 0x3ffff, flags 0x1
version:
        arch:     ARM64
        platform: QEMU_VIRT
        target:   QEMU_VIRT
        project:  MAGENTA_QEMU_ARM64
        buildid:  _LOCAL
INIT: cpu 0, calling hook 0xffff00000003d430 (vm_preheap) at level 0x3ffff, flags 0x1
initializing heap
calling constructors
INIT: cpu 0, calling hook 0xffff00000003d480 (vm_post_ctors) at level 0x40000, flags 0x1
INIT: cpu 0, calling hook 0xffff00000003d490 (vm) at level 0x50000, flags 0x1
initializing mp
initializing threads
initializing timers
initializing ports
INIT: cpu 0, calling hook 0xffff000000023c80 (intel_hda_init) at level 0x60000, flags 0x1
INIT: cpu 0, calling hook 0xffff00000002f6b0 (debuglog) at level 0x6ffff, flags 0x1
creating bootstrap completion thread
[00000.093] K top of bootstrap2()
[00000.093] K INIT: cpu 0, calling hook 0xffff00000003b788 (virtio) at level 0x70000, flags 0x1
[00000.094] K INIT: cpu 0, calling hook 0xffff000000048570 (global_prng) at level 0x70000, flags 0x1
[00000.096] K WARNING: System has no entropy source.  It is completely unsafe to use this system for any cryptographic applications.
[00000.096] K INIT: cpu 0, calling hook 0xffff00000050ea8 (magenta) at level 0x70000, flags 0x1
[00000.109] K INIT: cpu 0, calling hook 0xffff0000000cf810 (dpc) at level 0x70000, flags 0x1
[00000.116] K creating idle thread for cpu 1
[00000.119] K creating idle thread for cpu 2
[00000.121] K creating idle thread for cpu 3
[00000.123] K initializing platform
[00000.147] K initializing target
[00000.147] K calling apps_init()
[00000.147] K INIT: cpu 0, calling hook 0xffff000000039ad0 (userboot) at level 0xaffff, flags 0x1
[00000.147] K userboot: console init
[00000.150] K userboot: bootfs          2093056 @ 0xffff000000128000
[00000.150] K userboot: ramdisk           40960 @ 0xffff000000003000
[00000.183] K userboot: userboot-rodata        0 @ [0x1000000,0x1001000)
[00000.185] K userboot: userboot-code    0x1000 @ [0x1001000,0x1003000)
[00000.186] K userboot: vdso-rodata           0 @ [0x1003000,0x1005000)
[00000.187] K userboot: vdso-code        0x2000 @ [0x1005000,0x1006000)
[00000.197] K userboot: entry point             @ 0x1001778
[00000.208] U userboot: searching bootfs for "bin/devmgr"
```

Published on August 17, 2016

MAGENTA - HOW TO RUN YOUR OWN C SOURCE CODE BINARY

On August 15th, Google unveiled a new operative embedded system called Fuchsia based on Magenta kernel. Here below you will find a step by step guide to compile and run this new kernel under qemu. Moreover a manual tool-chain configuration in order to compile your own C-source example code.

The target architecture chosen is Arm64 CPU type arm926ej-s and the host is Ubuntu Linux 14.04 x86-64. The list of package in the following may not be fully exhaustive, especially if you install Ubuntu from scratch.

## *Compiling and providing the buildtools*

```
sudo apt-get install git libc-dev-bin gcc-multilib curl

mkdir -p fuchsia

pushd fuchsia

git clone https://fuchsia.googlesource.com/buildtools

git clone https://fuchsia.googlesource.com/magenta

pushd buildtools/

./update.sh

popd
```

## *Compiling and providing the tool-chain for ARM target*

## *Available architectures: arm i386 aarch64 x86_64*

```
sudo apt-get install texinfo libglib2.0-dev autoconf libsdl-dev \

build-essential bison flex

TARGET_ARCH="aarch64"

declare -i N_CPUS=$(nproc)
```

```
SYSROOT=$PWD/buildtools/sysroot

git clone https://fuchsia.googlesource.com/third_party/gcc_none_toolchains

ln -sf gcc_none_toolchains toolchain

pushd toolchain

./doit -a $TARGET_ARCH -f -j$N_CPUS

popd

TCBIN_PATH=$(readlink -f $PWD/toolchain/$TARGET_ARCH-*/bin)

export PATH=$PATH:$TCBIN_PATH
```

Building Magenta for pc-x86-64 target

Available build targets: magenta-pc-x86-64 magenta-qemu-arm32

magenta-qemu-arm64 pc-x86-64-test pc-x86-test qemu-virt-a15-test

qemu-virt-a53-test rpi3-test

```
pushd magenta

make -j$N_CPUS magenta-qemu-arm64

popd
```

Compile qemu and make Magenta run with it

```
git clone https://fuchsia.googlesource.com/third_party/qemu

pushd qemu

git checkout fuchsia

./configure --target-list=$TARGET_ARCH-softmmu \

--prefix=$PWD/../qemu-runtime

make -j$N_CPUS install

export PATH=$PWD/../qemu-runtime/bin:$PATH

popd

pushd magenta/build-magenta-qemu-arm64

declare -i REAL_MEM_KB QEMU_MEM_MB N_CPUS_QEMU

N_CPUS_QEMU=$[(N_CPUS+1)/2]

REAL_MEM_KB=$(grep -e ^MemTotal: /proc/meminfo | tr -cd [0-9])

QEMU_MEM_MB=$[($REAL_MEM_KB+1024)/2048]

test $QEMU_MEM_MB -gt 512 && QEMU_MEM_MB=512

qemu-system-$TARGET_ARCH -m $QEMU_MEM_MB -nographic \

-machine virt -cpu cortex-a53 -kernel magenta.elf -append ''

#> ls /boot/bin

#> core-tests
#> thread-depth-test

#> dlog

# to terminate qemu

CTRL-A X

popd
```

## Manual preparation of the current toolchain

```
export CC=aarch64-elf-gcc

export CXX=aarch64-elf-g++

export LD=aarch64-elf-ld.gold

export AR=aarch64-elf-ar

export AS=aarch64-elf-as

export NM=aarch64-elf-nm

export STRIP=aarch64-elf-strip

export RANLIB=aarch64-elf-ranlib

export DLLTOOL=aarch64-elf-dlltool

export OBJDUMP=aarch64-elf-objdump

export RESCOMP=aarch64-elf-windres

export WINDRES=aarch64-elf-windres

CFLAGS="-Wall -Wextra -ffunction-sections -fdata-sections -fPIC -mcpu=cortex-a53 -std=c11"

CFLAGS="$CFLAGS -include config-global.h -include config-user.h"

CFLAGS="$CFLAGS $(for i in $(find ../global ../system ../third_party -name include); do echo -n "-I $i "; done)"

export CFLAGS

export HOSTING_CRT0=./ulib/crt1.o

LIB_PATH_1=$(dirname $TCBIN_PATH)/lib/gcc/$TARGET_ARCH-elf/5.3.0

LDFLAGS="-s -nostdlib -Lkernel -Lsystem -Lthird_party -z max-page-size=4096 --gc-sections -z combreloc -z relro -z now -z text --hash-style=gnu --eh-frame-hdr --build-id -pie -dynamic-linker ld.so.1 $HOSTING_CRT0"

export LDFLAGS

EXTRA_LIBS="$(find ./ulib/ -name \*.so.abi) $LIB_PATH_1/libgcc.a"
```

## Compile and run your own code

```
cp -f ../third_party/uapp/kilo/kilo.c chilo.c

wget -c roberto.foglietta.name/pub/mk/armstrong.c

rm -f chilo.o chilo armstrong.o armstrong extra.bootfs

$CC $CFLAGS -c armstrong.c -o armstrong.o

$CC $CFLAGS -c chilo.c -o chilo.o -fPIC -Wno-unused-parameter
```

```
$LD $LDFLAGS armstrong.o $EXTRA_LIBS -o armstrong

$LD $LDFLAGS chilo.o $EXTRA_LIBS -o chilo
```

Remake the extra.bootfs initrd image

```
file chilo armstrong

echo "bin/armstrong=./armstrong" > extra.manifest

echo "bin/chilo=./chilo" >> extra.manifest

../../buildtools/mkbootfs -o extra.bootfs extra.manifest

sync
```

Run a new qemu instance of Magenta with the new initrd image

```
qemu-system-$TARGET_ARCH -m $QEMU_MEM_MB -nographic \

-machine virt -cpu cortex-a53 -kernel magenta.elf -initrd extra.bootfs -append ''
```
