SoC Linux 道場【其ノ四】

GNU アプリケーションのコンパイル

Ver.13.1

2017 年 8 月 Rev.2

目次

1. はじめに

Busybox とは、主に組み込み Linux® で使われるオープン・ソースの“コマンド集”アプリケーションのことです。

通常、コンピュータ・プログラムにはそれぞれ個別のバイナリ・ファイル（実行ファイル）が存在します。Busybox は全体で 1 つのバイナリになっており、その中に多数のアプリケーションが含まれています。その実行ファイルは Linux 上で最小の実行ファイルとなるよう設計されています。これにより、各コマンドの実行ファイルを個別にインストールするのに比べ、ディスクの使用量を大幅に削減することができます。そのため、特定用途の Linux ディストリビューションや組み込み Linux システムに最適なアプリケーションとなっています。

httpd は、組み込み機器では比較的使用されている Web サーバ・ソフトです。

ユーザからの要求に応じてサーバ内に蓄積されたコンテンツを送信したり、CGI スクリプトを起動したりすることができます。

今回は、これらの GNU アプリケーションのコンパイルとインストールについて解説します。

尚、この資料の説明で使用している主な開発環境は以下の通りです。

【表 1.1】この資料の説明で使用している主な開発環境

項番	項目	内容
1	ホスト OS	Microsoft Windows 7 Professional sp1 日本語版（64 bit）
2	ゲスト OS	Vine Linux 6.5 x86_64 この資料では、Linux 開発環境として、Vine Linux ディストリビューションを使用します。 詳細については、SoC Linux 道場【其ノ弐】を参照ください。
3	仮想 OS 実行環境	OS を仮想的に実行するための環境です。 この説明では、「VMware Player for Windows」と呼ばれるフリーウェア・ソフトを使用しています。 詳細については、SoC Linux 道場【其ノ弐】を参照ください。
4	クロス・コンパイラ	ターゲット（Helio）向けの実行イメージを生成するためのコンパイラです。 この説明では、Linaro から提供されている 32-bit ARMv7 Cortex-A 向け Linux GNU クロス・ツールチェーンを使用しています。 詳細については、SoC Linux 道場【其ノ参】を参照ください。
5	BusyBox	主に組み込み Linux で使われるオープン・ソースの“コマンド集”アプリケーションです。 ■ BusyBox のダウンロード URL http://www.busybox.net/
6	thttpd	組み込み機器では比較的使用されている Web サーバ・ソフトです。 ■ thttpd のダウンロード URL http://acme.com/software/thttpd/

7	Helio ボード	動作確認でターゲット・ボードとして使用する、アルテラ Cyclone® V SoC を搭載した マクニカ Helio ボードです。 Helio には複数のリビジョンが存在しますが、この資料では、Rev1.2 または Rev1.3 を使用して 動作確認を行っています。 ■ Helio ボード Rev1.2 http://www.rocketboards.org/foswiki/Documentation/HelioResourcesForRev12 ■ Helio ボード Rev1.3 http://www.rocketboards.org/foswiki/Documentation/HelioResourcesForRev13

2. GNU アプリケーションのコンパイル

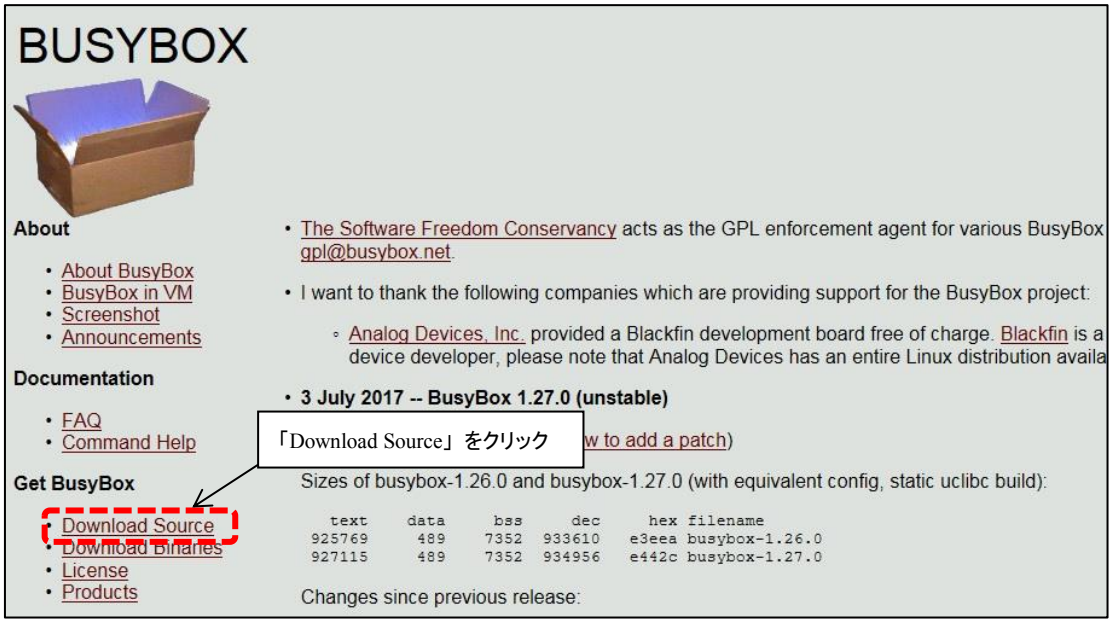
2-1. Busybox のクロス・コンパイルとインストール

Busybox を Vine Linux 上で使用するための手順を以下に説明します。

(1) 以下の Busybox のページに移動します。

http://www.busybox.net/

(2) 「Download Source」をクリックして、ダウンロード・ページに移動します。



【図 2-1.1】Busybox のサイト

(3) Busybox のソースをダウンロードします（この資料では「busybox-1.27.0.tar.bz2」を使用して動作確認を行っています）。

【図 2-1.2】Busybox のダウンロード

(4) ダウンロードできたら、このファイルを Samba サーバ経由で、Windows 上のエクスプローラから Vine Linux 上の一般ユーザ（この例では tori）のホーム・ディレクトリ（この例では /home/tori）に運びます。

【注記】

以降の説明は、SoC Linux 道場【其ノ弐】で説明した Linux マシン と Samba サーバが既に用意・設定されていて、使用できることを前提としています。

(5) **ホスト（Vine Linux）**でコマンド・ライン端末を起動して、一般ユーザのホーム・ディレクトリから tar コマンドを使用して、busybox のソースを /tmp ディレクトリに解凍します。

```
[tori@Vine65 ~]$ cd
[tori@Vine65 ~]$ tar jxvf busybox-1.27.0.tar.bz2 -C/tmp
```

(6) 以下のコマンドでクロス・コンパイラを設定し、busybox をコンフィギュレーションします。

```
[tori@Vine65 ~]$ cd /tmp/busybox-1.27.0
[tori@Vine65 busybox-1.27.0]$ make menuconfig
```

【注記】

make menuconfig を実行して、次のエラーが出て busybox のコンフィギュレーション画面が出ない場合は、下記 ①、② の手順を実行してから再度 make menuconfig を実行してください。

```
/usr/bin/ld: scripts/kconfig/lxdialog/checklist.o: シンボル 'stdscr' への未定義参照です
/lib64/libtinfo.so.5: error adding symbols: DSO missing from command line
collect2: エラー: ld はステータス 1 で終了しました
make[2]: *** [scripts/kconfig/lxdialog/lxdialog] エラー 1
make[1]: *** [menuconfig] エラー 2
make: *** [menuconfig] エラー 2。
```

① busybox-1.27.0/scripts/kconfig/lxdialog/Makefile をエディタで開いて以下の記述を追加します。

```
[tori@Vine65 busybox-1.27.0]$ leafpad scripts/kconfig/lxdialog/Makefile
```

```
HOST_EXTRACFLAGS += -DLOCALE -I/usr/include/ncurses   ← -I/usr/include/ncurses を追加
HOST_LOADLIBES   += -ltinfo -lncurses -L/usr/lib64/   ← この行を新規追加
```

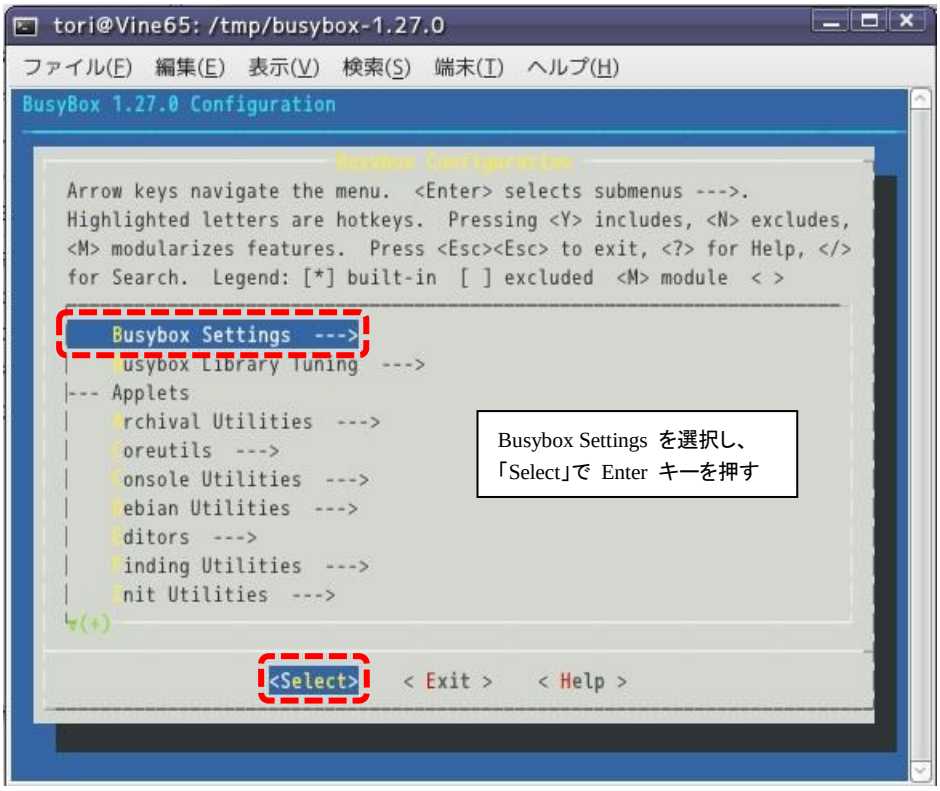
② busybox-1.27.0/scripts/kconfig/lxdialog/dialog.h ファイルをエディタで開き次のように変更します。

```
[tori@Vine65 busybox-1.27.0]$ leafpad scripts/kconfig/lxdialog/dialog.h
```

```
#include CURSES_LOC
          ↓
#include <ncurses.h>
```

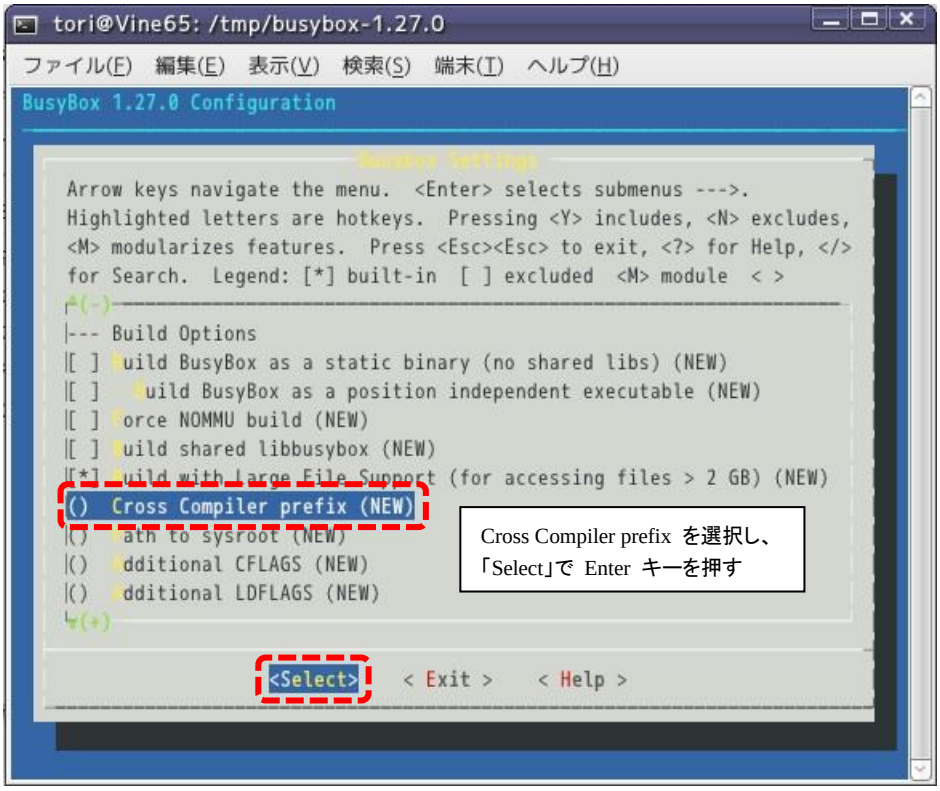
#include CURSES_LOC を **#include <ncurses.h> に変更**

(7) busybox のコンフィギュレーション画面が出たら、矢印キー(→／←／↓／↑)を使用して操作し、“Busybox Settings” を選択し、「Select」で Enter キーを押します。



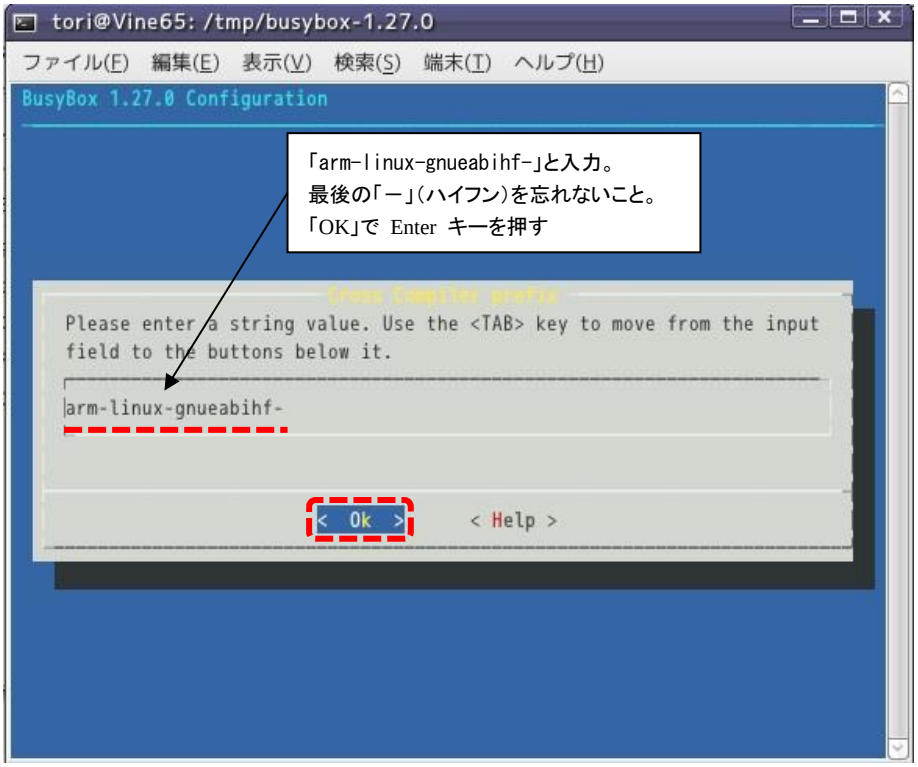
【図 2-1.3】“Busybox Settings” を選択

(8) “Cross Compiler prefix” を選択し、「Select」で Enter キーを押します。



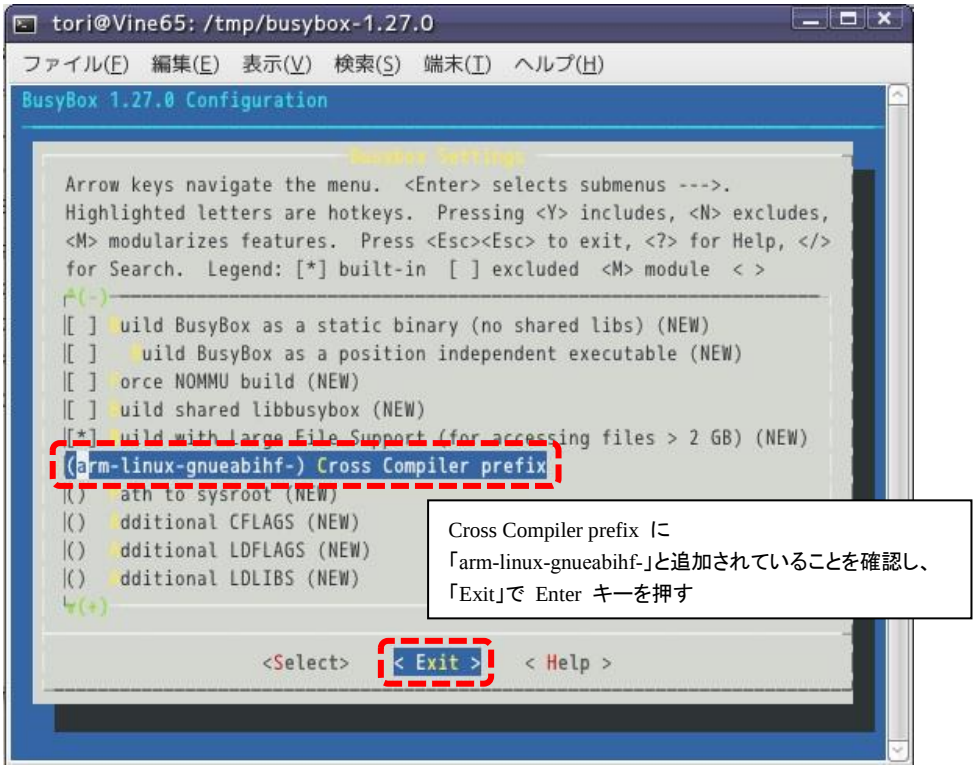
【図 2-1.4】“Cross Compiler prefix” を選択

(9) クロス・コンパイラを選択するためにテキスト入力欄に「arm-linux-gnueabihf-」と入力します。最後の「-」(ハイフン)を付けるのを忘れないでください。入力したら「OK」で Enter キーを押します。



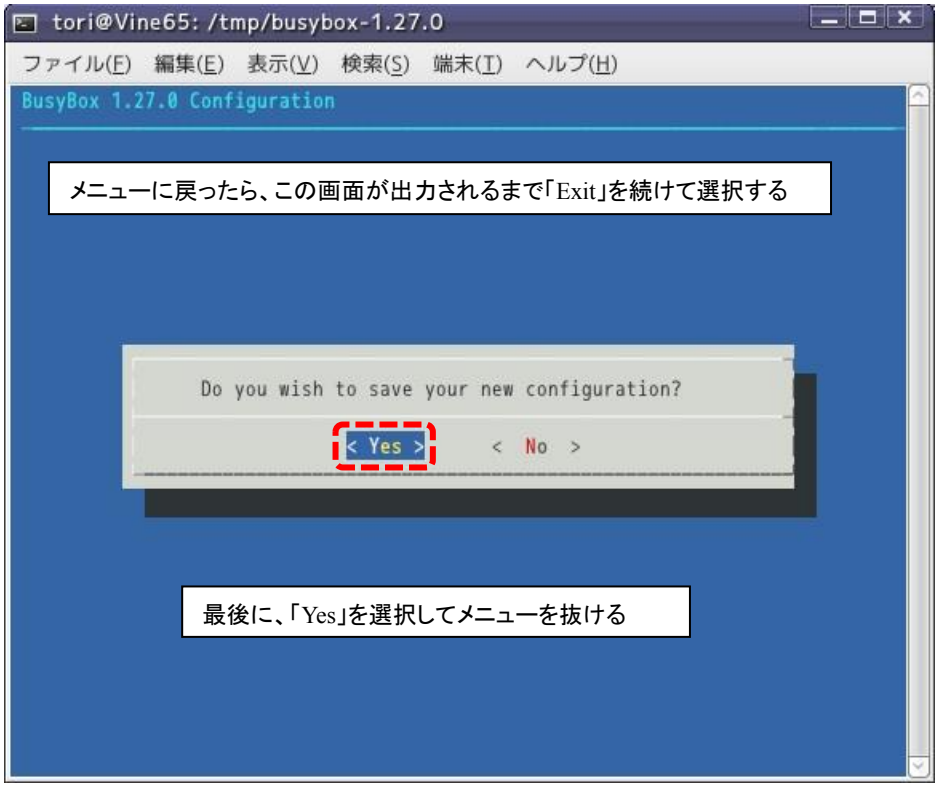
【図 2-1.5】クロス・コンパイラの選択

(10) メニュー表示に戻ったら、下図のように Cross Compiler prefix に「arm-linux-gnueabihf-」と追加されていることを確認し、「Exit」で Enter キーを押します。



【図 2-1.6】Cross Compiler prefix への「arm-linux-gnueabihf-」の追加

(11) 下図の画面が出力されるまで、続けて「Exit」を選択します。最後に、「Yes」を選択してメニューを抜けます。



【図 2-1.7】Busybox のメニューの終了

(12) 以下の make コマンドで busybox をクロス・コンパイルします。

```
[tori@Vine65 busybox-1.27.0]$ make
（コンパイルのログが出力）
```

(13) PC のスペックによりますが、クロス・コンパイル終了までには少々時間がかかります。クロス・コンパイルに成功したら、以下のコマンドで直下に生成されたファイルを確かめます。

```
[tori@Vine65 busybox-1.27.0]$ file busybox
busybox: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter
/lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32,
BuildID[sha1]=95b26672c3d28a7ef41e704c7a39d1a210c0acb0, stripped
```

(14) 次にターゲット側に FTP で転送するために、一般ユーザのホーム・ディレクトリ（この例では、/home/tori）の下に、でき上がった busybox をコピーします。

```
[tori@Vine65 busybox-1.27.0]$ cp busybox ~tori
```

(15) FTP サーバを使って busybox を Helio に運びます。以下のコマンドを **Helio 側** で実行します。

【注記】

以降の説明は、既に SoC Linux 道場【其ノ弐】で説明した Linux マシンが用意され、DHCP サーバ および FTP サーバ が設定されていて、使用できることを前提としています。

特に、DHCP サーバから Helio に対して IP アドレスが付与されていて、Helio と Vine Linux 間の通信が行える状態にしておいてください。

確認方法としては、Helio 側 から「ping 192.168.1.2」を実行して Vine Linux との接続を確認し、接続できない場合は、Helio 側 から「udhcpc」コマンドを実行して IP アドレスが付与されることを確認してから、再度 ping を試してみてください。ping で接続できたら下記のコマンドを実行してください。

```
root@socfpga:/# cd /tmp
root@socfpga:/tmp# wget ftp://tori:toriumi@192.168.1.2/busybox
--2013-11-05 06:07:14--  ftp://tori:*password*@192.168.1.2/busybox
           => 'busybox'
Connecting to 192.168.1.2:21... connected.
Logging in as tori ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> SIZE busybox ... 683388
==> PASV ... done.    ==> RETR busybox ... done.
Length: 683388 (667K) (unauthoritative)

100%[======================================>] 683,388      --.-K/s   in 0.03s

2013-11-05 06:07:25 (24.4 MB/s) - 'busybox' saved [683388]
```

(16) 以下のように、busybox に chmod コマンドで実行権限を与えてから実行します。

```
root@socfpga:/tmp# chmod 755 busybox
root@socfpga:/tmp# ./busybox ls /
bin         home        lost+found  proc        usr
boot        init        media       sbin        var
dev         lib         mnt         sys         www
etc         linuxrc     mount_dir   tmp
```

これで busybox コマンドを実行することに成功しました。

(17) SoC Linux 道場【其ノ弐】の 3-6 で、NFS のマウントがうまく行かなかったことを覚えているでしょうか？この新しい busybox コマンドを使ってマウントしてみましょう。

以下のようにコマンドを実行してマウントを試みます。

```
root@socfpga:/tmp# ./busybox mount -t nfs -o nolock 192.168.1.2:/opt/Helio /mount_dir/
```

これで NFS マウントできるようになりました。

(18) 今度は**ホスト側**で busybox コマンドを /opt/Helio にコピーして、Helio 上で動かせるようにしてみましょう。

```
[tori@Vine65 busybox-1.27.0]$ cp busybox /opt/Helio/
```

(19) 次に**ターゲット(Helio)側**で以下のように busybox コマンドを実行します。
「BusyBox v1.27.0」のバージョン番号が確認できれば OK です。
バージョン 1.27.0 の Busybox が生成できたことになります。

```
root@socfpga:/tmp# /mount_dir/busybox
BusyBox v1.27.0 (2017-07-08 01:33:41 JST) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.
```

```
Currently defined functions:
        [, [[, acpid, add-shell, addgroup, adduser, adjtimex, arp, arping, ash,
        awk, base64, basename, beep, blkdiscard, blkid, blockdev, bootchartd,
        brctl, bunzip2, bzcat, bzip2, cal, cat, chat, chattr, chgrp, chmod,
        chown, chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm,
        conspy, cp, cpio, crond, crontab, cryptpw, cttyhack, cut, date, dc, dd,
        deallocvt, delgroup, deluser, depmod, devmem, df, dhcprelay, diff,
        dirname, dmesg, dnsd, dnsdomainname, dos2unix, dpkg, dpkg-deb, du,
        dumpkmap, dumpleases, echo, ed, egrep, eject, env, envdir, envuidgid,
        ether-wake, expand, expr, factor, fakeidentd, fallocate, false,
        fatattr, fbset, fbsplash, fdflush, fdformat, fdisk, fgconsole, fgrep,
        find, findfs, flock, fold, free, freeramdisk, fsck, fsck.minix,
        fsfreeze, fstrim, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty,
        grep, groups, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid,
        hostname, httpd, hush, hwclock, i2cdetect, i2cdump, i2cget, i2cset, id,
        ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init, insmod,
        install, ionice, iostat, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink,
        ipneigh, iproute, iprule, iptunnel, kbd_mode, kill, killall, killall5,
        klogd, last, less, link, linux32, linux64, linuxrc, ln, loadfont,
        loadkmap, logger, login, logname, logread, losetup, lpd, lpq, lpr, ls,
        lsattr, lsmod, lsof, lspci, lsscsi, lsusb, lzcat, lzma, lzop, makedevs,
        makemime, man, md5sum, mdev, mesg, microcom, mkdir, mkdosfs, mke2fs,
        mkfifo, mkfs.ext2, mkfs.minix, mkfs.vfat, mknod, mkpasswd, mkswap,
        mktemp, modinfo, modprobe, more, mount, mountpoint, mpstat, mt, mv,
        nameif, nanddump, nandwrite, nbd-client, nc, netstat, nice, nl, nmeter,
        nohup, nproc, nsenter, nslookup, ntpd, od, openvt, partprobe, passwd,
        paste, patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root,
        pkill, pmap, popmaildir, poweroff, powertop, printenv, printf, ps,
        pscan, pstree, pwd, pwdx, raidautorun, rdate, rdev, readahead,
        readlink, readprofile, realpath, reboot, reformime, remove-shell,
        renice, reset, resize, rev, rm, rmdir, rmmod, route, rpm, rpm2cpio,
        rtcwake, run-parts, runlevel, runsv, runsvdir, rx, script,
        scriptreplay, sed, sendmail, seq, setarch, setconsole, setfont,
        setkeycodes, setlogcons, setpriv, setserial, setsid, setuidgid, sh,
        sha1sum, sha256sum, sha3sum, sha512sum, showkey, shred, shuf, slattach,
        sleep, smemcap, softlimit, sort, split, ssl_client, start-stop-daemon,
        stat, strings, stty, su, sulogin, sum, sv, svc, svlogd, swapoff,
        swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset,
        tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top,
        touch, tr, traceroute, traceroute6, true, truncate, tty, ttysize,
        tunctl, ubiattach, ubidetach, ubimkvol, ubirename, ubirmvol, ubirsvol,
        ubiupdatevol, udhcpc, udhcpd, udpsvd, uevent, umount, uname, unexpand,
        uniq, unix2dos, unlink, unlzma, unshare, unxz, unzip, uptime, users,
        usleep, uudecode, uuencode, vconfig, vi, vlock, volname, w, wall,
        watch, watchdog, wc, wget, which, who, whoami, whois, xargs, xxd, xz,
        xzcat, yes, zcat, zcip
```

Busybox のコマンドは上記のように Busybox そのものを実行すると、どのようなコマンドが利用可能かも教えてくれます(「Currently defined functions」以下に示されています)。

(20) ちなみに Helio に元々入っている Busybox のバージョン番号を調べてみましょう。以下のコマンドで確認できます。

```
root@socfpga:/tmp# /bin/busybox
BusyBox v1.20.2 (2013-11-04 23:53:29 CST) multi-call binary.
Copyright (C) 1998-2011 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
   or: busybox --list
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

Currently defined functions:
        [, [[, ar, ash, awk, basename, bunzip2, bzcat, cat, chattr, chgrp,
        chmod, chown, chroot, chvt, clear, cmp, cp, cpio, cut, date, dc, dd,
        deallocvt, df, diff, dirname, dmesg, dnsdomainname, dpkg-deb, du,
        dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset, fdisk,
        fgrep, find, flock, free, fsck, fsck.minix, fuser, grep, groups,
        gunzip, gzip, halt, head, hexdump, hostname, hwclock, id, ifconfig,
        ifdown, ifup, insmod, ip, kill, killall, klogd, less, ln, loadfont,
        loadkmap, logger, logname, logread, losetup, ls, lsmod, md5sum,
        microcom, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, modprobe,
        more, mount, mv, nc, netstat, nohup, nslookup, od, openvt, patch,
        pidof, ping, ping6, pivot_root, poweroff, printf, ps, pwd, rdate,
        readlink, realpath, reboot, renice, reset, rm, rmdir, rmmod, route,
        run-parts, sed, seq, setconsole, sh, sleep, sort, start-stop-daemon,
        strings, stty, swapoff, swapon, switch_root, sync, sysctl, syslogd,
        tail, tar, tee, telnet, test, tftp, time, top, touch, tr, traceroute,
        true, tty, udhcpc, udhcpd, umount, uname, uniq, unzip, uptime, users,
        usleep, vi, watch, wc, wget, which, who, whoami, xargs, yes, zcat
```

(21) Busybox は busybox コマンドにコマンド名を引数として渡すのと同じことを、実は各コマンドが busybox コマンドへのシンボリック・リンクを張ることで実現しています。
実際 Helio の /bin を見てみると、そのようになっています。

```
root@socfpga:/tmp# ls -l /bin
lrwxrwxrwx    1 root     root            14 Mar  3  2014 addgroup -> /bin/tinylogin
lrwxrwxrwx    1 root     root            14 Mar  3  2014 adduser -> /bin/tinylogin
-rwxr-xr-x    1 root     root         11356 Nov  5 05:56 arping
lrwxrwxrwx    1 root     root             7 Mar  3  2014 ash -> busybox
-rwxr-xr-x    1 1000     1000        592824 Nov  5 05:15 bash
-rwsr-xr-x    1 root     root        371972 Nov  5 05:58 busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 cat -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 chattr -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 chgrp -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 chmod -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 chown -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 cp -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 cpio -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 date -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 dd -> busybox
lrwxrwxrwx    1 root     root            14 Mar  3  2014 delgroup -> /bin/tinylogin
lrwxrwxrwx    1 root     root            14 Mar  3  2014 deluser -> /bin/tinylogin
lrwxrwxrwx    1 root     root             7 Mar  3  2014 df -> busybox
lrwxrwxrwx    1 root     root            16 Mar  3  2014 dmesg -> dmesg.util-linux
-rwxr-xr-x    1 root     root         18312 Nov  5 05:33 dmesg.util-linux
lrwxrwxrwx    1 root     root            23 Mar  3  2014 dnsdomainname -> dnsdomainname.net-tools
lrwxrwxrwx    1 root     root            18 Mar  3  2014 dnsdomainname.net-tools -> hostname.net-tools
lrwxrwxrwx    1 root     root            20 Mar  3  2014 domainname -> domainname.net-tools
lrwxrwxrwx    1 root     root            18 Mar  3  2014 domainname.net-tools -> hostname.net-tools
lrwxrwxrwx    1 root     root             7 Mar  3  2014 dumpkmap -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 echo -> busybox
lrwxrwxrwx    1 root     root            10 Mar  3  2014 egrep -> egrep.grep
-rwxr-xr-x    1 root     root        121952 Nov  5 05:57 egrep.grep
lrwxrwxrwx    1 root     root             7 Mar  3  2014 false -> busybox
lrwxrwxrwx    1 root     root            10 Mar  3  2014 fgrep -> fgrep.grep
-rwxr-xr-x    1 root     root         67852 Nov  5 05:57 fgrep.grep
lrwxrwxrwx    1 root     root             9 Mar  3  2014 grep -> grep.grep
-rwxr-xr-x    1 root     root        122900 Nov  5 05:57 grep.grep
lrwxrwxrwx    1 root     root             7 Mar  3  2014 gunzip -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 gzip -> busybox
lrwxrwxrwx    1 root     root            18 Mar  3  2014 hostname -> hostname.net-tools
-rwxr-xr-x    1 root     root          8400 Nov  5 05:55 hostname.net-tools
lrwxrwxrwx    1 root     root             7 Mar  3  2014 ip -> busybox
lrwxrwxrwx    1 root     root            15 Mar  3  2014 kill -> kill.util-linux
-rwxr-xr-x    1 root     root         14492 Nov  5 05:33 kill.util-linux
-rwxr-xr-x    1 root     root         56928 Nov  5 05:32 kmod
lrwxrwxrwx    1 root     root             7 Mar  3  2014 ln -> busybox
lrwxrwxrwx    1 root     root            14 Mar  3  2014 login -> /bin/tinylogin
-rwxr-xr-x    1 root     root         39752 Nov  5 05:42 login.shadow
lrwxrwxrwx    1 root     root             7 Mar  3  2014 ls -> busybox
```

```
lrwxrwxrwx    1 root     root            10 Mar  3  2014 lsmod -> lsmod.kmod
lrwxrwxrwx    1 root     root            11 Mar  3  2014 lsmod.kmod -> ../bin/kmod
lrwxrwxrwx    1 root     root             7 Mar  3  2014 mkdir -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 mknod -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 mktemp -> busybox
lrwxrwxrwx    1 root     root            15 Mar  3  2014 more -> more.util-linux
-rwxr-xr-x    1 root     root         26652 Nov  5 05:33 more.util-linux
lrwxrwxrwx    1 root     root            16 Mar  3  2014 mount -> mount.util-linux
-rwsr-xr-x    1 root     root         57400 Nov  5 05:33 mount.util-linux
lrwxrwxrwx    1 root     root            19 Mar  3  2014 mountpoint -> mountpoint.sysvinit
-rwxr-xr-x    1 root     root          4392 Nov  5 05:58 mountpoint.sysvinit
lrwxrwxrwx    1 root     root             7 Mar  3  2014 mv -> busybox
lrwxrwxrwx    1 root     root            17 Mar  3  2014 netstat -> netstat.net-tools
-rwxr-xr-x    1 root     root         67912 Nov  5 05:55 netstat.net-tools
lrwxrwxrwx    1 root     root            23 Mar  3  2014 nisdomainname -> nisdomainname.net-tools
lrwxrwxrwx    1 root     root            18 Mar  3  2014 nisdomainname.net-tools -> hostname.net-tools
lrwxrwxrwx    1 root     root            14 Mar  3  2014 pidof -> pidof.sysvinit
lrwxrwxrwx    1 root     root            14 Mar  3  2014 pidof.sysvinit -> /sbin/killall5
lrwxrwxrwx    1 root     root            12 Mar  3  2014 ping -> ping.iputils
-r-sr-xr-x    1 root     root         27240 Nov  5 05:56 ping.iputils
lrwxrwxrwx    1 root     root            13 Mar  3  2014 ping6 -> ping6.iputils
-r-sr-xr-x    1 root     root         28056 Nov  5 05:56 ping6.iputils
lrwxrwxrwx    1 root     root             7 Mar  3  2014 ps -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 pwd -> busybox
-rwxr-xr-x    1 root     root           190 Nov  5 05:19 reset
lrwxrwxrwx    1 root     root             7 Mar  3  2014 rm -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 rmdir -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 sed -> sed.sed
-rwxr-xr-x    1 root     root         76192 Nov  5 05:56 sed.sed
lrwxrwxrwx    1 root     root             4 Mar  3  2014 sh -> bash
lrwxrwxrwx    1 root     root             7 Mar  3  2014 sleep -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 stty -> busybox
lrwxrwxrwx    1 root     root            14 Mar  3  2014 su -> /bin/tinylogin
lrwxrwxrwx    1 root     root             7 Mar  3  2014 sync -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 tar -> busybox
-rwsr-xr-x    1 root     root         32068 Nov  5 05:58 tinylogin
lrwxrwxrwx    1 root     root             7 Mar  3  2014 touch -> busybox
-rwxr-xr-x    1 root     root          7076 Nov  5 05:56 tracepath
-rwxr-xr-x    1 root     root          8012 Nov  5 05:56 tracepath6
-r-sr-xr-x    1 root     root         10188 Nov  5 05:56 traceroute6
lrwxrwxrwx    1 root     root             7 Mar  3  2014 true -> busybox
lrwxrwxrwx    1 root     root            17 Mar  3  2014 umount -> umount.util-linux
-rwsr-xr-x    1 root     root         43424 Nov  5 05:33 umount.util-linux
lrwxrwxrwx    1 root     root             7 Mar  3  2014 uname -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 usleep -> busybox
lrwxrwxrwx    1 root     root             7 Mar  3  2014 vi -> busybox
lrwxrwxrwx    1 root     root            22 Mar  3  2014 ypdomainname -> ypdomainname.net-tools
lrwxrwxrwx    1 root     root            18 Mar  3  2014 ypdomainname.net-tools -> hostname.net-tools
lrwxrwxrwx    1 root     root             7 Mar  3  2014 zcat -> busybox
```

(22) では、Busybox への各コマンドのシンボリック・リンクのファイルはどのように作成するのでしょうか？

自分の手でリンク作成のためのコマンドを打っていると、これだけコマンドの数があると間違ってしまいそうです。

そこで以下のように**ホスト・マシン（Vine Linux）上**で「make install」コマンドを実行して作成します（下記の**【注意点】**をよく読んでから「make install」コマンドを実行してください）。

```
[tori@Vine65 busybox-1.27.0]$ make CONFIG_PREFIX=/tmp/busybox_command install
```

【注意点】

ここで注意しなければいけない点は、

① CONFIG_PREFIX を使用してインストール先を指定すること

② この操作を root で行わないこと

です。

何故このようにしなければいけないかというと、このコマンドはホスト・マシン（Vine Linux）上で実行しています。

従って ① の CONFIG_PREFIX を指定しないと、Vine Linux 上の /bin、/sbin、/usr/bin にインストールされます。つまり Intel® の CPU 上で実行可能なコマンドに、ARM® の CPU 上で実行可能なコマンド（リンク・ファイルですが）が上書きされてしまいます。

なので ② の root で実行しなければ、万が一 CONFIG_PREFIX のタイプ・ミスをしても、root 権限では無いので上書きされることはありません。

せっかく苦労してホスト・マシンの環境構築を行ってきたのに、ここですべてを無駄にするわけにはいきません。是非ここは良く理解して、慎重にコマンドを実行してください。万が一上書いてしまった場合には、最初からホストの Linux マシンをインストールし直す必要があるので気を付けましょう。

(23) 上記の例では /tmp/busybox_command の下に生成しました。この生成したコマンドを、ターゲット・ボード（Helio）にインストールすれば OK です。以下のように /opt/Helio にコピーします。

```
[tori@Vine65 busybox-1.27.0]$ cd /tmp
[tori@Vine65 tmp]$ cp -r busybox_command /opt/Helio/
```

(24) **Helio 上**で新しく作成した busybox のコマンド(この例では ifconfig)を実行してみます。

```
root@socfpga:/tmp# /mount_dir/busybox_command/sbin/ifconfig -a
eth0      Link encap:Ethernet  HWaddr FE:EC:4C:72:AF:27
          inet addr:192.168.1.238  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::fcec:4cff:fe72:af27/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4499 errors:0 dropped:0 overruns:0 frame:0
          TX packets:706 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3260460 (3.1 MiB)  TX bytes:62772 (61.3 KiB)
          Interrupt:152

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

これで新しい busybox コマンドで正常に実行できることが確認できました。

2-2. thttpd のクロス・コンパイルとインストール

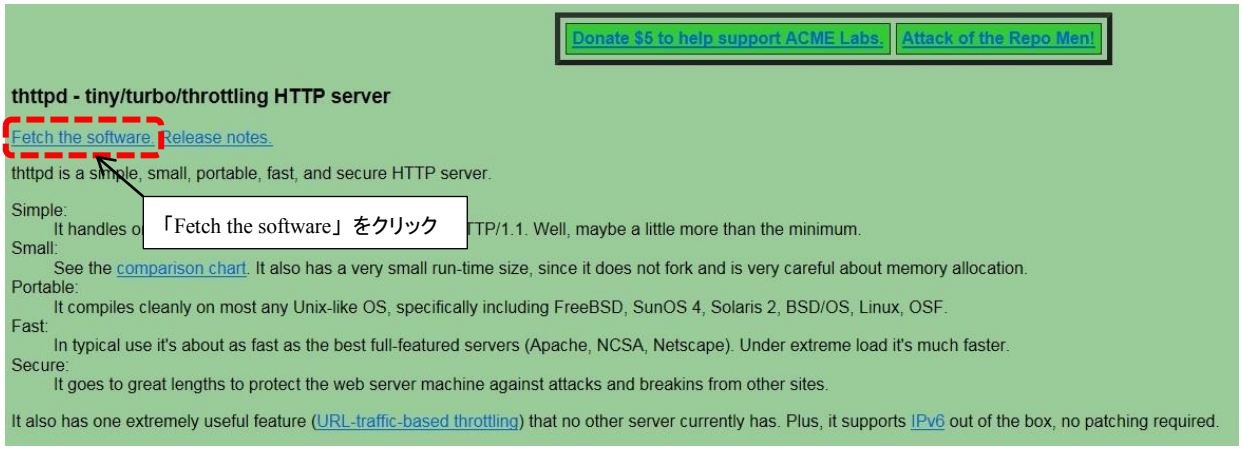
Helio にはあらかじめ Web サーバがインストールされています。これを使えば、HTML（Hyper Text Markup Language）だけではなく CGI（Common Gateway Interface）も実行できます。

この章では GNU アプリケーションのインストールを行うのが目的ですので、ここはあえて thttpd という組み込み機器では割と使用されている Web サーバ・ソフトをインストールすることにします（他の組み込み Linux ボードにあらかじめ thttpd がインストールされていない時にも有用です）。

(1) 以下の thttpd のページに移動します。

http://acme.com/software/thttpd/

(2) 下図のページで、「Fetch the software」をクリックして thttpd のソース・ファイルをダウンロードし、保存します。



【図 2-2.1】thttpd のサイト

(3) ダウンロードできたら、このファイルを Samba サーバ経由で、Windows 上のエクスプローラから Vine Linux 上の一般ユーザ（この例では tori）のホーム・ディレクトリ（この例では /home/tori）にコピーします。

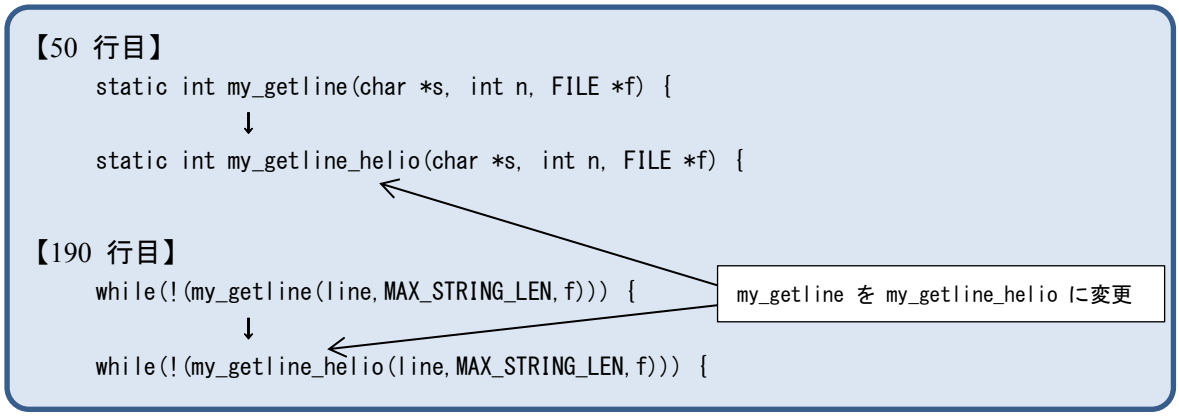
(4) コピーが完了したら以下の tar コマンドでファイルを展開し、ARM 用の thttpd ファイルをコンパイルします（この例では、ダウンロードした thttp ファイルは、thttpd-2.27.tar.tar を使用しています）。

```
[tori@Vine65 ~]$ cd ~
[tori@Vine65 ~]$ tar zxvf thttpd-2.27.tar.tar -C/tmp
[tori@Vine65 ~]$ cd /tmp/thttpd-2.27
[tori@Vine65 thttpd-2.27]$ CC=arm-linux-gnueabihf-gcc ./configure --host=arm-linux
（ログが出力される）
```

(5) このままだと、htpasswd.c ファイルの中で getline に関わる箇所でエラーになるので、Leafpad を使用して、あらかじめ**【リスト 2-2.1】**のように修正しておきます。

下記のように sudo を付けてスーパ・ユーザで実行します。

```
[tori@Vine65 thttpd-2.27]$ sudo leafpad extras/htpasswd.c
[sudo] password for tori:  ←  一般ユーザのパスワード（この例では、toriumi）を入力
```



【リスト 2-2.1】htpasswd.c ファイルの中身

編集が終了したら、Leafpad でファイルを保存して閉じます。

【参考】

Leafpad で**行番号を表示**させるには、Leafpad の「オプション」メニューから「行番号を表示」にチェックを入れます。
また、**指定した行にジャンプ**するには、Leafpad の「検索」メニューから「行へジャンプ」を選択し、行番号を指定して[ジャンプ]をクリックします。

(6) 以下のように make コマンドを実行して、thttpd をコンパイルします。

```
[tori@Vine65 thttpd-2.27]$ make
（ログが出力される）
```

【注記】

SoC Linux 道場【其ノ参】の「2-2. コマンド・パスの設定」で説明したクロス・コンパイラがインストールされて、パスが設定されていて使用できることを前提としています。

ここで、「./configure」と「make」を実行してエラーが出るようであれば、
which arm-linux-gnueabihf-gcc コマンドを実行して、クロス・コンパイラのパスが通っているか確認してください。
クロス・コンパイラのパスが正しく通っていない場合は、再度 SoC Linux 道場【其ノ参】を参照して必要な設定を行ってください。

(7) 特にエラーなく終了していれば、カレント・ディレクトリに「thttpd」というファイルができ上がっているはずです。

念のため **file** コマンドで確認してみます。以下のように ARM 用にコンパイルされていることが確認できるはずです。

```
[tori@Vine65 thttpd-2.27]$ file thttpd
thttpd: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter
/lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32,
BuildID[sha1]=204dbf54aa8aede1b3b18a79f85
```

この出力結果を見ると、ARM CPU で実行可能なことがわかります

(8) この thttpd のディレクトリの中にあるファイルのうち、以下のファイルを Helio にコピーします。

① thttpd（コンパイルしたディレクトリ：/tmp/thttpd-2.27 の下に存在）

② index.html（コンパイルしたディレクトリ：/tmp/thttpd-2.27 の下に存在）

③ thttpd.conf（コンパイルしたディレクトリ：/tmp/thttpd-2.27 の下の contrib/redhat-rpm に存在）

④ printenv（コンパイルしたディレクトリ：/tmp/thttpd-2.27 の下の cgi-bin に存在）

コピーする前にあらかじめ thttpd.conf ファイルのパーミッションと中身を変更しておきます。

```
[tori@Vine65 thttpd-2.27]$ chmod 666 contrib/redhat-rpm/thttpd.conf
[tori@Vine65 thttpd-2.27]$ leafpad contrib/redhat-rpm/thttpd.conf
```

今回は既にターゲット・ボードで動作している Web サーバがあるので、ポート番号は 8080 にしておきます。これで、同一マシン上に異なる Web サーバを動作させることができます。

```
dir=/home/httpd/html
nochroot
user=httpd
logfile=/var/log/thttpd.log
pidfile=/var/run/thttpd.pid
port=8080
cgipat=**.cgi
charset=EUC-JP
```

【リスト 2-2.2】thttp.conf の変更後の中身

作成したら、Leafpad でファイルを保存して閉じます。

次に NFS 経由で、前述した 4 つのファイルを /opt/Helio にコピーします。

```
[tori@Vine65 thttpd-2.27]$ cp thttpd /opt/Helio/
[tori@Vine65 thttpd-2.27]$ cp index.html /opt/Helio/
[tori@Vine65 thttpd-2.27]$ cp contrib/redhat-rpm/thttpd.conf /opt/Helio
[tori@Vine65 thttpd-2.27]$ cp cgi-bin/printenv /opt/Helio/printenv.cgi
```

printenv だけはコピー時に、
printenv.cgi にファイル名を変更しておく

(9) 次に Helio 上で thttpd の設定を行います。

まず、NFS 経由で見えている 4 つのファイル(thttpd, thttpd.conf, index.html, printenv.cgi)を /tmp ディレクトリにコピーしておきます。

```
root@socfpga:/# cd /tmp
root@socfpga:/tmp# cp /mount_dir/[tip]* .
```

← 最後の「. 」(ドット)を忘れないこと

(10) 以下のように Helio 上で必要なユーザやファイルを作成します。

```
root@socfpga:/tmp# adduser httpd
Changing password for httpd
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
Enter new password:
Bad password: too simple.

Warning: weak password (continuing).
Re-enter new password:
Password changed.
root@socfpga:/tmp# mkdir /home/httpd/html
root@socfpga:/tmp# mkdir /home/httpd/html/cgi-bin
root@socfpga:/tmp# cp index.html /home/httpd/html/
root@socfpga:/tmp# chmod 644 /home/httpd/html/index.html
root@socfpga:/tmp# cp printenv.cgi /home/httpd/html/cgi-bin
root@socfpga:/tmp# chmod 755 /home/httpd/html/cgi-bin/printenv.cgi
```

適当なパスワード(例えば、toriumi)を入力します(非表示)

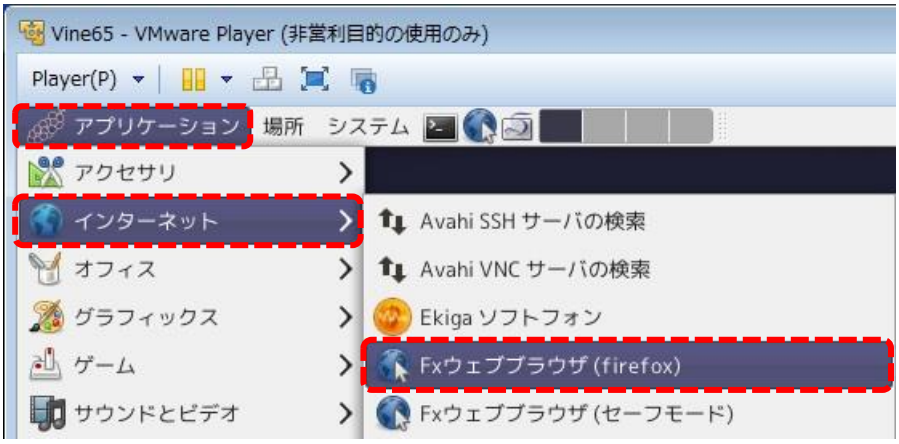
(11) 最後に thttpd を以下のコマンドで起動します。その後、ps コマンドでプロセスが起動していることを確認します。

```
root@socfpga:/tmp# ./thttpd -C thttpd.conf
root@socfpga:/tmp# ps
  PID USER       VSZ STAT COMMAND
    1 root      1312 S    init [5]
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]

     (途中省略)

  704 httpd     2388 S    ./thttpd -C thttpd.conf
  705 root      1944 R    ps
root@socfpga:/tmp#
```

← thttpd を起動

← プロセスが起動していることを確認

(12) ホスト・マシン（Vine Linux）上で Web ブラウザ（Firefox）を起動します。

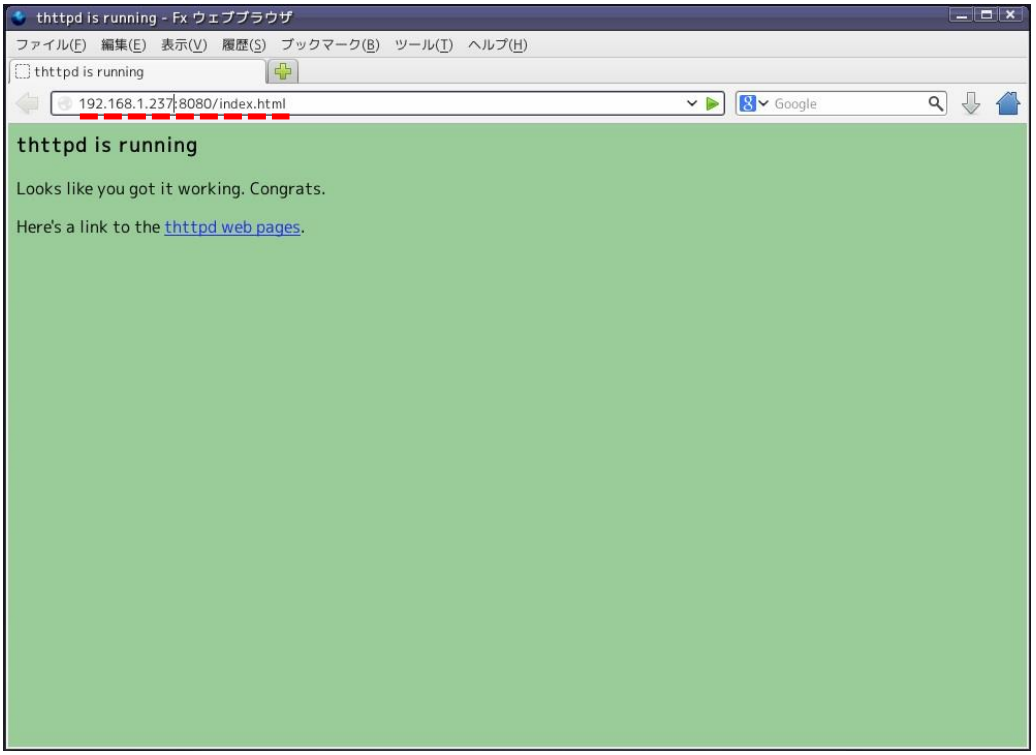


【図 2-2.2】Web ブラウザ（Firefox）を起動

(13) 例えば DHCP サーバから付与された Helio の IP アドレスが「192.168.1.237」の場合は、

```
http://192.168.1.237:8080/index.html
```

を URL に指定して、下図のように表示されることを確認します。

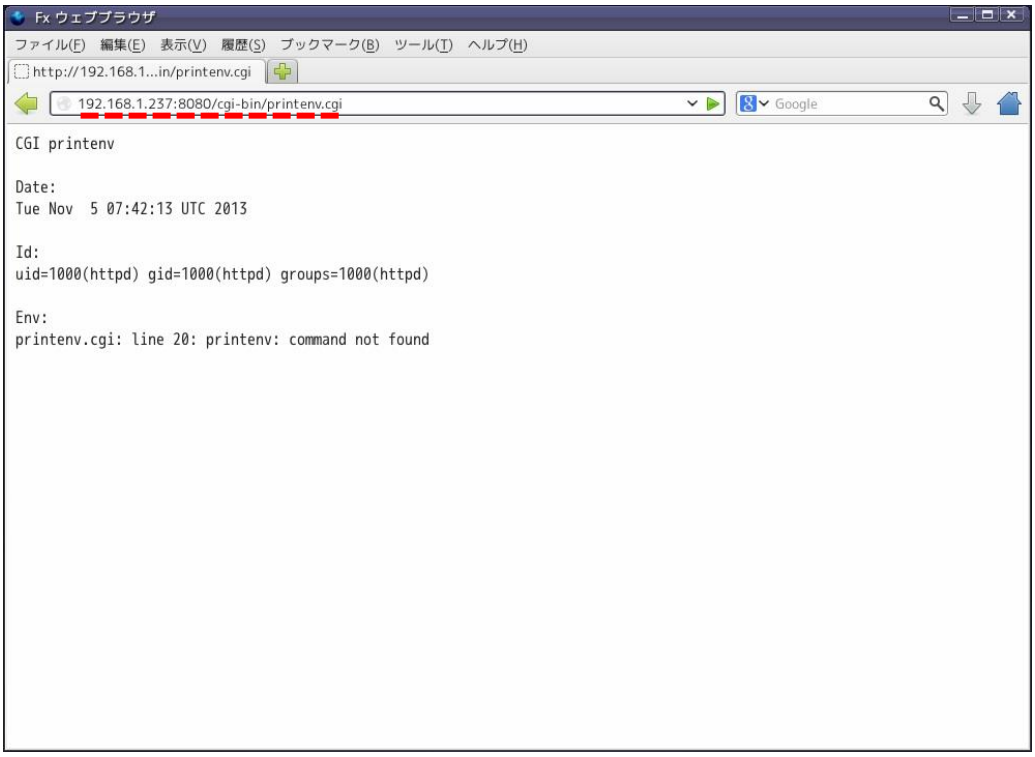


【図 2-2.3】index.html の表示

(14) 同様に DHCP サーバから付与された Helio の IP アドレスが「192.168.1.237」の場合は、

```
http://192.168.1.237:8080/cgi-bin/printenv.cgi
```

を URL に指定して、下図のように表示されることを確認します。



【図 2-2.4】printenv.cgi の表示

もしうまく表示されない場合は、

① thttpd.conf のファイルの中身を見直す（特に nochroot になっていないと CGI がうまく動作しません）

② index.html のパーミッションは 644、printenv.cgi のパーミッションは 755 になっているかを確かめる

③「httpd」という名前でユーザ名を作成しているか？

④ /home/httpd の下に「html」と「html/cgi-bin」の名前になっているか？

などを良く確かめてください。

Web サーバを自力で（といってもソースは GNU ですが）立てられるところも GNU の魅力だと思います。

次回【其ノ五】では、Linux カーネルの入手とコンパイルの手順について解説します。

Linux カーネルの入手を行い、その後、カーネル・コンフィギュレーションとコンパイルを実行します。

また、「カスタム・ドライバの作成とコンパイル（その 1）」として、簡単な Hello ドライバ（Hello メッセージを表示するドライバ）をコンパイルして Helio ボードに NFS 経由で転送してから実行してみます。

改版履歴

Revision	年月	概要
1	2015 年 1 月	新規作成
2	2017 年 8 月	① Sourcery CodeBench Lite Edition for ARM GNU/Linux の配布終了に伴い、クロス・コンパイル環境として Linaro Toolchain を使用する説明に変更 ② ゲスト OS を Vine Linux 6.2.1 i686 から Vine Linux 6.5 x86_64 に変更