Embedded GNU/Linux

Krzysztof Mazur

26 czerwca 2014

0.1 autoconf

```
$ ./configure --build=BUILD-TYPE \
--host=HOST-TYPE \
--target=TARGET-TYPE
```

Forma kanoniczna: *machine-os-release*

```
arm-linux-elf
arm-softfloat-none-eabi
armv5tel-softfloat-linux-gnueabi
i386-mingw32
i686-pc-linux-gnu
i686-pc-cygwin
alpha-netbsd
m68k-coff-gcc
```

0.2 Dodanie katalogu z bibliotekami

```
$ export TOPDIR=/home/user/grinn
$ export LD_LIBRARY_PATH="$(TOPDIR)/lib"
$ export LD_RUN_PATH="$(TOPDIR)/lib"
$ export LDFLAGS="-L$(TOPDIR)/lib"
$ export CFLAGS="$(CFLAGS) -I$(TOPDIR)/include"
$ export CXXFLAGS="$(CXXFLAGS) -I$(TOPDIR)/include"
$ export CPPFLAGS="$(CPPFLAGS) -I$(TOPDIR)/include"
$ export PKG_CONFIG_PATH="$(TOPDIR)/lib/pkgconfig" $ ./configure
```

1 GNU toolchain

1.1 binutils

```
$ wget ftp://ftp.gnu.org/gnu/ \
binutils/binutils-2.23.2.tar.bz2
$ wget ftp://ftp.gnu.org/gnu/ \
binutils/binutils-2.23.2.tar.bz2.sig
$ gpg --recv-keys A4E55E93
$ gpg --verify binutils-2.23.2.tar.bz2.sig
$ tar xfv binutils-2.23.2.tar.bz2.sig
$ export arm_sysroot=/ścieżka/do/gentoo
$ export arm_target=armv5tel-softfloat-linux-gnueabi
$ mkdir build-arm
$ ../buntils-2.23.2/configure --prefix=PREFIX
--with-build-sysroot="$arm_sysroot"
--with-sysroot="$arm_sysroot"
--target="armv5tel-softfloat-linux-gnueabi"
$ make -j5
$ make install
```

1.2 gcc

```
$ wget ftp://ftp.gnu.org/gnu/gcc/ \
gcc-4.7.4/gcc-4.7.4.tar.bz2
$ wget ftp://ftp.gnu.org/gnu/gcc/ \
gcc-4.7.4/gcc-4.7.4.tar.bz2.sig
$ gpg --recv-keys A4E55E93
$ gpg --verify gcc-4.7.4.tar.bz2
$ tar xfv gcc-4.7.4.tar.bz2
$ export arm_sysroot=/ścieżka/do/gentoo
$ export arm_target=armv5tel-softfloat-linux-gnueabi
$ mkdir gcc-arm
$ ../gcc-4.7.4/configure --prefix=PREFIX \
--target="$arm_target" \
--with-build-sysroot="$arm_sysroot" \
--with-sysroot="$arm_sysroot" \
--with-gnu-as --with-gnu-ld \
--disable-multilib --enable-nls \   --disable-libstdcxx-pch --enable-languages=c,c++
```

1.3 gdb

```
$ wget ftp://ftp.gnu.org/gnu/gdb/\
gdb-7.7.1.tar.bz2
$ wget ftp://ftp.gnu.org/gnu/gdb/\
gdb-7.7.1.tar.bz2.sig
$ tar xfv gdb-7.7.1.tar.bz2
$ export arm_target=armv5tel-softfloat-linux-gnueabi
$ mkdir gdb-arm
$ ../gdb-7.7.1/configure --prefix=PREFIX\
--target="$arm_target"
```

2 U-Boot

2.1 Konfiguracja U-Boot

1. Ustawić bootowanie BOOT na „SPI flash 0”.
2. Podłączyć się do portu szeregowego, włączyć płytkę i nacisnąć dowolny klawisz by przerwać automatyczne uruchamiane systemu prze uboota.
3. Ustawić adres IP używany przez U-Boota:

   ```
   U-Boot > set ipaddr 172.16.0.17
   ```

4. Ustawić adres IP serwera TFTP:

   ```
   U-Boot > set serverip 172.16.0.7
   ```

5. Ustawić argumenty kernela:

   ```
   U-Boot > set bootargs console=ttyS2,115200n8 mem=32M
   ```

6. Ustawić skrypt bootowania:

```
U-Boot > set bootcmd "sf probe 0; sf read 0xc0700000 0x80000 0x380000; bootm
0xc0700000"
```

7. Zapisać konfigurację ubboota do flasha:

```
U-Boot > saveenv
```

2.2 Ładowanie systemu z tftp

Poza ładowaniem firmware z pamięci flash U-Boot obsługuje także ładowanie firmware'u po sieci za pomocą protokołu TFTP. W tym celu należy odpowiednio ustawić adres serwera TFTP w konfiguracji U-Boot (zobacz. 2.1) oraz umieścić plik uImage na serwerze TFTP w głównym katalogu.

Następnie przy starcie U-Boota należy na konsoli nacisnąć jakiś przycisk by przerwać automatyczne bootowanie oraz wpisać następujące komendy:

```
U-Boot > tftpboot
U-Boot > bootm
```

3 Pobranie kluczy GPG

```
$ gpg --keyserver hkp://pgp.mit.edu --recv-keys B305467D 6092693E
```

4 Kompilacja narzędzi potrzebnych do kompilacji oprogramowania

4.1 Instalacja wymaganych pakietów w Ubuntu

```
$ sudo apt-get install libgmp-dev
$ sudo apt-get install libmpfr-dev
$ sudo apt-get install libmpc-dev
$ sudo apt-get install uboot-mkimage
```

4.2 Instalacja cross-kompilatorów

```
$ mkdir ~/omap
$ cd ~/omap
$ wget ftp://ftp.podlesie.net/pub/omap/omap-build-toolchain-0.0
$ wget ftp://ftp.podlesie.net/pub/omap/omap-build-toolchain-0.0.sig
$ gpg --verify omap-build-toolchain-0.0.sig
$ bash ./omap-build-toolchain-0.0
```

Skrypt ten instaluje cross-compilatory domyślnie do katalogu ~/omap. W tym przypadku przy kompilacji innych elementów trzeba ustawić

```
$ export PATH="$HOME/omap/bin:$PATH"
```

5 Kompilacja poszczególnych projektów

5.1 Linux

```
$ mkdir ~/omap/
$ cd ~/omap
$ wget ftp://ftp.kernel.org/pub/linux/kernel/v3.x/linux-3.6.10.tar.xz
$ xz -d linux-3.6.10.tar.xz
$ wget ftp://ftp.kernel.org/pub/linux/kernel/v3.x/linux-3.6.10.tar.sign
$ gpg --verify linux-3.6.10.tar.sign
$ tar xf linux-3.6.10.tar
$ wget ftp://ftp.podlesie.net/pub/omap/linux-omap.tar.gz
$ wget ftp://ftp.podlesie.net/pub/omap/linux-omap.tar.gz.sig
$ gpg --verify linux-omap.tar.gz.sig
$ cd linux-3.6.10
$ tar xfz ../linux-omap.tar.gz
$ patch -p1 < linux-omap/0001-*.patch
$ patch -p1 < linux-omap/0002-*.patch
$ make ARCH=arm O=linux-omap oldconfig
$ cd linux-omap
$ make ARCH=arm CROSS_COMPILE=armv4tl-softfloat-linux-gnueabi- uImage
```

Zmiana konfiguracji:

```
$ cd linux-omap
$ make ARCH=arm menuconfig
```

Instalacja:

```
$ scp arch/arm/boot/uImage root@plytka
$ ssh root@plytka fw-update uImage
```

6 Instalacja cross-compilatora

6.1 Pobieranie Gentoo

Pobieranie:

```
$ wget http://distfiles.gentoo.org/releases/arm/
autobuilds/current-stage3-armv5tel/
stage3-armv5tel-20140605.tar.bz2
$ wget http://distfiles.gentoo.org/releases/arm/
autobuilds/current-stage3-armv5tel/
stage3-armv5tel-20140605.tar.bz2.DIGESTS
$ wget http://distfiles.gentoo.org/releases/arm/
autobuilds/current-stage3-armv5tel/
stage3-armv5tel-20140605.tar.bz2.DIGESTS.asc
```

Weryfikacja:

```
$ gpg --recv-keys 2D182910
$ gpg --verify *.asc
```

Odpakowanie:

```
# tar xfvp stage3-armv5tel-20140605.tar.bz2
```

Na potrzeby budowy cross-compilatora wystarczy:

```
$ tar xfv stage3-armv5tel-20140605.tar.bz2
```

6.2 u-boot

```
$ git clone git://git.podlesie.net/u-boot.git
$ cd u-boot
$ git checkout km2-genesis
$ export PATH=PREFIX/bin:"$PATH"
$ make CROSS_COMPILE=$arm_target- distclean
$ make CROSS_COMPILE=$arm_target- chilisom_config
$ make CROSS_COMPILE=$arm_target- -j5
```

6.3 openocd

```
$ cat > a.cfg << EOF
jtag_rclk 50;
init;
ftdi_set_signal PWR_RST 1;
scan_chain;
EOF
$ openocd -f interface/ftdi/xds100v2.cfg \
-f board/ti_am335xevm.cfg -f a.cfg
```

6.4 openocd

```
$ armv5tel-softfloat-linux-gnueabi-gdb u-boot
> target remote localhost:3333
> monitor reset halt
> monitor arm mcr 15 0 1 0 0 0
> monitor reset halt
> load
> cont
```

```
> loadb 0x82000000
Przesłanie MLO
> nand erase 0x0 0x20000
> nand write 0x82000000 0x0 0x20000
```

```
> loadb 0x82000000
Przesłanie u-boot
> nand erase 0x80000 0x80000
> nand write 0x82000000 0x80000 0x80000
```

6.5 buildroot

```
$ git clone git://git.buildroot.net/buildroot
$ cd buildroot
$ make menuconfig
$ make
```

```
> fatload mmc 0:1 0x82000000 uImage
> ext2load mmc 0:2 0x82000000 /boot/uImage
```

6.5.1 Ładowanie systemu z NAND

```
> set bootargs console=ttyO0,115200n8 root=/dev/mtdblock8 rootfstype=jffs2 rw
mem=128M
> set bootcmd "nand read 0x82000000 0x280000 0x500000; bootm 0x82000000"
> setenv
```

6.5.2 Ładowanie systemu z MMC

```
> set bootargs console=ttyO0,115200n8 root=/dev/mmcblk0p2 rw mem=128M rootwait
> set bootcmd "fatload mmc 0:1 0x82000000 uImage; bootm 0x82000000"
> setenv
```