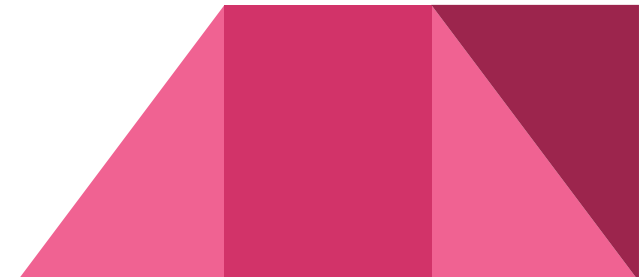


# OpenWrt for embedded development

Ohio LinuxFest - September 30, 2017

# About Me

- Linux user since 1998
  - Software Engineering
  - Systems Administration
  - Network Engineering
  - Security
- Worked at Battelle Memorial Institute for 4 years
  - Software Information Engineering group
  - CBRNE Defense group
  - Health & Analytics group
  - Energy group
- Lead Software Architect at Armada Power
  - Spun out of Battelle in 2015

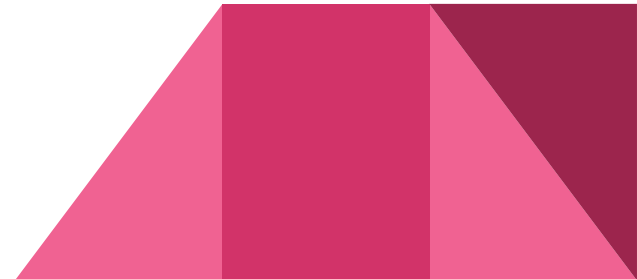


# What I'm going to talk about

- History of the project
- Setting up your development environment
- Building images and packages
- Flashing images and installing packages
- Debugging

Not covered: porting OpenWrt to new platforms

Slides and sample files: <http://xaox.net/olf>



# Project History

- Linksys WRT54G Router Linux source released
- DD-WRT, Tomato, and OpenWrt projects begin
- January 2006, OpenWrt 0.9 (White Russian)
- June 2007, OpenWrt 7.06 (Kamikaze)
- Current stable release was released in March 2016, OpenWrt 15.05.1 (Chaos Calmer)
- May 2016, frustrated with project process, a group of developers fork and start the LEDE project (Linux Embedded Development Environment)
- June 2017 the LEDE developers vote to merge with the OpenWrt project
- September 30, 2017 OLF speaker lifts OpenWrt history and picture from Wikipedia.



# Why OpenWrt?

- You want an open OS to upgrade your home router
- You are building the next awesome IoT device
- You are replacing the awful, buggy, insecure firmware on someone else's IoT device
- You think that 32MB of RAM ought to be enough for anybody

## **Best if used for systems with...**

- System on chip (SoC) hardware
- Limited RAM
- Limited non-wear leveled flash (NAND/NOR)
- No display

# OpenWrt system tour

- Flash Partitions
  - “Partition Table” built into the kernel
  - Overlay file system
- /tmp and /var are RAM disks
- Busybox for userspace
- Dropbear for SSH server
- UCI for configuration
- Luci for a router web interface

Bootloader (u-boot) 128kB
Kernel 1280kB
“ROM” SquashFS 1536kB
“Overlay” JFFS2 5184kB
“Art” 64kB (not mounted)

Example from TP-Link WR1043ND

# Setting up your development environment

Instructions here: <https://wiki.openwrt.org/doc/howto/buildroot.exigence>

- Suggest using Debian
- `apt-get install git-core build-essential libssl-dev libncurses5-dev unzip gawk zlib1g-dev`
- (optional) `apt-get install subversion mercurial`
- `git clone https://git.lede-project.org/source.git`

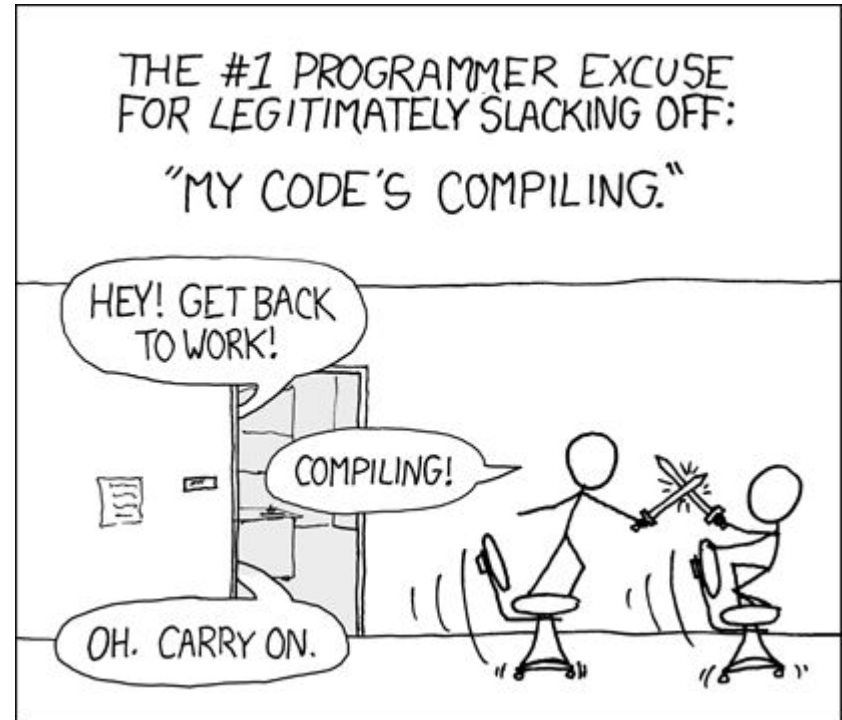
That gives you a bare bones system. More than likely you will want other libraries:

- `cd openwrt`  
`./scripts/feeds update -a`  
`./scripts/feeds search <for what your heart desires>`  
`./scripts/feeds install <what makes you happy>`

# Building it!

<https://wiki.openwrt.org/doc/howto/build>

- make menuconfig
  - Set the "Target System"
  - Set the "Subtarget"
  - Set the "Target Profile"
  - Choose packages to build / include
- make
- Read a few XKCDs
- Build output is in bin/targets/<platform>
  - Firmware image
  - "packages" directory





# Demo - Building OpenWrt

# A program to package

Our killer IoT app (helloworld.c):

```
#include <stdio.h>

int main() {
    printf("Hello Embedded World!\n");
    return 0;
}
```

Need a Makefile too:

```
helloworld: helloworld.o
    $(CC) $(LDFLAGS) $? -o $@

clean:
    rm *.o helloworld
```

# OpenWrt Packaging Magic

package/utils/helloworld:

Makefile (eye chart on right)

src/helloworld.c (previous slide)

src/Makefile (previous slide)

```
include $(TOPDIR)/rules.mk

# Name and release number of this package
PKG_NAME:=helloworld
PKG_VERSION:=1.0.0
PKG_RELEASE:=1
PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/package.mk

define Package/helloworld
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=Helloworld -- prints an awesome message
endef

define Package/helloworld/description
    Test program for testing OpenWrt build system
endef

define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
    $(MAKE) -C $(PKG_BUILD_DIR)
$(TARGET_CONFIGURE_OPTS)
endef

define Package/helloworld/install
    $(INSTALL_DIR) $(1)/bin
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/helloworld
$(1)/bin/
endef

$(eval $(call BuildPackage,helloworld))
```

# Package Makefile - stuff on top

```
include $(TOPDIR)/rules.mk

# Name and release number of this package
PKG_NAME:=helloworld
PKG_VERSION:=1.0.0
PKG_RELEASE:=1
PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/package.mk
```

# Package Makefile - package info

```
define Package/helloworld
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=Hello world -- prints an awesome message
endef
```

```
define Package/helloworld/description
    Test program for testing OpenWrt build system
endef
```

# Package Makefile - the real magic

```
define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
    $(MAKE) -C $(PKG_BUILD_DIR) $(TARGET_CONFIGURE_OPTS)
endef

define Package/helloworld/install
    $(INSTALL_DIR) $(1)/bin
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/helloworld $(1)/bin/
endef

$(eval $(call BuildPackage,helloworld))
```



Demo - Build the package

# What if it is a C++ application?

Using uClibc++:

```
include $(TOPDIR)/rules.mk

# Name and release number of this package
PKG_NAME:=helloucpp
PKG_VERSION:=1.0.0
PKG_RELEASE:=1
PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/uclibc++.mk
include $(INCLUDE_DIR)/package.mk

define Package/helloucpp
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=Helloucpp -- prints an awes...
    DEPENDS:=+uclibcxx
endef...
```

Using libstdc++:

```
...
define Package/hellocpp
    SECTION:=utils
    CATEGORY:=Utilities
    TITLE:=Hellocpp -- prints an awes...
    DEPENDS:=+libstdc++
endef
...
```



# Handy build commands:

Build just one package:

```
make package/helloworld/compile
```

Clean and build one package:

```
make package/helloworld/{clean,compile}
```

See all the build output:

```
make package/helloworld/compile V=s
```

Build with debug symbols:

```
make package/helloworld/{clean,compile} CONFIG_DEBUG=y
```

# Debugging?

<https://wiki.openwrt.org/doc/devel/gdb>

In menuconfig enable “Advanced configuration” -> “Toolchain Options” -> Build gdb

Core file:

- Target system: `ulimit -c unlimited`
- Build system: `./scripts/remote-gdb <corefile> build_dir/target-<build target>/<package dir>/<executable>`

Remote debugging:

- Enable / Deploy “Development” -> gdbserver to target
- Target side: `gdbserver :9000 <program to debug>`
- Build system: `./scripts/remote-gdb <target IP>:9000 build_dir/target-<build target>/<package dir>/<executable>`

# Thank You!

Slides and sample files: <http://xaox.net/olf>

Contact: [mikej@xaox.net](mailto:mikej@xaox.net)

Questions?

