

# Linuxによる組込みシステム開発

⑤

港湾職業能力短期大学校 神戸校

マイコンボード：T-SH7706LSR(TAC製)

マイコン：SH7706(SH3ファミリ)：ルネサスエレクトロニクス

この教材は みついわゆきお氏のmes2OSサイト

<http://mes.sourceforge.jp/mes2/>

およびLinux記事

<http://gihyo.jp/dev/serial/01/micom-linux/0001>

および、名もないページさんサイト

<http://wave2.iobb.net/doc/summary/sh3wiki/wifky.cgi>

の内容を参考に作成しています

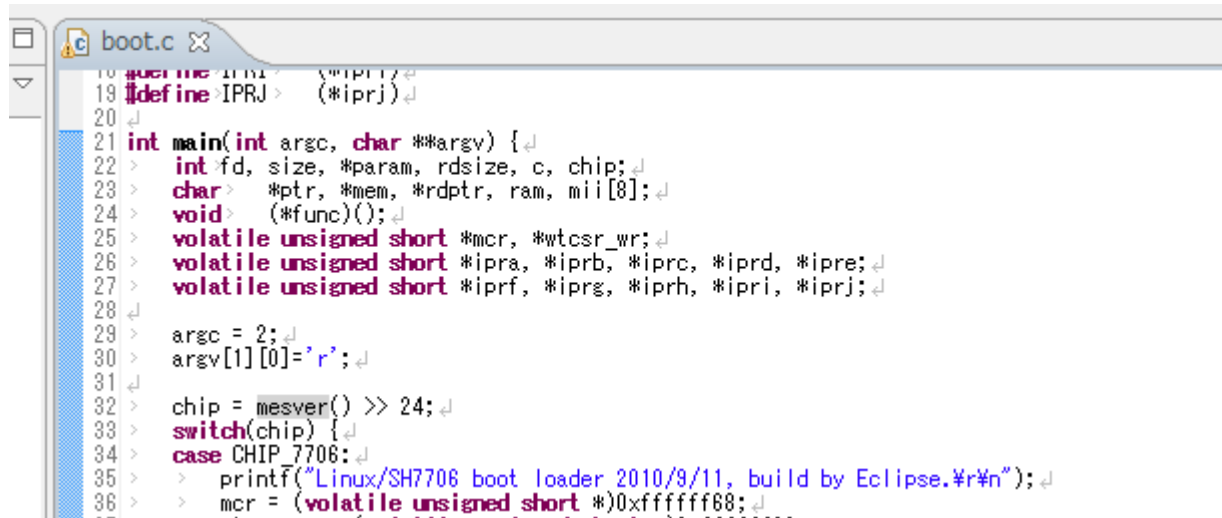
# RAMディスク仕様への書換(1)

これまでのSDカード上のファイルシステムで作成していたものをrootfs.imgに組み込み、電源断にも影響のないシステムに書き換える。ここではsgledtestのプログラムをtelnetとhttpdで動作させる。

1. マイコンの自動起動をrootfs.imgルートのものとする。②の記事を参考にautoboot.motを編集してもよいが、ここではboot.cに手を加える。

windowsマシンでeclipseを起動し、boot.cを以下に編集する。

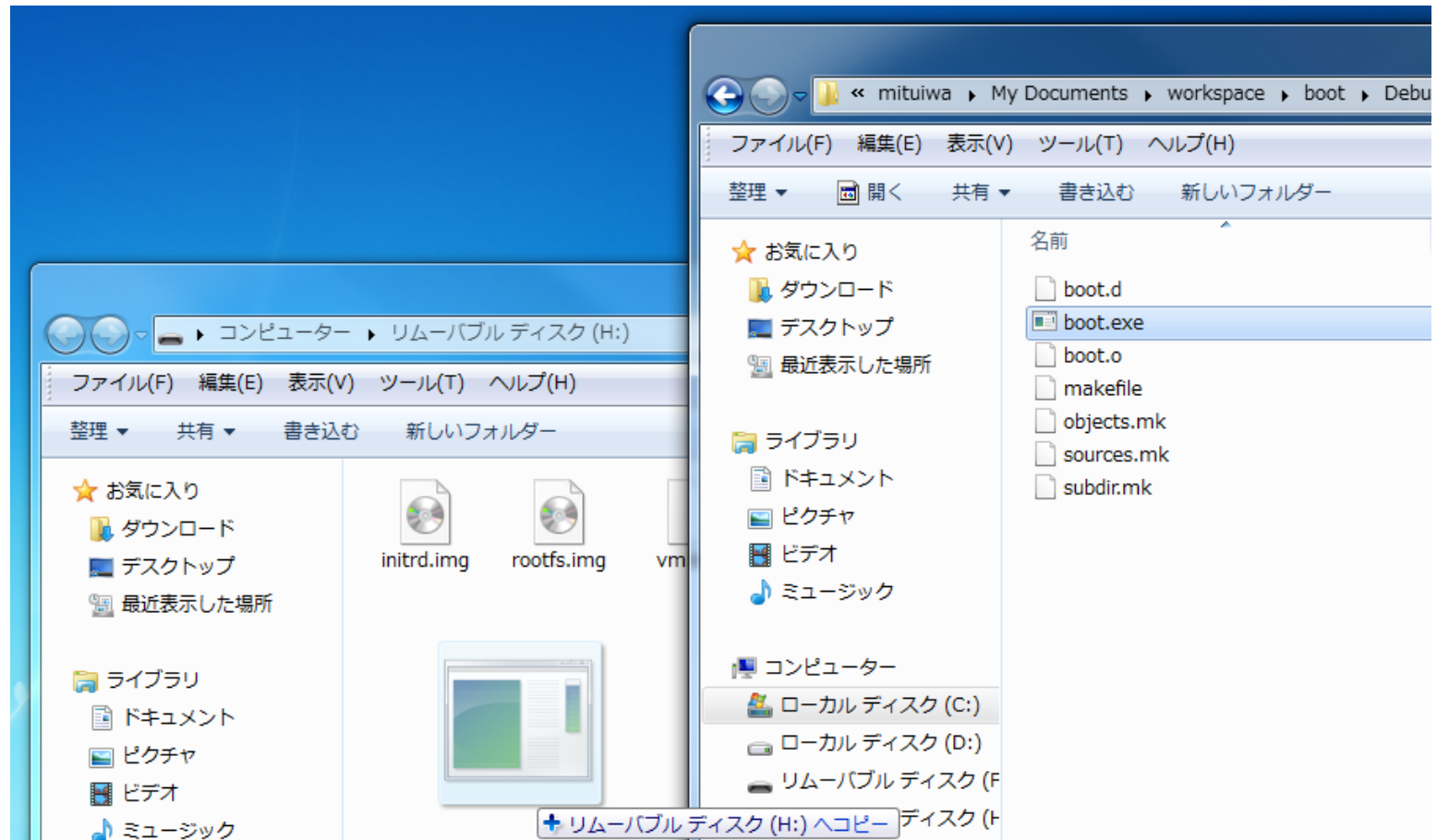
argc=2; と argv[1][0]='r'; を以下のように追記する。



```
18 #define IPR1 (&ipr1)
19 #define IPRJ (&iprj)
20
21 int main(int argc, char **argv) {
22 > int fd, size, *param, rdsi, c, chip;
23 > char *ptr, *mem, *rdptr, ram, mii[8];
24 > void (*func)();
25 > volatile unsigned short *mcr, *wtcsr_wr;
26 > volatile unsigned short *ipra, *iprb, *iprc, *iprd, *ipre;
27 > volatile unsigned short *iprf, *iprg, *iprh, *ipri, *iprj;
28
29 > argc = 2;
30 > argv[1][0] = 'r';
31
32 > chip = mesver() >> 24;
33 > switch(chip) {
34 > case CHIP_7706:
35 > > printf("Linux/SH7706 boot loader 2010/9/11, build by Eclipse.%r\n");
36 > > mcr = (volatile unsigned short *)0xfffff68;
```

## RAMディスク仕様への書換(2)

2. コンパイルしたのち、作成されたboot.exe をSDカードのFATフォーマット部へコピーする。



## RAMディスク仕様への書換(3)

3. カーネルが入れ替わっているのでメッセージが異なっているかもしれないが、rootでログイン後、パスワード"system0"にてプロンプト画面へ。

A terminal window with a black background and white text. The text reads "shlinux login: root" followed by "Password:" and a white cursor block.

```
shlinux login: root
Password: █
```

4. # ./helloで Hello C!のメッセージが返る。

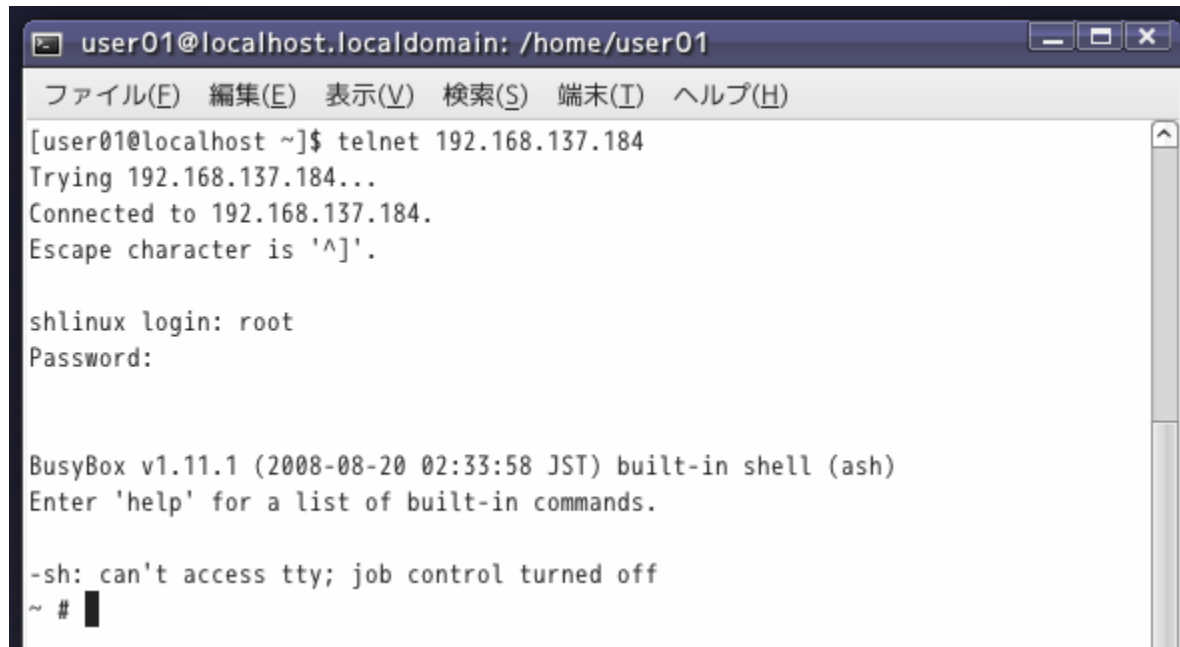
A terminal window titled "COM4:115200baud - Tera Term VT" with a menu bar (File, Edit, Settings, Control, Window). The terminal shows a sequence of commands and outputs: "# ls" followed by "hello" in green, "# ./hello" followed by "Hello C!", and finally "# █" with a cursor.

```
COM4:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W)
~ # ls
hello
~ # ./hello
Hello C!
~ # █
```

## RAMディスク仕様への書換(4)

5. #ifconfig eth0 192.168.137.184 を入力し、# telnetdを起動すればLinuxマシンからのtelnet接続も可能となる。

```
~ # ifconfig eth0 192.168.137.184
~ # telnetd
~ # █
```



```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.

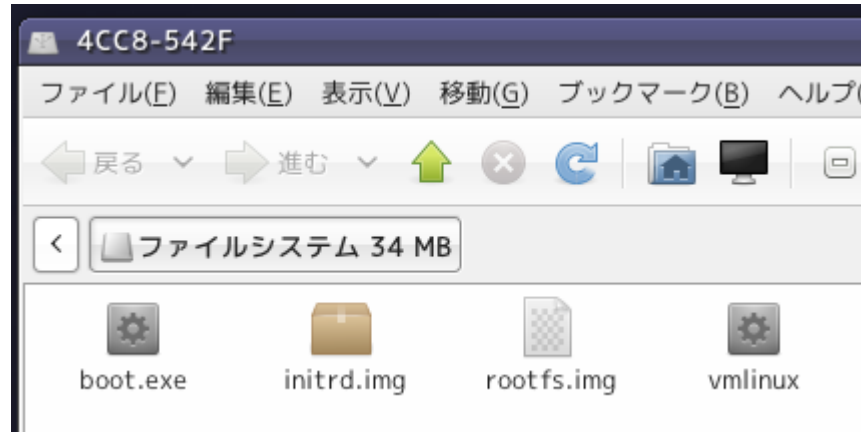
shlinux login: root
Password:

BusyBox v1.11.1 (2008-08-20 02:33:58 JST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

-sh: can't access tty; job control turned off
~ # █
```

## RAMディスク仕様への書換(5)

6. 今回はNFSは使えないのでデータのやり取りはSDカードでおこなう。  
IPアドレス設定とtelnetd起動を毎回設定するのはめんどろななので自動起動するよう設定する。SDカードをとりはずしLinuxマシンに認識させる。FATシステムは「ACC8-542F」であることがわかる。



7. rootfs.imagを①でおこなったように/mntにマウントする。

```
user01@localhost.localdomain: /mnt
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[user01@localhost ~]$ su
パスワード:
[root@localhost user01]# cd /media/4CC8-542F/
[root@localhost 4CC8-542F]# mount -o loop rootfs.img /mnt
[root@localhost 4CC8-542F]# cd /mnt
[root@localhost mnt]# ls
bin/   dev/   home/  lib/   lost+found/  proc/  sbin/  usr/
boot/  etc/   initrd/ linuxrc@ mnt/       root/  tmp/   var/
[root@localhost mnt]#
```

## RAMディスク仕様への書換(6)

8. /mnt/etc/rc.d/rc.sysinit を編集する。くれぐれもLinuxマシン本体側のファイルと取り違ふことのないよう注意のこと。

```
user01@localhost.localdomain: /mnt/etc/rc.d
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost mnt]# cd ../etc/rc.d/
[root@localhost rc.d]# vi rc.sysinit
```

9. rc.sysinitではhostname shlinuxの次の処理で記述する。

```
mount -t devpts devpts /dev/pts
#swapon /dev/shmmc3

hostname shlinux

ifconfig eth0 192.168.137.184
telnetd

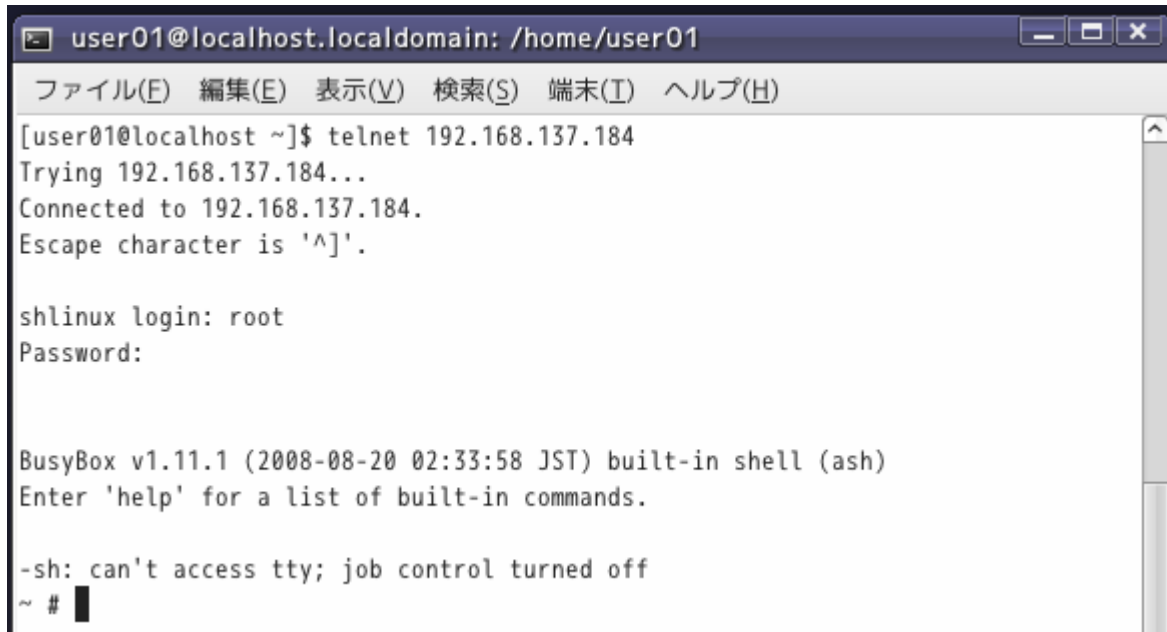
# for item in `cat /proc/devices`;
```

10. "Esc"→":"→"w"→"q"にて保存のあと、アンマウント処理し、SDカードをターゲットボードに装着して起動する。

```
user01@localhost.localdomain: /
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost rc.d]# cd /
[root@localhost /]# umount /mnt
[root@localhost /]# █
```

# RAMディスク仕様への書換(7)

11. マイコン側のログイン画面の確認ののち、Linuxマシンにてtelnet接続が確認できる。



```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.

shlinux login: root
Password:

BusyBox v1.11.1 (2008-08-20 02:33:58 JST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

-sh: can't access tty; job control turned off
~ #
```



## RAMディスク仕様への書換(8)

12. roofs.img のもつファイル容量は小さいためsgledtest の容量を小さくする。

-static オプションをつけたものは438kバイトである。

なお、sgledtest.cはタイマカウントを100で終了するように手を加えたものになっている。

```
user01@localhost.localdomain: /public/KNL_work/sgled
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost sgled]# ls -l|grep sgledtest
-rwxr-xr-x 1 root root 438981  4月 30 09:51 sgledtest*
-rw-r--r-- 1 1000 1000   884  4月 30 09:51 sgledtest.c
[root@localhost sgled]#
```

13. Makefileを編集し、ライブラリを使える設定にする。

```
user01@localhost.localdomain: /public/KNL_work/sgled
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost sgled]# mv sgledtest sgledtest.old
[root@localhost sgled]# vi Makefile
```

```
all:    $(TARGET) modules

$(TARGET): $(TARGET).c
          sh3-linux-gcc -o $@ $<

modules:
```

# RAMディスク仕様への書換(9)

14. #make で作成されたsgledtestは7kバイト程度まで減る。

```
user01@localhost.localdomain: /public/KNL_work/sgled
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost sgled]# ls -l|grep sgledtest
-rwxr-xr-x 1 root root 7113 5月 1 16:44 sgledtest*
-rw-r--r-- 1 1000 1000 884 4月 30 09:51 sgledtest.c
-rwxr-xr-x 1 root root 438981 4月 30 09:51 sgledtest.old*
[root@localhost sgled]#
```

15. 同様に agent2.exe loopcheck.exe も-static オプションを外して作成する。

```
user01@localhost.localdomain: /public/agent2
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost agent2]# mv agent2.exe agent2.exe.old
[root@localhost agent2]# sh3-linux-gcc -o agent2.exe agent2.c
[root@localhost agent2]# ls -l|grep agent2.exe
-rwxr-xr-x 1 root root 7132 5月 1 16:50 agent2.exe*
-rwxr-xr-x 1 root root 439069 5月 1 13:41 agent2.exe.old*
[root@localhost agent2]#
```

```
user01@localhost.localdomain: /public/loopcheck
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost loopcheck]# mv loopcheck.exe loopcheck.exe.old
[root@localhost loopcheck]# sh3-linux-gcc -o loopcheck.exe loopcheck.c
[root@localhost loopcheck]# ls -l|grep loopcheck.exe
-rwxr-xr-x 1 root root 7305 5月 1 16:52 loopcheck.exe*
-rwxr-xr-x 1 root root 440194 5月 1 13:54 loopcheck.exe.old*
[root@localhost loopcheck]#
```

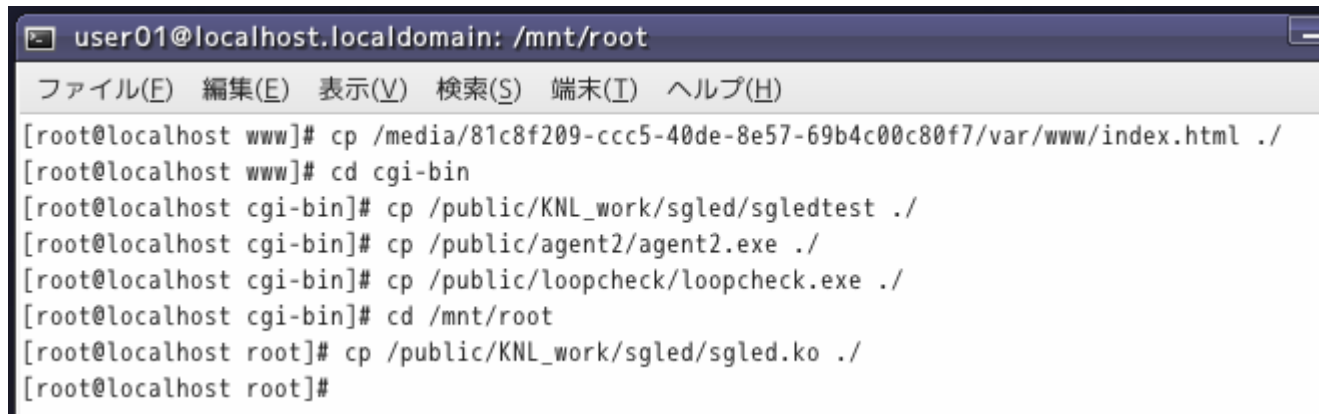
## RAMディスク仕様への書換(10)

16. SDカードのLinuxマシンに認識させ、rootfs.img を/mntにマウントさせる。  
/varフォルダに /www およびその下に /cgi-bin フォルダをそれぞれ作成する。



```
user01@localhost.localdomain: /mnt/var/www
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost user01]# mount -o loop /media/4CC8-542F/rootfs.img /mnt
[root@localhost user01]# cd /mnt/var
[root@localhost var]# mkdir www
[root@localhost var]# cd www
[root@localhost www]# mkdir cgi-bin
[root@localhost www]#
```

17. /mnt/var/www/cgi-bin に sgledtest,agent2.exe,loopcheck.exe をコピーし、  
/mnt /var/wwwにindex.html、mnt/root にsgled.koをコピーする。



```
user01@localhost.localdomain: /mnt/root
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost www]# cp /media/81c8f209-ccc5-40de-8e57-69b4c00c80f7/var/www/index.html ./
[root@localhost www]# cd cgi-bin
[root@localhost cgi-bin]# cp /public/KNL_work/sgled/sgledtest ./
[root@localhost cgi-bin]# cp /public/agent2/agent2.exe ./
[root@localhost cgi-bin]# cp /public/loopcheck/loopcheck.exe ./
[root@localhost cgi-bin]# cd /mnt/root
[root@localhost root]# cp /public/KNL_work/sgled/sgled.ko ./
[root@localhost root]#
```

index.html については同時に認識されたSDカードのフォルダからコピーする。

# RAMディスク仕様への書換(11)

18. rootfs.img をアンマウント後、SDカードをとりはずし、マイコンボードに差込み、ボードを立ち上げる。telnet 接続後、ドライバのノードを登録する。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
~ # mknod /dev/sgled c 240 3
~ #
```

19. 続けてsgled.koをロードする。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
~ # mknod /dev/sgled c 240 3
~ # insmod sgled.ko
~ # lsmod
Module                Size Used by    Not tainted
sgled                  960  0
~ #
```

20. sgledtest を実行して動作を確認する。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
~ # cd /var/www/cgi-bin/
/var/www/cgi-bin # ./sgledtest
```

## RAMディスク仕様への書換(12)

21. 7セグメントのカウン트가確認できる。



22. loopcheck.exe を実行し、頃合いをみて agent2.exe を実行する。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
/var/www/cgi-bin # ./loopcheck.exe &
/var/www/cgi-bin # ./agent2.exe
Content-type: text/html

<html><body bgcolor=white>
<div align=center><big>Reply Page</big></div>
<hr size=2 width=100%>
<p>Operate SgmentLED </p>
</body></html>
/var/www/cgi-bin # █
```

21同様、LEDのカウンタアップが実行される。フラグ動作がRAM内で操作されるためか、ダイナミック点灯のタイミング乱れは生じない。

## RAMディスク仕様への書換(13)

23. 個々のプログラムの動作が確認できたので、ブラウザ動作のまえにtelnet接続のポート80での起動を確認する。Loopcheck.exeは起動したままである。

httpd を起動する。

A terminal window showing the execution of the httpd command. The prompt is user01@localhost.localdomain: /home/u. The menu bar includes ファイル(E), 編集(E), 表示(V), 検索(S), 端末(I). The command # httpd -h /var/www is entered and executed, resulting in a # prompt.

```
user01@localhost.localdomain: /home/u
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I)
~ # httpd -h /var/www
~ #
```

24. telnet ポート80にて接続し、agent2.exe を起動させる。

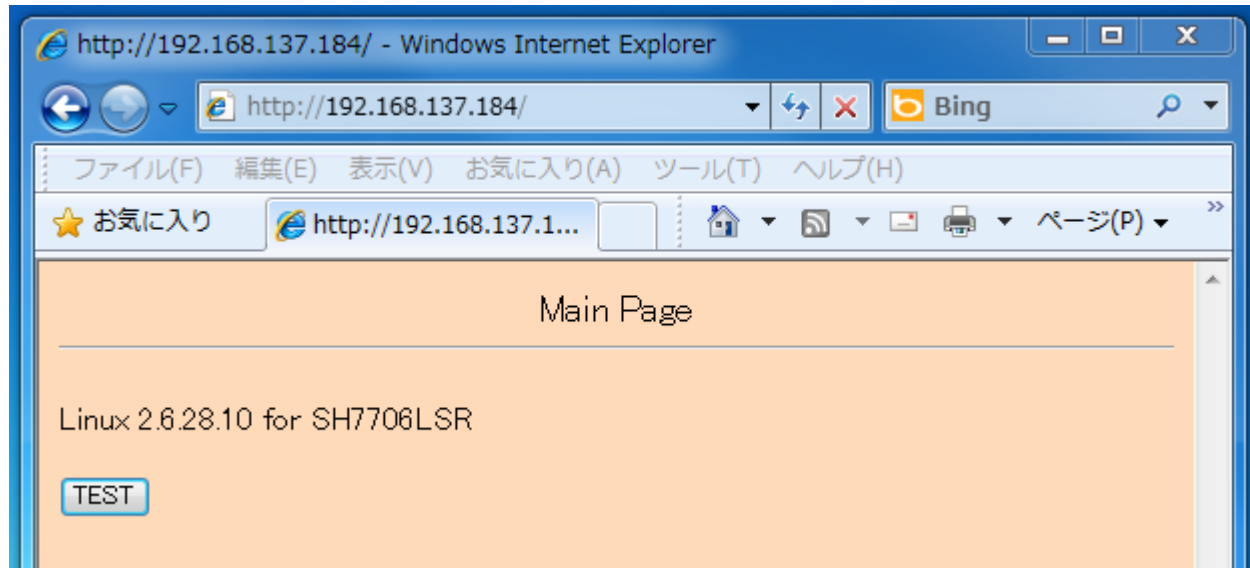
A terminal window showing a telnet connection to port 80 of 192.168.137.184. The prompt is user01@localhost.localdomain: /home/user01. The menu bar includes ファイル(E), 編集(E), 表示(V), 検索(S), 端末(I), ヘルプ(H). The command [user01@localhost ~]\$ telnet 192.168.137.184 80 is entered. The output shows the connection attempt, successful connection, and the execution of POST /cgi-bin/agent2.exe HTTP/1.0.

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184 80
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.
POST /cgi-bin/agent2.exe HTTP/1.0
```

リターン2回入力後、LEDのカウント動作を確認する。

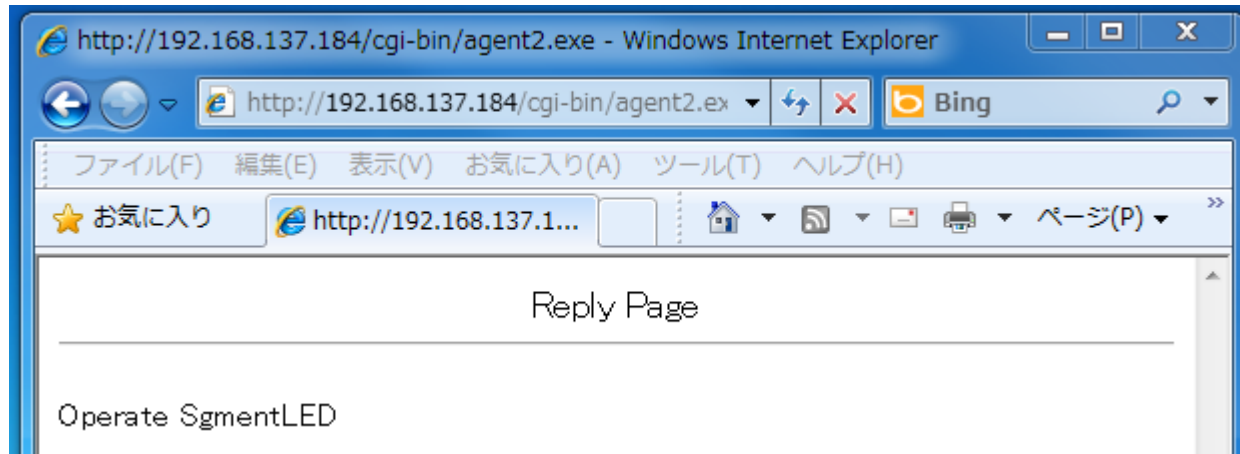
## RAMディスク仕様への書換(14)

25. Webブラウザを立ち上げ、URLに192.168.137.184を入力する。プロキシ設定している場合、設定をはずす。



## RAMディスク仕様への書換(15)

26. 「TEST」をクリックすると、下記返信画面が現れるとともに、LEDのカウントが確認できる。



22同様、ダイナミック点灯のタイミング乱れは生じない。



## RAMディスク仕様への書換(15)

27. 一連動作を自動起動に設定する。SDカードをLinuxマシンで認識させ、rootfs.imgを/mntにマウントする。



```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost user01]# mount -o loop /media/4CC8-542F/rootfs.img /mnt
[root@localhost user01]#
```

28. Linuxの慣習にしたがい、/mnt/lib/にmodules/フォルダとその下に2.6.28.10/フォルダを作成し、その中にsgled.koを保存する。



```
user01@localhost.localdomain: /mnt
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost mnt]# mkdir /mnt/lib/modules
[root@localhost mnt]# mkdir /mnt/lib/modules/2.6.28.10
[root@localhost mnt]# cp /public/KNL_work/sgled/sgled.ko /mnt/lib/modules/2.6.28.10/
[root@localhost mnt]# ls /mnt/lib/modules/2.6.28.10/
sgled.ko
[root@localhost mnt]#
```

## RAMディスク仕様への書換(16)

29. /mnt/etc/rc.d/rc.sysinit を編集して、ドライバのノード登録とドライバロードとhttpデーモン、loopcheck.exeの起動を登録する。

```
user01@localhost.localdomain: /mnt
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost mnt]# vi /mnt/etc/inittab
[root@localhost mnt]# vi /mnt/etc/rc.d/rc.sysinit
```

```
hostname shlinux
ifconfig eth0 192.168.137.184
telnetd
mknod /dev/sgled c 240 3
insmod /lib/modules/2.6.28.10/sgled.ko
httpd -h /var/www
/var/www/cgi-bin/loopcheck.exe &

# for item in `cat /proc/devices`;
# do
#   if [ $item = sfr ]; then
#     mknod /dev/sfr c 255 1
```

loopcheck.exeの実行時に"&"をオプションに入れないと、本行で停止し、ログオン画面まで進行しないので注意。

## RAMディスク仕様への書換(17)

30. loopcheck.cおよびagent.cのファイルオープンがフルパス指定にしないと動作しないのでこれに書き換える。コンパイル後。コピーする。

```
user01@localhost.localdomain: /mnt/var/www/cgi-bin
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost cgi-bin]# ls
agent2.exe* loopcheck.exe* select.txt sgledtest*
[root@localhost cgi-bin]#
```

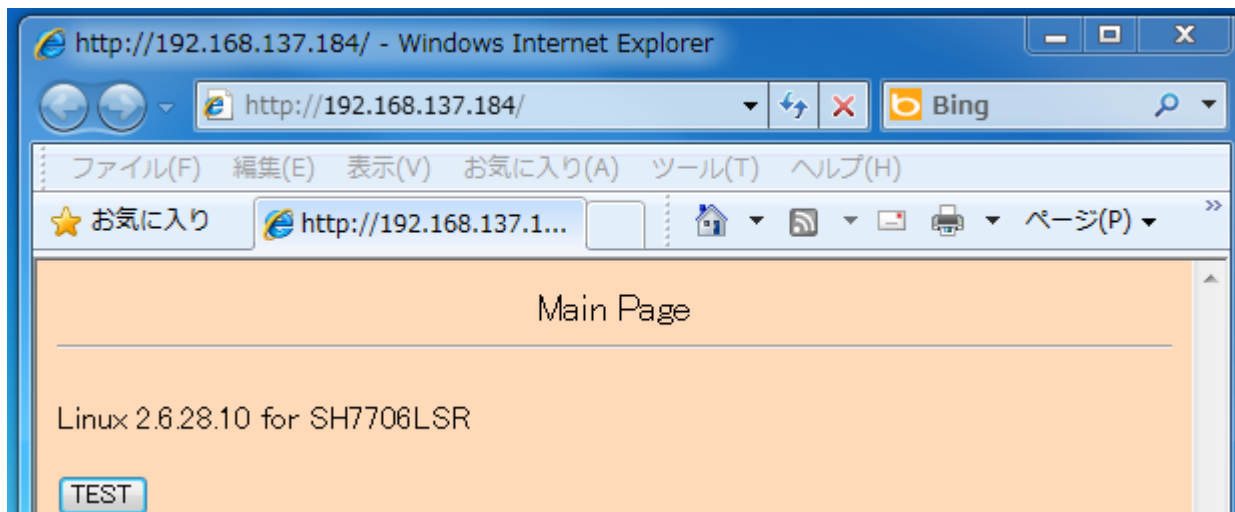
31. 内容を保存後、アンマウントし、SDカードを取り出しマイコンを起動する。  
telnet接続後 #psにて起動確認する。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
~ # ps
PID  USER  COMMAND
  1  root   init
  2  root   [kthreadd]

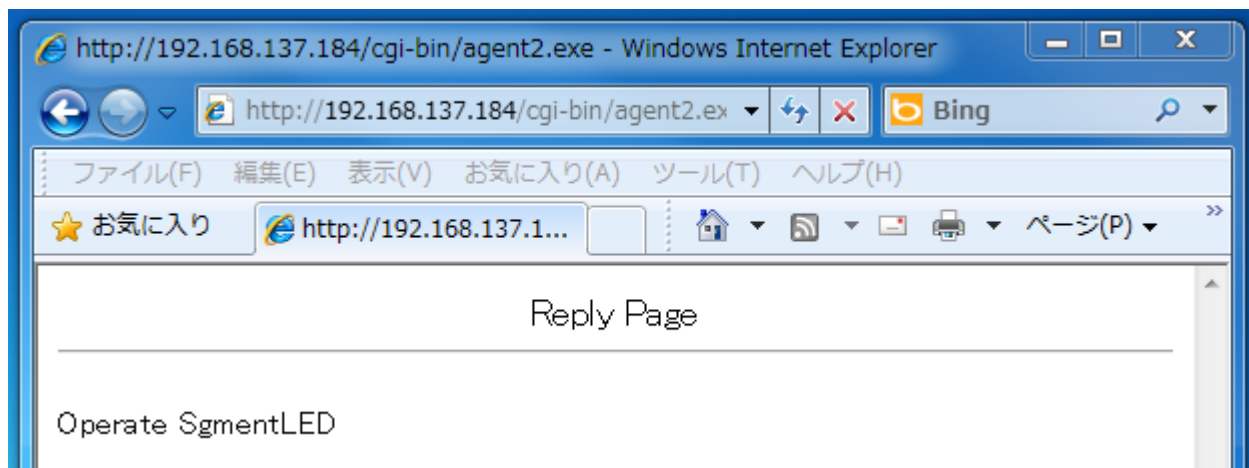
 39  root   telnetd
 43  root   httpd -h /var/www
 44  root   /var/www/cgi-bin/loopcheck.exe
 45  root   /sbin/getty -L ttySC1 115200 vt100
 46  root   -sh
 48  root   ps
~ # █
```

## RAMディスク仕様への書換(18)

31. Webブラウザを起動し、192.168.137.184に接続する。Proxy設定の場合は設定を解除する。



32. 「TEST」をクリックして下記画面とLEDのカウントを確認する。



## RAMディスク仕様への書換(19)

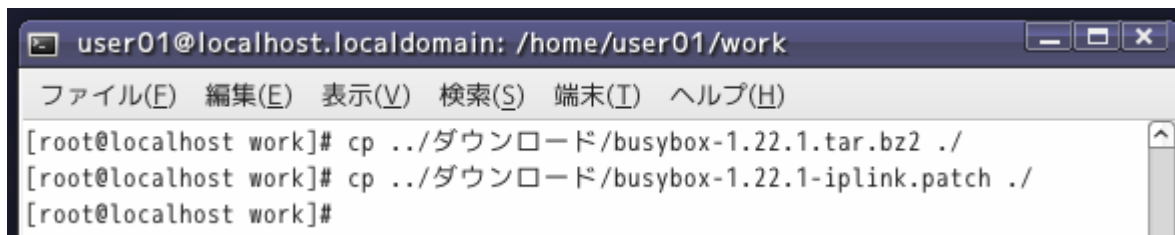
33. RAMをベースとしたファイルシステムのマイコンシステムは、このようにrootfs.imgの8Mバイトと、Linuxカーネルの4Mバイトを含めても12Mバイトのシステムとなる。一方、SDカードの容量は20Mバイトで十分だが、逆に1Gバイト以下のSDカード入手するほうが難しい状況である。
- 有り余るSDカードのメモリ容量をたとえば幾種類かのrootfs.imgを準備しておき、boot.exe でブートするrootfs.imgを切り替えるなど考えればよいかもしれない。

# Busyboxコマンドによる容量圧縮(1)

Busyboxの最小限必要な機能とsgledtest.c,loopcheck.cをソースサンプルとするbusyboxコマンドでのプログラム容量の圧縮を紹介する。

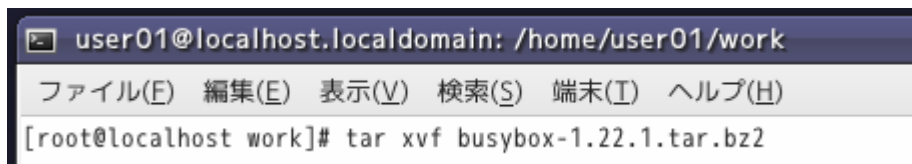
以下、<http://gihyo.jp/dev/serial/01/micom-linux/0005> の記事を参考にする。

1. 上記サイト記事は古いバージョンには対応していないので当記事作成時の最新版をダウンロードする。本記事作成時現在のバージョンは1.22.1であり、busybox-1.22.1.tar.bz2,busybox-1.22.1-iplink.patch をダウンロードし、所定のフォルダにコピーまたは移動する。以下の例では work/フォルダにコピーする。

A terminal window titled 'user01@localhost.localdomain: /home/user01/work'. The menu bar shows 'ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)'. The command history shows: [root@localhost work]# cp ../ダウンロード/busybox-1.22.1.tar.bz2 ./, [root@localhost work]# cp ../ダウンロード/busybox-1.22.1-iplink.patch ./, and [root@localhost work]#. The terminal has a scrollbar on the right side.

```
user01@localhost.localdomain: /home/user01/work
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost work]# cp ../ダウンロード/busybox-1.22.1.tar.bz2 ./
[root@localhost work]# cp ../ダウンロード/busybox-1.22.1-iplink.patch ./
[root@localhost work]#
```

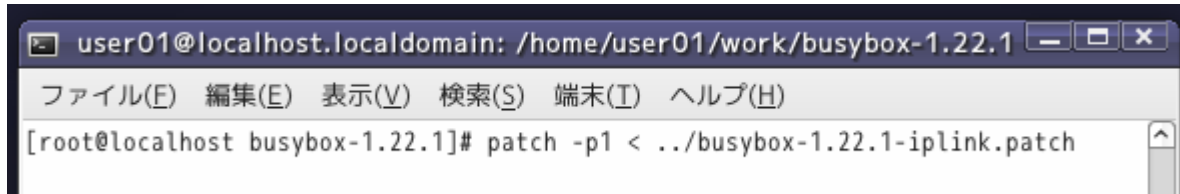
2. busyboxを展開する。

A terminal window titled 'user01@localhost.localdomain: /home/user01/work'. The menu bar shows 'ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)'. The command history shows: [root@localhost work]# tar xvf busybox-1.22.1.tar.bz2. The terminal has a scrollbar on the right side.

```
user01@localhost.localdomain: /home/user01/work
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost work]# tar xvf busybox-1.22.1.tar.bz2
```

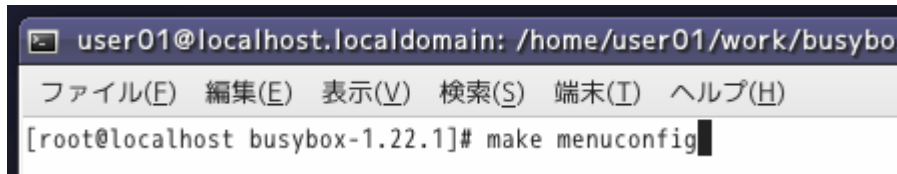
## Busyboxコマンドによる容量圧縮(2)

3. 展開したbusyboxフォルダに移り、パッチを当てる。



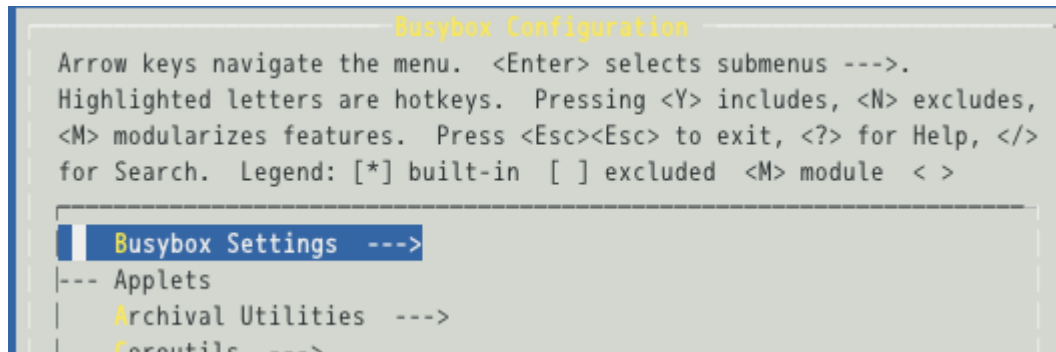
```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# patch -p1 < ../busybox-1.22.1-iplink.patch
```

4. # make menuconfig にて設定条件を確認する。



```
user01@localhost.localdomain: /home/user01/work/busybo
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make menuconfig
```

5. Busybox Settings から



```
Busybox Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

Busybox Settings --->
--- Applets
| Archival Utilities --->
| Coreutils --->
```

# Busyboxコマンドによる容量圧縮(3)

## 6. Build Optionsを選択

```
Busybox Settings
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

General Configuration --->
Build Options --->
Debugging Options --->
```

## 9. () Cross Compiler prefixを選択

```
[ ] Build BusyBox as a static binary (no shared libs) (NEW)
[ ] Build BusyBox as a position independent executable (NEW)
[ ] Force NOMMU build (NEW)
[ ] Build shared libbusybox (NEW)
[*] Build with Large File Support (for accessing files > 2 GB) (NEW)
() Cross Compiler prefix (NEW)
() Path to sysroot (NEW)
```

## 10. sh3-linux- と入力。<OK>

```
Cross Compiler prefix
Please enter a string value. Use the <TAB> key to move from
field to the buttons below it.

sh3-linux-

< Ok > < Help >
```



# Busyboxコマンドによる容量圧縮(4)

11. (sh3-linux-)の内容を確認する。

```
[ ] Build shared libbusybox
[ ] Build with Large File Support (for accessing files > 2 GB)
(sh3-linux) Cross Compiler prefix
[ ] Additional CFLAGS
```

12. <Exit>でBusybox Settingsに戻ってinstallation Optionsにて

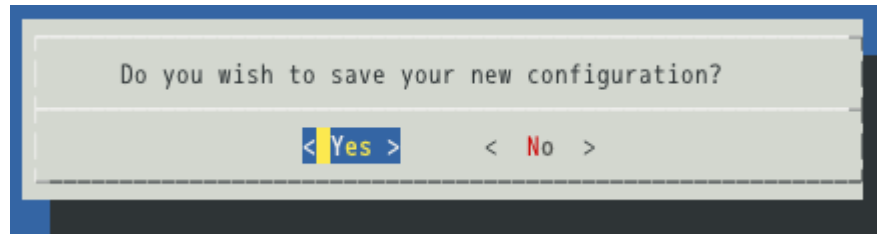
```
General Configuration --->
Build Options --->
Debugging Options --->
Installation Options --->
Busybox Library Tuning --->
```

13. デフォルトで\_installといったフォルダに作成されることを確認する。

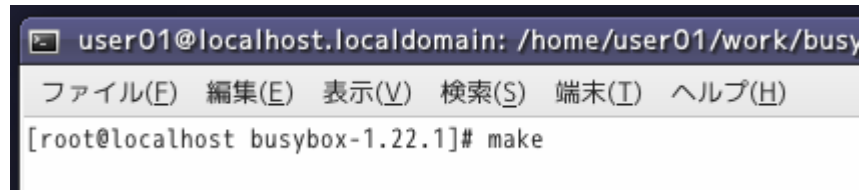
```
What kind of applet links to install (as soft-links) --->
(./_install) BusyBox installation prefix (NEW)
```

# Busyboxコマンドによる容量圧縮(5)

14. <Exit><Exit><Exit>を数回繰り返して、<yes>で保存する。



15. #make を実行



16. ionice.oでエラーになるので #make menuconfig でMiscellanc.....内のioniceのチェックをはずす。

```
ction)
make[1]: *** [miscutils/ionice.o] エラー 1
make: *** [miscutils] エラー 2
[root@localhost busybox-1.22.1]#
```

```
[ ] flash_unlock
[ ] flash_eraseall
[ ] ionice
[ ] inotifyd
[*] last
```

# Busyboxコマンドによる容量圧縮(6)

17保存後、再度 #make を実行

```
user01@localhost.localdomain: /home/user01/work/busybox
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make
```

18. nandwrite.oでエラーになるので #make menuconfig でMiscellanc.....内の nandwriteおよびnanddumpのチェックをはずす。

```
miscutils/nandwrite.c:110: warning: unused variable 'oob'
make[1]: *** [miscutils/nandwrite.o] エラー 1
make: *** [miscutils] エラー 2
[root@localhost busybox-1.22.1]#
```

```
^(-)
[*] Use 'tell me cursor position' ESC sequence to measure window
[*] Enable flag changes ('-' command)
[*] Enable dynamic switching of line numbers
[ ] nandwrite
[ ] nanddump
[ ] rfskill
[*] ...
```

# Busyboxコマンドによる容量圧縮(7)

## 19. 保存後、再度 #make を実行

```
user01@localhost.localdomain: /home/user01/work/busybox
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make
```

## 20. ubi\_tool.oでエラーになるので #make menuconfig でMiscellanc.....内のubi\*\*\*のチェックをはずす。

```
function)
make[1]: *** [miscutils/ubi_tools.o] エラー 1
make: *** [miscutils] エラー 2
[root@localhost busybox-1.22.1]#
```

```
[ ] ytkill
[*] setserial
[ ] ubiattach
[ ] ubidetach
[ ] ubimkvol
[ ] ubirmvol
[ ] ubirsvol
[ ] ubiupdatevol
[*] wall
```

# Busyboxコマンドによる容量圧縮(8)

## 21. 保存後、再度 #make を実行

```
user01@localhost.localdomain: /home/user01/work/busybox
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make
```

## 22. udhcp.oでエラーになるので #make menuconfig でnetworking.....内のudhcp server、udhcp clientのチェックをはずす。

```
make[1]: *** [networking/udhcp/dhccp.o] エラー 1
make: *** [networking/udhcp] エラー 2
[root@localhost busybox-1.22.1]#
```

```
^(-)-----
[*] tunctl
[*] Support owner:group assignment
[ ] udhcp client for DHCPv6 (udhccp6)
[ ] udhcp server (udhcpd)
[ ] udhcp client (udhcpc)
[*] udpsvd
[*] vconfig
[*] wget
[*] Enable a nifty process meter (+2k)
```

## Busyboxコマンドによる容量圧縮(9)

23. SH3マイコンに対応していない機能ははずせば、エラーなく終了する。

```
Trying libraries: crypt m
Library crypt is not needed, excluding it
Library m is needed, can't exclude it (yet)
Final link with: m
DOC    busybox.pod
DOC    BusyBox.txt
DOC    busybox.1
DOC    BusyBox.html
[root@localhost busybox-1.22.1]#
```

このあと、何度か#make defconfig で何度かデフォルト設定を繰り返すので、この機能ははずしておく。また、デフォルト設定ではほとんどの機能が含まれているので、最終的には容量を減らす上で不必要な機能ははずしておくべきである。ここではデフォルト機能を含んだままの設定する。

24. 続けて # make install を実行

```
user01@localhost.localdomain: /home/user01/work/busybox
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make install
```

# Busyboxコマンドによる容量圧縮(10)

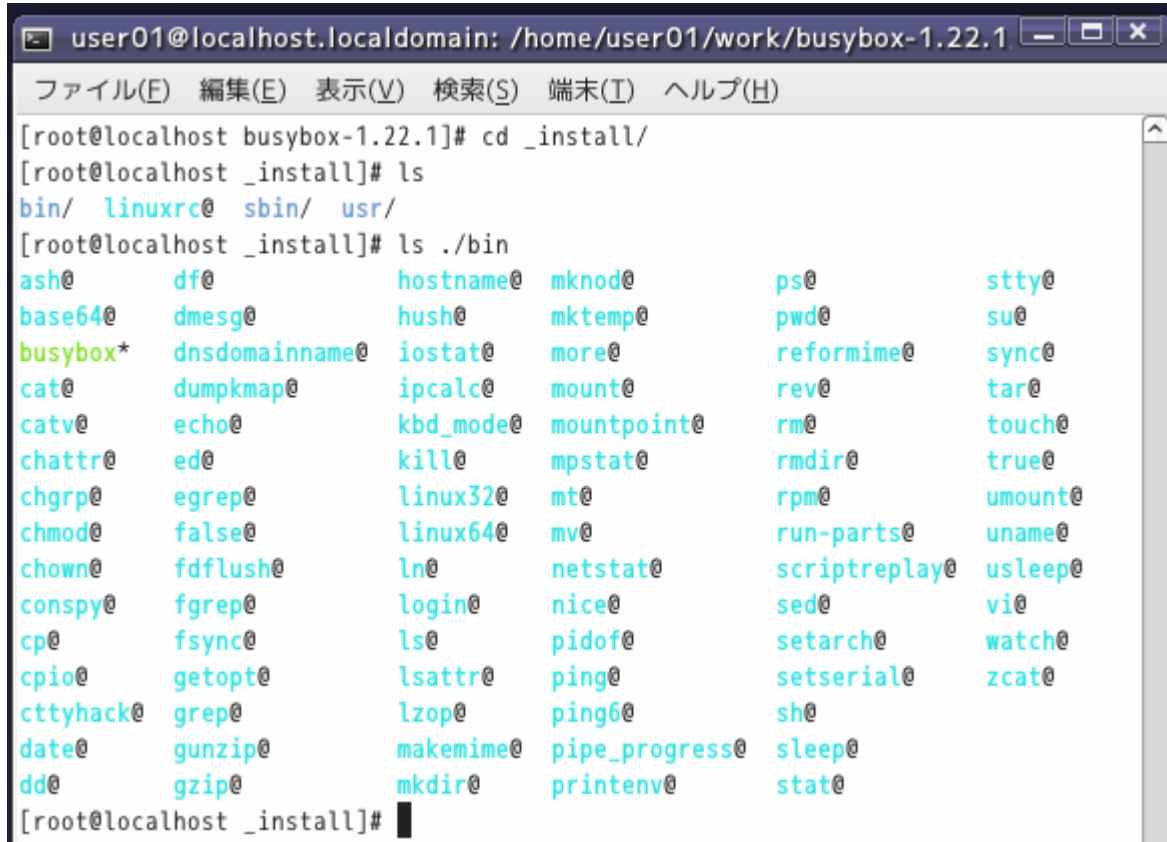
25. Busyboxフォルダに #make install により \_install/ フォルダが作成される。

```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make install
```

```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# ls
AUTHORS          applets/        debianutils/    miscutils/
Config.in        applets_sh/     docs/           modutils/
INSTALL          arch/           e2fsprogs/     networking/
LICENSE          archival/       editors/        printutils/
Makefile         busybox*        examples/      procps/
Makefile.custom  busybox.links   findutils/     runit/
Makefile.flags   busybox_unstripped* include/        scripts/
Makefile.help    busybox_unstripped.map init/           selinux/
README           busybox_unstripped.out libbb/          shell/
TODO             configs/        libpwdgrp/     sysklogd/
TODO_unicode     console-tools/ loginutils/     testsuite/
_install/        coreutils/      mailutils/     util-linux/
[root@localhost busybox-1.22.1]#
```

# Busyboxコマンドによる容量圧縮(11)

26. `_install/` の中はbusyboxへのリンクだけでできたコマンド体系である。



```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# cd _install/
[root@localhost _install]# ls
bin/ linuxrc@ sbin/ usr/
[root@localhost _install]# ls ./bin
ash@      df@      hostname@ mknod@   ps@      stty@
base64@   dmesg@   hush@     mktemp@  pwd@     su@
busybox*  dnsdomainname@ iostat@  more@    reformime@ sync@
cat@      dumpkmap@ ipcalc@  mount@   rev@     tar@
catv@     echo@    kbd_mode@ mountpoint@ rm@      touch@
chattr@   ed@      kill@     mpstat@  rmdir@  true@
chgrp@    egrep@   linux32@ mt@      rpm@     umount@
chmod@    false@   linux64@ mv@      run-parts@ uname@
chown@    fdflush@ ln@       netstat@ scriptreplay@ usleep@
consp@    fgrep@   login@    nice@    sed@     vi@
cp@       fsync@   ls@       pidof@   setarch@ watch@
cpio@     getopt@  lsattr@  ping@    setserial@ zcat@
cttyhack@ grep@    lzop@    ping6@   sh@
date@     gunzip@  makemime@ pipe_progress@ sleep@
dd@       gzip@    mkdir@   printenv@ stat@
[root@localhost _install]#
```

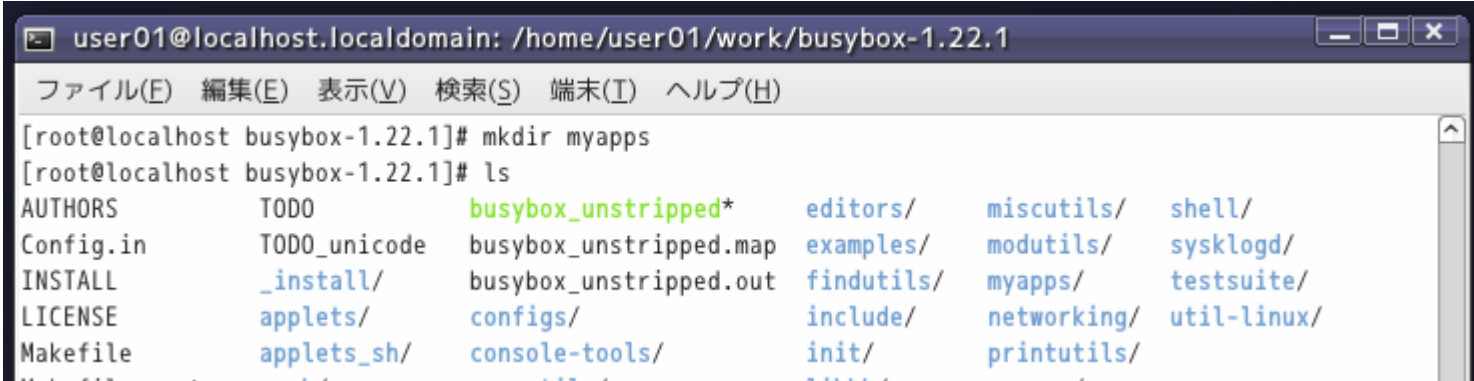
作成された `/bin /sbin /usr` を、現行の `/bin /sbin /usr` のものと入れ替えると busybox のみで構成されたシステムとなる。



## Busyboxコマンドによる容量圧縮(12)

27. busyboxのデフォルト設定でエラーなく作成できたことを確認できたので次にsgledtestをbusyboxインサイドコマンドとして作成する。

busyboxフォルダに戻り、新規なカテゴリとして “myapps”フォルダを作成する。



```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost busybox-1.22.1]# mkdir myapps
[root@localhost busybox-1.22.1]# ls
AUTHORS          TODO             busybox_unstripped*  editors/         miscutils/       shell/
Config.in        TODO_unicode    busybox_unstripped.map  examples/        modutils/         sysklogd/
INSTALL          _install/       busybox_unstripped.out  findutils/       myapps/           testsuite/
LICENSE          applets/        configs/               include/         networking/       util-linux/
Makefile         applets_sh/     console-tools/         init/            printutils/
..              .              ..
```

28. myappsフォルダに移り、sgledtest.c ソースコードをコピーする。



```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1/m
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost busybox-1.22.1]# cd myapps
[root@localhost myapps]# cp /public/KNL_work/sgled/sgledtest.c ./
```

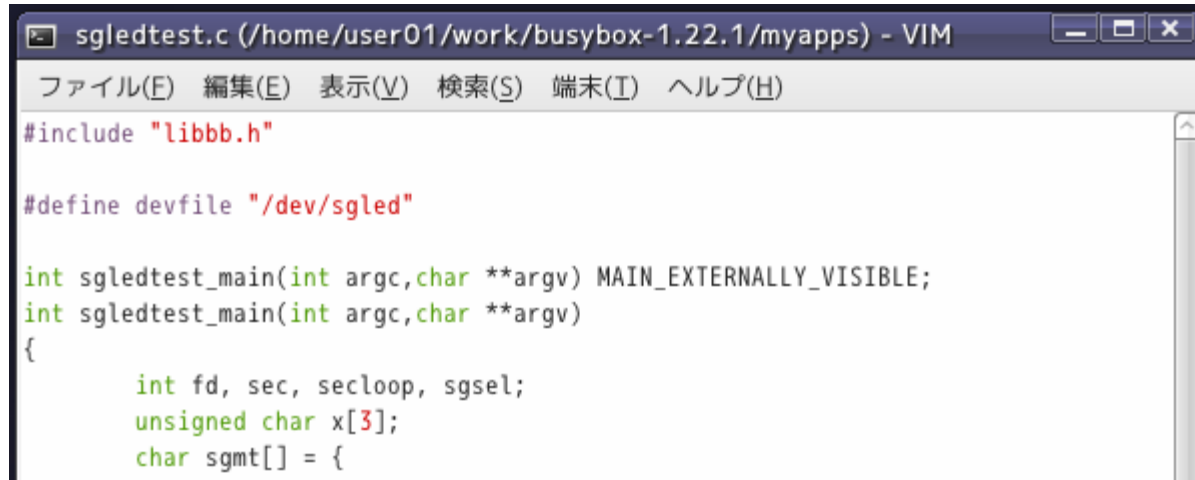
29. BusyBox向けにソースコードを書き換える。



```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost myapps]# vi sgledtest.c
```

## Busyboxコマンドによる容量圧縮(13)

30. 既存のヘッダ定義を取り除き、新たにlibbb.hを定義し、main関数をコマンド名\_mainに変え、さらにMAIN\_EXTERNALLY\_VISIBLE属性を追加する。以下の例ではコマンド名は”sgledtest”となる。



```
sgledtest.c (/home/user01/work/busybox-1.22.1/myapps) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
#include "libbb.h"

#define devfile "/dev/sgled"

int sgledtest_main(int argc, char **argv) MAIN_EXTERNALLY_VISIBLE;
int sgledtest_main(int argc, char **argv)
{
    int fd, sec, secloop, sgsel;
    unsigned char x[3];
    char sgmt[] = {
```

31. returnの成功時の値をEXIT\_SUCCESS、今回のソースコードには失敗時のreturnの記載はないが、ある場合は値をEXIT\_FAILUREにする。

```
        write(fd, x, 3);
        usleep(2000);
    }
    close(fd);

    return EXIT_SUCCESS;
}
```

# Busyboxコマンドによる容量圧縮(14)

32. myappsフォルダから抜け、トップフォルダのConfig.inを編集し、myappsを登録する。

```
user01@localhost.localdomain: /home/user01/work
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost myapps]# cd ..
[root@localhost busybox-1.22.1]# vi Config.in

comment "Applets"

source archival/Config.in
source coreutils/Config.in
source console-tools/Config.in
source debianutils/Config.in
source editors/Config.in
source findutils/Config.in
source init/Config.in
source loginutils/Config.in
source e2fsprogs/Config.in
source modutils/Config.in
source util-linux/Config.in
source miscutils/Config.in
source networking/Config.in
source printutils/Config.in
source mailutils/Config.in
source procs/Config.in
source runit/Config.in
source selinux/Config.in
source shell/Config.in
source syslogd/Config.in

source myapps/Config.in
```

# Busyboxコマンドによる容量圧縮(15)

33. myappsフォルダにConfig.srcファイルを作成する。

```
user01@localhost.localdomain: /home/user01/wo
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# cd myapps/
[root@localhost myapps]# vi Config.src
```

```
Config.src (/home/user01/work/busybox-1.22.1/myapps) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
menu "My Applications"

INSERT

config  SGLEDTEST
        bool "sgledtest"
        default y
        help
            a simple H/W test program.
endmenu
~
```

# Busyboxコマンドによる容量圧縮(16)

34. トップフォルダに移動し、Makefileを編集する。libs-yにmyappsを追加する。

```
user01@localhost.localdomain: /home/user01/work/bu
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost myapps]# cd ..
[root@localhost busybox-1.22.1]# vi Makefile
```

```
Makefile (/home/user01/work/busybox-1.22.1) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)

scripts_basic: include/autoconf.h

# Objects we will link into busybox / subdirs we need to visit
core-y      := \
             applets/ \

libs-y      := \
             archival/ \
             archival/libarchive/ \
             -----

             shell/ \
             syslogd/ \
             util-linux/ \
             util-linux/volume_id/ \
             myapps/\

endif # KBUILD_EXTMOD
```

# Busyboxコマンドによる容量圧縮(17)

35. myappsフォルダ内にてKbuild.srcを編集し、コンパイル実行内容を記述する。

```
user01@localhost.localdomain: /home/user01/work/bus
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# cd myapps
[root@localhost myapps]# vi Kbuild.src
```

```
Kbuild.src + (/home/user01/work/busybox-1.22.1/myap
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
lib-y :=
INSERT
lib-$(CONFIG_SGLED-TEST) += sgledtest.o
~
```

# Busyboxコマンドによる容量圧縮(18)

36. Busyboxフォルダにあるincludeフォルダ内のapplets.src.hにリンクの登録をする。

```
user01@localhost.localdomain: /home/user01/work/b
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost myapps]# cd ../include
[root@localhost include]# vi applets.src.h
```

```
IF_LESS(APPLET(less, BB_DIR_USR_BIN, BB_SUID_DROP))
IF_SETARCH(APPLET_ODDNAME(linux32, setarch, BB_DIR_BIN, BB_SUID_DROP, linux32))
IF_SETARCH(APPLET_ODDNAME(linux64, setarch, BB_DIR_BIN, BB_SUID_DROP, linux64))

IF_SGLEDTEST(APPLET(sgledtest, BB_DIR_BIN, BB_SUID_DROP))

IF_LN(APPLET_NOEXEC(ln, ln, BB_DIR_BIN, BB_SUID_DROP, ln))
IF_LOAD_POLICY(APPLET(load_policy, BB_DIR_USR_SBIN, BB_SUID_DROP))
IF_LOADFONT(APPLET(loadfont, BB_DIR_USR_SBIN, BB_SUID_DROP))
IF_LOADKMAP(APPLET(loadkmap, BB_DIR_SBIN, BB_SUID_DROP))
IF_LOGGER(APPLET(logger, BB_DIR_USR_BIN, BB_SUID_DROP))
/* Needs to be run by root or be suid root - needs to change uid and gid: */
```

第1引数はコマンド名、第2引数は通常の実行ファイルでBB\_DIR\_USR\_BINを指定、第3引数は特権ユーザーの実行が不要なのでBB\_SUID\_DROPとする。

## Busyboxコマンドによる容量圧縮(19)

38. Busyboxフォルダにあるincludeフォルダ内のusage.src.hにヘルプテキストを登録する。ヘルプテキストがないとコンパイルできない。何らかのテキストを入力する。sgledtest\_trivial\_usageは簡単なヘルプテキスト、sgledtest\_full\_usageは詳細なヘルプテキストを記述する。

```
user01@localhost.localdomain: /home/user01/wor
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost include]# vi usage.src.h

INSERT

#define busybox_notes_usage \
    "Hello world!\n"

#define sgledtest_trivial_usage \
    "[times]\n" \
    "A simple H/W test program"
#define sgledtest_full_usage \
    "[times]"

#endi
```



## Busyboxコマンドによる容量圧縮(20)

38. トップフォルダに戻り、`#make defconfig` を実行する。

```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost myapps]# cd ..
[root@localhost busybox-1.22.1]# make defconfig
```

39. `#make menuconfig` を実行し、My Applications を探す。

```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make menuconfig
```

```
^(-)
| Print Utilities --->
| Mail Utilities --->
| Process Utilities --->
| Runit Utilities --->
| Shells --->
| System Logging Utilities --->
| My Applications --->
| ---
| Load an Alternate Configuration File
| Save Configuration to an Alternate File
```

# Busyboxコマンドによる容量圧縮(21)

40. My Applications には sgledtest がデフォルトでチェックが入る。

```
My Applications
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

[*] sgledtest
```

41. <help>のメッセージは

```
sgledtest
CONFIG_SGLEDTEST:
a simple H/W test program.
Symbol: SGLEDTEST [=y]
Prompt: sgledtest
Defined at myapps/Config.in:5
Location:
-> My Applications
```

Config.srcに記述したhelpが現れる。

## Busyboxコマンドによる容量圧縮(22)

42. sh3-linux-(CROSS\_COMPILE=sh3-linux-を指定していれば不要)記述と, 一連のエラー機能ははずして#make を実行する。

```
Cross
Please enter a string value.
field to the buttons below it
-----
|sh3-linux-
|
```

```
[*] Enable flag changes ('-' command)
[*] Enable dynamic switching of line numbers
[ ] nandwrite
[ ] nanddump
[ ] rftkill
```

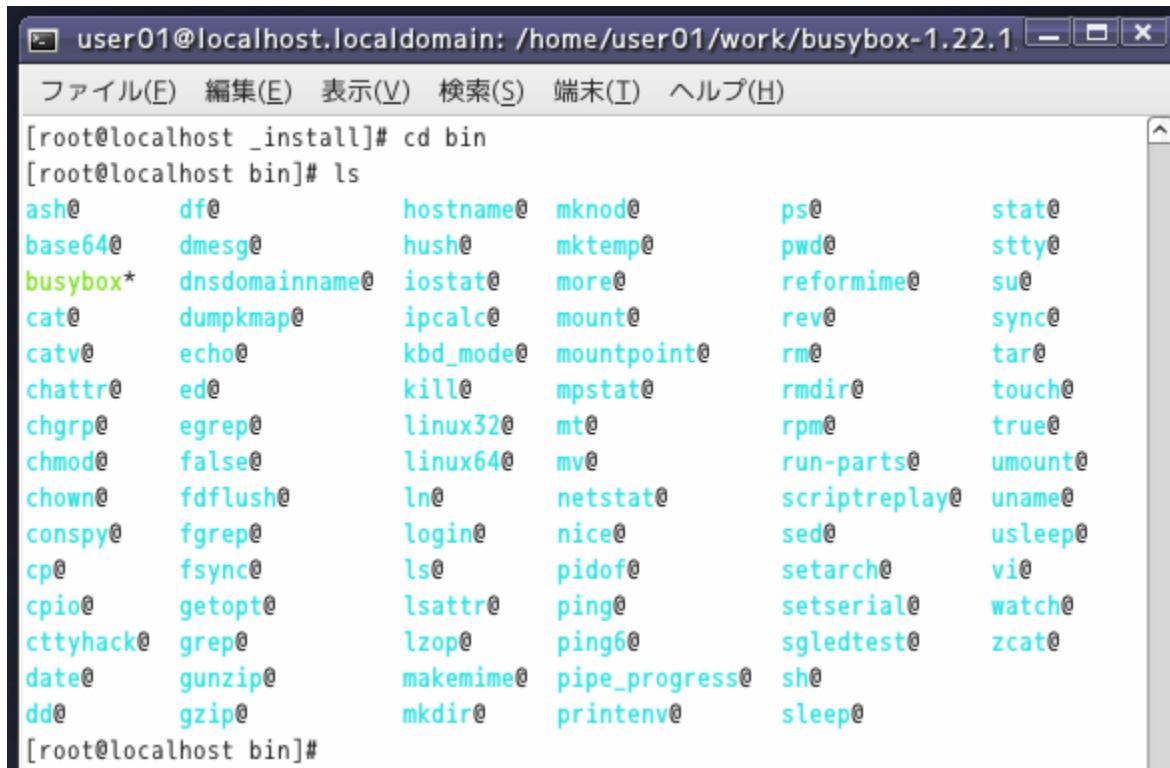
```
user01@localhost.localdomain: /home/user01/work/busy
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# make
```

43. エラーなく終了すれば、#make install を実行すれば、\_installにbusyboxシステムが作成される。

```
user01@localhost.localdomain: /home/user01/work/bu
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost busybox-1.22.1]# cd _install/
[root@localhost _install]# ls
bin/ linuxrc@ sbin/ usr/
[root@localhost _install]#
```

# Busyboxコマンドによる容量圧縮(23)

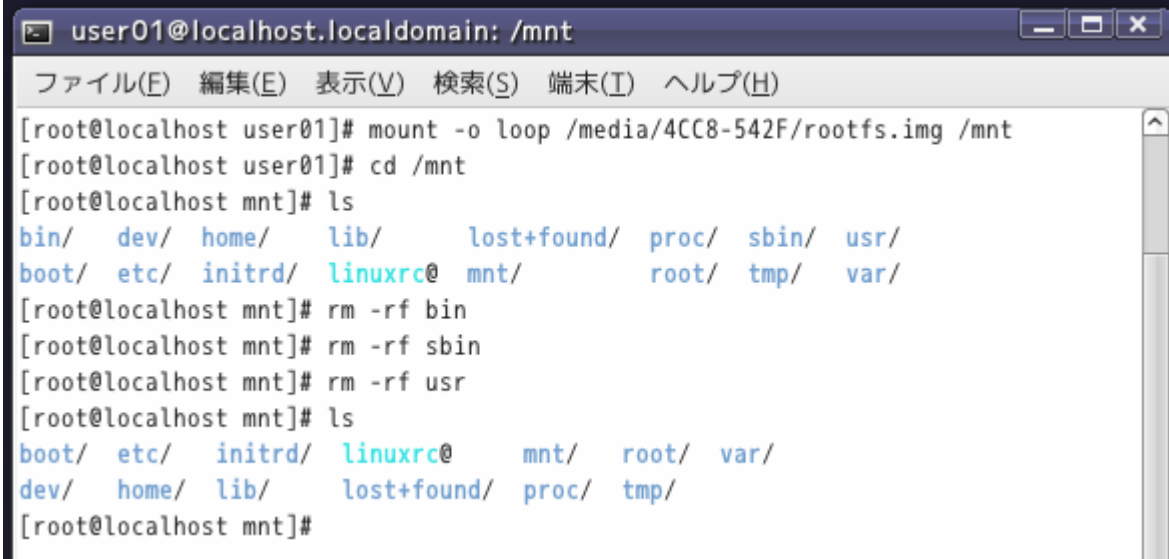
44. 作成された bin/の中に sgledtestコマンドが確認できる。



```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost _install]# cd bin
[root@localhost bin]# ls
ash@      df@      hostname@ mknod@   ps@      stat@
base64@   dmesg@   hush@     mktemp@  pwd@     stty@
busybox*  dnsdomainname@ iostat@  more@    reformime@ su@
cat@      dumpkmap@ ipcalc@   mount@   rev@     sync@
catv@     echo@    kbd_mode@ mountpoint@ rm@      tar@
chattr@   ed@      kill@     mpstat@  rmdir@  touch@
chgrp@    egrep@   linux32@  mt@      rpm@     true@
chmod@    false@   linux64@  mv@      run-parts@ umount@
chown@    fdflush@ ln@       netstat@ scriptreplay@ uname@
conspy@   fgrep@   login@    nice@    sed@     usleep@
cp@       fsync@   ls@       pidof@   setarch@ vi@
cpio@     getopt@  lsattr@   ping@    setserial@ watch@
cttyhack@ grep@    lzop@    ping6@   sgledtest@ zcat@
date@     gunzip@  makemime@ pipe_progress@ sh@
dd@       gzip@    mkdir@    printenv@ sleep@
[root@localhost bin]#
```

## Busyboxコマンドによる容量圧縮(24)

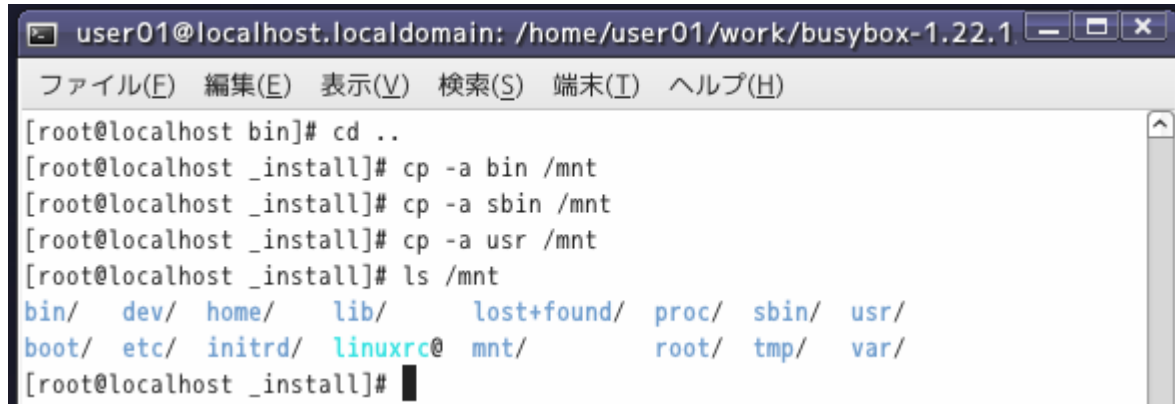
45. 作成された sgledtestコマンド動作を確認するため、SDカードをLinuxマシンに認識させrootfs.img を/mntにマウントし、rootfs.img の中のbin/ ,sbin/, usr/と入れ替えるため消去する。なお以下の例ではSDカードのフォルダ名は"4CC8-542f"で個々に異なる。



```
user01@localhost.localdomain: /mnt
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost user01]# mount -o loop /media/4CC8-542F/rootfs.img /mnt
[root@localhost user01]# cd /mnt
[root@localhost mnt]# ls
bin/  dev/  home/  lib/  lost+found/  proc/  sbin/  usr/
boot/  etc/  initrd/  linuxrc@  mnt/  root/  tmp/  var/
[root@localhost mnt]# rm -rf bin
[root@localhost mnt]# rm -rf sbin
[root@localhost mnt]# rm -rf usr
[root@localhost mnt]# ls
boot/  etc/  initrd/  linuxrc@  mnt/  root/  var/
dev/  home/  lib/  lost+found/  proc/  tmp/
[root@localhost mnt]#
```

## Busyboxコマンドによる容量圧縮(25)

46. Busyboxで作成した bin/,sbin/,usr/フォルダを/mntにコピーする。



```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost bin]# cd ..
[root@localhost _install]# cp -a bin /mnt
[root@localhost _install]# cp -a sbin /mnt
[root@localhost _install]# cp -a usr /mnt
[root@localhost _install]# ls /mnt
bin/  dev/  home/  lib/  lost+found/  proc/  sbin/  usr/
boot/ etc/  initrd/ linuxrc@ mnt/  root/  tmp/  var/
[root@localhost _install]#
```

47. アンマウントし、SDカードを取り出し、マイコンに差込み起動させる。rootfs.imgの入れ替えたフォルダ以外はそのままなのでドライバの読込およびIPアドレスも自動設定され、Linuxマシンからtelnet接続する。

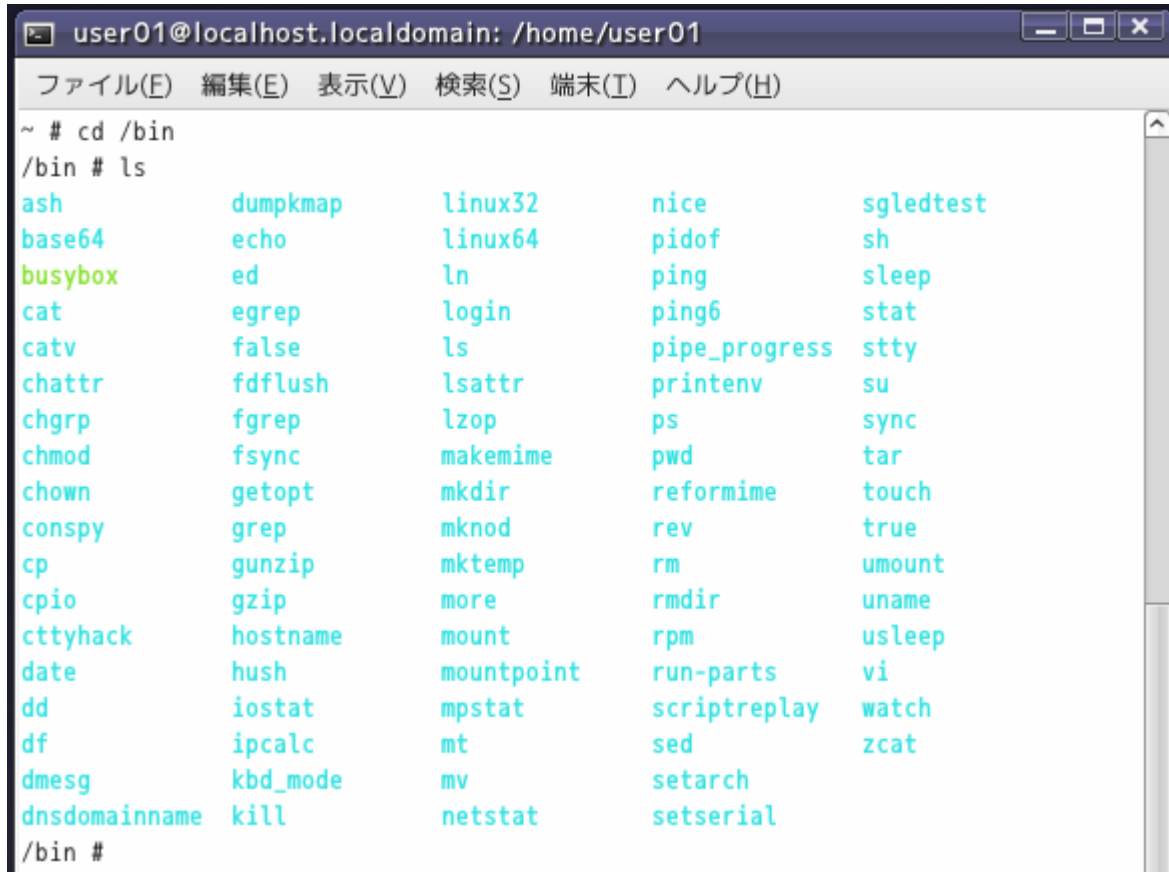


```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.

shlinux login: root
Password:
-sh: can't access tty; job control turned off
~ #
```

# Busyboxコマンドによる容量圧縮(26)

48. bin/の中にsgledtestが リンクコマンドとして確認できる



```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
~ # cd /bin
/bin # ls
ash          dumpkmap    linux32     nice        sgledtest
base64       echo        linux64     pidof       sh
busybox      ed          ln          ping        sleep
cat          egrep       login       ping6       stat
catv         false       ls          pipe_progress stty
chattr       fdflush     lsattr      printenv    su
chgrp        fgrep       lzop        ps          sync
chmod        fsync       makemime    pwd         tar
chown        getopt      mkdir       reformime   touch
conspy       grep        mknod       rev         true
cp           gunzip      mktemp      rm          umount
cpio         gzip        more        rmdir       uname
cttyhack     hostname    mount       rpm         usleep
date         hush        mountpoint  run-parts   vi
dd           iostat      mpstat      scriptreplay watch
df           ipcalc      mt          sed         zcat
dmesg        kbd_mode    mv          setarch
dnsdomainname kill         netstat     setserial
/bin #
```

## Busyboxコマンドによる容量圧縮(27)

49. # ./sgledtest を実行。セグメントLEDにカウント表示が確認できる。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
/bin # ./sgledtest
```



50. sgledtestと同様にloopcheck.cについてもコマンド化する。コマンド名はloopcheckとする。loopcheck.cをmyappsフォルダにコピーする。

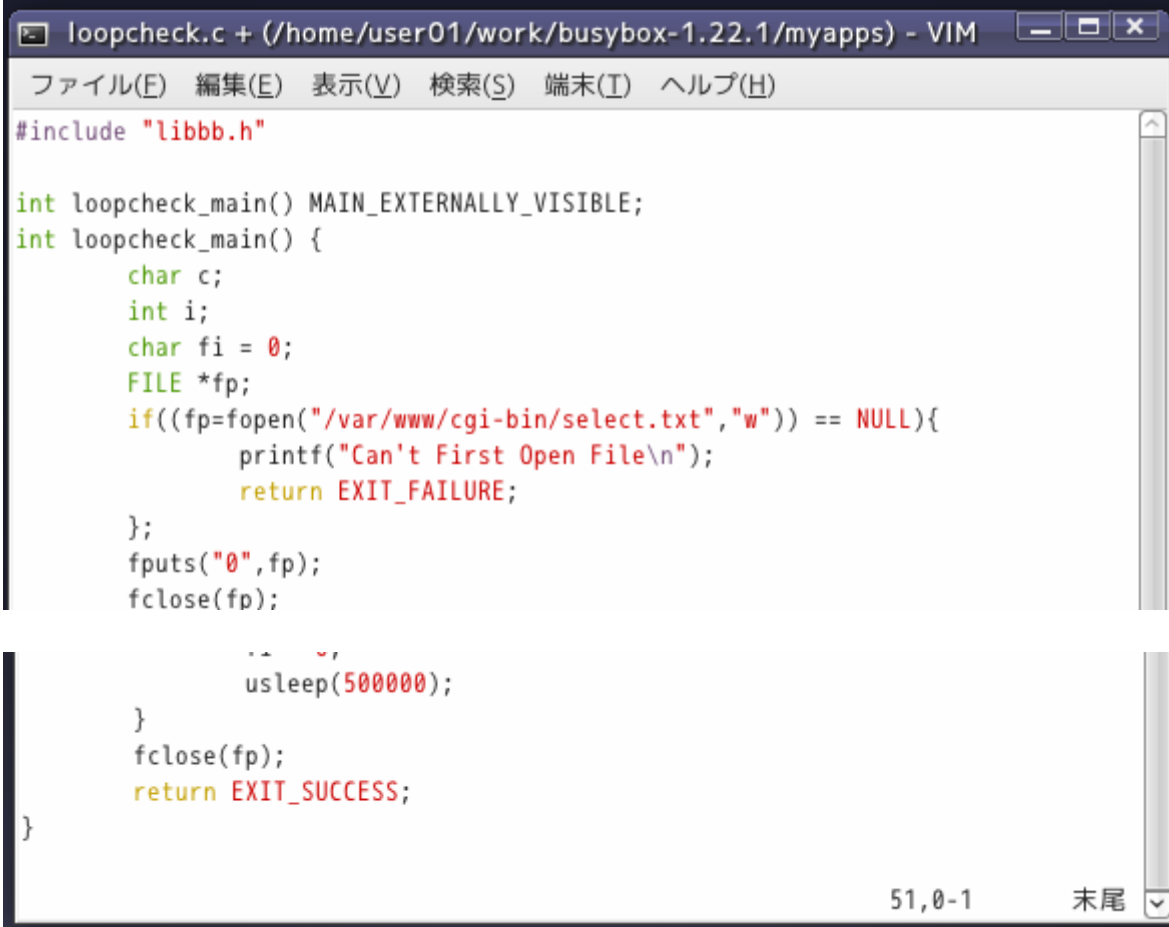
```
user01@localhost.localdomain: /home/user01/work/busybox-1.2
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost myapps]# cp /public/loopcheck/loopcheck.c ./
[root@localhost myapps]#
```



## Busyboxコマンドによる容量圧縮(28)

51. loopcheck.cを編集する。既存のヘッダ定義を取り除き、新たにlibbb.hを定義し、main関数をコマンド名\_mainに変え、さらにMAIN\_EXTERNALLY\_VISIBLE属性を追加する。以下の例ではコマンド名は"loopcheck"となる。

returnの成功時の値をEXIT\_SUCCESS、失敗時をEXIT\_FAILUREにする。



```
loopcheck.c + (/home/user01/work/busybox-1.22.1/myapps) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
#include "libbb.h"

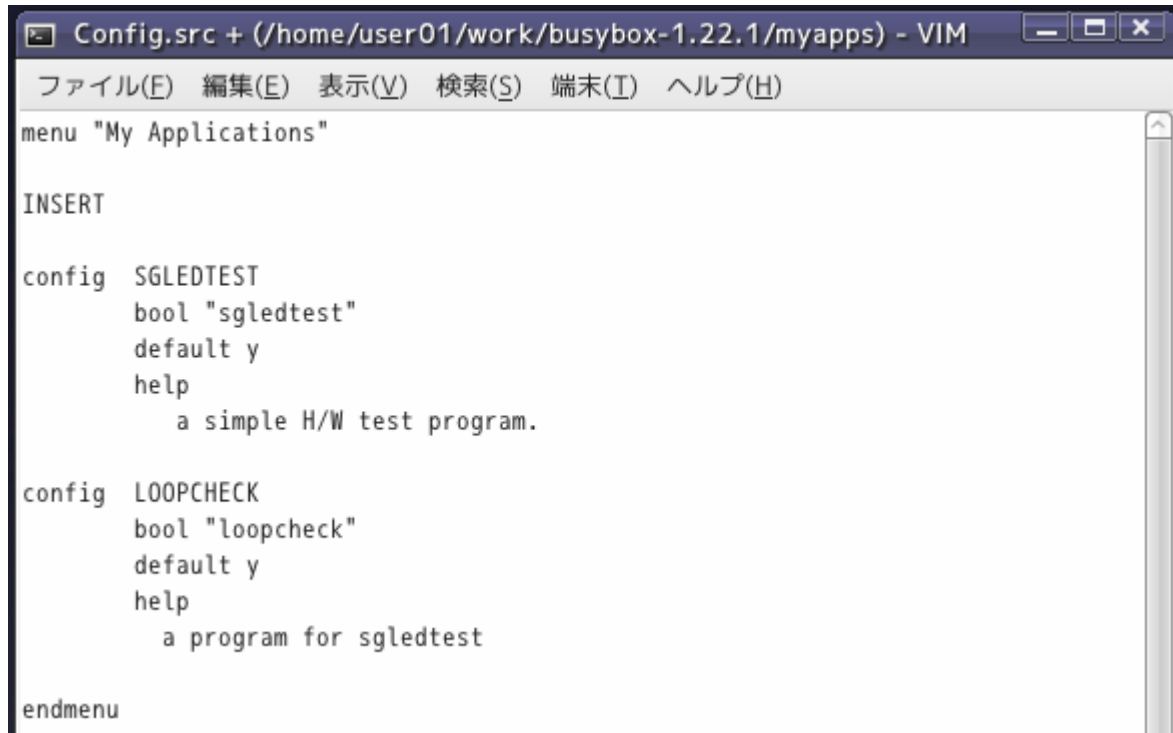
int loopcheck_main() MAIN_EXTERNALLY_VISIBLE;
int loopcheck_main() {
    char c;
    int i;
    char fi = 0;
    FILE *fp;
    if((fp=fopen("/var/www/cgi-bin/select.txt","w")) == NULL){
        printf("Can't First Open File\n");
        return EXIT_FAILURE;
    };
    fputs("0", fp);
    fclose(fp);

    usleep(500000);
}
fclose(fp);
return EXIT_SUCCESS;
}
```

51,0-1 末尾

## Busyboxコマンドによる容量圧縮(29)

52. myappsフォルダ内のConfig.srcを編集する。SGLEDTESTの下に追記する。



```
Config.src + (/home/user01/work/busybox-1.22.1/myapps) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
menu "My Applications"

INSERT

config SGLEDTEST
    bool "sgledtest"
    default y
    help
        a simple H/W test program.

config LOOPCHECK
    bool "loopcheck"
    default y
    help
        a program for sgledtest

endmenu
```

# Busyboxコマンドによる容量圧縮(30)

52. myappsフォルダ内のKbuild.srcを編集する。CONFIG\_SGLEDTESTの下に追記する。



```
Kbuild.src + (/home/user01/work/busybox-1.22.1/myapps) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
lib-y :=
INSERT
lib-$(CONFIG_SGLEDTEST) += sgledtest.o
lib-$(CONFIG_LOOPCHECK) += loopcheck.o
~
```

53. Includeフォルダのapplets.src.hにリンク内容を追加する。

```
IF_SETARCH(APPLET_ODDNAME(linux64, setarch, BB_DIR_BIN, BB_SUID_DROP, linux64))

IF_SGLEDTEST(APPLET(sgledtest, BB_DIR_BIN, BB_SUID_DROP))
IF_LOOPCHECK(APPLET(loopcheck, BB_DIR_BIN, BB_SUID_DROP))

IF_LN(APPLET_NOEXEC(ln, ln, BB_DIR_BIN, BB_SUID_DROP, ln))
IF_LOAD_POLICY(APPLET(load_policy, BB_DIR_USR_SBIN, BB_SUID_DROP))
IF_LOADFONT(APPLET(loadfont, BB_DIR_USR_SBIN, BB_SUID_DROP))
IF_LOADKMAP(APPLET(loadkmap, BB_DIR_USR_SBIN, BB_SUID_DROP))
```

# Busyboxコマンドによる容量圧縮(31)

54. Includeフォルダのusage.src.hにヘルプテキストを登録する。

```
usage.src.h + (/home/user01/work/busybox-1.22.1/include) - V
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)

#define NOUSAGE_STR "\b"

INSERT

#define busybox_notes_usage \
    "Hello world!\n"

#define sgledtest_trivial_usage \
    "[times]\n" \
    "A simple H/W test program"
#define sgledtest_full_usage \
    "[times]"

#define loopcheck_trivial_usage \
    "[times]\n" \
    "A program for sgledtest"
#define loopcheck_full_usage \
    "[times]"

#endif
```

## Busyboxコマンドによる容量圧縮(32)

55. #make defconfig のあと #make menuconfig で登録内容を確認後、既定の処理をおこない#makeを実行する。

```
logger (LOGGER) [Y/n/?] (NEW) y
*
* My Applications
*
sgledtest (SGLEDTEST) [Y/n/?] (NEW) y
loopcheck (LOOPCHECK) [Y/n/?] (NEW) y
[root@localhost busybox-1.22.1]#
```

```
for search: legend [ ] state on [ ] exclude all modes [ ]
[*] sgledtest
[*] loopcheck
```

56. #make installののち\_installフォルダ内のbin/にloopcheckがあることを確認する。

```
busybox*  dnsdomainname@  iostat@         mktemp@         pwd@            stty@
cat@      dumpkmap@       ipcalc@         more@           reformime@      su@
catv@     echo@           kbd_mode@      mount@          rev@            sync@
chattr@   ed@             kill@           mountpoint@     rm@             tar@
chgrp@    egrep@          linux32@        mpstat@         rmdir@          touch@
chmod@    false@          linux64@        mt@             rpm@            true@
chown@    fdflush@        ln@             mv@             run-parts@      umount@
conspy@   fgrep@          login@          netstat@        scriptreplay@   uname@
cp@       fsync@          loopcheck@      nice@           sed@            usleep@
--
```

## Busyboxコマンドによる容量圧縮(33)

57. SDカードをLinuxマシンに認識させrootfs.imgを/mntにマウントする。rootfs.imgの中のbin/,sbin/,usr/と入れ替えるため消去し、\_installのものをコピーする。

```
user01@localhost.localdomain: /mnt
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost user01]# mount -o loop /media/4CC8-542F/rootfs.img /mnt
[root@localhost user01]# cd /mnt
[root@localhost mnt]# ls
bin/  dev/  home/  lib/      lost+found/  proc/  sbin/  usr/
boot/ etc/  initrd/ linuxrc@ mnt/      root/  tmp/  var/
[root@localhost mnt]# rm -rf bin
[root@localhost mnt]# rm -rf sbin
[root@localhost mnt]# rm -rf usr
[root@localhost mnt]# ls
boot/  etc/  initrd/  linuxrc@  mnt/  root/  var/
dev/   home/  lib/     lost+found/  proc/  tmp/
[root@localhost mnt]#
```

```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost _install]# ls
bin/  linuxrc@  sbin/  usr/
[root@localhost _install]# cp -a bin /mnt
[root@localhost _install]# cp -a sbin /mnt
[root@localhost _install]# cp -a usr /mnt
```

# Busyboxコマンドによる容量圧縮(34)

58. SDカード(rootfs.img)のbin/ にloopcheckが確認できる。

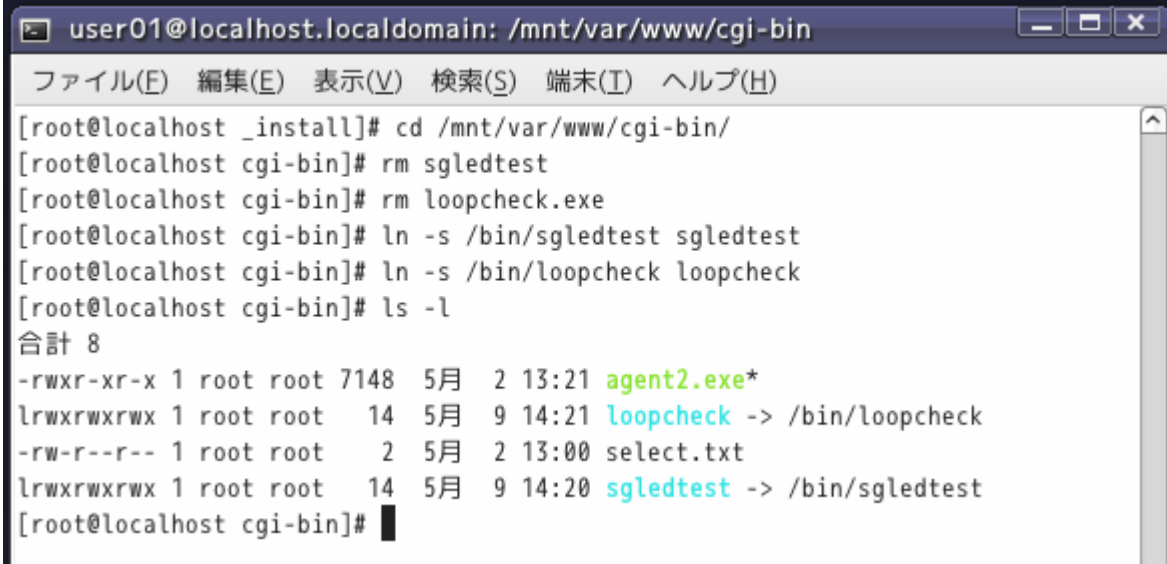


```
user01@localhost.localdomain: /home/user01/work/busybox-1.22.1
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(I) ヘルプ(H)
[root@localhost _install]# ls
bin/ linuxrc@ sbin/ usr/
[root@localhost _install]# cp -a bin /mnt
[root@localhost _install]# cp -a sbin /mnt
[root@localhost _install]# cp -a usr /mnt
[root@localhost _install]# ls /mnt/bin
ash@      df@      hostname@ mkdir@     printenv@ sleep@
base64@   dmesg@   hush@     mknod@    ps@       stat@
busybox*  dnsdomainname@ iostat@  mktemp@   pwd@      stty@
cat@      dumpkmap@ ipcalc@   more@     reformime@ su@
catv@     echo@    kbd_mode@ mount@     rev@      sync@
chattr@   ed@      kill@     mountpoint@ rm@       tar@
chgrp@    egrep@   linux32@ mpstat@   rmdir@   touch@
chmod@    false@   linux64@ mt@       rpm@     true@
chown@    fdflush@ ln@       mv@       run-parts@ umount@
conspy@   fgrep@   login@    netstat@  scriptreplay@ uname@
cp@       fsync@   loopcheck@ nice@     sed@     usleep@
cpio@     getopt@  ls@       pidof@    setarch@ vi@
cttyhack@ grep@    lsattr@   ping@     setserial@ watch@
date@     gunzip@  lzop@     ping6@    sgledtest@ zcat@
dd@       gzip@    makemime@ pipe_progress@ sh@
[root@localhost _install]#
```

## Busyboxコマンドによる容量圧縮(35)

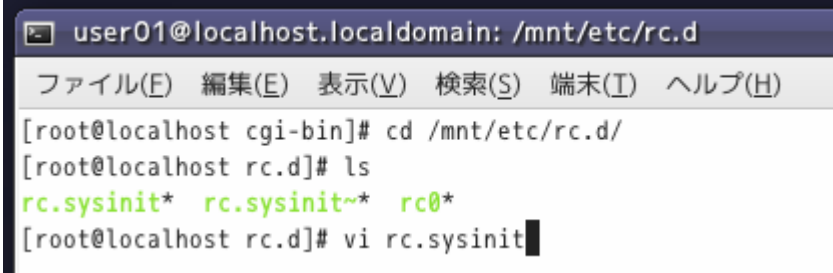
59. SDカード(rootfs.img)の/mnt/var/www/cgi-bin/の実行ファイルを消去し、/binの中のbusyboxへのリンクに書き換える。

agent2.exeについてはhtmlテキストがあるため、自由な編集ができるようコマンド化はやめておく。



```
user01@localhost.localdomain: /mnt/var/www/cgi-bin
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost _install]# cd /mnt/var/www/cgi-bin/
[root@localhost cgi-bin]# rm sgledtest
[root@localhost cgi-bin]# rm loopcheck.exe
[root@localhost cgi-bin]# ln -s /bin/sgledtest sgledtest
[root@localhost cgi-bin]# ln -s /bin/loopcheck loopcheck
[root@localhost cgi-bin]# ls -l
合計 8
-rwxr-xr-x 1 root root 7148  5月  2 13:21 agent2.exe*
lrwxrwxrwx 1 root root  14  5月  9 14:21 loopcheck -> /bin/loopcheck
-rw-r--r-- 1 root root  2  5月  2 13:00 select.txt
lrwxrwxrwx 1 root root  14  5月  9 14:20 sgledtest -> /bin/sgledtest
[root@localhost cgi-bin]#
```

60. /mnt/etc/rc.d/rc.sysinitを編集する。

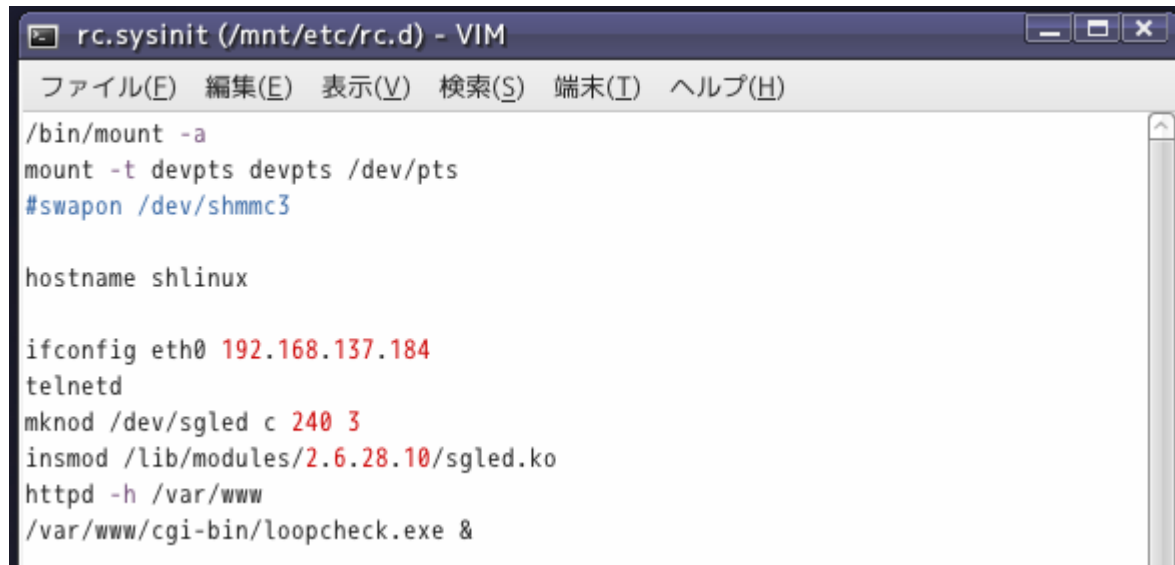


```
user01@localhost.localdomain: /mnt/etc/rc.d
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@localhost cgi-bin]# cd /mnt/etc/rc.d/
[root@localhost rc.d]# ls
rc.sysinit* rc.sysinit~* rc0*
[root@localhost rc.d]# vi rc.sysinit
```



# Busyboxコマンドによる容量圧縮(36)

61. loopcheck.exeからloopcheckに変更する。



```
rc.sysinit (/mnt/etc/rc.d) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
/bin/mount -a
mount -t devpts devpts /dev/pts
#swapon /dev/shmmc3

hostname shlinux

ifconfig eth0 192.168.137.184
telnetd
mknod /dev/sgled c 240 3
insmod /lib/modules/2.6.28.10/sgled.ko
httpd -h /var/www
/var/www/cgi-bin/loopcheck.exe &
```



```
rc.sysinit + (/mnt/etc/rc.d) - VIM
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
/bin/mount -a
mount -t devpts devpts /dev/pts
#swapon /dev/shmmc3

hostname shlinux

ifconfig eth0 192.168.137.184
telnetd
mknod /dev/sgled c 240 3
insmod /lib/modules/2.6.28.10/sgled.ko
httpd -h /var/www
/var/www/cgi-bin/loopcheck &
```

## Busyboxコマンドによる容量圧縮(37)

62. 保存後、rootfs.imgをアンマウントし、SDカードを取り出し、マイコンに差込み起動する。Linuxマシンからtelnetポート80で接続する。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184 80
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.
POST /cgi-bin/agent2.exe HTTP/1.0
```

リターンを2回にてメッセージが返り、セグメントLEDのカウントが確認できる。

```
user01@localhost.localdomain: /home/user01
ファイル(E) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[user01@localhost ~]$ telnet 192.168.137.184 80
Trying 192.168.137.184...
Connected to 192.168.137.184.
Escape character is '^]'.
POST /cgi-bin/agent2.exe HTTP/1.0

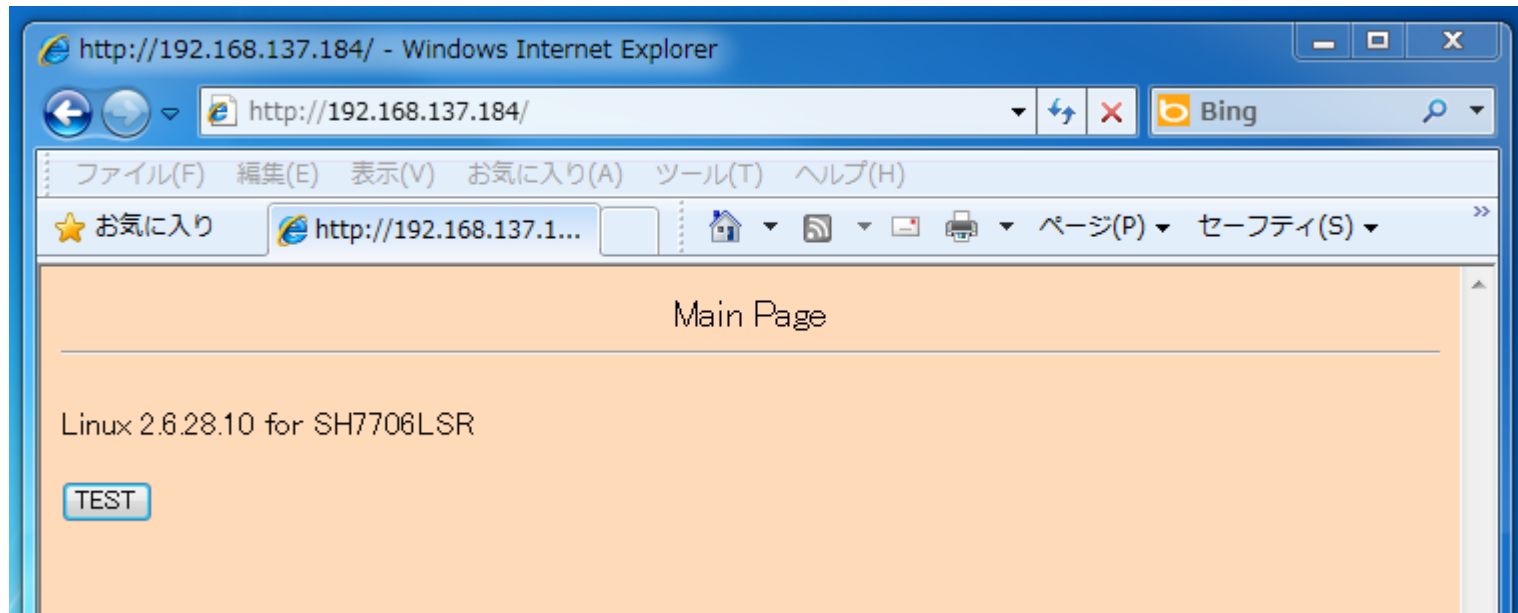
HTTP/1.0 200 OK
Content-type: text/html

<html><body bgcolor=white>
<div align=center><big>Reply Page</big></div>
<hr size=2 width=100%>
<p>Operate SgmentLED </p>
</body></html>
Connection closed by foreign host.
[user01@localhost ~]$
```



## Busyboxコマンドによる容量圧縮(38)

63. Webブラウザによる確認。192.168.137.184に入力し接続する。このときProxy設定されている場合は解除のこと。



## Busyboxコマンドによる容量圧縮(39)

64. 「TEST」をクリックすれば下記画面となりセグメントLEDのカウントが確認できる。

