Linux From Scratch アンケート結果・参考資料

本稿は１月２５日の OSC 大阪での講演 **Linux From Scratch 全てソースコードから構築するＯＳ** の出席者アンケートの集計結果、並びに configure と make カーネル構築についての簡単な紹介です。

アンケート集計結果　　回答 12 名

１． LFS を構築する（学ぶ）理由は何ですか？（複数回答可。一番大事な物に◎）

（12）OS についての理解を深める学習
（ 0）仕事の上で必要となっている
（ 1）柔軟性の高いシステムが欲しい
（ 2）高速作動するシステムが欲しい
（ 1）構築（configure, make）の標準的手法について調査している
（ 0）その他（具体的にお願いします）

複数回答で◎
（ 1）OS についての理解を深める学習
（ 1）高速作動するシステムが欲しい

２．普段開発に使用されているシステムは何ですか？（バージョンは不要）
複数ある場合は利用頻度の順で列記して下さい。

（略）

３．make はご存知ですか？普段お使いですか？

（ 5）知らない、耳にした程度
（ 3）普段開発で使っている。プロジェクトリーダーが書いた Makefile を実行
（ 4）自分で Makefile を書く
（ 0）複数の Makefile を階層的に使ったり、変数や条件分岐を使うなど make に習熟している

４．次の構築手順の経験はありますか?

```
tar xfz package.tar.gz
./configure
make
make check
make install
```

（ 1）全く無い
（ 6）一度～数度ある
（ 0）何度も行っているので慣れている
（ 2）configure のオプションを操作したことがある
（ 2）Makefile やソースコードを目的に応じて修正したことがある
（ 1）configure のオプションを操作＋Makefile やソースコードを修正したことがある

5. Linux カーネルを自分で構築したことはありますか？

（ 9）試みていない
（ 2）試行したが成功していない
（ 1）一二度成功している
（ 0）何度も行っているので慣れている

６．Linux のカーネルの初設定等の configure のドキュメントは英語が主体です。
日本語文献が無くて（古くて）困ることはありますか？

（ 4）日本語の文献が無いのがネックでソースコードからの構築はほとんどできない
（ 4）困ることが時々ある
（ 1）困ることはほとんどない
（ 2）無回答

７．日本の産業でも広く使われている品質管理の基本がベル研究所由来とご存知でしたか？

（ 2）知っていた
（ 2）「論語とコンピュータ」で知った
（ 8）知らなかった

８．Linux From Scratch をやってみようと思いますか？
（ 7）挑戦してみたい
（ 5）もう少し勉強してから挑戦したい
（ 0）一生しないだろう

ご回答頂いた皆様にお礼申し上げます。

make 学習について

make とは

更新されたソースファイルだけコンパイルし直すことによって実行形式生成に要する時間を削減することを目的に make は開発されました。実際の利用においては、ソースコードを更新したのにコンパイルをするのを失念して古いオブジェクトをリンクしてしまったというミスを防ぐのに有用で利用者が増えて行きました。現在では構築手順の記述ならびに全般的な構築管理の標準的方法になっています。

スクリプト言語を主体に仕事をされている方にはなじみが薄いかもしれませんが、検査の自動化など役立つ場面が多いと感じます。

LFS のプログラム、ライブラリのパッケージはごく少数の例外を除いて make で構築します。しかし重要なソフトはさまざまな環境に適応可能なように組まれているため Makefile は複雑で見通しが悪く、初学者の参考には向きません。LFS 収録のパッケージで例外的に単純で見通しの良いものを紹介します。

圧縮プログラム bzip2 の Makefile が最も単純とされます。
https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz

コマンドライン計算ソフト bc の Makefile も解読が容易です。
変数を活用しているので、上記 bzip2 より少々難しくなります。
https://github.com/gavinhoward/bc/archive/2.5.3/bc-2.5.3.tar.gz

bzip2 の Makefile の閲覧方法

ファイルを展開するので作業ディレクトリで行います。

```
tar xfz bzip2-1.0.8.tar.gz
cd bzip2-1.0.8
ls Makefile
```

ファイルを展開しないで閲覧する方法:

```
tar xfzO /src/other/bzip2-1.0.8.tar.gz bzip2-1.0.8/Makefile | less
```

make の試行実験は一般ユーザーとして行います。特権ユーザーとなる必要は make install 以外ありません。試行中は何かをインストールしてシステムを変更しないためにも一般ユーザーとしておくべくです。

configure の例

一般的に configure は非対話型のプログラムです。（主な例外は Linux カーネルと Netpbm。）

一般に調整はコマンドラインのオプションで行います。どういうオプションがあるか知りたい時は

```
configure --help
```

とします。

以下は失敗例です。LFS 教書の指定するシステム要件が満たされていなかったり、指示から逸脱した場合にこのような設定失敗に遭遇します。教書の通りにしていれば起きません。

```
checking for a BSD-compatible install... /bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of
Makefiles... no
checking for style of include used by make... GNU
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed

  … 略 …

checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs... immediate
checking whether stripping libraries is possible... yes
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... yes
checking for gmp.h... no
configure: error: gmp.h cannot be found or is unusable.
```

make の例

make も非対話型のプログラムです。実行すると以下のようにコンパイル、リンク命令が実行され、その状況が表示されます。以下の例はごく小さい LFS 外のパッケージです。一般的な LFS 収録のパッケージでは何千行も経過出力されます。

```
cc   -DVER_REVISION=\"7.3.4\" -DVER_DATE=\"2016-05-24\"
-DVER_AUTHOR=\"'Erwin Waterlander'\" -DDEBUG=0   -DD2U_UNICODE
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -DENABLE_NLS
-DLOCALEDIR=\"/usr/share/locale\" -DPACKAGE=\"dos2unix\" -O2 -Wall
-Wextra -Wconversion    -c dos2unix.c -o dos2unix.o
cc   -DVER_REVISION=\"7.3.4\" -DVER_DATE=\"2016-05-24\"
-DVER_AUTHOR=\"'Erwin Waterlander'\" -DDEBUG=0   -DD2U_UNICODE
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -DENABLE_NLS
-DLOCALEDIR=\"/usr/share/locale\" -DPACKAGE=\"dos2unix\" -O2 -Wall
-Wextra -Wconversion    -c querycp.c -o querycp.o
cc   -DVER_REVISION=\"7.3.4\" -DVER_DATE=\"2016-05-24\"
-DVER_AUTHOR=\"'Erwin Waterlander'\" -DDEBUG=0   -DD2U_UNICODE
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -DENABLE_NLS
-DLOCALEDIR=\"/usr/share/locale\" -DPACKAGE=\"dos2unix\" -O2 -Wall
-Wextra -Wconversion    -c common.c -o common.o
cc dos2unix.o querycp.o common.o     -o dos2unix
ln -sf dos2unix mac2unix
cc   -DVER_REVISION=\"7.3.4\" -DVER_DATE=\"2016-05-24\"
-DVER_AUTHOR=\"'Erwin Waterlander'\" -DDEBUG=0   -DD2U_UNICODE
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -DENABLE_NLS
-DLOCALEDIR=\"/usr/share/locale\" -DPACKAGE=\"dos2unix\" -O2 -Wall
-Wextra -Wconversion    -c unix2dos.c -o unix2dos.o
cc unix2dos.o querycp.o common.o     -o unix2dos
ln -sf unix2dos unix2mac
msgfmt -c po/da.po -o po/da.mo
msgfmt -c po/de.po -o po/de.mo
msgfmt -c po/eo.po -o po/eo.mo
msgfmt -c po/es.po -o po/es.mo
msgfmt -c po/fr.po -o po/fr.mo
msgfmt -c po/hu.po -o po/hu.mo
msgfmt -c po/ja.po -o po/ja.mo
msgfmt -c po/nb.po -o po/nb.mo
msgfmt -c po/nl.po -o po/nl.mo
msgfmt -c po/pl.po -o po/pl.mo
msgfmt -c po/ru.po -o po/ru.mo
msgfmt -c po/sr.po -o po/sr.mo
msgfmt -c po/sv.po -o po/sv.mo
msgfmt -c po/uk.po -o po/uk.mo
msgfmt -c po/vi.po -o po/vi.mo
msgfmt -c po/pt_BR.po -o po/pt_BR.mo
msgfmt -c po/zh_CN.po -o po/zh_CN.mo
msgfmt -c po/zh_TW.po -o po/zh_TW.mo
```

make check の例

検査はパッケージごとに方法が異なります。make check ではなく、make test の場合もあります。下記は類例の多い流儀で、個別項目の成功失敗は結果出力の途中に出現します。個別の成功失敗を確認するには出力をファイルに格納するなどします。下記は途中を抜き出したものです。

```
creating t-sub
make[4]: Leaving directory `/tmp/gmp-4.2.1/tests'
make  check-TESTS
make[4]: Entering directory `/tmp/gmp-4.2.1/tests'
PASS: t-bswap
PASS: t-constants
PASS: t-count_zeros
PASS: t-gmpmax
PASS: t-hightomask
PASS: t-modlinv
PASS: t-popc
PASS: t-parity
PASS: t-sub
==================
All 9 tests passed
==================
make[4]: Leaving directory `/tmp/gmp-4.2.1/tests'
make[3]: Leaving directory `/tmp/gmp-4.2.1/tests'
Making check in devel
```

Linux カーネル構築時の設定

Linux カーネルには膨大な数の設定項目があります。configure が数種類用意されています。LFS 教書では簡易 GUI ncurses による make menuconfig を指定します。対話形式の設定となります。多くの項目は要・不要、モジュール（常駐せず）の選択となります。以下に設定過程の画面を掲載します。

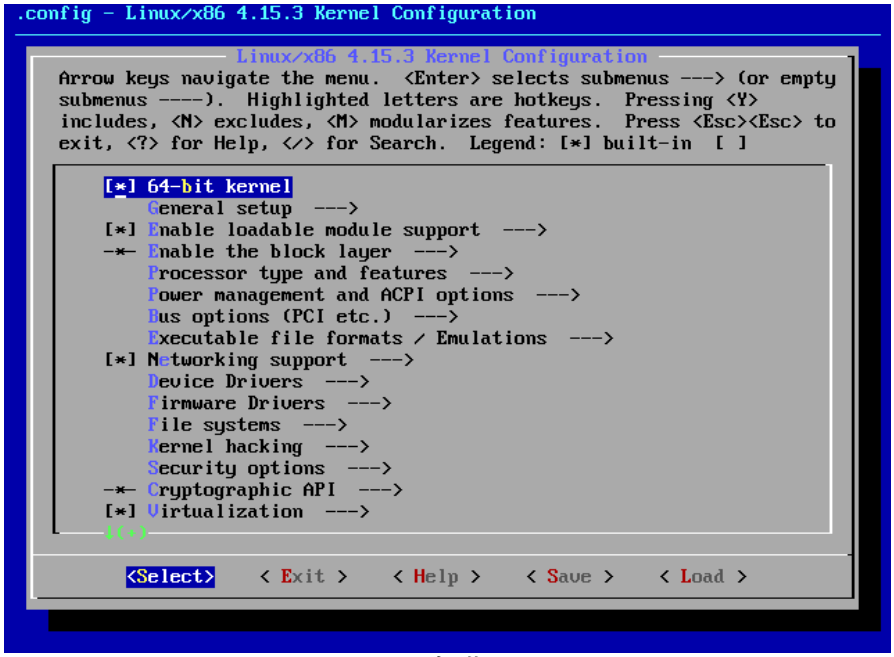

図 1: 初期画面

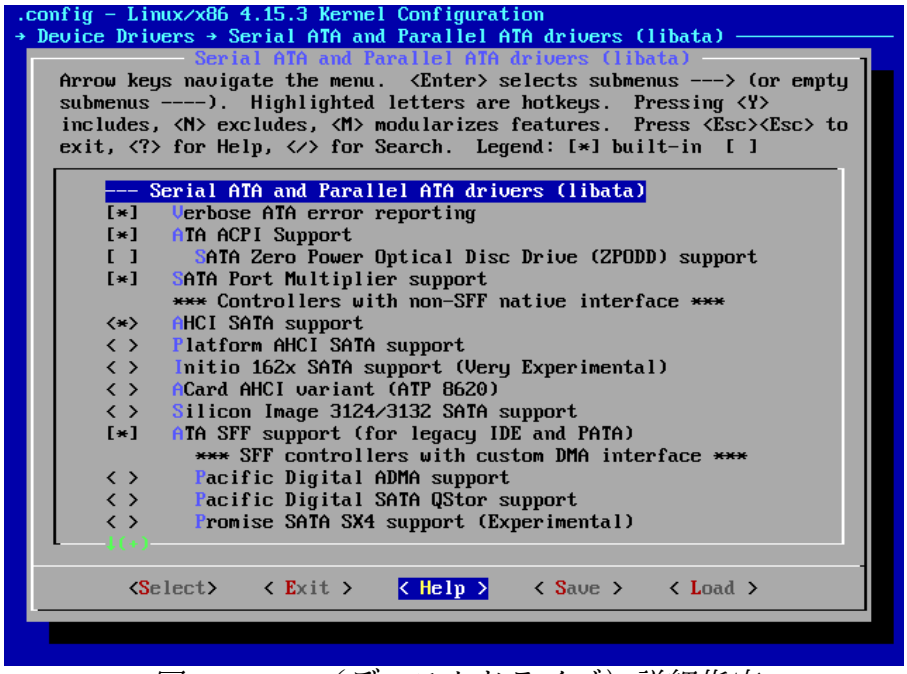

図 2: SATA （ディスクドライバ）詳細指定

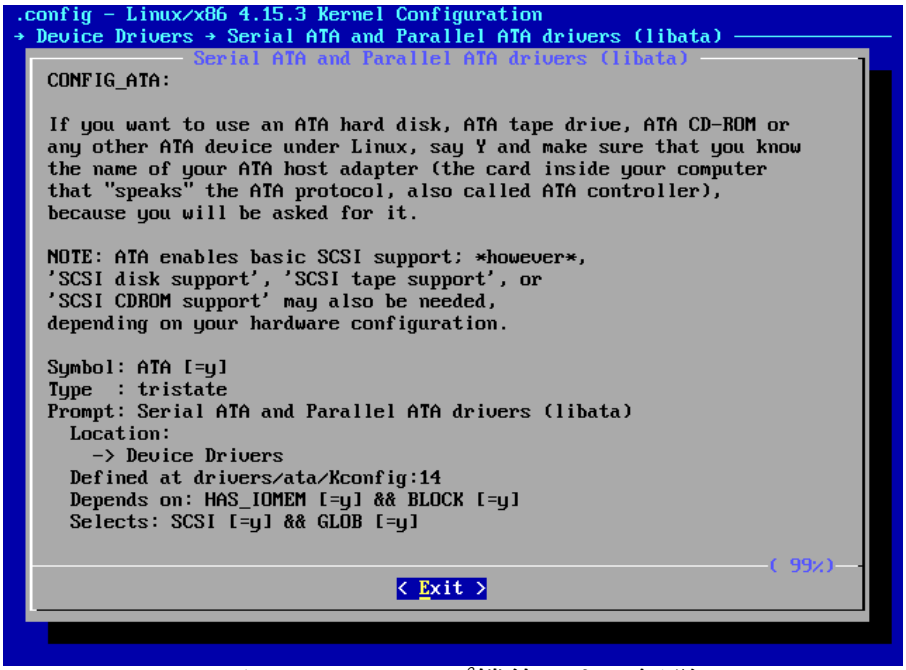

図 3: SATA ヘルプ機能による解説

ほとんどの設定項目に上のようなヘルプ解説がついています。

大多数の利用者にとって必須となる項目はその旨明記されています。特殊な用途でしか利用されない項目は「聞いたことなかったら君には不要だよ」等と書かれています。

余談　標本抽出による検査

プロジェクトの進捗管理は計画、実行、検証、調整のサイクルで行われます。基幹ソフトのパッケージでは検証は `make check` が重要です。LFS の進捗管理は各自が工夫して行います。

電話の機器のような量産工業製品では標本抽出による品質検査が有効とされていますが、この統計的手法はベル研究所で生まれたものです。

「標本（サンプル）を抽出して検査した方が全数検査するより正確」と言われます。意外な感じがしますが、全数検査すると 1 つの検体あたりにかける時間が少なくなり、正確さが損なわれてしまうと言うのです。限られたサンプルを詳しく、正確に測定した方が有効なのです。

この考え方によく当てはまる例が最近の新型肺炎の患者数のデータです。日本政府の手配したチャーター機で武漢を離れた 565 人のうち 9 人がウイルス陽性と判定されています。1/63 の割合です。一方、中国政府の発表では 2020 年 1 月末で武漢の感染者は 3000 人位、人口比にして 1/4000 位です。桁が違います。日本で行われた検査が丹念で精度が高かったからこの違いとなったのです。

世界の疫学の専門家は新型肺炎の感染力や致死率がまだよくわからず先が読みにくいと言っています。こういう時は標本抽出して詳しく調べるのが有効な方法です。

自動化によって全数検査の負担が大きくない場合は、人手による検査とは方針が違って来ます。プログラマは製造業の品質管理を普段は意識しないかもしれませんが、この方面の知識は仕事の開拓には有効なはずです。

参考書籍　資料　リンク

GNU Make

Robert Mecklenburg 著　矢吹達郎監訳　菊池彰訳
オライリー　第 3 版 2005 年 12 月　304 ページ
https://www.oreilly/.co.jp/books/4873112699

Linux From Scratch （英文）

https://www.linuxfromscratch.org/

LFS 日本語訳

https://lfsbookja.osdn.jp/

Pi LFS （英文）

https://testinate.com/pilfs/
ラズベリーパイの LFS

Linux From Scratch　全てソースコードから構築する OS

https://www.ospn.jp/odc2019/pdf/ODC2019_Netpbm.pdf
オープンデベロパーズカンファレンス（ODC）２０１９における講演資料

論語とコンピュータ

https://www.ospn.jp/osc2018-nagoya/pdf/OSC2018_Nagoya_netpbm%202.pdf
知識共有の古典思想、ベル研究所と品質管理、UNIX、GNU、Linux の歴史を概説しています。

2020 年 2 月　漆畑晶