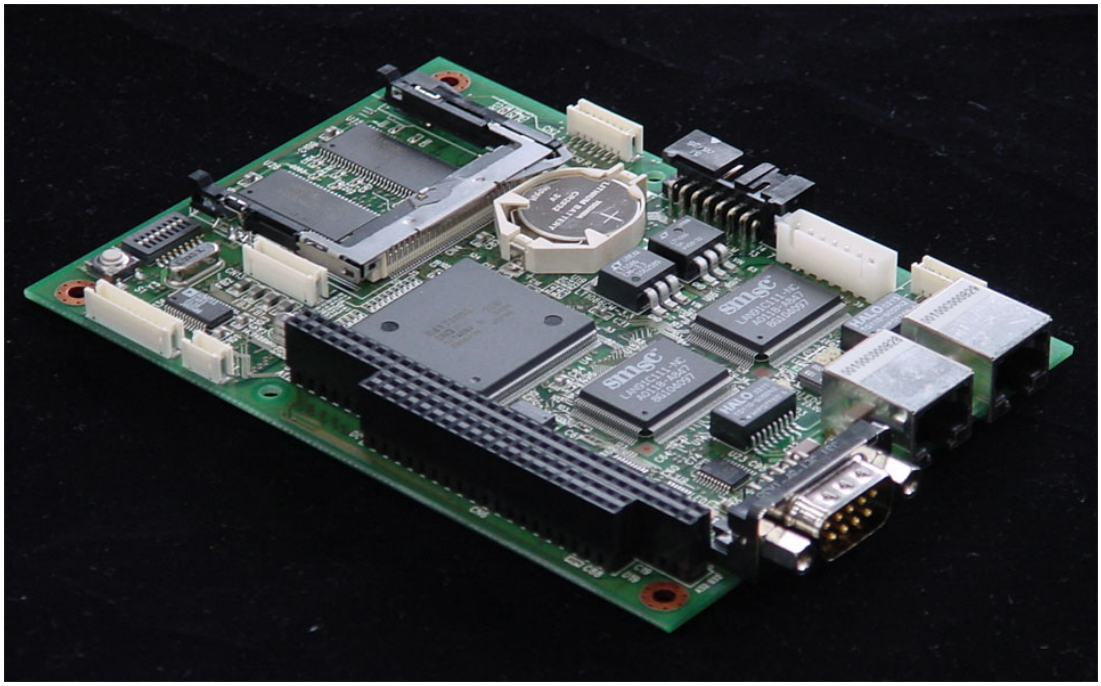
SH-Linux ユーザーズガイド

2002/9 Rev.1.0



ソフトウェアライセンスと保証

本書で使用しているソフトウェアは GNU GENERAL PUBLIC LICENSE（GPL)などに従い配布されているものです。ライセンスの条項については http://www.gnu.org/copyleft/gpl.ja.html などをご覧ください。

SH-2000,SH-2002 に搭載、付属ＣＤ－ＲＯＭに収録したソフトウェア、Ｗｅｂサイトなどで提供するアップデートされたソフトウェアなど一切のソフトウェアは完全無保証です。従いまして、これらのソフトウェアをご使用になった結果については当社および該当ソフトウェアの開発元は一切の保証を致しません。

GPLの日本語訳　http://www.key.ne.jp/Report/Counter/GPL.html

はじめに

インターネットが地球人に広く受け入れられ、先進国と呼ばれる国々に住む人々にとっては無くてはならない存在になりましたし、発展途上国に住む人々にとっては旧式の交換機を経る電話網などの無駄な投資を省いて最初から IP パケットによる通信網を確立できる技術が整いつつあります。

コンピュータやそれに関連する企業のみならず大概の企業では社内に Ethernet による LAN が張り巡らされインターネットに常時接続されているのは常識ですし、一般家庭でも光ファイバや ADSL モデムによる常時接続が普及し、Ethernet や無線 LAN による家庭内ネットワーク化も結構広まってきています。

コンピュータ同士を接続する機関技術として TCP/IP が開発されてから随分の年月を経てようやく誰もが TCP/IP の恩恵を受ける時代になりました。昔は UNIX を搭載したワークステーションと呼ばれるコンピュータと、いわゆるパソコンの間には大きな開きがありパソコンで TCP/IP プロトコルを扱うのは大変でしたが、現在ではハードウェア性能も OS 性能も全く遜色がありません。その中で Windows や MAC は受け入れやすいユーザーズインターフェースを持つためディスクトップ OS として広く普及しました。しかしインターネットを始め、ネットワーク上でサービスを提供するコンピュータとしては UNIX 系 OS を搭載したワークステーションやパソコンに強い支持があります。Windows2000/XP/NT などのファミリでサーバとよばれるエディッションを搭載したコンピュータもネットワーク上でサービスを提供するマシンとして動作できますが、サーバとしてそれ程必要としない GUI を持つため（GUI 部が切り離せないため）オーバヘッドが大きくなったり、ハードウェア性能の高いコンピュータを準備する必要があったりします。

UNIX 系の OS の中でもライセンス料が無料で全てのソースが公開されている Linux やフリーな xxxBSD などの支持が高くなってきています。基幹業務の OS に Linux が採用されていると言う話も今や珍しいことではありません。

一方、各種装置に組み込まれる CPU では最近まで OS を搭載せず、マルチタスクカーネルを採用するケースや、全くそういったものを含まず、スクラッチに全てのコードを開発することが普通に行われていました。でも状況は急速に変化してきました。ちょっとした装置でも LAN に接続したくなったり、接続しないと機能しなっかたり性能が低下する装置が増えてきたのです。

例えば時計を内蔵し、日時や時刻を使う各種装置ではシステム時計に精度を必要としますが、年差 1 秒とかの時計は実現不可能（困難）です。しかしこの装置がインターネットに接続されており、NTP（ネットワークタイムプロトコル）サーバに繋がっているなら他力本願（インターネット上に存在する NTP サーバの精度）ですが年差 1 秒の精度も十分可能です。また、装置が異常を検出したときに電子メールで知らせるなども簡単に出来てしまいます。インターネットと言う、望めば誰でもがリーズナブルに利用できる地球規模のネットワークが PC やワークステーションのみならず、CPU を内臓したインテリジェント

な各種装置の仕様や性能まで左右するのが今の時代だと言えると思います。
ソースファイルは Linux の kernel だけでもざっと 150 万行ほどありますし、SH-Linux で使用できる各種アプリケーションに限定しても数百万行のソースがあります。本書で、これらが無料で使用できる GNU GPL を始めとするフリーなソフトウェアの世界へシングルボードコンピュータのユーザ・プログラマの皆様をご招待できたら筆者としても幸いです。

目次

第１章　SH-Linux 開発環境構築

本章の役割

日立の Super-H を搭載し画面表示とキーボードなどユーザーインターフェースを持つコンピュータが無い訳ではありませんが、x86（Pentium3,4 や Athlon などを総称して）ベースのコンピュータと比べるとハードウェア性能が劣りますし、開発したソフトウェアを搭載するターゲットマシンとしては、高度なユーザーインターフェースを持たないボードコンピュータの場合が多いので、別の CPU を搭載したコンピュータ上でソフトウェアを開発することが一般的です。このような開発環境を「クロス開発環境」などと呼びます。

Linux マシン上でも Windows マシン上でもそのコンピュータ上で動作するソフトウェアをコンパイルすることは一般的ですし、このようなことを説明する文献も豊富ですが、全くアキテクチャーの違う CPU のための開発環境を準備するためには多少余分な知識が必要です。また、ホストとなるコンピュータの Linux も多数のディストリビューションが存在しますので、それら全てについて具体的なインストール方法を説明するのは難しいので、ここでは RedHat Linux7.x を搭載した x86 ベースのコンピュータに SH-Linux 用開発環境をインストールする方法を説明します。この開発環境をインストールしたコンピュータを本書では「開発マシン　や　ホストマシン」と呼ぶことにします。

RedHat は rpm（RedHat Package Manager）をベースにするディストリビューションです。この RedHat と同じく rpm をベースにしたパッケージには VineLinux や Laser5Linux などがあります。これらのディストリビューションでも概ね同じ方法で SH-Linux 開発環境がインストール可能です。

現在広く使用されている rpm にはバージョン 3 系とバージョン 4 系があり、上位互換性があります。rpm バージョン 4 では rpm バージョン 3 で作成されたパッケージも使用できますが、rpm Ver4 で作成した rpm パッケージは rpm Ver3 では使用できません。

１－０　最初にお断り

本書を執筆するにあたり、可能な限りの注意を払い、誤記の無いように勤めていますが、残念ながら筆者はそれほどハッカーではありませんので、詳しい説明を省いてお茶を濁したり、堂々と間違った説明をする可能性がありますので、前もってご了承の上本書をご利用ください。また本書および配布されているソフトウェアのご利用の結果については、いかなる動作も保証しない完全無保証ですので、すべて自己責任でお願いします。

「すべて自己責任で使え」と言われては躊躇したくなりますが、ちょっと待ってください。GNU GPL に代表されるフリーなソフトウェアはソースが完全に公開され、ほとんどのライセンス下ではソースを自由に改竄して使用する権利を保障するものです。ソフトウェアとして究極の透明性を持っているのがフリーソフトウェアだと言えると思います。またフリーソフトウェアだから信頼性が低いことはありません。１企業がソースを抱え込んで限られた人々が開発・デバッグに携わっている市販ソフトウェアよりも、全世界の雑多な環境下で使用・デバッグ・カスタマイズされているフリーソフトウェアの方がヘビーな

テストがなされている可能性が大ですし、問題が発生した場合にいち早く修正されるのがフリーソフトウェアの特徴とも言えます。もし、これらソフトウェアを販売するとしたら相当高価になるのは間違いないですし、当然ソースファイルは隠蔽したくなるはずです。フリーなソフトウェアの開発に携わっている多くの人々は「ソフトウェアが持つべき本来の姿」のため日夜努力しているのです。勿論生きていくための糧を得るため企業に属していたり公的な研究機関に属している人が多数でしょうが、大きく言えば人類全体の幸福に貢献するためソフトウェア開発に従事しているのだと思います。

フリー(自由)と表裏一体をなす自己責任の意味を理解したときに今までとは違う世界が開けるのだと思います。

１－１　ホストマシンに必用な条件

SH-Linux クロス開発環境をインストールするマシン上に一般ユーザ権限およびルート権限を持っている必要があります。Linux の経験が豊富な方はご自由にディストリビューションをお選びになり、何か問題が発生したら適切な対処を行ってインストールされれば良いと思いますが、もしこの本の読者が Linux 初心者ならコラム１のマシンを準備されると開発環境インストール時の問題が少ないと思います。

―――コラム１―――

ディストリビューションは RedHatLinux7.2FTP 版をフルインストールします。/usr を置くパーティションは 5GB 程度確保しておきます。/home パーティションは大きいに越したことありませんが 10GB 程度あれば当面問題ないと思います。rpm パッケージを作成する場合に/var/tmp が使用されます。/var 用のパーティションを別に取らなかった場合はインストール完了後にシンボリックリンクを張るなどして概ね 2GB 以上空きがある場所に移動してください。

―――コラム２―――

ホストマシンは x86 ベースの Linux マシンですし、SH-Linux は CPU が日立の SH3/4 用であることを除いて x86-Linux と何ら違いがありませんので、Linux になじみの無い方、Linux ビキナーな方は x86 ベースの Linux に関する書籍などをお手元に準備し作業を進める方が良いかも知れません。

本書を記述する際に使用したマシンはディストリビューションが RedHat7.2 で次のようにパーティションを分割しています。$ cat /etc/redhat-release

Red Hat Linux release 7.2 (Enigma)

$ df

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda2	497861	75917	396240	17%	/
/dev/hda1	69973	6162	60198	10%	/boot
/dev/hda8	9598348	1517360	7593420	17%	/home
none	252180	0	252180	0%	/dev/shm
/dev/hda7	2016016	33056	1880548	2%	/tmp
/dev/hda5	5044156	3044972	1742952	64%	/usr
/dev/hda6	2016016	115940	1797664	7%	/var
/dev/cdrom	626176	626176	0	100%	/mnt/cdrom

ホストマシンの準備ができたらいよいよ SH-Linux 用クロス開発環境をインストールします。

１－２　開発環境のインストール

これからインストールする binutils や gcc には SH3 リトルエンディアン用、SH3 ビッグエンディアン用、SH4 リトルエンディアン用、SH4 ビッグエンディアン用の 4 種類が含まれています。SH-Linux を使用する限りリトルエンディアン用だけで良く、SH-2002 や SH-2000 を使う場合 SH4 用も不要なのですが、ここでは将来使うかも知れませんのでそれら全てをインストールします。

本書執筆時点での最新バージョンは

binutils-2.11.2

gcc-3.0.4

glibc-2.2.4

ですが、SH 用開発環境は発展途上なので、より新しいものがある場合はそちらをインストールしても OK です。

ここでは SH-2002/SH-2000 に付属の CD-ROM を/mnt/cdrom にマウントした場合で path を表現します。別の場所から rpm をインストールする場合は適時読み替えてください。

インストールには root 権限が必要ですから、コマンド su または sudo で root になるか、root 権限で login してから行ってください。

コマンド sudo がインストールされている場合は/etc/sudoers に次のように記述すると“sodo　実行したいコマンド”とすると root 権限で実行できます。 ユーザ名　ALL = NOPASSWD:ALL 例　sh2002 ALL = NOPASSWD:ALL

SH-Linux 開発環境 CD-ROM をドライブに入れます。

automount デーモンが動作している場合は自動的に mount されます。

自動マウントにしていない場合は次のようにして mount します。

mount デバイスファイル　マウントポイント

/etc/fstab に次のような記述があれば　mount /mnt/cdrom でマウントできます。

/dev/cdrom　/mnt/cdrom　iso9660 noauto,owner,kudzu,ro 0 0

開発マシンコンソールのプロンプト

>$　一般ユーザ権限で行う作業の場合。

>#　root 権限が必要な作業の場合。

次のようにしてアセンブラ、リンカなど SH 用の binutils やコンパイラをインストールします。

>\# rpm -ivh /mnt/cdrom/RPMS/i386/RedHat7.1/binutils-sh-linux-2.11.2-6.i386.rpm
>\# rpm -ivh /mnt/cdrom/RPMS/i386/RedHat7.1/gcc-sh-linux-3.0.4-1.i386.rpm
>\# rpm -ivh /mnt/cdrom/RPMS/i386/RedHat7.1/gdb-sh-linux-20001217-4.i386.rpm
（使用する予定が無ければインストール不要です）
>\# rpm -ivh /mnt/cdrom/RPMS/i386/RedHat7.1/libgcj-sh-linux-3.0.4-1.i386.rpm
（使用する予定が無ければインストール不要です）

インストールが成功すると次のようなコマンドが使えるようになります。
binutils(アセンブラ、リンカなど)

```
sh-linux-as,       sh-linux-ld, etc       ... executable binaries
sh3-linux-as,      sh3-linux-ld, etc      ... for sh3-linux
sh3eb-linux-as, sh3eb-linux-ld, etc ... for sh3eb-linux
sh4-linux-as,      sh4-linux-ld, etc      ... for sh4-linux
sh4eb-linux-as, sh4eb-linux-ld, etc ... for sh4eb-linux
```

gcc(c,c++コンパイラ)

```
sh-linux-gcc          --- multilibed cross compiler for sh-linux
sh3-linux-gcc         --- non-multilib cross compiler for sh3-linux
sh3eb-linux-gcc       --- non-multilib cross compiler for sh3eb-linux
sh4-linux-gcc         --- non-multilib cross compiler for sh4-linux
sh4eb-linux-gcc       --- non-multilib cross compiler for sh4eb-linux
```

開発マシンで必要な SH 用のファイルをインストールします。
>\# rpm -ivh /mnt/cdrom/RPMS/noarch/host/*.rpm

クロス開発に必要な SH-Linux カーネルヘッダファイルは rpm パッケージに含まれていませんので、CD-ROM の中の tarball を展開して所定の場所へインストールします。
>\# cd /usr/src
>\# tar jxf /mnt/cdrom/linux-sh-2.4-020802.tar.bz2
（020802 は作成日です。より新しいものが CD-ROM に入っているかも知れません）

依存関係を解決しないと作成されないファイルがありますから、次のように config と dep を実行します。
>\# cd /usr/src/linux-sh-2.4
>\# cp config.sh2002 .config
>\# make menuconfig
>\# make dep

menuconfig は xconfig などお好きなもので OK です。予め作成されている config.sh2002 を.config にコピーしていますからコンフィグレーションは不要です。起動したらそのままファイルをセーブして終了してください。
次に dep を実行して依存関係を解決してください。
これで必用なヘッダファイルが全て作成されます。
カーネルコンフィグレーションの詳しいことは x86 ベース Linux の解説書などを参照してください。

これで開発マシンに SH 用クロス開発環境がインストールされました。/usr に次のようなディレクトリ構造で各種ファイルが追加されています。

```
usr --+-- bin                                          binaries, shell scripts
      +-- lib ---- gcc-lib --+--- sh-linux        cross compiler
      |                      +--- sh3-linux
      |                      +--- sh3eb-linux
      |                      +--- sh4-linux
      |                      +--- sh4eb-linux
      |
      +--  sh-linux    --+-- bin                  binary executables (binutils)
      |                    +-- include              include files
      |                    +-- lib                  libraries for sh3-linux (symlinks)
      |                        +-- m4                 libraries for sh4-linux (symlinks)
      |                        +-- mb                 libraries for sh3eb-linux (symlinks)
      |                            +-- m4             libraries for sh4eb-linux (symlinks)
      |
      +-- sh3-linux    --+-- bin                  symlink, shell script
      |                    +-- include              symlink to /usr/sh-linux/include
      |                    +-- lib                  libraries for sh3-linux
      |
      +-- sh3eb-linux   ...
      +-- sh4-linux     ...
      +-- sh4eb-linux   ...
      |
      +-- src --- linux-sh-2.4 --- include        kernel include files
```

ここまでは、一般的なアプリケーションをインストールする手順とほとんど同じでした。次にターゲット（SH-2002/SH-2000）用のパッケージをインストールします。ここでインストールする各種ファイルは SH-2002 などの上で実行されるものですから、開発マシン上にターゲット用 root ファイルシステムを構築することになります。

１－3　ターゲット用パッケージ＆カーネルのインストール

SH-2002 や SH-2000 にはフロッピーディスクや CD-ROM を接続することができませんので、別の場所からソフトウェアをインストールする方法はコンパクトフラッシュ（CF）カードを経由するかイーサネットを経由する必要があります。（SH-2002 には USB インターフェースが装備されているので将来 USB 対応 CD-ROM などからインスト

ールできる可能性もありますが、それには USB に対応した SH-Linux カーネルに加えて、BIOS に相当する sh-ipl の大幅な強化が必要です）

開発マシンに CF カードリーダー／ライタを接続して SH-200x 起動 CF の作成も勿論可能ですが、Linux で動作する CF リーダー／ライタが少ない、使用するリーダー／ライタによって必用なドライバが異なるなどがあります。

そのような訳でここでは、nfs サーバと dhcp サーバによる BOOTP&NFSroot で SH-200 x を起動させます。

少なくても RedHat7.x（もう少し範囲を広げても）を使っているなら、ＮＦＳサーバやＤＨＣＰサーバの設定は結構簡単ですから心配は要らないと思います。NFSroot で動作するようになれば、SH-200x 自身で起動用 CF が作成できますので CF リーダー／ライタを購入する必要もありません。

それでは、ターゲット用パッケージをインストールしましょう。インストールは機械的な作業ですが、インストール後にターゲットファイルシステムの/etc 以下に各種ファイルを作成したり、編集したりする必要があります。これが面倒な方は CD-ROM に入っているターゲット用 root ファイルシステムの tarball を展開するのも一つの手段です。

ここでは、ホームディレクトリ（/home/foo）以下に次のようなディレクトリ構造を作成します。（/home/foo は適時読み替えてください）

CDROM のトップディレクトリに入っているターゲット用 rpm インストールスクリプト setup_target.sh とカタログファイル target.file を自分のホムデイレクトリにコピーします。

cp /mnt/cdrom/setup_target.sh .
cp /mnt/cdrom/target.file .

setu_target.sh をオプション無しで起動するとファイルを/mnt/cdrom 以下から読み /home/foo/SH-2002 以下にインストールします。
インストーラの起動
./setup_target.sh
これで SH-2002 用 root ファイルシステムと SH-Linux カーネルソースがインストールさらます。なお、インストールには数分かかります。

/home/foo/SH-2002/export　　root ファイルシステムです。
/linux-sh-2.4　カーネルソースです。

/mnt/cdrom 以外からインストールする場合は./setup_target.sh に次のようなオプションを付けます。

./setup_target.sh　ソースパス（例　/home/foo/SH-2002-CD-ROM）
インストール先を/home/foo/SH-2002 以外にする場合は、ディストネーションパスを指定します。
例　./setup_target.sh　/mnt/cdrom sh-linux/cf_img

もし、インストール先を SH-2002/export 以外に変更した場合は、以下の説明のパスを適時読み替えてください。また、kernel ツリーのファイル（トップディレクトリの Makefile）を少し編集する必要があります。

１－４　インストール後の処理

前項でインストールした rpm には含まれないファイルが幾つかあります、また内容を修正する必要があるファイルもあるので、それらを新たに作成・編集します。

デバイスファイル

/dev 以下のファイルが必要ですから、CD-ROM の/Various/dev 以下のファイルをコピーしてください。

># cp –a /mnt/cdrom/Various/dev/* /home/foo/SH-2002/export/dev/

メモ：デバイスファイルのコピーの場合、ファイルが 1 つであってもスイッチ'–a'が必要です。

（システムの/dev/以下をコピーしても OK ですが余分なデバイスファイルがコピーされます。また一部 SH-200x 専用のデバイスファイルがありません）

システムの dev 以下をコピーした場合は追加で次の作業を行ってください。

console を ttySC0 にシンボリックリンクする

># cd /home/foo/SH-2002/export/dev

># ln –sf ttySC0 console

(SH-2002 のコンソールはシリアルポートに接続された端末ですから)

/etc/fstab 作成

/mnt/cdrom/Various からファイルをコピーするか新規に作成します。

/home/foo/SH-2002/export/etc/fstab.nfs

/dev/nfs	/	nfs	defaults	1 1
proc	/proc	proc	defaults	0 0
devpts	/dev/pts	devpts	gid=5,mode=620	0 0

/home/foo/SH-2002/export /etc/fstab.hda1

/dev/hda1	/	ext3	defaults,ro,noatime	1 1

```
proc            /proc     proc     defaults          0 0
devpts          /dev/pts  devpts   gid=5,mode=620    0 0
```

swap を追加する場合は次の行も作成しておいてください

```
/dev/hda?       swap      swap     defaults          0 0
```

?はタイプ 0x82 のパーティション

シンボリックリンクを張る

fstab は nfs ルート用と CF 用の２つを作成しましたので使用する方へのシンボリックリンクを作成しておきます。

```
cd /home/foo/SH-2002/export/etc
ln -s fstab.nfs fstab
```

inittab の編集

次の行の先頭に＃を付けてコメントアウトします。

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

SH-2002 のシリアルインターフェース（CN11=ttySC0,CN10=ttySC1）に接続した端末から login できるようにするため、次の行を追加します。

sh-ipl では ttySC0 がコンソールになっていますので特に支障がなければ SH-Linux のコンソールも ttySC0 を使用するのが良いでしょう。もし ttySC1 に端末以外の機器を接続するなら ttySC1 の行はコメントアウトしておいてください。

```
1:2345:respawn:/sbin/agetty 115200 /dev/ttySC0 -L
2:2345:respawn:/sbin/agetty 115200 /dev/ttySC1 -L
```

hosts 作成

/home/foo/SH-2002/export/etc/hosts を作成します。

```
127.0.0.1       localhost.localdomain   localhost
```

sysconfig/network 作成

/home/foo/SH-2002/export/etc/sysconfig/network を作成します。

NETWORKING=yes

resolv.conf 作成

環境に合わせて/home/foo/SH-2002/export/etc/resolv.conf を作成します。

search にはドメイン名を記載します。nameserver には dns サーバの IP アドレスを記載します。

search itonet.co.jp

nameserver 192.168.0.5

/etc/localtime をコピー

時刻にJSTを指定するため開発マシンの/etc/localtimeを　/home/foo/SH-2002/export/etc/にコピーします。

># cp /etc/localtime /home/foo/SH-2002/export/etc/

/etc/mime.types もコピー

http サーバのために mime.types も開発マシンからコピーしておきます。

># cp /etc/mime.types /home/foo/SH-2002/export/etc/

/sbin/update を作成

/sbin/update が起動スクリプトから呼ばれますが、存在しないのでエラーになります。それを避けるため形だけ作成しておきます。エディタで /home/foo/SH-2002/export/sbin/update を作成します。内容は、

#!/bin/sh

だけで OK です。作成したら chmod で実行権を付けておきます。

># chmod 755 /home/foo/SH-2002/export/sbin/update

各種シンボリックリンクの作成

必用なシンボリックリンクを幾つか作成します。

># cd /home/foo/SH-2002/export/usr/lib

># ln -s libdb2.so.3 libdb.so.3

># ln -s libcrack.so.2.7 libcrack.so.2

httpd の設定

SH-2002 起動後のお楽しみ？のため、apache のコンフィグレーションファイルを少し編集しておきます。

/home/foo/SH-2002/export/etc/httpd/conf/httpd.conf の 364 行目あたりにある

コメントアウトされた ServerName　localhost を有効（先頭の#を取る）にします。
ついでに CD-ROM のディレクトリ Various にある index.html を SH-2002/export/home/httpd/html/にコピーしておくと SH-2002 起動後に自我自賛？できます。

kernel を make してインストール

開発マシンの/usr/src/linux-sh-2.4 とは別に/home/foo/SH-2002/linux-sh-2.4 以下に SH-Linux のカーネルが入っていますので、これをコンパイルして実行ファイルを作成します。
メモ：もし/usr 以下のディスク容量が少ない場合は/usr/src/linux-sh-2.4 はこのカーネルツリーへのシンボリックリンクにしても大丈夫です。

カーネルのコンパイルは次の手順で行います。

>$ cd /home/foo/SH-2002/linux-sh-.24

>$ cp config.sh2002 .config

カーネルツリーのトップディレクトリに SH-2002 用コンフィグレーションファイルがファイル名 config.sh2002 で入っていますのでこれを.config にコピーします。

>$ make menuconfig

コンフィグレーションを変更する必要はありませんが、必ずセーブして終了してください。X をお使いなら menuconfig は xconfig でも OK です。

>$ make dep

>$ make clean

念のためです。（初めてのコンパイルなら不要です）

>$ make zImage

圧縮されたカーネルを構築します。

>$ make modules

モジュールをコンパイルします。

>$ sudo make modules_install

SH-2002 用 root ファイルシステムを/home/foo/SH-2002/export 以外にした場合はカーネルツリーのトップディレクトリの Makefile の 8 行目あたりを修正してください。

（INSTALL_MOD_PATH=$(HOME)/SH-2002/export
を環境に合わせて修正します。）

sudoers を設定していない場合は su で root になって実行してください。

>$ sudo cp arch/sh/boot/zImage ../export/vmlinuz

sudoers を設定していない場合は su で root になってコピーしてください。SH-2002 用 root ファイルシステムの位置が違う場合は適時読み替えてください。

ここまでの作業で容量約 80MB の SH-2002 の nfs ルートファイルシステムが一応出来上がりました。後は開発マシンの dhcp サーバ（sh-ipl が BOOTP でカーネルをロードして起動します）と nfs サーバ（開発マシンのディスクの一部を NFS で SH-2002 が root ファイルシステムとして使用します）を動かせば待望の SH-2002 初起動が待っています。

１－５　開発マシンの nfs サーバ,dhcp サーバを設定してターゲットの起動

nfs サーバと dhcp サーバが開発マシンにインストールされている必要がありますので、先ず確かめて見ましょう。

>$ rpm -qa | grep dhcp としたら

dhcp-2.0pl5-8 のような dhcp サーバパッケージが見つかれば OK です。

>$ rpm -qa | grep nfs

nfs-utils-0.3.1-13 ような nfs ユーティリティパッケージが見つかれば概ね OK です。

開発マシンの/etc に dhcpd.conf と exports を emacs などエディタを使用して作成します。

>\# emacs /etc/dhcpd.conf

内容の例

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  default-lease-time 600; max-lease-time 7200;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.0.255;
  option routers 192.168.0.254;
  option domain-name-servers 192.168.0.5;
  option domain-name "itonet.co.jp";
  deny unknown-clients;

  host sh2002 {
        hardware ethernet 00:10:0c:00:08:32;
        filename "/home/sh2002/SH-2002/export/vmlinuz";
        fixed-address 192.168.0.71;    ←この IP アドレスが SH-2002 に設定されます。
        option host-name "sh2002";
        option root-path "/home/sh2002/SH-2002/export";
        option option-128 "DODES";
        option      option-129      "console=ttySC0,115200      ether=32,0x300,0,0,eth0
```

ether=33,0x320,0,0,eth1 ide0=0x1f0,0x3f6,34";　←実際には 1 行で書きます。
　}
}
各 IP アドレスやドメイン名などは適時お使いの環境に合わせてください。
また、hardware ethernet に記載されているのは MAC アドレスです。お手元の SH-2002 の eth0 コネクタ上部に書いてあるアドレスを見て正しく設定してください。

ご注意：Ethernet からのブートは eth0 から行います。eth1 側からはブートできません。

```
>#  emacs /etc/exports
```
内容の例
/home/sh2002/SH-2002　　192.168.0.0/255.255.255.0(rw,no_root_squash)

ご注意：空白はエクスポートするディレクトリと提供先 IP アドレスの間だけに入れます。
他の部分に空白を入れると nfs サーバが期待に反した動きをします。
また、オプション(rw,no_root_squash)は必須です。

># /etc/rc.d/init.d/dhcpd start　として DHCP サーバを起動します。
># /etc/rc.d/init.d/nfs start
># /etc/rc.d/init.d/nfslock start　として NFS サーバを起動します。

何か異常がでたら設定ファイルなどを見直してください。

電源 ON で自動起動に設定するなら/sbin/chkconfig で行えます。
例　　/sbin/chkconfig –level 2345 dhcpd on
　　/sbin/chkconfig –level 2345 nfs on
　　/sbin/chkconfig –level 2345 nfslock on

無事開発マシンの DHCP サーバ、NFS サーバが起動したら SH-2002 を起動しましょう。
SH-2002 に所定の電源を接続し、eth0 に LAN ケーブル、起動コンソールに端末を接続します。
SH-2002 のディップ SW を全て OFF にし、電源を ON にすると sh-ipl が起動し、コマンド待ちになりますので、端末からイーサブートコマンド'e'を入力します。

ブートメッシージ
SH IPL+g version 1.02, Copyright (C) 2001 Free Software Foundation, Inc.

```
This software comes with ABSOLUTELY NO WARRANTY; for details type `w'.
This is free software, and you are welcome to redistribute it under
certain conditions; type `l' for details.

128Mbyte memory
> e
Booting from network!
SMSC LAN91C111 Driver (v2.0), (Linux Kernel 2.4 + Support for Odd Byte) 09/24/01
 -          by Pramod Bhardwaj (pramod.bhardwaj@smsc.com)

LAN91C111: SMC91C11xFD(rev:0) at 0x300 MEMSIZE:8192b ADDR:
00:10:0c:00:08:32
PHY=LAN83C183 (LAN91C111 Internal)Searching for server (BOOTP/DHCP)...
<sleep>
IP Address: 192.168.0.71
Server: 192.168.0.225, Gateway 192.168.0.254
Kernel to load: "/home/sh2002/SH-2002/export/vmlinuz"
HOSTNAME: sh2002
ROOT PATH: /home/sh2002/SH-2002/export
COMMANDLINE: console=ttySC0,115200 ether=32,0x300,0,0,eth0
ether=33,0x320,0,0,et
h1 ide0=0x1f0,0x3f6,34
Loading Kernel: /home/sh2002/SH-2002/export/vmlinuz .....
.....SUM: 594c3a7
done
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.4.18 (sh2002@h225.itonet.co.jp) (gcc version 3.0.4) #1 2002 年  7
月  31 日  水曜日  16:05:39 JST
＜途中省略＞
Starting kernel logger: [   OK   ]
Starting cron daemon: [   OK   ]
Starting sshd [   OK   ]
Starting httpd: [   OK   ]

sh2002 login:
```

login が表示されたら root でログインします。（root のパスワードは未設定です）

端末には次のようなものが使えます。 Linux マシンで cu を動かす。 Windows マシンで teraterm を動かす。など

１－６　SH-2002 起動後の設定

無事 SH-2002 が nfs root で起動できましたら、残りの設定を SH-2002 の端末から行います。

●パスワードファイルを shadow に変更

このシステムでは起動直後は shadow パスワードになっていませんのでコマンド pwconv を使って shadow パスワードに変更しましょう。

SH-2002 を起動して、SH-2002 のコンソールから

/usr/sbin/pwconv

で shadow パスワードに変更されます。

●ユーザの追加とパスワードの設定

/usr/sbin/useradd コマンドでユーザの追加ができます。

bash-2.05# /usr/sbin/useradd 追加するユーザ名

これでユーザが１つ追加されました。

確認して見ます。

```
bash-2.05# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
```

```
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
ユーザ名:x:500:500::/home/nakamura:/bin/bash
```

最後の行に新しいユーザが追加されています。
このユーザには未だパスワードが設定されていませんのでパスワードを設定しましょう。

```
bash-2.05# passwd　ユーザ名
Changing password for user ユーザ名
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
bash-2.05# passwd nakamura
```

安全のために root のパスワードも設定しておきましょう。

昔ネットワークが安全だった頃は telnet や ftp など平文のパスワードをネットワーク上に流しても平気でしたが、最近はクラッカー達が元気なのでできるだけ安全な環境で使用しましょう。ボードコンピュータ SH-2002 と言えど立派なＵＮＩＸマシンなのでクラックの踏み台にされる可能性もあります。そのような訳で inittab から起動される telnet（in.telnetd）などは動作させないが、良いと思います。ここでは LAN(WAN)経由の login に ssh（セキュア shell）を使います。

sshd の起動

```
bash-2.05# /etc/rc.d/init.d/sshd start
Generating SSH RSA host key: [  OK  ]
Generating SSH DSA host key: [  OK  ]
Starting sshd: [  OK  ]
```

最初の起動の場合 hostkey の作成に少し時間が必用です（概ね１分以内）

sshd を自動起動に設定するなら chkconfig コマンドで行います。

```
bash-2.05# /sbin/chkconfig --list | grep ssh
sshd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
bash-2.05# /sbin/chkconfig --level 2345 sshd on
bash-2.05# /sbin/chkconfig --list | grep ssh
```

sshd　　0:off　1:off　2:on　3:on　4:on　5:on　6:off

Windows 環境で ssh 対応のターミナルなら
寺西氏が作成された teraterm　http://www.vector.co.jp/authors/VA002416/
に teraterm 用セキュア shell　TTSSH（Eric A Young 氏作）
　http://www.zip.com.au/~roca/ttssh.html
を入れるのが良さそうです。
Linux からは　コマンド ssh で接続できます。

先程作成したユーザ名で SH-2002 にセキュア接続して見ましょう。
ここでは Linux マシンから ssh で接続します。

[sh2002@h225 SH-2002]$ ssh 192.168.0.71　←正しい IP アドレスを指定
The authenticity of host '192.168.0.71 (192.168.0.71)' can't be established.
DSA key fingerprint is 0c:5f:91:2d:97:bd:5e:b6:b1:9a:ba:64:87:51:18:61.
Are you sure you want to continue connecting (yes/no)? yes　←
Warning: Permanently added '192.168.0.71' (DSA) to the list of known hosts.
初めて接続する場合は上記のようなメッセージが出ます。問いには yes で答えます。
ユーザ名@192.168.0.71's password:
Last login: Wed Jul 31 17:00:54 2002 from naka.itonet.co.jp
bash-2.05$　←SH-2002 のプロンプトです。

コマンドｗで login ユーザを見てみる。
bash-2.05# w
　1:29pm　up 20:42,　3 users,　load average: 0.08, 0.02, 0.01
USER　　TTY　　FROM　　LOGIN@　IDLE　JCPU　PCPU　WHAT
root　　/dev/tty -　　Wed 4pm　0.00s　0.33s　0.10s　w
nakamura pts/0　　naka.itonet.co.j Wed 5pm 18:03m　0.18s　0.18s　-bash
bash-2.05#

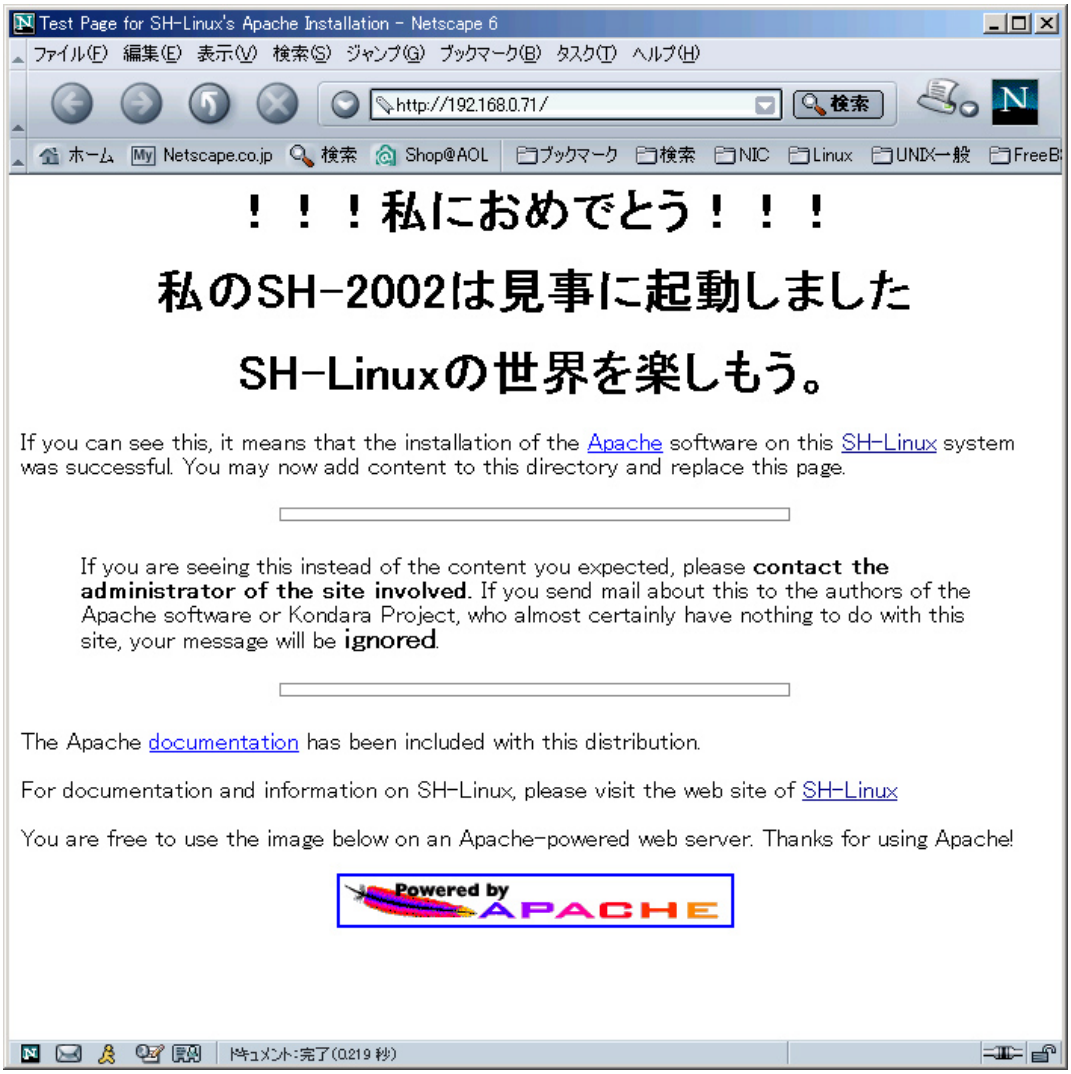
HTTP サーバのチェック
プロセスリストを見て httpd が幾つかあれば http サーバが動作しています。

```
bash-2.05# ps ax
（以下は適時省略して表示しています）
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:02 init
    2 ?        SW     0:00 [keventd]
  295 ttySC0   S      0:00 -bash
  296 ttySC1   S      0:00 /sbin/agetty 115200 /dev/ttySC1 -L
  297 ?        S      0:00 httpd
  298 ?        S      0:00 httpd
bash-2.05#
```

SH-2002 と同じネットワークに接続されたマシン（勿論開発マシンでも OK です）からブラウザで SH-2002 に接続して見ましょう。
（IP アドレスは開発マシンの dhcpd.conf で指定したものですが SH-2002 側からも確認できます）
例

```
bash-2.05# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:10:0C:00:08:32
          inet addr:192.168.0.71  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:106723 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58012 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:32 Base address:0x300 DMA chan:1
＜途中省略＞
bash-2.05#
```

ブラウザからアクセス（自我自賛版）



この辺りまでインストールが進むとほっと一息ブレークタイムでしょうか？。休憩したら折角起動した SH-2002root ファイルシステムのバックアップを取り、SH-2002 自身で起動用 CF の作成に入ります。

```
root ファイルシステムのバックアップ
># cd /home/foo/SH-2002
># tar cfz sh-2002-root-fs.tar.gz export
```

１－７　起動用 CF の作成

一旦 SH-2002 の電源を OFF にします。

メモ：特に何も起動していなければ nfs ルートの場合、shutdown 手続きを飛ばし

て直接電源を OFF にしても大丈夫です。

128MB 程度の容量の CF カードを SH-2002 に装着します。

メモ：CF カードは SanDisk 社製だけ動作確認済みです。IBM 社製マイクロドライブも動作確認されています。また、「SH-200x で動作しない CF がある」ことが報告されています。

開発マシンの SH-2002root ファイルシステムの適当な場所に CF 書き込み用ファイルシステム作成の準備を行います。

># cd /home/foo/SH-2002/export/home/ユーザ名

># tar zxf ../../../ sh-2002-root-fs.tar.gz export

バックアップを展開します。

># mv export cf_img （別にリネームしなくてもＯＫですが)

次の各ファイルを作成・編集します。

カレントディレクトリが「/home/foo/SH-2002/export/home/ユーザ名/cf_img」として記載

以下の作業は SH-2002 のコンソールからも行えますが、SH-2002 に入っているエディタは vi だけです。

etc/fstab（シンボリックリンクの張替え）

etc/sysconfig/network

etc/sysconfig/network-scripts/ifcfg-eth0

etc/sysconfig/network-scripts/ifcfg-eth1（必用に応じて）

etc/lilo.conf

boot/vmlinuz（/vmlinuz へのシンボリックリンクでも OK）

作業手順

># mv cf_img/vmlinuz cf_img/boot/

># cd cf_img/etc

># ln –sf fstab.hda1 fstab

># emacs sysconfig/network

NETWORKING=yes

HOSTNAME=sh2002.itonet.co.jp ←お使いの環境に合わせてください

GATEWAY=192.168.0.1　←お使いの環境に合わせてください

># emacs sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

BOOTPROTO=static

ONBOOT=yes

BROADCAST=192.168.0.255　←お使いの環境に合わせてください

IPADDR=192.168.0.71　←お使いの環境に合わせてください

NETMASK=255.255.255.0　←お使いの環境に合わせてください
NETWORK=192.168.0.0　←お使いの環境に合わせてください

>\# emacs sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
BROADCAST=192.168.100.255　←お使いの環境に合わせてください
IPADDR=192.168.100.71　←お使いの環境に合わせてください
NETMASK=255.255.255.0　←お使いの環境に合わせてください
NETWORK=192.168.100.0　←お使いの環境に合わせてください
>\# emacs lilo.conf
linear
install=/boot/boot.b
boot = /dev/hda
disk = /dev/hda
append="console=ttySC0,115200　mem=128M　ide0=0x1f0,0x3f6,34
ether=32,0x300,0,0,eth0 ether=33,0x320,0,0,eth1"　←実際は 1 行
image = /boot/vmlinuz
label = linux
root = /dev/hda1
read-only

SH-2002　128MB 版では mem=128M です。64MB 版では mem=64M としてください。

引き続き SH-2002 のコンソールで作業します。
●ＣＦカードの初期化
bash-2.05\# /sbin/fdisk /dev/hda

Command (m for help): p

Disk /dev/hda: 8 heads, 32 sectors, 980 cylinders
Units = cylinders of 256 * 512 bytes
新しい CF の場合 fat で領域が確保されている可能性があります。その場合領域を削除してから linux 用のパーティションを作成してください。

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-980, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-980, default 980):
Using default value 980

Command (m for help): p

Disk /dev/hda: 8 heads, 32 sectors, 980 cylinders
Units = cylinders of 256 * 512 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1               1         980      125424   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

● CF カードのフォーマット

SH-2002 では ext3(ジャナルファイルシステム)を使用できますので mke2fs にはスイッチ'-j'を付けます。

```
bash-2.05# /sbin/mke2fs -j /dev/hda1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
```

Fragment size=1024 (log=0)
31360 inodes, 125424 blocks
6271 blocks (5.00%) reserved for the super user
First data block=1
16 block groups
8192 blocks per group, 8192 fragments per group
1960 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 29 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
●CF をマウントします。
マウントポイント/mnt/cf を作成しておきます。
bash-2.05# mkdir /mnt/cf
CF のマウント
bash-2.05# mount /dev/hda1 /mnt/cf
do_page_fault: task=mount pc=0x8800b678 address=0x420000
do_page_fault: task=mount pc=0x8800b678 address=0x420000
現行の kernel ではページフォルトが出てしまいますが、マウントは正しく行われます。
bash-2.05# df
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/nfs 9598348 1517336 7593444 17% /
/dev/hda1 121459 4127 111061 4% /mnt/cf
bash-2.05# ls /mnt/cf/
lost+found
●ファイルツリーをコピーします。
bash-2.05# cp -a /home/ユーザ名/cf_img/* /mnt/cf/
bash-2.05# df
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/nfs 9598348 1517336 7593444 17% /
/dev/hda1 121459 77318 37870 67% /mnt/cf

●CF に ipl を書き込みます。
bash-2.05# /sbin/lilo -r /mnt/cf
Added linux *
bash-2.05# sync
bash-2.05# umount /mnt/cf
bash-2.05#
sync でダーティバッファをフラッシュして CF をアンマウントします。

lilo のスイッチ ‘-r’は lilo を実行する root(/)を変更するスイッチです。この場合/mnt/cf が/として扱われます。lilo の default コンフィグレーションファイルは/etc/lilo.conf ですが スイッチ’-C’で変更可能です。 -r /mnt/cf と共に -C /etc/lilo.sh2002 とした場合、/mnt/cf/etc/lilo.sh2002 がコンフィグレションファイルとして使用されます。

SH-2002 の電源を OFF/ON するか、リセットボタンを押します。

SH-2002 起動メッセージ（CF からブート時）
SH IPL+g version 1.02, Copyright (C) 2001 Free Software Foundation, Inc.

This software comes with ABSOLUTELY NO WARRANTY; for details type `w'.
This is free software, and you are welcome to redistribute it under
certain conditions; type `l' for details.

128Mbyte memory
> b ←CF からブートします。
Disk drive detected: SanDisk SDCFB-128 vde 1.10 293287L1709
Set Transfer Mode result: 51
Initialize Device Parameters result: 50
IDLE result: 50
LILO boot: first-image
Loading linux.....done.
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.4.18 (sh2002@h225.itonet.co.jp) (gcc version 3.0.4) #1 2002 年 7 月 31 日 水曜日 16:05:39 JST
LLG-2000 Setup...done
On node 0 totalpages: 32768

zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: ro root=301 console=ttySC0,115200 mem=128M ide0=0x1f0,0x3f6
,34 ether=32,0x300,0,0,eth0 ether=33,0x320,0,0,eth1
ide_setup: ide0=0x1f0,0x3f6,34

Setting GDB trap vector to 0x80000100
CPU clock: 199.99MHz
Bus clock: 66.66MHz
Module clock: 33.33MHz
Interval = 83331
Calibrating delay loop... 99.73 BogoMIPS
Memory: 127704k/131072k available (1180k kernel code, 3368k reserved, 42k data, 44k init)
Dentry-cache hash table entries: 16384 (order: 5, 131072 bytes)
Inode-cache hash table entries: 8192 (order: 4, 65536 bytes)
Mount-cache hash table entries: 2048 (order: 2, 16384 bytes)
Buffer-cache hash table entries: 8192 (order: 3, 32768 bytes)
Page-cache hash table entries: 32768 (order: 5, 131072 bytes)
CPU: SH7709A/SH7729
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
VFS: Diskquotas version dquot_6.4.0 initialized
Journalled Block Device driver loaded
pty: 256 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with no serial options enabled
SuperH SCI(F) driver initialized
ttySC0 at 0xfffffe80 is a SCI
ttySC1 at 0xa4000150 is a SCIF
ttySC2 at 0xa4000140 is a SCIF
Real Time Clock Driver v1.10e

```
block: 128 slots per queue, batch=32
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
Uniform Multi-Platform E-IDE driver Revision: 6.31
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
hda: SanDisk SDCFB-128, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 34
hda: 250880 sectors (128 MB) w/1KiB Cache, CHS=980/8/32
Partition check:
 hda: hda1
SMSC LAN91C111 Driver (v2.0), (Linux Kernel 2.4 + Support for Odd Byte) 09/24/01
 -
        by Pramod Bhardwaj (pramod.bhardwaj@smsc.com)
eth0: SMC91C11xFD(rev:0) at 0x300 IRQ:32 MEMSIZE:8192b NOWAIT:1 ADDR:
00:10:0c:0
0:08:32
eth1: SMC91C11xFD(rev:0) at 0x320 IRQ:33 MEMSIZE:8192b NOWAIT:1 ADDR:
00:10:0c:0
0:08:33
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 1024 buckets, 8Kbytes
TCP: Hash tables configured (established 8192 bind 16384)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
 hda: hda1
 hda: hda1
kjournald starting.  Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 44k freed
modprobe: modprobe: Can't locate module char-major-4
INIT: version 2.78 booting
                        Welcome to GNU/Linux on SuperH
                Press 'I' to enter interactive startup.
Mounting proc filesystem [  OK  ]
Configuring kernel parameters [  OK  ]
Setting clock : Thu Aug  1 14:27:33 JST 2002 [  OK  ]
```

```
Activating swap partitions [  OK  ]
Setting hostname sh2002.itonet.co.jp [  OK  ]
Checking root filesystem
/dev/hda1: clean, 19795/31360 files, 81295/125424 blocks
[/sbin/fsck.ext3 -- /] fsck.ext3 -a /dev/hda1
[  OK  ]
Remounting root filesystem in read-write mode [  OK  ]
Finding module dependencies [  OK  ]
Checking filesystems
Checking all file systems.
[  OK  ]
Mounting local filesystems [  OK  ]
chgrp: invalid group name `utmp'
                                        Enabling swap space [  OK  ]
INIT: Entering runlevel: 3
Entering non-interactive startup
Setting network parameters [  OK  ]
Bringing up interface lo [  OK  ]
Bringing up interface eth0 [  OK  ]
Bringing up interface eth1 [  OK  ]
Starting portmapper: [  OK  ]
Initializing random number generator [  OK  ]
Mounting other filesystems [  OK  ]
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting cron daemon: [  OK  ]
Starting sshd [  OK  ]
Starting httpd: [  OK  ]

sh2002.itonet.co.jp login: root   ←login します。
―――――――――――――――――――起動メッセージここまで―――――――――――――――――――
bash-2.05# df   ←確認すると CF から正しく起動しています。
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/hda1                121459     77343     37845  67% /
bash-2.05# /sbin/ifconfig   ←ifconfig でネットワークを見てみる
eth0      Link encap:Ethernet  HWaddr 00:10:0C:00:08:32
```

```
          inet addr:192.168.0.71  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:32 Base address:0x300 DMA chan:1

eth1      Link encap:Ethernet  HWaddr 00:10:0C:00:08:33
          inet addr:192.168.100.71  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:149 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:33 Base address:0x320 DMA chan:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  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

bash-2.05# /sbin/route  ←route でルーティングを見てみる
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.100.71  *               255.255.255.255 UH    0      0        0 eth1
192.168.0.71    *               255.255.255.255 UH    0      0        0 eth0
192.168.100.0   *               255.255.255.0   U     0      0        0 eth1
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         gate.itonet.co. 0.0.0.0         UG    0      0        0 eth0
bash-2.05#
```

標準で起動しているプロセス

```
bash-2.05# ps ax
  PID TTY      STAT   TIME COMMAND
```

```
    1 ?        S      0:02 init
    2 ?        SW     0:00 [keventd]
    3 ?        RWN    0:00 [ksoftirqd_CPU0]
    4 ?        SW     0:00 [kswapd]
    5 ?        SW     0:00 [bdflush]
    6 ?        SW     0:00 [kupdated]
    7 ?        SW     0:00 [rpciod]
  191 ?        S      0:00 portmap
  244 ?        S      0:00 syslogd -m 0
  252 ?        S      0:00 klogd
  265 ?        S      0:00 crond
  275 ?        S      0:02 sshd
  289 ?        S      0:00 httpd
  295 ttySC0   S      0:00 -bash
  296 ttySC1   S      0:00 /sbin/agetty 115200 /dev/ttySC1 -L
  297 ?        S      0:00 httpd
  298 ?        S      0:00 httpd
  299 ?        S      0:00 httpd
  300 ?        S      0:00 httpd
  301 ?        S      0:00 httpd
  302 ?        S      0:00 httpd
  303 ?        S      0:00 httpd
  304 ?        S      0:00 httpd
  331 ?        SW     0:00 [kjournald]
  333 ttySC0   R      0:00 ps ax
bash-2.05#
```

SH-Linux 開発環境構築のまとめ

ここまでの作業で x86 ベースの Linux マシンが SH-Linux クロス開発用のホストマシンになり、カーネルぐらいはコンパイルできることが実証されました。またターゲット（SH-2002）用の NFS ブート環境も正常に動作するようになりました。最終的にターゲットは CF から起動されて使用されることが多いと思いますが、筆者は開発中は NFS ルートでターゲットをデバッグするのが効率的で結構気に入っています。

手順を踏んで構築した NFS ブートは案外簡単だと感じられたのではないでしょうか？。これは sh-ipl が BOOTP によるネットワークブートをサポートしているおかげで、サーバ側の設定だけで高度なネットワークブートが実現できてしまうからそんな感じがするのか

も知れません。

ネットワークブートが標準でサポートされるので SH-200x 用起動 CF カードの作成も SH-200x 自身で行え、その結果余計な問題が発生しなくて快適です。ここでも先人の方々の努力に敬意を表し感謝するばかりです。

SH-200x のソフトウェアで未だ説明していない sh-ipl のコンパイル&インストールの方法、開発マシンで CF カードを作成する方法などを次の章で解説します。

第２章　SH-2002 起動 CF と開発環境の整備

この章で出てくる話は SH-200x 固有の部分もありますが SH-Linux≒Linux に関する部分が主ですから、不明な部分や詳細はインターネット上のサイトや書籍をご参照ください。

２－１　開発マシンで起動 CF を作成する

開発マシンで SH-200x 起動用コンパクトフラッシュ（CF）カードを作成する上でのボトルネックは開発マシンに CF リーダ／ライタをインターフェースすることかも知れません。

開発マシンとしてノートＰＣをお使いの場合は、アダプタを介してＰＣＭＣＩＡスロットに CF を装着できますが、ディスクトップ機の場合はＰＣＭＣＩＡスロット無しが一般的ですから、ディスクトップ機ではＵＳＢ接続のＣＦリーダ／ライタを使うことにします。

ＰＣＭＣＩＡスロットにＣＦを装着した場合は、IDE デバイスの１つとして見えるはずです。またＵＳＢに装着した場合は仮想ＳＣＳＩディスクとしてアクセスします。開発マシンの kernel が、お使いのデバイスをサポートしていないと CF の作成ができません。その場合は、必要なものをコンフグレーションでイネーブルして kernel を再構築してください。

> RedHat7.x に入っている kernel や、そのバージョンアップされた kernel は CF を作成するのに必要なデバイスが入っています。もし kernel を tar ball から作成する場合は、rpm で提供されている kernel ソースに入っている.config をコピーし、それを元に kernel をコンフィグレーションする方が手間がかかりません。

ＰＣＭＣＩＡスロットにＣＦを装着した場合。

ここではテストＰＣとしてＶＡＩＯノートＰＣＧ－Ｃ１ＶＪを例に説明します。

/proc 以下のファイルを見てみます。
CF 未装着の場合 HDD として hda だけが見えています。

```
[nakamura@c1vj nakamura]$ ls /proc/ide
drivers  hda  ide0  ide2  piix
```

ＰＣＭＣＩＡスロットにＣＦカードを装着すると、IDE ドライブ“hde”が増えていることがわかります。（デバイス名はお使いの PC により変わります）

```
[nakamura@c1vj nakamura]$ ls /proc/ide
drivers  hda  hde  ide0  ide2  piix
```

ファイルを cat して確認してみます。

```
[nakamura@c1vj nakamura]$ cat /proc/ide/hde/model
```

SanDisk SDCFB-128

今回使用する CF の容量は 128MB なので全領域を hde1 に割り当てます。
fdisk でパーティションを作成します。（この CF カードの場合、購入時に FAT16 でパーティションが切られているので一旦削除し、Linux のネイティブなパーティションを作成します）

＊ここでの説明では root 権限が必要な操作は sudo を使っています。（1 章参照）

```
[nakamura@c1vj nakamura]$ sudo /sbin/fdisk /dev/hde
コマンド (m でヘルプ): p
ディスク /dev/hde: ヘッド 8, セクタ 32, シリンダ 980
ユニット = シリンダ数 of 256 * 512 バイト
 デバイス ブート     始点        終点   ブロック     ID  システム
/dev/hde1   *           1         979      125296      6  FAT16
コマンド (m でヘルプ): d
領域番号 (1-4): 1
コマンド (m でヘルプ): n
コマンドアクション
   e   拡張
   p   基本領域 (1-4)
p
領域番号 (1-4): 1
最初 シリンダ (1-980, 初期値 1): ↓
初期値 1 を使います
終点 シリンダ または +サイズ または +サイズ M または +サイズ K (1-980, 初期値 980): ↓
初期値 980 を使います
コマンド (m でヘルプ): p
ディスク /dev/hde: ヘッド 8, セクタ 32, シリンダ 980
ユニット = シリンダ数 of 256 * 512 バイト
 デバイス ブート     始点        終点   ブロック     ID  システム
/dev/hde1               1         980      125424      83  Linux
コマンド (m でヘルプ): w
領域テーブルは交換されました！
ioctl() を呼び出して領域テーブルを再読込みします。
警告: DOS 6.x 領域を作成、または変更してしまった場合は、
fdisk マニュアルページにある追加情報を参照してください。
```

ディスクを同期させます。

[nakamura@c1vj nakamura]$

太字が入力する値です。また↓はリターンキーを表します。

次にフォーマットを行います。スイッチ“-j“はファイルシステムを ext3（ジャーナルファイルシステム）にするスイッチです。

```
[nakamura@c1vj nakamura]$ sudo /sbin/mke2fs -j /dev/hde1
mke2fs 1.23, 15-Aug-2001 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
31360 inodes, 125424 blocks
6271 blocks (5.00%) reserved for the super user
First data block=1
16 block groups
8192 blocks per group, 8192 fragments per group
1960 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[nakamura@c1vj nakamura]$
```

SH-2002 ファイルシステムの tar ball を開発マシン上に展開します。

展開する tar ball は SH-2002 に付属している CD-ROM に入っているものや、SH-200x SH-Linux の配布元（http://sh2000.sh-linux.org）などから得てください。もちろん 1 章で作成したファイルシステムを使うこともできます。

ここでは http://sh2000.sh-linux.org/stock/にある sh2002-cf_img-20020920.tar.gz をダウンロードして使用します。

ディレクトリ tmp を作成してダウンロードした sh2002-cf_img-20020920.tar.gz を展開し

ます。

```
[nakamura@c1vj SH-2002]$ mkdir tmp
[nakamura@c1vj SH-2002]$ cd tmp
[nakamura@c1vj tmp]$ sudo tar zxf ../ sh2002-cf_img-20020920.tar.gz
[nakamura@c1vj tmp]$ ls cf_img/
bin   dev   home    lib   opt   root   tmp   var
boot  etc   initrd  mnt   proc  sbin   usr
[nakamura@c1vj tmp]$
```

展開してできたディレクトリ cf_img 以下が SH-2002 の root ファイルシステムになります。ここでの説明で扱うファイルはこのディレクトリ以下にある各ファイルです。

開発マシンの lilo コマンドのバージョンを確かめておきます。

```
[nakamura@c1vj tmp]$ /sbin/lilo -V
LILO version 21.4-4
[nakamura@c1vj tmp]$
```

lilo のコンフィグレーションファイルを必用に応じて作成／修正します。（次のものは lilo.cross として tar ball に入っているものです）

```
①       inear
②       install=/boot/boot-v214.b
③       boot = /dev/hde
③       disk = /dev/hde
④          bios=0x80
        delay = 30
⑤       append="console=ttySC0,115200 mem=128M ide0=0x1f0,0x3f6,34
        ether=32,0x300,0,0,eth0 ether=33,0x320,0,0,eth1"
        image = /boot/vmlinuz
                label = linux
                root = /dev/hda1
                read-only
```

① linear は SH-IPL+g が LBA しかサポートしないので必要です。
② default では/boot/boot.b が使用されます。開発マシン上の lilo のバージョンが 21.4 の場合は boot-v214.b を指定します。
③ HOST 上で CF として認識されているデバイスで、SH-2002(ターゲット上)では、/dev/hda がブートデバイスとなっています。
④ bios=0x80 の指定は、ブートデバイスを /dev/hda にするための記述です。
⑤ オリジナルの SH-LILO では、append は無視されますが、rpm 版では

append に記述したパラメータがカーネルに渡されます。64MB 版の場合、mem=の指定は 64M にします。

環境に合わせて編集するファイル
cf_img/etc/resolv.conf
search itonet.co.jp　←お使いの環境のドメイン名を記述します。
nameserver 192.168.0.5←お使いの環境のネームサーバを記述します。
cf_img/etc/sysconfig/network
NETWORKING=yes　←ネットワークが不要な場合以外は yes です。
HOSTNAME=sh2002.itonet.co.jp　←SH-2002 に与える名前を記述します。
GATEWAY=192.168.0.1　←　default ゲートウェイの IP アドレスを記述します。
cf_img/ etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0　←　デバイス名
BOOTPROTO=static　←　静的に IP アドレスを指定します。
ONBOOT=yes　←　ブート時から起動します。
BROADCAST=192.168.0.255　←　お使いの環境に合わせてください。
IPADDR=192.168.0.71　←　お使いの環境に合わせてください。
NETMASK=255.255.255.0　←　お使いの環境に合わせてください。
NETWORK=192.168.0.0　←　お使いの環境に合わせてください。
cf_img/ etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
BROADCAST=192.168.100.255
IPADDR=192.168.100.71
NETMASK=255.255.255.0
NETWORK=192.168.100.0

eth1 側も eth0 と同様に設定してください。eth0 とは別のセグメントになるよう ip アドレスとネットマスクを指定します。また、eth1 を使用しない場合は、ifcfg-eth1 を削除するか ONBOOT=no としてください。

必要なファイルの修正が完了したら CF にファイルをコピーします。前もって/mnt 以下のマウントポイント cf を作成しておきます。(sudo mkdir /mnt/cf)
[nakamura@c1vj tmp]$ **sudo cp -a cf_img/* /mnt/cf/**
lilo を実行する。
[nakamura@c1vj tmp]$ **sudo /sbin/lilo -r /mnt/cf -C /etc/lilo.cross**
Added linux *
lilo のスイッチ　-r で/（ルート）を変更します
-C でコンフィグレーションファイルを指定します。（デフォルトは lilo.conf です）
補足：-r で lilo を実行する/（ルート）を変更していますから、-C で指定するファイルや lilo.conf(lilo.cross)に記載されているデバイスファイルは、チェンジルートしたディレクトリ以下に必要です（上記では/mnt/cf)。

USB 接続 CF リーダ／ライタの場合。

ここでは、USB 接続の CF リーダ／ライタとして SanDisk 社の SDDR-31 を使用します。

Linux と CF リーダ／ライタの相性は結構くせもので Linux の USB ドライバで動作しないものが多いのが現状です。

リーダライタが正しく認識されているか確認できます。


```
[nakamura@wine nakamura]$ ls /proc/scsi/
scsi  usb-storage-0
[nakamura@wine nakamura]$ cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: SanDisk   Model: ImageMate II
                        Rev: 1.30
  Type:    Direct-Access
                  ANSI SCSI revision: 02
[nakamura@wine nakamura]$
```

リーダライタに CF を装着して、fdisk を行います。/dev/sda は他に scsi を使っている機器がある場合は sdb,sdc･･･と変化します。

```
[nakamura@wine nakamura]$ sudo /sbin/fdisk /dev/sda
コマンド (m でヘルプ): p
ディスク /dev/sda: ヘッド 8, セクタ 32, シリンダ 980
ユニット = シリンダ数 of 256 * 512 バイト
 デバイス ブート     始点        終点   ブロック     ID   システム
/dev/sda1                1         980      125424      6   FAT16
コマンド (m でヘルプ): d
領域番号 (1-4): 1
コマンド (m でヘルプ): n
コマンドアクション
   e   拡張
   p   基本領域 (1-4)
p
領域番号 (1-4): 1
最初 シリンダ (1-980, 初期値 1):
初期値 1 を使います
終点 シリンダ または +サイズ または +サイズ M または +サイズ K (1-980,
初期値 980):
```

```
初期値 980 を使います
コマンド (m でヘルプ): p
ディスク /dev/sda: ヘッド 8, セクタ 32, シリンダ 980
ユニット = シリンダ数 of 256 * 512 バイト
 デバイス ブート     始点         終点   ブロック     ID   システム
/dev/sda1                1         980      125424     83   Linux
コマンド (m でヘルプ): w
領域テーブルは交換されました！
ioctl() を呼び出して領域テーブルを再読込みします。
警告: DOS 6.x 領域を作成、または変更してしまった場合は、
fdisk マニュアルページにある追加情報を参照してください。
ディスクを同期させます。
[nakamura@wine nakamura]$
```

ext3 でフォーマットします。

```
[nakamura@wine nakamura]$ sudo /sbin/mke2fs -j /dev/sda1
mke2fs 1.26 (3-Feb-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
31360 inodes, 125424 blocks
6271 blocks (5.00%) reserved for the super user
First data block=1
16 block groups
8192 blocks per group, 8192 fragments per group
1960 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[nakamura@wine nakamura]$
```

PCMCIA スロットに装着した場合と同じように必要に応じて幾つかのファイルを編集し、CF にコピー後 lilo を実行します。
/dev/sda が CF の場合の lilo.conf と lilo の実行例

```
inear
install=/boot/boot-v214.b
boot = /dev/sda
disk = /dev/sda
   bios=0x80
delay = 30
append="console=ttySC0,115200 mem=128M ide0=0x1f0,0x3f6,34
ether=32,0x300,0,0,eth0 ether=33,0x320,0,0,eth1"
image = /boot/vmlinuz
        label = linux
        root = /dev/hda1
        read-only

[nakamura@wine nakamura]$ sudo /sbin/lilo -r /mnt/cf -C /etc/lilo.cross
Added linux *
[nakamura@wine nakamura]$ sync
[nakamura@wine nakamura]$ sudo umount /mnt/cf
[nakamura@wine nakamura]$ sudo /sbin/modprobe –r usb-storage
```

＊上記のように使用を終了したらモジュール usb-storage を抜いておく方がベターです。

PCMCIA スロット、USB 接続 CF リーダライタどちらの場合でも作成した CF を SH-2002 に装着して起動したら OK です。

２－２　rpm パッケージの追加

SH-200x で提供している現行の RPM 版ディストリビューションでは、セルフ（SH-200x 自身）で rpm パッケージインストールなど rpm 管理ツールが動作しません。ファイルシステムを CF に入れて SH-200x で動作する場合に限らず、nfs ルートで SH-200x にファイルシステムを提供する場合も rpm パッケージの管理は開発マシンの rpm ツールを使用して行います。（第 1 章で root ファイルシステムを作成したのと同じ手順です）
新しいパッケージをインストールする場合。
/bin/rpm --root SH-2002 の root ファイルシステムの位置を絶対パス -Uvh --force --

nodeps --ignorearch –noscripts　パッケージ名

スイッチの説明

--root	ここを root として rpm を実行する
--force	強制的にインストールする
--nodeps	依存関係を無視する
--ignorearch	ターゲットアーキテクチャを無視する
--noscripts	スクリプトを実行しない

例

rpm --root /home/nakamura/SH-2002/cf_img -Uvh --force --nodeps --ignorearch --noscripts nfs-utils-0.3.1-1.sh3.rpm

現在インストールされているパッケージを見る

/bin/rpm --root /home/foo/SH-2002/cf_img –qa

ファイルが属している rpm パッケージを知る。

/bin/rpm --root /home/foo/SH-2002/cf_img –qf　絶対パスで指定するファイル名

例

[nakamura@wine SH-2002]$ rpm --root /home/nakamura/SH-2002/cf_img -qf /usr/sbin/httpd

apache-1.3.12-2

rpm パッケージに含まれるファイルを見る。

/bin/rpm –qlp rpm ファイル名

例

[nakamura@wine SH-2002]$ rpm -qlp nfs-utils-0.3.1-1.sh3.rpm

/etc/rc.d/init.d/nfs

/etc/rc.d/init.d/nfslock

/sbin/rpc.lockd

/sbin/rpc.statd

/sbin/rpcdebug

/usr/sbin/exportfs

/usr/sbin/nfsstat

/usr/sbin/nhfsstone

/usr/sbin/rpc.mountd

/usr/sbin/rpc.nfsd

/usr/sbin/rpc.rquotad

/usr/sbin/showmount

/var/lib/nfs
/var/lib/nfs/etab
/var/lib/nfs/rmtab
/var/lib/nfs/xtab

rpm の詳しい説明は開発マシン上で man rpm などでガイドを見てください。
rpm をインストールする場合、スイッチが結構必要なので次のようなスクリプトを作っておくのが便利かも知れません。

rpm クロスインストールのスクリプト

>$ cat addinst

```
#! /bin/sh
    INST_PATH=/SH-2002/cf_img
/bin/rpm --root $HOME/$INST_PATH -Uvh --force --nodeps --ignorearch --noscripts $1
```

> cat addCF

```
#! /bin/sh
    INST_PATH=/mnt/cf
/bin/rpm --root $INST_PATH -Uvh --force --nodeps --ignorearch --noscripts $1
```

エディタでファイルを作成し、実行権を付けておきます。（例　chmod 755 addinst）

２－３　開発マシンから nfs で CF をマウントする

nfs-utils を SH-2002 にインストールすると SH-2002 の CF 全体や指定したディレクトリを開発マシンで mount することができます。
nfs-utils は CD-ROM に含めていないので次の URL から取得してください。

http://sh2000.sh-linux.org/stock/RPMS/sh3/

開発マシンに SH-2002 起動 CF を装着し、/mnt/cf に mount します。

＜NFS サーバのインストール＞

上記 URL から nfs-utils-0.3.1-1.sh3.rpm を取ってきます。
スクリプト addCF が作成してあるなら、

./addinst nfs-utils-0.3.1-1.sh3.rpm

で CF に nfs サーバがインストールされます。
＜/etc/exports の作成＞

ネットワークが 192.168.?.?で SH-2002 の CF 全体を export するなら次の内容で exports を作成します。

```
sh-2002:~# cat /etc/exports
/mnt/cf              192.168.0.0/255.255.0.0(rw,no_root_squash)
```

インストールとファイルの作成が完了したら開発マシンから umount して SH-2002 に CF を装着して起動します。

SH-2002 が起動したら root で login してテストして見ましょう。
SH-2002 上の nfs サーバの起動は、スクリプトを実行して行います。また、自動起動に設定するなら/sbin/chkconfig で行ってください。

```
bash-2.05# /etc/rc.d/init.d/nfs start
Starting NFS services:  [  OK  ]
Starting NFS quotas: [  OK  ]
Starting NFS daemon: [  OK  ]
Starting NFS mountd: [  OK  ]
bash-2.05# /etc/rc.d/init.d/nfslock start
Starting NFS file locking services:
Starting NFS lockd: [  OK  ]
Starting NFS statd: [  OK  ]
bash-2.05#
```

開発マシンで SH-2002 の CF を nfs mount します。この例では SH-2002 はネームサーバに sh-2002.local.ss-net.net で登録されており、同一のドメイン上のマシンから mount しています。

```
[nakamura@wine SH-2002]$ sudo mount sh-2002:/ mnt
[nakamura@wine SH-2002]$ df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/md2              495844      183365    286879  39% /
途中省略
sh-2002:/             121459       75491     39697  66% /home/nakamura/SH-2002/mnt
```

SH-2002 の CF 全体(/以下)が/home/nakamura/SH-2002/mnt に mount されています。
SH-2002 をシャットダウン／リブートする場合は開発マシンで umount してから行うことをお勧めします。

```
[nakamura@wine SH-2002]$ sudo umount mnt
```

２－４　SAMBA を使う

筆者は、ほとんどの作業を Wincows マシン上の VNC や Teraterm での接続と、WindowsExplorer＋秀丸でファイル編集でやっていますのでほとんどの UNIX（Linux）マシンで SAMBA を動作させています。SH-2002 でも SAMBA が動作するので必要な方

はインストールしてみてください。
samba-2.0.7 が CD-ROM に入っていますので、これを前項の nfs mount でインストールします。（注意　samba は 20MB 近くディスクを消費します）
前項の addinst をちょっと変更してインストール先を CF の mount ポイントにします。

```
#! /bin/sh
    INST_PATH=/SH-2002/mnt
/bin/rpm --root $HOME/$INST_PATH -Uvh --force --nodeps --ignorearch --noscripts $1

[nakamura@wine SH-2002]$ sudo ./addinst /mnt/cdrom/RPMS/sh3/samba-2.0.7-2.sh3.rpm
```

このパッケージはファイルオーナとグループが作成者のままになっていますので警告が沢山でますが特に気にしなくて大丈夫です。
SAMBA のコンフィグレーションファイル smb.conf を編集します。（mnt/etc/smb.conf）
例

```
[global]
        coding system = SJIS
        client code page = 932
        workgroup = LOCAL
        interfaces = eth0
        server string = Samba %v
        encrypt passwords = Yes
        log file = /var/log/samba/log.%m
        max log size = 50
        deadtime = 15
        read size = 65536
        socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
        dns proxy = No
        level2 oplocks = Yes

[homes]
        comment = Home Directories
        writeable = Yes
        browseable = No
```

smb.conf の設定や smbpasswd など SAMBA の各設定は Windows 側でユーザ認証をドメ

イン（アクティブディレクトリ）コントローラがある／無い、ドメイン名（ワークグループ名）などお使いの環境によって設定が異なりますので詳細は man smbd man nmbd や samba のサイトをご覧ください。

SH-2002 自身で NETBIOS（SAMBA）のユーザ認証を行う場合、ほとんどの Windows 環境がエンクリプトパスワードを使用していますから SH-2002 に/etc/smbpasswd ファイルが必要です。
smbpasswd にユーザ名とパスワードを登録するためには/etc/passwd にユーザ名が存在する必要があります。もし root 以外が未登録なら先に Linux んもユーザ登録を行ってください。
SH-2002 のコンソールから

bash-2.05# **/usr/sbin/useradd 新しいユーザ名**

bash-2.05# **passwd 新しいユーザ名**

/etc/smbpasswd にユーザを登録

bash-2.05# **smbpasswd -a 新しいユーザ名**

load_client_codepage: filename /etc/codepages/codepage.932 does not exist.

load_unicode_map: filename /etc/codepages/unicode_map.850 does not exist.

New SMB password:**パスワード**

Retype new SMB password: **パスワード**

Added user **新しいユーザ名**.

SAMBA の準備ができたので起動してみます。

bash-2.05# **/etc/rc.d/init.d/smb start**

[OK] SMB services: [OK]

正常に起動できたら Windows マシンから確認します。
Explorer のツール：ネットワークドライブの割り当て、または、ＣＭＤウインドで net use を使用してドライブを割り当てます。
Explorer 上で「Samba 2.0.7-ja-1.3(sh-2002)の nakamura(Z:)」のようにマイコンピュータに表示されれば正常に動作しています。

２－５　ｔｅｌｎｅｔの設定

SH-2002 root ファイルシステムではネットワークからの login は ssh（セキュアシェル）を default で動作させていますが、もちろん telnet も動作可能です。
telnet は inetd から起動しますので、現在コメントアウトされている telnet の行を有効にし、inetd を起動すれば OK です。
SH-2002 の CF が開発マシンに nfs mount されているなら emacs で編集できます。SH-2002 にエディタは vi しかありません。（telnet の先頭の#を削除します）

[nakamura@wine SH-2002]**$ sudo emacs　mnt/etc/inetd.conf**

```
#
# These are standard services.
#
#ftp      stream   tcp      nowait   root     /usr/sbin/tcpd   in.ftpd -l -a
telnet   stream   tcp      nowait   root     /usr/sbin/tcpd   in.telnetd
#
# Shell, login, exec, comsat and talk are BSD protocols.
#
```

SH-2002 のコンソールから

bash-2.05#　**/etc/rc.d/init.d/inet start**

これで telnet での login が可能です。例によって自動起動にするなら chkconfig で inet を ON にします。

Windows では Teraterm+ttssh と言うフリーソフトウェアがありますし、Linux では ssh が使えるのでセキュリティ上不安のある telnet を特に使用する価値は無いかも知れません。

２－６　ファイル共有と前項までのまとめ

前項のパッケージを追加していない BASIC な SH-2002root ファイルシステムでは rsync、nfs クライアントと ftp サーバがインストールされています。SH-2002 で採用している ftp サーバは proftpd です。初期状態では停止していますので、使用する場合は、次のように起動します。

/etc/proftpd.conf の編集

環境に合わせて編集してください。default で Group が nogroup に設定されていますが、SH-2002 では nogroup が無いので/etc/group に nogroup を追加するか proftpd.conf の Group を nobody などに変更してください。

proftpd.conf の一部

```
# Set the user and group that the server normally runs at.
User                                    nobody
Group                                   nobody
#Group                                  nogroup
```

proftpd を起動する

bash-2.05# **/etc/rc.d/init.d/proftpd start**

rsync の tips

一般的に r で始まるコマンドはセキュリティが弱く、使用するべきではありませんが、rsync は、セキュアシェル ssh の下で動作できますから安全な通信が可能です。

例　開発マシンのファイル honya を SH-2002 の/home/foo/にユーザ名 foo の権限でコピーする

rsync –avz –e ssh honya foo@sh-2002:/home/foo/

スイッチ z や v は任意です。man rsync で確認してください。

root 権限でもコピーできます。

kernel やデバイスドライバを開発中はもとより、アプリケーション開発中も SH-2002 の root ファイルシステムを nfs で行うのが最も効率的ですが、SH-2002 に nfs サーバをインストールすることにより CF を root ファイルシステムにしても結構開発効率を高めることができます。

CF 上の kernel を入れ替えた場合は lilo の実行が必須です

lilo 自身はファイルシステムを持っていませんので kernel（ファイル名 vmlinuz）をロードするのに直接ハードディスク（CF）のセクタ番号でアクセスします。（lilo を実行した時に物理的な位置を取得し、それを使用して kernel を HDD(CF)から取り出します）
kernel を入れ替えた場合、lilo 実行を忘れるとその CF から起動できなくなります。
（最近のブートマネージャに Grub がありますが、これはファイルシステムを持っているので vmlinuz を名前でアクセスできます。しかし結構大きいプログラムなのでディスク（CF）の小さい組み込みマシンには不適合かも知れません）

nfs ルートで起動する場合は開発マシン（別のマシンでも OK ですが）の dhcp サーバが vmlinuz を SH-2002 に提供するので kernel を入れ替えてリブートするだけで OK です。

samba パッケージの追加で SH-2002 の CF を開発マシンで nfs mount して行いましたが、これを開発マシン中の cf_img などを対象にして行うと rpm パッケージインストール後に手動で複数のファイルをコピーするか、あるいは cf_img 以下を全コピー（rsync で差分＆update されたファイルだけのコピーもできますが）し、その後 lilo を実行する手間がかかります。これを省く方法として例示しました。

このように環境を整備するとボードコンピュータ用のソフトウェア開発も結構簡単で効率的に行えることがお分かり頂けましたでしょうか。

SAMBA 経由でファイル編集する場合の注意

smb 共有で Windows マシンから Windows アプリケーションを使用してファイルの編集（コピーも？）行った場合、改行コードが 0xd,0xa（キャリッジリターン、ラインフィード）になる場合があります。一方 UNIX 系テキストファイルの改行コードは 0xa（ラインフィード）だけです。改行コードが異なっても問題が発生しない場合もありますが、スク

リプトファイルなどはエラーとなりますので、Windows 上からテキストファイルを保存する場合は、改行コードがラインフィードだけになるようにしてください。また、アプリケーションで漢字を使う場合はコード体系を euc にするのが安全です。（宣伝ではありませんが秀丸などはこの機能があります）

２－７　sh-ipl+g のコマンドとカスタマイズ

SH-200x では IPL に新部裕氏が作成された、GNU GPL ソフトウェア sh-ipl+g に幾つかの拡張を施して採用しています。

sh-ipl のコマンド

? --- Show this message (HELP)
b --- Boot the system
g --- Invoke GDB stub
l --- Show about license
w --- Show about (no)warranty
e --- Ether Boot

オンボードディップ SW の設定により sh-ipl のコマンド入力待ちをスキップして起動することができます。

DipSW-5=on DipSW-4=off　CF ブート
DipSW-5=oFF DipSW-4=on　nfs ブート

ソースファイルの展開

tar ball を展開するとディレクトリ sh-ipl 以下にソースファイルがインストールされます。

```
>$ tar zxf sh-ipl+g-020901.tar.gz
```

コンパイル&リンクは、次のように行います。

```
>$ cd sh-ipl
>$ make clean (バイナリオブジェクトを削除します)
>$ make sh-stub.bin　（バイナリ形式を作成する場合）
>$ make sh-stub.hex （インテル HEX を作成する場合）
```

＊ コンパイラやアセンブラなどは kernel のコンパイルと同じ sh3-linux-xxx が使用されます。
＊ make に成功するとトップディレクトリに sh-stub.bin（sh-stub.hex）が出来上がります。
＊ sh-stub.hex は P-ROM ライタで EPROM などに書き込む際に利用されます。sh-stub.bin は SH-200x のオンボードフラッシュ ROM に Linux

ベースの書き込みツールで書き込む場合に使用します。

2-1-1　sh-ipl の再構築が必要な場合の例

SH-200x はハードウェアにキーボードと VGA(画面表示)を持たないため、起動コンソールに１つのシリアルインターフェースを割り当てています。出荷時の IPL では SCI（SH-Linux のデバイス名では ttySC0）を使い、115.2kbps8 ﾋﾞｯﾄ 1 ｽﾄｯﾌﾟﾋﾞｯﾄの設定になっています。これを SCIF2（SH-Linux のデバイス名では ttySC1）に変更したりボーレートなどを変更する場合は sh-ipl+g の再構築と SH-200x フラッシュ ROM への書き込み（または PC104 準拠バスに付ける外部 ROM から起動）が必要です。

ここでの変更が有効な範囲は sh-ipl が出すメッセージの出力先とコマンド入力元だけです。SH-200x の起動コンソールは起動のステージよって異なる場所で制御されています。

IPL 中　　sh-ipl+g

Linux 起動当初のメッセージ　lilo のコンソール指定

init に制御が渡った後　　シンボリックリンク/dev/console のリンク先

login 前のプロンプト　　agetty+login

なお、ipl 中は sh-ipl のシリアル IO ドライバ、Linux が起動を始めたら Linux のシリアルドライバが入出力を制御します。

2-1-2　sh-ipl の FlashROM への書き込み

SH-2002 のフラッシュ ROM は Linux の MTD(メモリテクノロジーデバイス)でサポートしています。また mtd 用ツール（mtd-utils-20011106-1.sh3.rpm）が入っています。

次の手順で FlashROPM 上の ipl を update できます。

```
/sbin/modprobe iflash
/sbin/modprobe sh2000-flash writable=1
/sbin/erase /dev/mtd0 0 8
cp /sh-stub.bin /dev/mtd0
/sbin/modprobe -r sh2000-flash
/sbin/modprobe -r iflash
```

第 3 章　SH-Linux kernel＆デバイスとドライバ

Linux はリーナスさんの手によりインテル社の x86（ここでは 80386 以降をまとめて x86 と呼ぶ）搭載のパソコンに乗せる OS として開発されたが、その後何種類もの CPU 上で稼動するように拡張がなされた。日立 SuperH の中で SH3 以上の CPU には MMU（メモリマネジメントユニット）が入っているので仮想記憶を使用する x86 や Sparc と同じ仕様の Linux を動作させることができる。

Linux を新しい CPU に移植するためにはその CPU 用のコンパイラ(gcc)が必須であり、gcc と共に使用するアセンブラ(as)やリンカを含む binutils も必要である。また、狭義の Linux は kernel だけを指すが、コンピュータ装置として機能するためには各種アプリケーション類の移植が必要となる。このアプリケーション類にライブラリを提供する glibc の移植も必須なものである。

これらの最新版(2002/9 現在)は、

gcc が 3.04
binutils が 2.11.2
glibc が 2.2.5

である。

開発マシンに使用している x86 ベースの Linux をコンパイルしている gcc のバージョンは 2.96 あたりである。SH-Linux をコンパイルするための開発環境に最新のものを使用している理由は、SH-Linux 全体がまだまだ EXPERIMENTAL なものである証と考えて頂きたい。kernel についても同様で、本書執筆時点の最新が 2.4.19、SH-2002 のリリースは 2.4.18 であるが、逐次新しいバージョンがリリースされるし、SH-2002kernel のビルド番号は毎日のように更新している。皆さんの開発ツールや SH-2002kernel そしてライブラリ、各アプリケーションもフットワーク軽くバージョンアップに追従して頂き、より一層安定なシステムを目指してください。また、バグを修正／発見されたり、新しいドライバの開発や改良がなされたら是非コミュニティで公開して頂き、より一層完成度高い OS を目指そうではありませんか。

３－１　kernel の話

kernel ツリーを展開するとトップディレクトリに arch があります。その下に各 CPU 毎のコードが入っています。したがって SH-Linux 固有のコードは arch/sh 以下にあります。また、トップディレクトリの include の中に各ＣＰＵ毎のインクルードファイルがディレクトリ名 asm-CPU 名以下にあります。
各デバイスのドライバはトップディレクトリの drivers 以下にカテゴリ毎のサブディレクトリに分かれて入っています。

arch/sh 以下は SH 専用なので比較的自由に改竄しても将来のバージョンアップで破綻を来たすことは少ないと思われますが、場合によっては drivers 以下のファイルなどに手

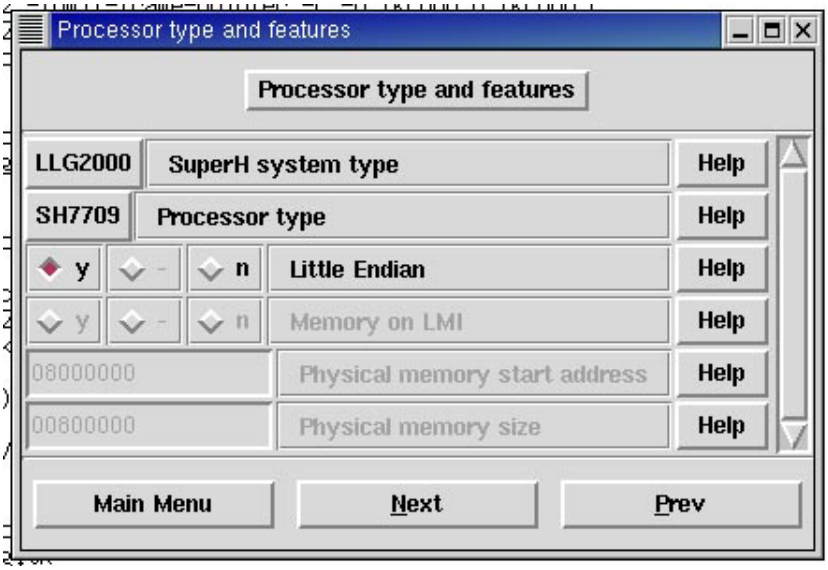
を入れる事態が起こるかも知れません。このような場合でも将来のバージョンアップで破綻を来たさないソースの書き方が結局自分のシステムを将来に渡って安定に保つ秘訣かも知れません。

SH-2002 用 kernel とモジュールを手早く作るには、トップディレクトリにある config.sh2002 を.config にコピーして次のように make します。
make menuconfig（修正が無くても一回は起動し、ファイルをセーブする必要があります）
make dep (セーブされた.config をベースに依存関係を解決し、必要なシンボリックリンクなども生成されます)
make zImage（圧縮された kernel が arch/sh/boot/zImage として作成されます）
make modules（menuconfig(xconfig)でモジュール指定されたオブジェクトが作成されます）
make modules_install（モジュールをインストールします。インストール先はトップディレクトリの INSTALL_MOD_PATH=で指定されたところになります。SH-2002 用カーネルでは INSTALL_MOD_PATH=$(HOME)/SH-2002/export と設定していますが、変更は自由です）

重要：SH-2002 用 kernel は menuconfig→Processor type で LLG2000 を指定します。
（LLG2000 は SH-2002 の ITO 社内品番です）

xconfig でのプロセッサタイプ指定

ボードは LLG2000
CPU は SH7709
エンディアンは
リトルを指定します



Processor type に LLG2000 を指定すると、トップディレクトリの.config に

CONFIG_SH_LLG2000=y が生成されます。これが、どのハードウェア用 kernel を生成するかのベースになります。
例えば arch/sh/kernel/Makefile ではハードウェア依存コードを生成する部分に次のような記述があります。

```
obj-$(CONFIG_SH_LLG2000)        += setup_llg2000.o io_llg2000.o sh2002_io.o
machine-specific-objs           += setup_llg2000.o io_llg2000.o sh2002_io.o
```

CPU に同じ SH3 を使用したボードでも IO のマッピングなどが異なるため、各ハードウェアの固有部分は別のソースになっています。

IO ポートのこと
x86 系 CPU はメモリアクセス以外に IO アクセス用の別命令があり、アドレスバスの下位 16 ビットと ior/iow コマンドで IO デバイスにアクセスします。しかし SH3/4 を含めて大半の CPU はメモリアクセスと IO アクセスを区別しません（アドレスで分けるだけです）
Linux のデバイスドライバは x86 ベースで記述されているので（例外もあるかも知れません）例えば IDE は 0x3f6 とかシリアルインターフェースのチップは 0x3f8 など 16 ビットで表されています。（構造体などで IO アドレスを保持している変数も 16 ビット長の場合がある）。sh3/4 の場合アドレス長は 32 ビットあり、SH-2000 や SH-2002 の場合、cs5 エリアをボード内部 IO エリア、cs6 を PC104 準拠バスの IO エリアとしています。デバイスドライバを全面的に書き換えることなど手間はもちろん、互換性からもできませんから、io_llg2000.c に次のような変換ルーチンを持っています。

```
#define IDE_OFFSET      0xb6200000
#define NIC_OFFSET      0xb0400000
#define EXTBUS_OFFSET 0xba000000
#define USB_OFFSET      0xb0800000
#define PORT2ADDR(x) llg2000_isa_port2addr(x)
unsigned long llg2000_isa_port2addr(unsigned long offset)
{
        if((offset & ~7) == 0x1f0 || offset == 0x3f6)
                return IDE_OFFSET + offset;
        else if((offset & ~0x3f) == 0x300)
                return NIC_OFFSET + offset;
        else if((offset & ~0x3f) == 0x1000)
                return USB_OFFSET + offset;
        return EXTBUS_OFFSET + offset;
}
```

IO ファンクション（out?や in?）はこのアドレス変換ルーチンでアドレス変換されます。

kernel の詳細な説明は興味深いところが多いと思いますが、筆者の知識限界を超えていますし、本書の趣旨からもずれていくかも知れません。kernel を理解する最も良い資料はお手元のマシンにあるカーネルツリーのソースファイルです。じっくりと取り組まれたら幾つもの発見があるに違いありません。

３－２　オンボードディップ SW と LED

SH-2002 オンボードディップスイッチの読み出しと、LED の制御を行うデバイスドライバが組み込まれています。

デバイスファイル（/dev/sh2002）が無い場合は、メジャー番号 10,マイナ番号 102、キャラクタデバイスを/dev/に作成してください。

mknod –m 666 /dev/sh2002 c 10 102

このデバイスドライバはディップスイッチ(S2)の読み出しと、LED1 および CN1-5 番ピンの制御を行えます。（LED2 はオンボード IDE(CF)アクセスまたは nfs サーバアクセスに連動しています）

ディップ SW

ハードウェア・sh-ipl での割り当て

SW-No	割り当てられた機能	ON	OFF
1	未定義		
2	未定義		
3	未定義		
4	OS ブート	nfs ブート	共に OFF なら IPL コマンド待ち
5		CF ブート	
6	JTAG デバッガ	使用	ノーマル
7	FlashROM 書き込み禁止	書き込み禁止	書き込み許可
8	IPL デバイス	PC104 から IPL	オンボード FlashROM から IPL

デバイスドライバではスイッチ OFF で 0 が読み出されます。

テストプログラム

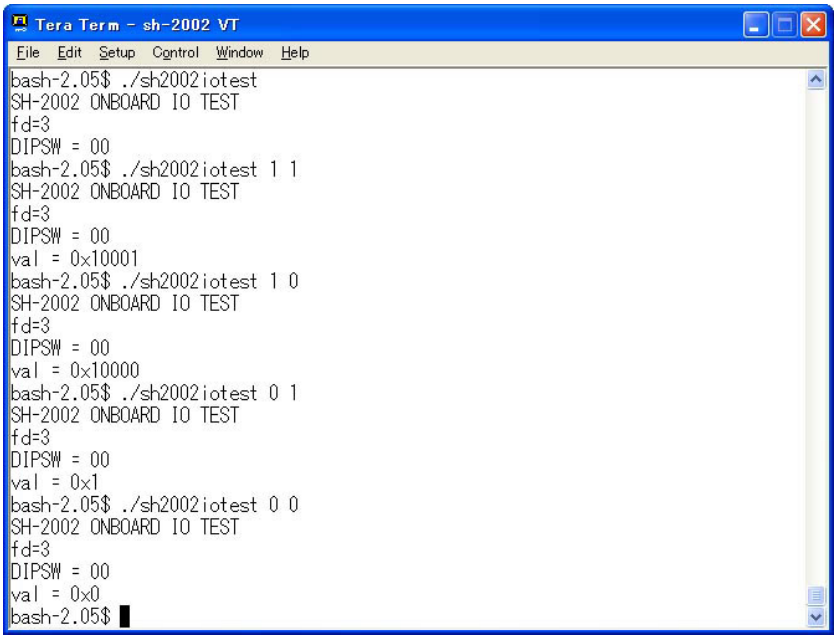
コンパイルは、sh3-linux-gcc –O2 –o out-file in-file.c

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
```

```
#define PIO_READ                0xc001
#define PIO_WRITE               0xc002

int main(int ac, char **av)
{
        unsigned long val;
        int fd, i, pin, pinref, j;
        unsigned long n;

        printf("SH-2002 ONBOARD IO TEST\n");
        pinref = 0;
        fd = open("/dev/sh2002", O_RDONLY);
        printf("fd=%d\n", fd);
        if(fd < 0)
        {
                perror("/dev/sh2002");
                exit(1);
        }

        val = 0;
        if(ioctl(fd, PIO_READ, &val) < 0)
        {
                perror("PIO_READ");
        }
        printf("DIPSW = %02x\n", val & 0xff);

        if (ac >= 3)
        {
                val = atoi(av[1]) << 16 | atoi(av[2]);
                if(ioctl(fd, PIO_WRITE, &val) < 0)
                {
                        perror("PIO_WRITE");
                }
                printf("val = 0x%x\n", val);
        }
        return 0;
}
```

このデバイスドライバは kernel ツリーの arch/sh/kernel/sh2002_io.c　です。このドライバは open,release と ioctl しか持っていません（read,write は未実装ですし、割り込みも使用しないので割り込みハンドラもありません）

実行例



1 つ目の引数　1=LED1　0=CN1-Pin5

２つ目の引数　1=H を出力　0=L を出力

その他の LED

LED3～6 は Ethernet コントローラ(LAN91C111L)の LED ポートに接続されています。
またこのポートは、バッファを経由して CN1 にも接続されています。

			初期値
LED3	CN1-Pin1	eth0－LEDA	0
LED4	CN1-Pin2	eth0－LEDB	5
LED5	CN1-Pin3	eth1－LEDA	0
LED6	CN1-Pin4	eth1－LEDB	5

0	nPLED3+ nPLED0 – Logical OR of 100Mbps Link detected 10Mbps Link detected(default)
1	Reserved
2	nPLED0 - 10Mbps Link detected
3	nPLED1 – Full Duplex Mode enabled
4	nPLED2 – Transmit or Receive packet occurred
5	nPLED3 - 100Mbps Link detected
6	nPLED4 – Transmit packet occurred
7	nPLED5 – Receive packet occurred

Ethernet の LED 機能を変更したい場合は/proc に値を書き込むことでできます。
例　eth0　LEDA を送受信で点灯させる。

bash-2.05# **echo 4 > /proc/sys/dev/eth0/leda**

３－３　時計

SH-2002 は CPU 内臓の時計がバックアップされています。Linux は起動時にこの時計を読んでシステム時計に設定します。Linux のシステム時計は CPU 内蔵時計とは無関係にタイマ割り込みにより処理されています。

システム時計の合わせ方１

bash-2.05# **date -s "2002/9/2 12:01:23"**

“”で囲んだ引数をシステム時計に設定します。

システム時計の合わせ方２

bash-2.05# **/usr/sbin/ntpdate ntp サーバ名**

ntp サーバが使用できる場合

どちらの場合もシステム時計に新しい時刻が設定されるだけで CPU 内臓の時計は更新されません。

CPU 内臓時計の確認と更新

確認

```
bash-2.05# /sbin/hwclock
Mon Sep 23 11:42:21 2002    -0.159087 seconds
```

更新

bash-2.05# /sbin/hwclock --systohc

システム時計の内容が CPU 内臓時計に設定されます。

システム時計は ntp デーモンで他の ntp サーバと同期することもできます。

謝辞

Linux を開発された Linus Torvalds 氏始め多くのプログラムマの方々、新部氏、小島氏を始め SH-Linux 開発に多大な貢献をされた方々に感謝致します。

参照 URL

http://sh-linux.org/index-j.html

http://sh2000.sh-linux.org/index.html

http://www.m17n.org/linux-sh/

http://dodo.nurs.or.jp/~kkojima/

http://sourceforge.net/projects/linuxsh

株式会社アイ・ティオ・オー

info@sh-linux.org