Linux From Scratch

漆畑 晶

Linux From Scratch

全てソースコードから構築するOS

Linux From Scratchとは何か

創始

Linux From Scatch (LFS)はジェラード・ビークマンズ(Gerald Beekmans)氏が自分で全てソースコードからOSを構築した経験を詳細にまとめた教書を1999年に発表したのが始まりです。

From Scratchの語義

From scratchとは「スタートラインから」という意味の英語俗語表現です。Scratchとは「引っ掻く」という意味です。運動会で地面に棒で掻いた線をスタートラインとしたのが言葉の由来です。

From scratchは今日では手作りの料理を言うのによく使われます。"My mother makes pizza from scratch." とは、生地は小麦粉をこねて発酵させ、ソースは生のトマトから作り、瓶詰めの市販のソース等は使わない徹底的な手作りという意味です。

LFS構築の目的

LFSを構築する理由として多いのは学習目的です。作ることを通して、UNIX系OSの全体像を把握しようというものです。ビークマンズ氏もその意義を感じて発表しました。

その他に設定を操作しやすい柔軟性のあるシステムや高速なシステムが欲しいというのもあります。

LFSの構成部品がどのように構築されているかをみれば構築方法の業界標準や最近の動向を知ることができます。

ここで言うLinuxはカーネルだけではありません。むしろカーネル以外の部分に重点があります。GNU/Linux From Scratchと呼ぶ方が正確です。

LFSはどのようなシステムか

完成したLFSは自己完結したシステムです。

自由ソフトウエアの提唱者、フリーソフトウエア財団の創設者リチャード・ストールマン博士は４つの自由を唱えています。

0. プログラムを実行する自由
1. 研究し、改変する自由
2. 再配布する自由
3. 改変したものを配布する自由

LFSは以上の４つが自由にできます。開発用のソフト一式とエディタが整っていて、基本的な開発作

業が可能です。配布は SD カード等の着脱可能な記憶媒体で行えます。電子メールやファイル交換用ソフトは基本システムに含むまれませんが、LAN、インターネット接続まではできているので少しの手間で追加できます。言語は c,c++,Awk,Perl,Python が標準装備です。

LFS には何が必要か

一版的なインテルか AMD の CPU を実装したパーソナルコンピュータで構築できます。ARM と PowerPC も一部対応しています。

必要なソフトウエアとそのバージョンのリストがありますが、GNU/Linux ディストリビューションの最新版のインストール時に「開発用」のパッケージ集を指定していればほとんどのソフトウエアは揃うと思われます。

MacOS からは 2004 年に直接構築に成功したとの報告がありますが、通常はまず GNU/Linux ディストリビューションをインストールしてそこから作業します。

LFS は自己完結していますから、LFS から同じ版の LFS、あるいは将来の版の LFS を構築することが可能です。

LFS の関連プロジェクト

LFS の関連プロジェクトには BLFS CLFS ALFS があります。また独立した PiLFS というプロジェクトもあります。BLFS は一部日本語訳されています。

BLFS　Beyond Linux From Scratch

発展編。電子メールの送受信や WWW 閲覧、音声、CD-ROM の読み書きのためのソフトウエアを構築します。X Window は本編ではなく BLFS に含まれます。Apache,Git,Libre Office, Firefox など定番ソフトが構築できます。

Firefox など大型アプリケーションは必要な追加パッケージが多く、LFS 本編の完成までに要するのと同じくらいの時間を費やして辿り着く感じです。依存関係の樹形図を作成してから計画的に作業を進めます。

CLFS　Cross Linux From Scratch

クロス・コンパイル。例えば 32 ビット型の機材で 64 ビット型のためのシステムを構築します。難易度は高く、面倒ですが、おもしろいと言えます。上級者向けです。

ALFS　Automated Linux From Scratch

構築手順の主要部分を自動化。学習目的には向いていません。

LFS を構築したら、使い勝手がよく使い続けていきたいという人が 2 度目以降の構築に使うに向いています。

Pi LFS　Raspberry Pi Linux From Scratch

ラズベリーパイのシステムをソースコードから構築。ブートローダーだけは非公開なので構築できません。LFS とは独立したプロジェクトです。ALFS のように自動化されています。

LFS 構築のしくみ

通常のインストーラのしくみ

GNU/Linux を利用される方はインストール用の CD－ROM などをお使いになっていると思います。

LFS がどのようなプロセスかを考えるのに、通常の GNU/Linux インストーラーの働きを想起すると良いと思います。インストーラの機能は大きく分けて 3 つあると言えます。

1. ハードディスクの区画
2. パッケージを順次インストール
3. 設定

通常のインストーラではハードディスクの区画、各パーティションの利用方法指定の対話型のインタフェイスが用意されています。LFS では cfdisk, fdisk などを使ってハードディスクにパーティションを作成し、mkfs でファイルシステムを、mkswap でスワップ領域をそれぞれ作成します。

パッケージのインストールは通常は実行型式ファイルを集めたアーカイブを展開する形で行われています。LFS でどうなっているかはこの後で詳しく述べます。

設定には起動設定、ユーザーとパスワードの登録、インターネット接続のための設定などがあります。とくに起動設定が重要です。通常のインストーラでは対話型のインタフェイスで質問項目に答える形で設定していくものですが、LFS では設定ファイルを直接編集します。いずれも雛形が用意されていて、必要に応じて修正する形となります。

一般的なソースパッケージの構築・インストールの手順

LFS では約 70 のパッケージをソースから構築してインストールします。多くのパッケージの構築は以下の手順に従います。

```
./configure
make
make install
```

configure はパッケージごとについているシェルスクリプトです。その大きな役割はインストール先のシステムの機能詳細の自動判定です。オプション機能の選択指定、インストール先の指定もします。

configure を始めたのはリチャード・ストールマン博士です。Linux カーネルの発表される前の時代でストールマン博士の率いるフリーソフトウエア財団の GNU プロジェクトのソフトウエアは色々な商用 UNIX、そして MS-DOS にインストールされていました。それぞれの OS の違いに対応するのが目的でした。

configure スクリプトは autoconf というプログラムにより自動生成されているものが多く、インストール先の指定方法、デフォルトのインストール先はどこも同じです。

インストール先を特に指定したい場合は次のように --prefix を使います。次の例では全体を/usr/local/apricot にインストールします。

```
./configure --prefix=/usr/local/apricot
```

この--prefix を指定しなければインストール先は/usr/local/です。次のように指定したのと同じことになります。

```
./configure --prefix=/usr/local/
```

新しいパーティションにインストールしていく方法

それでは新しくパーティションを作成し、どこかにマウントし格納先を調節しながらソースパッケージを展開・構築していけば、新しい OS が出来上がるでしょうか。

具体的にはまず主要なディレクトリを作成：

```
mkdir /mnt/new-os
mount /dev/sda2 /mnt/new-os

mkdir /mnt/new-os/etc
mkdir /mnt/new-os/tmp
mkdir /mnt/new-os/sys
```

大事な設定ファイルもコピー：

```
cp /etc/fstab /mnt/new-os/etc/fstab  # 内容は必要に応じて後で修正
cp /etc/hosts /mnt/new-os/etc/hosts
…
```

各ソースパッケージについて次のようにして行きます：

```
./configure --prefix=/mnt/new-os
make
make install
```

イメージとしてはだいたい合ってはいますが、このままでは上手く動作しません。

第一にデフォルトでは./configure は元のシステムのライブラリや実行ファイルを検索して、そこで見つかったものと連携して上手く作動するよう諸設定をします。ライブラリなら/lib /usr/lib /usr/local/lib を検索します。これらはアクセスできなくなる旧システムのもので、新システムとは違いがあります。微妙な違いでも動作に大きく影響するものです。

第二に一部のパッケージはバイナリファイルの中に外部プログラム等の絶対パスを書き込みます。例えば perl を使うパッケージがあったとします。「perl は/mnt/new-os/usr/bin/perl を利用せよ」と何らかの方法で指示できたとします。そうすると作業用の/mnt/new-os/の部分までが新しいファイルの中に書き込まれてしまうのです。

第三に一部のパッケージは一部の機能をソースからコンパイルするのではなく、システムの中に利用可能なバイナリ形式があるとそれをコピーしてしまいます。LFS が目的とする、全てソースからの構築はこれでは実現しません。

以上の問題を回避するため LFS では以下のような２段階構築をします。

まず新しい領域をマウントし、その中に/tools というディレクトリを作成します。

ソフトリンクによって、古い OS、新しい OS どちらからも/ （ルート）直属の/tools として参照されるようにします。

```
mkdir  /mnt/lfs
mount /dev/sda2 /mnt/lfs
mkdir  /mnt/lfs/tools
ln -s   /mnt/lfs/tools  /
```

この設定ができたら/tools に作業用の仮システムを作って行きます。約 30 個のパッケージを順次構築します。尚、この段階ではカーネルは構築しません。

```
./configure --prefix=/tools
make
make install
```

いくつかのパッケージは./configure の前に調整します。たいがい sed で修正します。

これが完成したら chroot 命令で新しいパーティションの中に入り、シェルを新規起動します。目的とする環境の中で構築作業を行う事になります。chroot は特殊な状況でしか使わない命令ですが、カーネルはそのままで、その他は新しい OS として動くことだと覚えておいて下さい。

ところで X-Window システムでいくつかのアプリケーションを動かしている中で、一つの xterm コンソールで chroot を行なったらどうなるでしょうか？chroot を行ったコンソールだけが他と違う特殊な環境で動作します。通常利用しているブラウザで HTML 形式の教書を見ながら、chroot 環境のコンソールにコマンドを打ち込んで作業をすることが可能です。

chroot 環境の中で約 70 のパッケージを構築していきます。

```
./configure --prefix=/usr
make
make check       #   省略可
make install
```

やはりいくつかのパッケージは./configure の前に調整します。小規模な修正なら sed で行いますが、一部はパッチによります。

プログラムによっては/usr/bin より/bin、ライブラリによっては同様に/usr/lib より/lib に置くべきものがあります。これらは mv コマンドで移動します。

make check は必須ではありませんが、構築を確認する手段として行う人が多いようです。ただし構築途上のシステムの制約から報告されるエラーもあります。

いくつかのパッケージはドキュメントは make install でインストールされず、別途操作が必要です。

本システムが完成したら/tools は消去します。

この作業は chroot 環境の中で行うことが大事です。教書は途中でシェルからログアウトしない前提で書かれています。一旦作業を中断して再起動する場合は再度正しい手順で chroot を必要があります。

ブートローダの grub はコンピュータ起動時に最初に作動するプログラムで BIOS が正しく検出するよう特別なインストール手順を必要とします。

カーネル構築

Linux カーネルは最後に構築します。カーネル構築は調査すべき項目も多く、時間がかかるものですが LFS の指示は淡白で一頁にまとまっています。LFS 教書は他のパッケージについは手取り足取り指導していますがカーネルは全く違った方針です。カーネルはドキュメントが充実しているのでそれでも問題無いと言えますが、そのドキュメントの多くは日本語に訳されていないので英語が不得意な方は苦労しています。

実際に構築して気づくのは、システムの他の構成要素との干渉が少ない点です。LFS の事情からカーネルに入れなければならない必須機能はありますが、ごく少数です。

構築の方法は他のパッケージと少し違います。

```
make mrproper
make defconfig
make menuconfig
make
make modules_install  # カーネルモジュールの有る構成のみ
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-v.vv.vv-lfs-x.x
cp -iv .config /boot/config-v.vv.vv
```

make mrproper

カーネルのソースの清掃を行うものです。カーネル構築の最初に唱える浄化の呪文と思っておけば良いでしょう。

make defconfig

諸設定をデフォルトの状態にします。ソースを管理している人にとってのデフォルトのようですが、ハードディスクなど主要な機器については現在主流の物に適合した設定になっていると期待できます。設定は.config というテキストファイルに書かれ、次の段階で修正します。.config を生成するには make oldconfig など他の方法もおあります。

make menuconfig

他のパッケージの./configure と違って対話形式の設定です。ncurses という簡易 GUI を使っています。この設定は調査をしながら進めるので時間がかかります。シンボルの依存関係は精査されています。項目の設定状況によっては別の関連項目が現れたり消えたりします。.config は諸設定の記録となりますので必ず保存します。一回構築に成功したらその.config ファイルをベースに必要な修正を加えて自作カーネルを改良して行きます。

お使いのシステムの設定ファイルを持ってきてそのまま使う方法もありますが、これは勉強にならないので推奨できません。Debian、Ubuntu などは initramfs を使って 2 段階で起動しています。この initramfs を除去した構成に作り直すのには手間がかかると思います。

make defconfig で作成された.config ファイルを make menuconfig にかけて確認と設定を行う場合、表示される全ての設定項目に目を通しておくことが理想ですが、膨大な英文ドキュメントを読むことになり、時間がかかります。全部目を通さない場合は主要なハードウエア、システム要件についてはメニューから探し出すようにすると良いでしょう。

CPU はインテル、AMD の x86-32, x86-64 系列なら細かな型式選定が可能です。商品名 Pentium は古い用法と新しい用法がありますが、ここでは 1990 年代の古い方です。新しい Pentium、Core は指定すべき型式の特定が難しいと思います。CPU の機種型番検出には lscpu 命令を使います。

バスは PCI と ISA が選択できます。PCMCIA カードを使えるようにする設定はこの近くにあります。

ハードディスクコントローラは SCSI、IDE-PATA、SATA から選択できます。最近は SATA がデフォルトの libata と呼ばれるデバイスドライバが主に使われています。旧式の IDE-PATA 用のインターフェイスも用意されていますが普通は使いません。新しい libata は IDE も対応可能です。一般に CD/DVD はハードディスクと同じコントローラで libata が設定されていれば動くはずです。ハードディスクコントローラの設定が正しくないとディスクが検出されず起動しません。大多数のパソコンはデフォルトの設定で動くはずですがここはしっかり確認したい箇所です。

ハードディスクコントローラ等の重要なデバイスドライバーは汎用の物と特定チップセット用の専用の物とが用意されています。専用の物を使えれば効率がよくなります。ハードウエアの検出、製造元や型番特定には dmesg と lspci の両コマンドが有用です。

キーボード、マウス、USB の設定を確認します。USB 記憶装置は設定項目が多岐にわたりますが外付けディスクなどの利用に必要になります。最近のラップトップパソコンには SD カードを挿入するスロットがありますが、これはバスに直結されている独自の機器で MTD(Memory Technology Device) と呼ばれます。ラズベリーパイは SD が主ディスクですので MTD ドライバは必須です。

ハードウエアドライバは秋葉原でも入手できないような専門的な装置のためのものが多数用意されています。ネットワークプロトコルも特殊な装置でしか利用しないものがかなりあります。これらをはずせばカーネルが軽量化できます。最初自信が無ければこれらは触るのは最小限にしておき、稼働するカーネルができてから合理化すると良いでしょう。

「よくわからないけど便利かもしれない」とオプションの機能を追加すべきではありません。これらには開発者がデバッグに使うものがあります。デバッグ情報保存先として新たな装置を定義して主ディスク装置の名前を例えば/dev/sda から/dev/sdb に変えてしまい、起動不良になる場合もあります。

ファイルシステムも選択設定します。ext2 系列の ext4 が現在主流です。CD-ROM や DVD などの光磁気ディスクをマウントするには ISO-9660、USB メモリや SD カードを（フォーマットを改めることなく）マウントするには vfat が必要です。読み込み専用ファイルシステム romfs、圧縮機能を追加した cramfs と squashfs は組み込み装置の開発をされる方は意識すべきです。その他に初めて名前を聞くファイルシステムがたくさん出てきますが、多くは他の OS の物で不要です。Network File System(NFS)は昔 SUN でよく使用されていましたが単体パソコンではまず不要です。半導体主記憶をディスクのように使う ramfs は積極的な設定は不要で、ブートパラメータの指定だけで利用できますが、メモリを浪費するので推奨できません。この種の機能をお求めなら tmpfs をお勧めします。

```
make
```

コンパイルは他のパッケージと変わりません。ただ時間はかかります。設定状況によりコンパイルが通らないのはほとんど経験したことがありません。

```
make modules_install
```

カーネルモジュールを利用するならインストールします。最初はモジュール無しの構成が無難です。

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-v.vv.vv-lfs-x.x
```

```
cp -iv .config /boot/config-v.vv.vv
```

カーネルのインストールは make install では行わず、直接コピーして行います。.config はカーネ

ルの近くに保存しておきます。

構築ツールの核心部分：コンパイラ

ソースパッケージ構築はコンパイラ無しでは不可能です。「コンパイル」という語には二義あり、狭義ではソースコードのオブジェクトコードへの変換、広義では実行形式の生成をいいます。gcc は、実行形式まで作成してくれますが、自身で行っているのは狭義のコンパイルで、その先は外部のアセンブラとリンカを起動しています。プログラムでは printf()や malloc()などの標準関数が使われていますので標準ライブラリも用意する必要があります。構築途上の LFS のように、標準ライブラリが複数ある場合どちらをデフォルトとするか決める必要があります。

コンパイラは gcc、標準ライブラリは glibc、アセンブラとリンカは binutils(Binary Utilities)の一部です。これらは GNU プロジェクト由来のパッケージです。

LFS の中で特に製作者が工夫しているのは相互に依存するこの三者を構築する方法の作り込みです。./configure 段階で細かな設定が目につきます。昔から離れ技みたいなこともやっていました。

完成した gcc が希望のリンカ、ライブラリと連動してきちんと作動しているかどうか、検査手順が教書に記述されています。gcc を常用している人が見ても謎めいた手順ですが、至って単純です。必ず教書の通りに検査して下さい。

LFS 構築成功の秘訣

LFS 構築に際して注意すべき点

第一は教書から逸脱しないことです。教書掲載のソフトウエアパッケージのより新しい版が発表されている場合がありますが、こういう時は教書通りの版を使います。特に gcc、glibc、binutils は相性問題が起きやすいとされています。

複数のパッケージでも構築を並行して行うのは避けるべきです。

第二にプロジェクト進捗管理を導入することです。教書は管理ができる人を前提としていて、何も教えてくれません。進捗管理の基本は後で述べますが、実際にどうするかは自分の力量や目的にあわせて考案する必要があります。

この進捗管理は上記の「教書からの逸脱は避ける」と矛盾する一面があります。インストール後の確認は教書に書かれておらず、これを行うのは厳密に言えば教書からの逸脱となります。コマンドにはシステムに変更を加えるものとそうでないものとがあります。システムへの影響の無い操作を選ぶ必要があります。

構築途中の LFS とくに chroot 環境では普通なら当然使える道具が未整備の場合があります。思わぬことも起きると覚悟しておくべきです。この想定外の事態はいい勉強になります。制約のあるシステムなので構築途上は凝ったシェル芸は避けるべきです。

第三に危険な操作を把握し、慣れない操作は練習をしておくことです。事故は慣れない操作の最中に起きます。また何か様子がおかしかったら、一旦立ち止まってよく確認する事です。

第四に未知の事柄に遭遇したら調査を行うことです。LFS は自分の知識が不足しているのがどこか明らかにする効果があります。オープンソースカンファレンス（OSC）や勉強会に参加するなどして、わからない事について相談できる人を求めると良いでしょう。

プロジェクト進捗管理の基本

一般に品質管理は Plan 計画 ・Do 実行・ See 検証 ・Adjust 調整 のサイクルで行うのが良いとされます。略して PDSA と言います。

詳細な教書がありますから計画とはこれから何をするのか把握し、その結果何が起きるのかを予想することです。実行は記述されたコマンドを打ち込み実行するだけです。検証は教書には記述がありません。プログラムが正しくインストールされたか、ということになりますが、この部分は自分で考えます。調整とは、検証して問題が見つかった場合、対策を施すことですが、たいがいは手戻りして構築し直すことになります。

手戻りを予見して節目の所でバックアップを取って置きます。どこでバックアップを行うべきかは教書に記述があります。

システムをよく理解している人は手戻りが小くなります。理解できていないと手戻りは大きくなり、最悪、最初から全部やり直しとなります。

上記の PDSA 管理サイクルは戦後の昭和期に盛んに唱えられたものです。終戦直後、アメリカ人の統計学者 W.E.デミング博士が日本に伝えたものです。デミング博士は来日前、ベル研究所で品質管理の研究をした経歴の持ち主です。ベル研究所では UNIX が生まれました。今でも UNIX 系 OS が根強い人気を保っているのはベル研究所の品質管理に対する思い入れが大きな要因だと思います。UNIX 自体、OS とコンピュータ言語の研究目的の他は通信網デジタル化に際して、品質管理のための情報収集と分析が主な用途だったと思います。

事前練習による技能習得

LFS 構築にはいくつかの技能が必要です。自信が無い部分は事前調査と練習をしておくべきです。
LFS 構築は自己の力量を越えていると判断される方も一つでも技能を身につけるべく以下をご検討下さい。

LFS では避けて通れないソースパッケージからの構築、インストールを何回か練習しておくと良いでしょう。初めは思い入れのあるプログラム、小さめのものを選び、徐々に大きなパッケージに挑戦して行けばいいでしょう。

ソースパッケージをダウンロードした際、伝送ミスが起きなかった確認するために md5sum などのチェックサムで確認します。md5sum -c で一括してチェックする方法を知っておくべきです。

Linux カーネルの自前での構築もやってみると良いでしょう。カーネルは各ディストリビューションで構築の流儀があって方法を手引き書が用意されているはずです。

カーネルのブートパラメータを調査するのも意義があります。ただ数が膨大なのでテーマを絞る必要があります。カーネルを自前で構築する場合、ブートメッセージの表示間隔を広げる（遅くする）方法は知っておくべきです。

パソコン一台に複数の OS をインストールし、デュアルブートの設定も良い練習です。2 つ目以降はなるべく手動でできるところは手動で設定し、正常に作動するようになってからも様々な設定を変えてみると勉強になります。起動設定が記述される/boot/grub/grub.cfg とパーティションの設定が記述される/etc/fstab はよく理解する必要があります。起動不良は多く、この 2 つの記述に原因があります。

バックアップも大事な技能です。バックアップはファイルシステム全体のコピーという見方もできます。システム領域とデータ領域ではデータ領域のバックアップの方が簡単です。システム領域をコ

ピーする時は対象システムの停止中とし、マウントは読み込み専用にします。dd コマンドでファイルシステムの入ったパーティションごと複製することも可能です。マウントはしません。dd や再帰的コピーでは書き込み先を少し間違えると既存のファイルを上書きしてしまいます。慎重に確認をしながら作業を進める必要がありますが、これはただ「注意する」だけでは不十分だと感じます。ミスを防止し、失敗した際の被害を軽減する作業手順を編み出し、身につけるべきです。

ext4 ファイルシステムの詳細設定は調査しておく必要があります。フォーマット時の設定次第ではマウント時にオプションを指定する必要が生じます。

romfs など読み込み専用ファイルシステムは主に組み込み装置で使われます。アーカイブとしても利用できます。機能が増えた tar や zip より単純で理解しやすいもので、展開と使用後のファイル削除が不要というメリットもあります。試用してみて下さい。

パッケージによっては make の段階に多大な時間を要するものがあります。ラズベリーパイなど小型で高密度のハードウエアでは構築作業中の発熱暴走が起きることがあります。冷却装置を付加するのも一つの予防策ですが、CPU の動作周波数やディスクからのデータ転送レートを抑制してコンピュータを意図的に遅くする方法を知っておくと良いでしょう。

LFS は自己完結したシステムです。どんな知識や技能を身につけるべきか、目標を定めるために LFS 教書を一読してみるのも良いででしょう。

参考リンク

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/

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。

この文書は２０１９年８月２４日の東京都大田区でのオープンデベロパーズカンファレンス（ＯＤＣ）における講演を元に加筆したものです。

2019 年９月　漆畑晶

著者紹介

漆畑 晶

プロフィール

- 画像処理ソフトウェアユーティリティ群 Netpbm の主要開発者
- FSF 創設者のリチャード・ストールマン博士と親交あり
- 中国の大学（上海復旦大学、南京大学、天津大学、中国科学院大学等）で講演多数

信州大学経済学部在学中に「国富論」やアメリカ産業史の本に加えて、「論語」「老子」などの古典を読んで引用する力を磨く一方で、R 言語による多変量回帰分析プログラムや、駅すぱーとの核心部である最短距離経路探索のプログラムを自作していました。

オープンソース・ソフトウェアについて

なにか良いものをタダで貰ったり使わせてもらったときは、必ずお礼を言うものです。お礼を欠いていると、おかしなことになってしまいます。誰でも自由に利用できるオープンソース・ソフトウェアも同じです。感謝を欠かしてはならないのです。

Linux From Scratch

著　者：漆畑 晶

発行者：（特非）オープンソースソフトウェア協会
http://www.ossaj.org/
info@ossaj.org

発行日：2024年2月1日