Free Software, Free Society:
Selected Essays of Richard M. Stallman

Introduction by Lawrence Lessig

Edited by Joshua Gay

GNU Press
www.gnupress.org
Free Software Foundation
Boston, MA USA

First printing, first edition.

ISBN 1-882114-98-1
Published by the Free Software Foundation
59 Temple Place
Boston, MA Tel: 1-617-542-5942
Fax: 1-617-542-2652
Email: gnu@gnu.org
Web: www.gnu.org

GNU Press is an imprint of the FSF.
Email: press@gnu.org
Web: www.gnupress.org
Please contact the GNU Press for information regarding bulk purchases for classroom or user group use, reselling, or any other questions or comments.

Original artwork by Etienne Suvasa. Cover design by Jonathan Richard.

目次

編集にあたって

20世紀末は、オーウェルの悪夢のような日々だった。ソフトウェアの科学的な研究の発表を妨害する法律、ソフトウェアの共有を妨害する法律、開発を阻害するソフトウェア特許の氾濫、ユーザーから所持、秘密保護、共有、ソフトウェアの動作原理の理解などのあらゆる自由を奪うエンドユーザーライセンス契約。リチャード・M・ストールマンによるこの評論、講演集は、これらの問題の多くを取り上げる。何よりもまず、ストールマンは、フリーソフトウェア運動の哲学を論じる。この運動は、ソフトウェアの自由の理想を普及させるために、連邦法による抑圧や、悪意に満ちたエンドユーザーライセンス契約と戦う。

フリーソフトウェアは、GNUソフトウェアとGNU/Linuxオペレーティングシステムの開発に携わる数十万人のプログラマの力でInternetをコントロールするサーバという地点を死守してきた。そして、デスクトップコンピュータ市場への浸透が進むとともに、Microsoftを始めとする私有ソフトウェア企業にとっては脅威になりつつある。

本書の評論は、広い範囲の読者を対象として書かれている。本書で展開されている哲学や思想を理解するために、コンピュータ科学の基礎知識は不要である。ただし、専門家ではない読者のために、「ソフトウェアについてのコメント」では、コンピュータ科学でよく使われる専門用語や概念を解説した。また、適宜脚注が加えられている。

評論の多くは、最初に発表された形態から書き換えられ、訂正されている。1つ1つの評論には、本文に一切の変更を加えない複製の再頒布を許可する文が付加されている。

これらの評論は、18年を越える長期にわたってそれぞれ独立に書かれたものなので、並べられている順番には特別な意味はなく、このように読まなければならないという順番もない。第1部「GNUプロジェクトとフリーソフトウェア」は、

フリーソフトウェアとGNUプロジェクトの歴史、哲学に親しんでいただくことを意図してまとめてある。また、この部分は、プログラマ、教育者、ビジネスピープルが、それぞれの集団、業務、生活にプラグマティックにフリーソフトウェアを組み込むためのロードマップになるはずである。第2部「コピーライト、コピーレフト、特許」は、著作権、特許制度の哲学的、政治的基礎と、過去数百年の間にそれらがどのように変化してきたかを論じる。また、特許と著作権に関連する現行法と各種規制が、ソフトウェア、音楽、映画などの媒体の消費者やエンドユーザーの利益を最大限に守るものにはなっていないことも指摘する。むしろ、これらの法律は、企業や政府が個人の自由を破壊することを助けるように作られていることを論じる。第3部「自由、社会、ソフトウェア」は、引き続き自由と権利の議論を展開し、私有ソフトウェア、著作権法、グローバリズム、「トラステッド(信託)コンピューティング」などの社会的に有害な規則、規制、政策が、自由と権利をいかに阻害しているかを論じる。産業界や政府は、特定の権利や自由を諦めるように人々を誘導するための手段の1つとして、情報、アイディア、ソフトウェアの共有が悪であるかのようなニュアンスを持つ用語を使っている。そこで、この部分には、混乱を起こしやすく、避けたほうがよい単語を説明する評論を加えた。第4部の「ライセンス」には、GNUプロジェクトの基礎であるGNU一般公開使用許諾書、GNU準一般公開使用許諾書[*1]、GNU自由公開文書使用許諾書を収録した。

自分用、授業用、あるいは頒布用に本書を購入したい場合には、sales@fsf.orgまたはhttp://order.fsf.org/のフリーソフトウェア財団(FSF)に申し込んでいただきたい。また、ソフトウェアの自由のためにさらに助力したいと考えておられる場合には、http://donate.fsf.org/かdonations@fsf.org(趣旨を詳しく書きたい場合)を通じてFSFに寄付を提供することをご検討いただきたい。FSFには、+1-617-542-5942という電話番号でも連絡を取ることができる。

GNUプロジェクトへの貢献に感謝を捧げるべき人々は、おそらく数千人を超えるだろう。しかし、1枚のリストにそれらの人々の名前をすべてまとめることは

*1 【訳注】本書に掲載のGPLの翻訳は引地信之さん・美恵子さん、LGPLの翻訳は八田真行さんが手がけられたもので、訳者は一切手を触れていない。そのため、各タイトルも訳者が関わった部分とは異なる訳語(それぞれ「GNU一般公有使用許諾書」「GNU劣等一般公衆利用許諾契約書」)になっている。

不可能である。そのため、私の謝意は、それら無名のすべてのハッカーたち、フリーソフトウェアの開発、推進、世界中への普及を助けたすべての人々に及ぶものと考えていただきたい。

本書が世に出たことについて、以下の方々に感謝の意を捧げたい。

ジュリー・サスマン、P.P.A. は、発展過程のさまざまな段階で複数のバージョンを編集し、「本書について」を執筆し、句読点の付け方から章の順番に至るまで、さまざまなアイディアを提供してくれた。

リサ（オーパス）ゴールドスタインとブラッドリー・M・クーンは、構成、校正など、本書を実現するために必要なこと全般を助けてくれた。

クレア・H・アビタビル、リチャード・バックマン、トム・シュネル、そして（特に）ステファン・コンパルは、本書全体を綿密に校正してくれた。

カール・ベリー、ボブ・チャッセル、マイケル・マウントニー、M・ラマクリシュナンは、本書を TEXinfo（http://www.texinfo.org/）で整形、編集する上で、それぞれの熟練した技術を提供してくれた。

マッツ・ベンソンは、Lilypond（http://www.gnu.org/software/lilypond/）でフリーソフトウェアの歌を整形するのを助けてくれた。

エチエンヌ・スヴァサは、以前からアートの分野で FSF に力を貸してくれているが、本書でも、各部冒頭の挿画を描いてくれた。

メラニー・フラナガンとジェイソン・ボランは、一般の読者について貴重なヒントを提供してくれた。また、自動車の変速機について詳しく教えてくれた Paul's Transmission Repair のボブ・トッキオには、特に感謝している。

さらに、人はよって立つ理想のために生きるべきだということを教えるとともに、2 人の兄弟、3 人の姉妹を生むことによって共有の重要性を初めて学ぶ機会を与えてくれた父母、ウェインおよびジョアン・ゲイに謝意を捧げたい。

最後になってしまったが、GNU の哲学、すばらしいソフトウェア、優れた論文を世界中で共有できるようにしてくれたもっとも重要な人、すなわちリチャード・M・ストールマンその人に感謝の意を捧げたい。

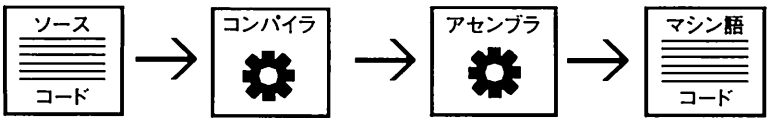
ジョシュア・ゲイ
josh@gnu.org

ソフトウェアについてのコメント

本節は、コンピュータ科学の技術的側面についてほとんど、あるいはまったく知識を持たない読者のために用意されたものである。本書の評論や講演を理解するために本節を読まなければならないということはないが、プログラミングやコンピュータ科学についてまわる専門用語をよく知らない読者にとっては役に立つかもしれない。

コンピュータプログラマは、ソフトウェア、すなわちコンピュータプログラムを書く。プログラムは、特定の仕事をこなすために何をすべきかをコンピュータに指示するコマンドで書かれたレシピのようなものである。おそらく、読者はさまざまなプログラムをご存知だろう。Web ブラウザ、ワープロ、電子メールクライアントなどといったものがそれである。

通常、プログラムはまず、ソースコードという形を取る。この比較的高い水準のコマンドは、C とか Java といったプログラミング言語で書かれる。ソースコードが書かれると、コンパイラと呼ばれるツールを使って、これをアセンブリ言語というそれよりも低い水準の言語に翻訳する。次に、アセンブラと呼ばれる別のツールを使って、アセンブリ言語のコードをマシン語という最終段階（最低水準）のコンピュータがネーティブに（母語として）理解する言語に変換する。



たとえば、C を学ぶ人が最初に書くプログラムで、（コンパイル、実行すると、）

画面に「Hello World!」と出力する[*1]「Hello World!」プログラムについて考えてみよう。

```
int main(){
    printf("Hello World!");
    return 0;
}
```

Java 言語では、同じプログラムを次のように書く。

```
public class hello {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

しかし、マシン語では、次のようになる（これは全体のごく一部である）。

```
11000111101110101001010010010010101011110
01101010100110000011110010110101011111101
01001111111111100101101100000000010100100
01001000011001010110110001101100011011111
00100000010101110110111101110010011011100
01100100001000010100001001101111011011111
```

上記のマシン語の形式は、バイナリ（2 進）というもっとも基本的な表現である。コンピュータのすべてのデータは、0 または 1 の値の連続から構成されているが、人間がそのようなデータを理解するのは非常に難しい。バイナリに簡単な変更を加えるためには、特定のコンピュータがマシン語をどのように解釈するか

---

*1 Scheme などの他のプログラミング言語では、Hello World!プログラムは、通常、最初のプログラムではない。Scheme で最初に書くプログラムは、次のようなものになることが多い。

```
(define (factorial n)
    (if (= n 0)
        1
        (* n (factorial (- n 1)))))
```

このプログラムは、与えられた数値の階乗を計算する。たとえば、(factorial 5) を実行すると、5*4*3*2*1 を計算して 120 と出力する。

についての詳細な知識を持っていなければならない。上記の例のような小さなプログラムであれば、そのようなことも可能かもしれないが、およそ面白い意味を持つようなプログラムでは、わずかな変更を加えるだけでもとてつもない労力が必要になる。

たとえば、C言語で書かれた先ほどの「Hello World!」プログラムに変更を加え、英語で「Hello World!」と出力する代わりにフランス語を使う場合について考えてみよう。この変更は簡単であり、新しいプログラムは、次のようになる。

```
int main() {
    printf("Bonjour, monde!");
    return 0;
}
```

Java言語で書かれたプログラムの直し方も同じようなものだと簡単に類推できるだろう。しかし、バイナリ表現を書き換えようとすると、多くのプログラマでさえ、どこから手を付けたらよいのかわからなくなってしまう。私たちが「ソースコード」というとき、それはコンピュータしか理解できないマシン語のことではない。CとかJavaといったより高い水準の言語のことを指している。人気の高いプログラミング言語としては、そのほかにC++、Perl、Pythonなどが挙げられる。理解しやすいものとしにくいもの、プログラムが書きやすいものとそれほどでもないものはあるが、プログラムがコンパイル、アセンブリされたあとの複雑なマシン語と比べれば、どれもはるかに操作しやすい。

理解すべきもう1つの重要概念は、**オペレーティングシステム**（OS）とは何かということである。オペレーティングシステムとは、入力と出力、メモリの割り当て、タスクのスケジューリングを処理するソフトウェアである。一般に、GUI（グラフィカルユーザーインターフェイス）などのよく使われるプログラム、便利なプログラムは、オペレーティングシステムの一部とみなされている。GNU/Linuxオペレーティングシステムには、GNUソフトウェアと非GNUソフトウェアの両方が含まれており、Linuxと呼ばれるカーネルが含まれている。カーネルは、入出力やタスクのスケジューリングなど、アプリケーションが必要とする低水準の仕事を処理する。GNUソフトウェアは、GCC（さまざまな言語に対応している汎用コンパイラ）、GNU Emacs（非常に大量の機能を持つ拡張性の高いテキスト

エディタ)、GNOME (GNU デスクトップ)、GNU libc (カーネル以外のすべてのプログラムがカーネルと通信するために使わなければならないライブラリ)、Bash (コマンドラインを読む GNU コマンドインタープリタ) などのオペレーティングシステムのカーネル以外の多くの部分から構成されている。これらのプログラムの多くは、GNU プロジェクトの初期の段階でリチャード・ストールマンの主導で開発されたもので、現代のあらゆる GNU/Linux オペレーティングシステムに付属している。

重要なのは、たとえ自分では与えられたプログラムのソースコードを書き換えたり、これらのツールを直接使いこなしたりすることができなくても、それができる人を見つけるのは比較的簡単だということである。そのため、プログラムのソースコードを持っていれば、通常、それを変更、訂正、カスタマイズしたり、プログラムについて学習したりする力を手にすることができる。これは、ソースコードが与えられなければ掴めない力である。ソースコードは、ソフトウェアをフリーにするために必要なものの 1 つである。他の要件は、本書を読み進めるうちに、フリーの哲学、発想法とともに明らかになってくるだろう。楽しみにしていただきたい。

リチャード・E・バックマン
ジョシュア・ゲイ

本書について

本書の評論や演説は、異なる時期に異なる層の人々に宛ててなされたものなので、重なり合う部分がかなり含まれており、いくつかのテーマは複数の箇所で取り上げられている。しかも、索引を作ることができなかったので、章のタイトルから書かれている位置がわかる場合を除けば、以前に読んだテーマにあとから戻ろうと思っても、簡単にはいかないだろう。

そこで、大雑把で不完全ながらも（それぞれの項目に関連したすべてのトピック、議論を網羅しているわけではない）、このガイドを作った。読者が読みたいと思っている思想や説明についてヒントが見つかれば幸いである。

ジュリー・サスマン、P.P.A.

概要

1 章は、本書で取り上げられているソフトウェア関連のすべてのトピックについて概要を示している。20 章も同様である。

ソフトウェアとは無関係なトピックについては、後述の「プライバシーと個人の自由」、「知的財産権」、「著作権（コピーライト）」の項を参照していただきたい。

GNU プロジェクト

GNU プロジェクトの歴史については、1 章と 20 章を参照していただきたい。

20 章には、GNU という再帰的な頭字語（GNU's Not Unix、グニューと発音する）の起源と発音についての楽しい説明がある。

後述の「Linux、GNU/Linux」の項も参照していただきたい。

フリーソフトウェア財団（FSF）

FSFについては、1章、20章と、18章の「フリーソフトウェアへの資金提供」を参照していただきたい。

フリーソフトウェア

11、12、13、16、17、19章を除くすべての章がフリーソフトウェアを扱っているので、本書でフリーソフトウェアについて議論しているすべての箇所を示そうとは思わない。

フリーソフトウェアの歴史（フリーソフトウェアから私有ソフトウェアへ、そして再びフリーソフトウェアへ）については、1章を参照していただきたい。

フリーソフトウェアの定義については、3章を参照していただきたい。同じ定義は、他のいくつかの章でも繰り返されている。

「フリー」という単語の曖昧さ、「フリー（無料）ビール」という意味ではなく、「フリースピーチ（言論の自由）」という意味でこの単語を使っている理由については、1章と6章の「曖昧さ」を参照していただきたい。

後述の「ソースコード、ソース」、「オープンソース」、「コピーレフト」の項も参照していただきたい。

21章では、フリーソフトウェアという単語を21か国語[1]に訳してある。

ソースコード、ソース

ソースコードについては、フリーソフトウェアの議論全体で言及されている。ソースコードというのが何のことかわからない読者は、「ソフトウェアについてのコメント」を参照していただきたい。

Linux、GNU/Linux

Linuxの起源、およびLinux（オペレーティングシステムカーネル）とGNU/Linux（オペレーティングシステム全体）の違いについては、1章の「LinuxとGNU/Linux」という短い説明と、20章の長い説明を参照していただきたい。

---

[1] 【訳注】http://www.gnu.org/philosophy/words-to-avoid.html より3つ追加して24か国語分ある。

オペレーティングシステムを指すときに、Linux と省略せず、GNU/Linux と呼ぶ理由については、5 章と 20 章を参照していただきたい。

プライバシーと個人の自由

私たちが長い間存在して当然と考えてきた個人の自由、プライバシー、書かれたものに対するアクセスが失われることについての警告に関しては、11、13、17 章を参照していただきたい。これらの議論は、すべての読者を対象としている。

オープンソース

オープンソース運動とフリーソフトウェア運動の違いについては、6 章を参照していただきたい。このことについては、1 章（「オープンソース」）と 20 章でも触れている。

知的財産権

「知的財産権」という用語がいわゆる「知的財産権」問題を考える上で誤解の原因にも障害にもなっている理由については、21 章で説明してある。16 章の冒頭も参照していただきたい。

特定のタイプの「知的財産権」については、後述の「著作権（コピーライト）」と「特許」の項を参照していただきたい。

著作権（コピーライト）

注意：著作権についての言及の大半は、ソフトウェアについてのものではない。

著作権の歴史、目的、実施、効果、および著作権に対する妥当な方針については、12、19 章を参照していただきたい。私たちが属するデジタル時代において非常に重要な意味を持つ、e-book、DMCA（Digital Millennium Copyright Act：デジタルミレニアム著作権法）などのトピックもここで扱っている。

特許と著作権の違いについては、16 章を参照していただきたい。

フリーソフトウェアとフリードキュメンテーションの推進のために著作権を活用することについては、次の「コピーレフト」の項を参照していただきたい。

コピーレフト

コピーレフトの説明とそれが著作権制度を使ってフリーソフトウェアを推進している仕組みについては、1章（「コピーレフトとGNU GPL」）、14章、20章を参照するとともに、次の「ライセンス」の項も参照していただきたい。

15章では、コピーレフトが実際的で効果的であるとともに理想的であるという議論をしている。

9章は、フリーソフトウェアに付属するフリーマニュアルを論じている。

ライセンス

ソフトウェアとマニュアルのコピーレフトを推進するために活用できるGNUライセンスについては14章で紹介されている。また、第4部は、すべてライセンスに充てられている。

特許

特許と著作権の違い、およびソフトウェアの特許に反対する議論、ソフトウェアが他の特許対象の発明と異なる理由については16章を参照していただきたい。この章では、米国以外の国々でのソフトウェア特許政策についても論じてある。

ハッカー vs. クラッカー

これらの用語の正しい使い方については、1章の冒頭を参照していただきたい。

序文

時代には、必ずそれを代表する哲学者がいる。すなわち、時代の想像力を掴んだ作家や芸術家のことである。彼がそのような存在としてすぐに認められることがないわけではないが、時代とのつながりがはっきりとわかるまでには時間がかかることが多い。しかし、認められようがそうでなかろうが、時代の刻印を押すのは、詩のささやきとして、あるいは政治運動の嵐という形で、自らの理想を語った者である。

私たちの時代にも、哲学者がいる。彼は芸術家ではなく、職業的な作家でもない。彼はプログラマである。リチャード・ストールマンは、MITの研究所で、オペレーティングソフトウェアを構築するプログラマ、アーキテクトとして仕事を始めた。彼は、「コード」によって支配されつつある世界の中で、自由を求める運動を確立したプログラマ、アーキテクトとして、公的なキャリアを積んできた。

「コード」とは、コンピュータを動かすためのテクノロジーである。ソフトウェアとして書かれているか、ハードウェアの中に焼き付けられているかにかかわらず、コードはマシンの動作を指示する命令のコレクションで、最初は言葉によって書かれる。これらのマシン（コンピュータ）は、次第に私たちの生活を規定し、支配するようになってきている。コードは、電話をどのようにつなぐかとか、テレビに何を表示するかといったことを決める。ブロードバンドリンクを介してビデオをコンピュータに流し込めるようにするかどうかを決める。そして、コンピュータが、それぞれのメーカーに報告することを決める。マシンが私たちを動かし、コードがマシンを動かす。

私たちは、このようなコードをどのように制御すべきなのだろうか。どのような理解が必要なのか。コードが実現する力に見合うだけの自由としてはどのようなものが必要なのだろうか。それはどのような力なのか。

これらは、ストールマンが生涯をかけて取り組んできた問題だった。彼は、仕事や言葉を通じて、コードの「フリー」を保つことの重要性を私たちに示してきた。フリーと言っても、コードの作者にお金を払わないという意味ではない。コード

の作者が構築した支配力の透明性を万人に対して保証するとともに、誰もがその力を自分のものとし、自分のニーズに合わせて書き換えられるようにすることである。これが「フリーソフトウェア」である。「フリーソフトウェア」は、コードで組み立てられた世界に対する1つの答えである。

「フリー」。ストールマンは、自らの用語の曖昧さを嘆いている。しかし、嘆く必要はない。パズルは、人々に考えることを強いる。そして、この「フリー」という言葉は、パズルのこのような機能を非常によく果たしている。現代のアメリカ人の耳には、「フリーソフトウェア」は空想的で不可能なことのように聞こえるかもしれない。「フリー」なものなど、たかが昼食まで含めてもどこにもありはしない。世界を動かすもっとも重要なマシンのもっとも重要な言葉がどうやったら「フリー」になるのだろうか。健全なる社会がどうやったらそのような理想を追い求めることができるのだろうか。

「フリー」という単語が不協和音を発するのは、単語自体の問題ではなく、私たちの問題である。「フリー」には別の意味があり、「無料」はいくつもある意味の中の1つに過ぎない。ストールマンによれば、「フリー」という言葉のより根源的な意味は、「フリースピーチ（自由討論）」とか「free labor：自由労働（奴隷労働の反対語）」という言葉の中に含まれている。価格がないという意味ではなく、他者による支配が制限されているという意味でのフリーである。フリーソフトウェアとは、透明性が保証されており、変更に対して開かれている支配力である。free law、すなわち「自由社会：free society」の法律が、その規制内容を了解可能にするとともに、変更に対して開かれているのと同じである。ストールマンの「フリーソフトウェア運動」の目的は、できる限り多くのコードを「フリー」にすることにより、透明性を保証し、変更できるようにすることである。

これを実現するためのメカニズムは、GPLと呼ばれるライセンスを通じて構成される「コピーレフト」という恐ろしく巧妙な装置である。「フリーソフトウェア」は、著作権（コピーライト）法の力を使い、自らの公開性、変更可能性を確保するだけではなく、「フリーソフトウェア」を利用、援用する他のソフトウェア（および専門用語で「派生的な仕事」と呼ばれるもの）自体もフリーになることを保証する。フリーソフトウェアプログラムを利用し、改良し、その改良版を広くリリースするなら、その新バージョンも、オリジナルバージョンと同様にフリー

でなければならない。これは義務であり、違反すれば著作権法に抵触することになる。

「フリーソフトウェア」は、自由な社会と同様に敵を持つ。Microsoft は、GPL を「危険な」ライセンスと喧伝し、GPL に対して戦いを仕掛けてきた。しかし、Microsoft が言うところの危険は、ほとんど幻想に過ぎない。変更版もフリーでなければならないという GPL の条件を「強圧的」と非難する者もいる。しかし、条件は強制ではない。数百万ドル（おそらく）を支払わなければ Microsoft Office の変更版の頒布をユーザーに認めようとしない Microsoft の姿勢が強圧的でないのなら、フリーソフトウェアの変更版もフリーでなければならないと主張することは強圧的ではないだろう。

そして、ストールマンのメッセージを過激だと言う人たちがいる。しかし、過激ということはないはずだ。実際、ストールマンの仕事は、コード以前の世界で私たちの伝統が築き上げてきた自由の概念の単純な翻訳である。「フリーソフトウェア」は、コードに支配された世界が、コード以前の世界を築き上げてきた伝統と同じくらい自由になることを保証するはずである。

たとえば、「自由社会」は、法律によって規制されている。しかし、自由社会が法律によって加えられる規制には、限界がある。法律を秘密にしている社会を自由社会と呼ぶことはできない。規制される人々から規制内容を隠すような体制は、未だかつて存在したことはない。法律が支配する。しかし、法律は、目に見える形でしか支配しない。そして、法律が目に見えると言えるのは、規制されている人々、あるいはその代理人（法律家や議会）が条文を知ることができ、コントロールできるときだけである。

法律のこの条件は、議会の仕事の範疇に留まるものではない。アメリカの法廷における法律の運用について考えてみよう。弁護士は、顧客の利益確保を助けるために顧客に雇われる。利益は、訴訟を通じて追求される場合がある。訴訟が提起されると、弁護士は摘要書を書く。摘要書は、裁判官が書く判決に影響を与える。判決は、特定の訴訟において勝利者を決めたり、特定の法律が憲法に違反していないかどうかを決める。

この過程におけるすべての文献は、ストールマンが言う意味でフリーである。摘要書は公開で、他者が自由に使うことができる。議論は透明であり（良いと言っ

ているわけではない)、理由書は最初に文書を書いた弁護士の許可を得ずに引用できる。弁護士が書いた理由書は、その後の摘要書で引用できる。他の摘要書、理由書は、それらを複製、統合することができる。アメリカ法にとっての「ソースコード」は、設計上、また原則上、公開で誰もが自由に利用できる。弁護士がしていることを真似よう。何しろ、摘要書の優劣は、過去に発生した事件をうまく再利用して創造的な内容になっているかどうかによって判断されるのだから。ソースはフリーであり、創造性と経済性はそれを基礎としている。

このフリーコードの経済（ここでは、法律の世界のフリーコードのことを意味する）は、弁護士の仕事を奪ったりはしない。弁護士事務所は、誰もが摘要書を自由に引用、複製できるにもかかわらず、優れた摘要書を書く意欲を保っている。弁護士は職人であり、その製品は公開されている。しかし、工芸は慈善事業ではない。弁護士には報酬が与えられる。社会は、無償でそのような仕事を求めようとはしない。それどころか、この経済は、古い仕事に新しい仕事が追加されることによって繁栄している。

これとは異なる法律運用を想像することもできるだろう。摘要書と理由書が秘密にされ、判決は結果を発表するだけで理由を述べない。法律は警察によって管理され、それ以外の人間には公開されない。規則を説明せずに規制が運用されている状態である。

このような社会を想像することはできるが、それを「フリー」だと言うことは考えられないだろう。そのような社会は、より活気に満ち、より効率的に力を配分できるかもしれないし、そうでないかもしれないが、そのような社会をフリーと呼ぶことはできない。自由な社会では、自由の理想は、効率的な運用よりも重視される。公開性と透明性は、その中で法律のシステムを構築するための制約であって、指導者にとって便利であれば付け加えられるようなオプションではない。ソフトウェアコードによって支配される社会は、それ以下であってはならないだろう。

コード開発は、訴訟ではない。訴訟よりも良いものであり、豊かであり、生産的である。しかし、法律は、製品の生産に対する完全な支配が創造性や意欲に影響を与えないことをはっきりと示す例である。ジャズ、小説、建築などと同様に、法律は、以前に行われた仕事を基礎として構築されている。創造性が宿るのは、

いつもこの追加や変更の部分である。そして、自由社会とは、このような意味でもっとも重要な資源の自由を保証する社会のことである。

本書は、リチャード・ストールマンの評論や講演を集め、その微妙な意味や説得力を明確にすることを意図した初めての試みである。評論は、著作権からフリーソフトウェア運動の歴史まで、広い範囲にまたがっている。その中には、デジタルの世界における著作権に懐疑の目を向けさせることになった環境の変化についての示唆に富む論述など、あまりよく知られていないテーマを扱ったものが多く含まれている。これらは、このとても強力な（他のことは差し置いても、発想、情熱、高潔さにおいて強力な）人物の思想を理解したい人々にとって貴重な資料になるだろう。また、彼の考え方を受け入れ、その上に新たな思想を構築しようとする人々に刺激を与えることだろう。

私は、ストールマン氏のことをよく知らない。好きになるのは大変そうな人物だということを知っている程度である。彼は衝動的で、しばしば短気になる。彼の怒りは、敵と同じように友にも向かう。彼は妥協を知らず、頑固である。この2つのことについては病気のようなものだ。

しかし、私たちの社会がコードの威力と危険性を理解するようになったら、また、コードは法律や政府と同じく透明でフリーなものでなければならないということがわかったら、私たちはこの妥協を知らない頑固なプログラマのことを振り返り、彼が現実化しようとして闘ってきたビジョン――自由と見識の前にコンパイラが跪く世界というビジョンを思い出すだろう。そして、次世代の社会が手にするはずの自由を獲得するために、その言動を通じて最大限の努力を示したのが、彼以外の誰でもないことを知るはずである。

私たちはまだその自由を手にしていない。それどころか、その自由を守ることに失敗するかもしれない。しかし、その成否にかかわらず、本書にはその自由がどのようなものになり得るかが描かれている。そして、これらの言葉と仕事を残した人物の中には、この自由の獲得のためにストールマンと同じように闘おうとするすべての人々を鼓舞してやまないひらめきがある。

ローレンス・レッシグ
スタンフォードロースクール法学教授

第1部
GNUプロジェクトと
フリーソフトウェア

第1章 GNUプロジェクト

最初のソフトウェア共有コミュニティ

1971年にMIT人工知能研究所（AIラボ）で働き始めたとき、私は長年にわたって存在してきたソフトウェア共有コミュニティの一員になった。ソフトウェアの共有は、別に私たちのコミュニティのみに限定されていたわけではない。レシピの共有が料理と同じだけの歴史を持つのと同じように、コンピュータと同じだけの歴史を持っていた。しかし、私たちの共有の進め方は、他のコミュニティ以上だったと言ってよいだろう。

AIラボは、スタッフのハッカーたちが設計し、Digital PDP-10（当時のメジャーなコンピュータの1つ）のアセンブリ言語で書いた、ITS（Incompatible Timesharing System：互換性のないTSS）と呼ばれるタイムシェアリングオペレーティングシステムを使っていた。このコミュニティの一員として、またAIラボスタッフのシステムハッカーとしての私の仕事は、このシステムを改良することだった。

私たちは自分たちのソフトウェアを「フリーソフトウェア」とは呼んでいなかったが、それはまだその用語が存在していなかったからで、このソフトウェアはまさにフリーソフトウェアだった。ほかの大学や企業の人たちがプログラムを移植、利用することを望めば、私たちは喜んでそれを認めた。誰かが見慣れない面白そうなプログラムを使っていたら、いつでもソースコードを見せてくれと頼むことができた。そうすれば、それを読み、書き換えることができるし、一部を引っこ抜いて新しいプログラムを書くことができる。

「ハッカー」という用語を「セキュリティ破壊者」の意味で使うのは、マスメディアの誤解である。私たちハッカーは、そのような意味付けを拒否し、「プログラムを書くことを愛し、そのための工夫を楽しむ人々」という意味でハッカーという単語を使い続ける[1]。

コミュニティの崩壊

1980年代初めにAIラボハッカーコミュニティが崩壊し、PDP-10コンピュータが製造中止になると、状況は一変した。

1981年に、SymbolicsというAIラボ出身者によって設立された企業がAIラボのほぼすべてのハッカーを雇い入れ、メンバーを失ったコミュニティは、自らを維持することができなくなった（Steven Levy, "*Hackers*", Dell, 1985：邦訳『ハッカーズ』工学社、1987年）は、AIラボコミュニティの最盛時の姿をくっきりと描くとともに、この間の経緯を明らかにしている）。AIラボが1982年に新しいPDP-10を購入したとき、管理者は、ITSではなく、Digitalのフリーではないタイムシェアリングシステムを新マシンに導入することを決めた。

それからまもなく、Digitalは、PDP-10シリーズの製造を中止した。PDP-10のアーキテクチャは、60年代にはエレガントで強力だったが、80年代に実用化されたより広いアドレス空間に合わせて自然に拡張することは不可能だった。つまり、ITSを構成するほぼすべてのプログラムが時代遅れになったということである。ITSにとっては、これが命取りになった。15年の仕事が水の泡となったのである。

VAXや68020などの当時の最新コンピュータは、それぞれのオペレーティングシステムを持っていたが、それらはどれ1つとしてフリーソフトウェアではなかっ

---

*1　ハッキングのように多様な内容を持つものについて単純な定義を与えるのは難しいが、私はほとんどの「ハック」に共通しているものは、遊び、工夫、探求心だと思う。つまり、ハッキングとは、遊び心に満ちた工夫によって可能なことの限界を探求することである。セキュリティ破壊については「クラック」という用語を使い、セキュリティ破壊とハッキングを区別するだけで、誤解を解くための一助になる。セキュリティ破壊を行う人物は「クラッカー」である。クラッカーの一部には、チェスプレイヤーやゴルファーが含まれているのと同じように、ハッカーが含まれているかもしれないが、ほとんどはそうではない（"*On Hacking*", RMS, 2002）。

た。実行可能コードを入手するだけのためにも、NDA（nondisclosure agreement：非開示契約）を結ばなければならなかった。

つまり、コンピュータを使うときの第一歩が、隣人を助けないということを約束することだったのである。協力し合うコミュニティは禁止された。私有ソフトウェアの所有者が作った規則は、「隣人と共有したら、あなたは著作権侵害者である。変更を希望するなら、私たちにそうしてくれるよう懇願せよ」というものだった。

私有*2ソフトウェア制度（ユーザーにソフトウェアの共有や変更を認めない制度）が反社会的で、非倫理的で、単純に間違っているという思想は、一部の読者には驚きかもしれない。しかし、一般の人々を分断し、ユーザーを無援の状態に放置することを基礎とする制度に対して、ほかに何と言うことができるだろうか。私の考え方に驚いた読者は、この私有ソフトウェア制度を当然のものと考えているか、私有ソフトウェア企業が提示している約款を判断基準にしていのだろう。ソフトウェア企業は、この問題に対する考え方は1つしかないと人々が思い込むように、古くから精力的に活動してきている。

ソフトウェア企業がそれぞれの「権利」を「強制する」とか「海賊版の流通を防止する」と言うとき、彼らが実際に「言っている」ことは副次的なことである。これらの言明の本当のメッセージは、彼らが当然と考えている実際には書かれていない仮定にある。一般の人々は、それを無批判的に受け入れるものと考えられているのである。では、それらの仮定について考えてみよう。

1つ目の仮定は、ソフトウェア企業がソフトウェアを所有する疑問の余地のない自然権を持っており、すべてのユーザーにその権力を行使できるということである（もしこれが自然権なら、公共のためにいかに有害であっても、私たちはそれに反対することはできないところである）。興味深いことに、合衆国憲法と法律への伝統は、この考え方を付けている。つまり、著作権は自然権ではなく、政策的に設けられた人為的な独占権であって、コピーをするというユーザーの自然権

*2　【訳注】言語は proprietary で、辞書には「所有者の」「独占的な」「著作権を持つ」等の訳語が書かれているが、本書では「私有」という訳語を選んだ。それは、GNU の「共有」の思想との対照がもっとも際立つと考えたからである。もっとも、「共有」の原語である public の反対語は private であり、「私有財産」も private property と表現されることが多い。

に制約を課するものなのだ。

もう1つの書かれていない仮定は、ソフトウェアで重要なことは、それによってどのような仕事ができるようになるかだけだということである。コンピュータユーザーは、どのような社会を持つことを認められているかということについて考えてはならないものとされている。

第3の仮定は、企業にユーザーへの支配権を差し出さなければ、使えるソフトウェア（あるいは、あれこれの特定の仕事をするプログラム）は作られないというものである。この仮定は、フリーソフトウェア運動が、ユーザーを縛らずに大量の役に立つソフトウェアを提供できることを実証するまで、もっともらしく見えていたかもしれない。

私たちがこれらの仮定を鵜呑みにすることなく、ユーザーを第1に考えながら、通常の常識的な倫理に基づいてこれらの問題を判断するなら、まったく異なる結論に達する。コンピュータユーザーは、それぞれのニーズに合わせてプログラムを書き換える自由やソフトウェアを共有する自由を持たなければならない。なぜなら、他人を助けることは、社会の基礎だからである。

倫理的に厳しい選択

コミュニティが消えてから、以前と同じように生きることは不可能になった。私は倫理的に厳しい選択を迫られた。

簡単な選択肢は、NDAに署名し、仲間のハッカーを助けないことを約束して、私有ソフトウェアの世界に入ることだった。その場合、おそらくNDAのもとでリリースされるソフトウェアを開発し、他人にも仲間に対する裏切りを迫ることになるだろう。

こういうやり方でも、稼ぐことはできただろうし、コードを書くことを楽しむこともおそらくできただろう。しかし、生涯を終えようとするときに、人々を分断する壁を築くために過ごしてきた年月を振り返り、世界を悪くするために人生を費やしてきたのかと感じるだろうことはわかっていた。

私はすでにNDAを押し付けられる側になることは経験していた。それは、私とAIラボがある人物にプリンタ制御プログラムのソースコードの提供を拒否さ

れたときのことである（このプログラムがある機能を持っていないがために、このプリンタを使うのはとてもイライラすることだった）。そのため、私はNDAを罪のないものだとはとても思えなかったのである。私は、彼がソースコードの共有を拒否したときに非常に怒った。逆の立場になって、他のすべての人たちに対して同じことをすることは不可能だった。

簡単だが面白くない選択肢として、コンピュータの世界から離れるというものもあった。そうすれば、私の技能が悪用されることはないが、無駄になることに変わりはなかった。私自身はコンピュータユーザーを分断し、制約を押し付ける罪を免れることができるが、同じ問題は間違いなく発生する。

そこで、私は、プログラマが善に貢献するための道を探した。コミュニティを復活させるために書けるプログラムはないか、自問自答した。

答えははっきりしていた。まず最初に必要なものは、オペレーティングシステムだった。オペレーティングシステムがあればさまざまなことができるが、なければコンピュータを動かすことさえままならない。フリーオペレーティングシステムがあれば、協力し合うハッカーたちのコミュニティを再び築くことができるだろう。コミュニティに加わるように誘うこともできるだろう。そして、まず友達をなくすことを強いられずに、誰もがコンピュータを自由に使えるようになるはずだ。

私は、オペレーティングシステム開発者として、この仕事をするための適切な技能を持っていた。成功を当然のこととして見込むことはできないものの、私は自分がこの仕事をするために選ばれた者だと確信した。私は、Unix互換システムを作ることにした。そうすれば、移植性を確保できるし、Unixユーザーが簡単に移って来れる。GNUという名前は、ハッカーの伝統に従い、"GNU's Not Unix"という再帰的な（入れ子状の）頭字語として選ばれたものである。

オペレーティングシステムとは、単なるカーネルのことではない。他のプログラムを動かせれば充分というわけではないのだ。1970年代、オペレーティングシステムの名に値するシステムは、コマンドプロセッサ、アセンブラ、コンパイラ、インタープリタ、デバッガ、テキストエディタ、メーラなどのプログラムを含んでいた。ITS然り、Multics然り、VMS然り、Unix然り。GNUオペレーティングシステムも、それらを持つものにしよう。

その後、私はヒレル[3]のものだとされる次の言葉を聞いた。

「私が自分のために動かなければ、誰が私のために動いてくれるというのだ。私が私だけのために動くのだとすれば、私は一体何者なのだ。今やらなければ、いつやる？」

GNUプロジェクトを立ち上げることにしたのも、同じような考え方からだった。

私は無神論者なので、宗教界の指導者には従わないが、彼らの中の誰かが言ったことを尊敬することはある。

自由という意味でのフリー

「フリーソフトウェア」という用語は、誤解されることがある。価格とは無関係だ。フリーとは自由のことである。そこで、フリーソフトウェアの定義を下しておくことにしよう。以下の条件を満たすプログラムは、そのユーザーにとって、フリーソフトウェアである。

- ユーザーが任意の目的のためにプログラムを実行する自由を持っている。
- ユーザーが自らのニーズに合わせてプログラムを書き換える自由を持っている（この自由を現実的に有効なものにするためには、ソースコードにアクセスできなければならない。なぜなら、ソースコードを持たないプログラムに変更を加えるのは、極度に難しいことだからである）。
- ユーザーが無料、あるいは有料でコピーを再頒布する自由を持っている。
- ユーザーがプログラムの変更バージョンを頒布する自由を持っており、コミュニティがそのユーザーによる改良から利益を受け取ることができる。

「フリー」が価格のことではなく、自由であることを意味しているので、コピーを売ることとフリーソフトウェアであることの間には何らの矛盾もない。実際、コピーを販売する自由は、非常に重要である。CD-ROM化されたフリーソフト

---

[3] 【訳注】キリストと同時代のユダヤ教ラビ。

ウェアのコレクションはコミュニティにとって重要であり、それを販売することは、フリーソフトウェアの開発資金を稼ぐための重要な手段である。そのため、このようなコレクションに自由に組み込めないプログラムは、フリーソフトウェアではない。

「フリー」という用語は曖昧なので、人々はずっと代わりになる言葉を探してきたが、適切なものを見つけた人はまだいない。英語は他の言語よりも多くの単語とニュアンスを持っているが、自由としての「フリー」を意味する単純で曖昧さのない単語を持っていない。もっとも近い意味を持つ単語は、「unfettered」である。「liberated」、「freedom」、「open」などの言葉は、誤った意味を兼ね備えていたり、他の欠点を持っていたりする。

GNUソフトウェアとGNUシステム

システム全体を開発するのは、非常に大規模なプロジェクトである。目標を手が届くところに置くために、私は可能な限り既存のフリーソフトウェアを利用、修正することにした。たとえば、私はごく初期の段階で、テキストフォーマッタとして主にTEXを使うことにした。数年後、GNUのために新しいウィンドウシステムを書く代わりに、X Window Systemを使うことにした。

このような決定のために、GNUシステムとすべてのGNUソフトウェアのコレクションは同じではない。GNUシステムには、GNUソフトウェアではないプログラム、他の人々や他のプロジェクトがそれぞれの目的のために開発したプログラムが含まれている。しかし、それらはフリーソフトウェアなので、私たちはそれを使うことができるのだ。

プロジェクトの立ち上げ

私は、1984年1月にMITの仕事を辞め、GNUソフトウェアの開発を開始した。MITがフリーソフトウェアとしてのGNUの頒布に干渉できないようにするために、MITを辞めることは必要だった。私がスタッフとして残っていたら、MITは私の仕事の所有権を主張したり、独自の頒布条件を課したり、私の仕事を私有ソ

フトウェアパッケージに化けさせたりすることさえできていただろう。新しいソフトウェアコミュニティの創出という当初の目的にまったく添わない結果を招くために、大仕事をするつもりはなかった。

もっとも、ウィンストン教授（当時のMIT AIラボ所長）は、ラボの設備をそのまま使えるようにはからってくれた。

第一歩

GNUプロジェクトを立ち上げる直前に、私はFree University Compiler Kit——別名VUCK（オランダ語の「フリー」は、Vから始まる）というものの話を聞いた。これは、CやPascalなど、複数の言語を処理し、複数のターゲットマシンをサポートするコンパイラだった。私はGNUでそれを使えるかどうかを尋ねるために、作者に手紙を書いた。

返事は嘲笑的なものだった。大学はフリー（自由）だが、コンパイラは違う（フリーコンパイラではない）というのである。そこで、GNUプロジェクトの最初のプログラムは、マルチ言語マルチプラットフォームコンパイラにすることにした。

コンパイラ全体を自分で書かなくても済むとよいと思い、私はローレンスリバーモア研究所で開発されたマルチプラットフォームコンパイラ、Pastelのソースコードを入手した。Pastelは、システムプログラミング言語になるようにPascalを拡張した言語で書かれており、その拡張Pascalをサポートしていた。私は、Cフロントエンドを追加し、Motorola 68000コンピュータへの移植作業を開始した。しかし、このコンパイラが何Mバイトものスタック領域を必要とするのに対し、68000のUnixシステムが64Kバイトのスタック領域しか認めていないことを知り、移植を諦めなければならなかった。

その後、Pastelコンパイラは、入力ファイル全体を走査して構文木を構築し、構文木全体を「命令」のチェインに変換し、出力ファイル全体を生成しており、その間、メモリを一切解放していないことがわかった。これがわかった時点で、私は0から新しいコンパイラを書かなければならないという結論に達した。その新コンパイラは、今、GCCと呼ばれているものである。Pastelコンパイラのコードは一切使われていないが、自分で書いたCフロントエンドは何とか修正して活用

した。しかし、その仕事に取り掛かったのは、数年後のことである。まず、私は GNU Emacs を書いた。

GNU Emacs

私は 1984 年 9 月に GNU Emacs の仕事を始めた。そして、1985 年始めには、使えるものになり始めていた。これで、私は Unix システムを使って編集の仕事をできるようになった。私は、vi や ed の使い方を学ぶ気にはなれなかったので、そのときまでは Unix 以外のマシンで編集をしていたのである。

この頃から、人々は GNU Emacs を使いたがるようになったが、そのことは、ソフトウェアをどのように流通させるかという問題を引き起こした。もちろん、自分が使っていた MIT のコンピュータの anonymous FTP サーバから入手できるようにはした（そのため、このコンピュータ、prep.ai.mit.edu は、GNU のメイン ftp サイトになった。数年後、このマシンが廃棄処分になったときには、新しい ftp サーバに同じ名前を移した）。しかし、その当時、GNU に関心を持つ人々の多くは Internet に接続しておらず、ftp ではコピーを入手できなかった。そこで、そのような人々にどう言ったらよいかが問題になった。

「Internet に接続できてコピーを作れる友達を探してください」と言ってもよかったし、オリジナルの PDP-10 Emacs のときと同じように「私にテープと SASE（切手を貼った返信用封筒）を郵送していただければ、Emacs をコピーして送り返します」と言ってもよかった。しかし、私は無職だったし、フリーソフトウェアで稼ぐ方法を探していた。そこで、私は、150 ドルで希望者全員にテープを郵送すると発表した。このようにして私はフリーソフトウェア流通ビジネスを始め、現在、Linux ベースの GNU システムを販売している企業の先駆者になったのである。

すべてのユーザーに対してフリーなソフトウェア

作者の手を離れたときにフリーソフトウェアであったとしても、そのプログラムは、コピーを持っているすべての人々にとってフリーソフトウェアであるとは限らない。たとえば、PDS（パブリックドメインソフトウェア、すなわち著作権が放棄されているソフトウェア）はフリーソフトウェアだが、誰もがそれに変更を加えた私有バージョンを作ることができる。同様に、多くのフリープログラムは、著作権を放棄してはいないものの、変更が加えられた私有バージョンを認めるような単純で寛大なライセンスのもとで流通されている。

この問題を非常によく示している例がX Window Systemである。MITで開発され、寛大なライセンスのもとでフリーソフトウェアとしてリリースされたX Windowは、すぐにさまざまなコンピュータメーカーに採用された。これらの企業は、私有しているUnixシステムにバイナリ形式のみでXを追加し、同じNDAを適用した。これらのXは、Unixと同様にフリーソフトウェアではなくなってしまったのである。

X Window Systemの開発者たちは、これを問題だとは考えなかった。むしろ、こうなることを予想し、意図していたのである。彼らの目標は自由ではなく、「多くのユーザーの獲得」と定義される「成功」だった。彼らは、ユーザーが自由を持つかどうかに注意を払わなかった。ユーザーを増やすことしか考えていなかったのである。

これは、ずいぶん逆説的な話である。自由の量を計る方法が2通りあり、「このプログラムはフリーか」という問いに対してそれらが異なる答えを出しているのである。MITリリースの頒布条件が与える自由に基づいて判断するなら、Xはフリーソフトウェアだったと言える。しかし、Xの平均的なユーザーの自由度を計るなら、Xは私有ソフトウェアだと言わなければならない。ほとんどのXユーザーは、フリーバージョンではなく、Unixシステムに付属している私有バージョンを実行していたのである。

コピーレフトとGNU GPL

GNUの目標は、単にポピュラーになることではなく、ユーザーに自由を与えることだった。そこで、私たちは、GNUソフトウェアが私有ソフトウェアに化けるのを防ぐ頒布条件を必要としていた。そして、私たちが今使っている方法は、コピーレフト（copyleft）というものである。

コピーレフトは著作権（copyright）法を利用しているが、通常とは逆の目的に奉仕するように、逆立ちさせた使い方をしている。著作権法は、ソフトウェアを私有化するための手段ではなく、ソフトウェアの自由を守る手段になったのである。

コピーレフトという考え方の中心は、プログラムを実行し、プログラムをコピーし、プログラムを書き換え、書き換えられたバージョンを流通させることをすべての人に認める一方で、独自の制限事項の追加を禁止することにある。つまり、「フリーソフトウェア」がよって立つ由縁であるところの自由が、コピーを持つすべての人に対して保証される。自由は、絶対に奪うことのできない権利になったのである。

コピーレフトの効力を保つためには、書き換えられたバージョンもフリーでなければならない。こうすれば、私たちのコミュニティは、私たちの仕事を基礎とする仕事が公表されたときに、必ずその仕事を利用できる。プログラマとしての仕事を持つプログラマがボランティアでGNUソフトウェアを改良したときに、その雇い主が「君が加えた変更は、プログラムの私有バージョンを作るためにうちの会社で使うつもりだから、共有に付するわけにはいかない」などと言えないようにするのがコピーレフトである。

プログラムのすべてのユーザーに対して自由を保証したければ、変更点もフリーにするよう要求することは非常に重要である。X Window Systemを私有化した企業は、通常、それぞれのシステムやハードウェアにX Window Systemを移植するために、何らかの変更を加えている。これらの変更点は、Xの規模の大きさから比べればごく少量だが、ごくわずかというわけでもない。変更を加えたことが、ユーザーの自由を否定する言い訳になるのであれば、誰もがその言い訳を利用するようになるだろう。

フリープログラムとフリーではないコードを結合するときにも、同様の問題が発

生する。そのような結合は、確実にフリーではないコードを作るだろう。フリーではない部分で欠けている自由は、全体として欠けている自由でもある。そのような結合を認めれば、船を沈めるに足る穴を開けることになるだろう。コピーレフトは、絶対にこの穴を塞ぐ必要がある。コピーレフトの対象プログラムに何かを追加、結合した場合、結合後の大きなバージョンもまた、フリーでコピーレフトの対象になっていなければならない。

私たちがほとんどの GNU ソフトウェアのために使っているコピーレフトの具体的な条文は、GNU 一般公開使用許諾書（GNU GPL）である。私たちは、特定の状況で使われる別の種類のコピーレフトも持っている。たとえば、GNU マニュアルもコピーレフトの対象だが、マニュアルには GNU GPL の複雑さは不要なので、ずっと単純な種類のコピーレフトを使っている。

1984 年か 85 年のことだが、ドン・ホプキンス（非常に想像力豊かな人物）が私に手紙を送ってきた。その封筒には、いくつかの面白いことが書かれていたが、「Copyleft--all rights reversed.」（コピーレフト――すべての権利が逆になっている）というのもその 1 つだった。私は、このコピーレフトという言葉を当時構想していた流通概念の名前として使うことにした。

フリーソフトウェア財団

Emacs を使うことに対する関心が高まるにつれて、GNU プロジェクトには、私以外の人々が関わるようになってきた。そこで、私たちは再び資金集めを追求すべきだと考え、1985 年にフリーソフトウェア開発のための非課税慈善団体として、フリーソフトウェア財団（FSF）を設立した。FSF は、Emacs のテープ頒布ビジネスも引き継ぎ、その後、テープに他のフリーソフトウェア（GNU および非 GNU）を追加し、フリーマニュアルの販売も開始して、事業を拡大した。

FSF は寄付も受け付けているが、その収入の大半は、フリーソフトウェアのコピーや他の関連サービスの販売から得ている。現在、FSF はソースコードの CD-ROM、バイナリを収めた CD-ROM、美しく印刷されたマニュアル（どれも、再頒布、変更の自由を伴う）、およびデラックスディストリビューション（顧客が選んだプラットフォームのためにソフトウェアの全コレクションを構築する）を

販売している。

FSFのスタッフは、たくさんのGNUソフトウェアパッケージを開発し、保守してきた。特筆すべきは、Cライブラリとシェルである。GNU Cライブラリは、GNU/Linuxシステムで動作するすべてのプログラムがLinuxとやり取りするために使っているものである。このライブラリは、FSFのスタッフ、ローランド・マクグラスによって開発された。ほとんどのGNU/Linuxシステムで使われているシェルはBASH（Bourne Again Shell）だが、これはFSFのスタッフであるブライアン・フォックスによって開発された。

私たちがこれらのプログラムの開発に資金提供したのは、GNUプロジェクトが単なるツールや開発環境ではないからである。私たちの目標は、完全なオペレーティングシステムを作ることであり、これらのプログラムはその目標のために必要だったのだ。

Bourne Again Shellは、Unixで通常に使われているBourne Shellというシェルの名前を借りたジョークである。

フリーソフトウェアのサポート

フリーソフトウェアの哲学は、広く普及しているビジネスのある特定の形態に反対するが、ビジネスそのものに反対するわけではない。ビジネスがユーザーの自由を尊重するとき、私たちはそのビジネスの成功を望む。

Emacsのコピーの販売は、フリーソフトウェアビジネスの1つの形態を実証するものである。FSFがその事業を引き継いだとき、私は生計を立てるための別の手段を見つけなければならなくなった。そして、私が開発したフリーソフトウェアに関連したサービスの販売を思いついた。たとえば、GNU Emacsのプログラム方法やGCCのカスタマイズ方法といったテーマについて教えることやソフトウェア開発である。ちなみに、ソフトウェア開発のほとんどはGCCの新プラットフォームへの移植だった。

今日、これらのフリーソフトウェアビジネスは、数社によって展開されている。CD-ROMでフリーソフトウェアコレクションを販売する企業、ユーザーの質問への回答から、バグフィックス、大きな新機能の開発に至るまでのサポートサー

ビスを販売する企業がある。新しいフリーソフトウェア製品の開発を事業の軸に据えるフリーソフトウェア企業さえ生まれ始めている。

しかし、注意していただきたい。「オープンソース」という言葉を掲げているいくつかの企業は、実際には、フリーソフトウェアと併用される非フリーソフトウェアに事業の基礎を置いている。これらは、フリーソフトウェア会社ではなく、ユーザーを自由から引き離そうとする製品を作っている私有ソフトウェア会社である。彼らはそれを「付加価値」と称するが、それは彼らが私たちに受け入れさせようとする価値にほかならない。つまり、自由よりも便宜性を高く評価する価値観である。自由により高い価値を認めるなら、そのようなものは「自由を抜き取った」製品と呼ぶべきだろう。

技術的な目標

GNUの第1の目標は、フリーソフトウェアになることだった。GNUがUnixと比べて技術的に優れていなかったとしても、ユーザーの協力を認めるという社会的なメリット、ユーザーの自由を尊重するという倫理的なメリットはある。

しかし、標準として認められている優れた作業習慣に従うのは当然のことである。たとえば、適当にサイズの上限を決めるのではなく、動的にデータ構造を確保するとか、意味がある場合には、すべての8ビットコードを処理するといったことである。

さらに、私たちは16ビットマシンをサポートしないことにして（GNUシステムが完成する頃には、32ビットマシンが標準になることは明らかだったので）、Unixの小さなメモリサイズに対するこだわりを捨てた。また、1Mバイトを超えない限り、メモリの利用を抑制しようとはしなかった。非常に大きなファイルを処理することがそれほど重視されないプログラムでは、入力ファイル全体をメモリに読み込み、入出力の心配をせずにその内容を走査することをプログラマたちに勧めた。

これらの決定により、多くのGNUプログラムは、信頼性とスピードの面でUnixの同等のプログラムよりも優れたものになった。

寄付されたコンピュータが抱えていた問題

GNUプロジェクトの評価が上がってくるにつれて、プロジェクトにはUnixマシンが寄付されるようになってきた。GNUコンポーネントのもっとも簡単な開発方法は、Unix上でUnixのコンポーネントを1つずつ取り替えていくことだったので、これは非常に役に立った。しかし、このことはある倫理的な問題を引き起こした。それは、私たちがUnixのコピーを持つことは正しいかどうかということである。

Unixは私有ソフトウェアだった（今もそうである）が、GNUプロジェクトの哲学は、私有ソフトウェアを使わないとしていた。しかし、自己防衛のための暴力が正当化されるのと同じ理由から、他人が私有パッケージを使うのをやめるのを助けるフリーバージョンを開発するために必要不可欠なのであれば、私有パッケージを使うことは正当化されるという結論を下した。

しかし、必要悪であろうが、悪は悪である。現在はフリーオペレーティングシステムで置き換えたので、私たちはUnixのコピーを持っていない。あるマシンのオペレーティングシステムをフリーオペレーティングシステムに置き換えられない場合、私たちはマシンのほうを取り替える。

GNUタスクリスト

GNUプロジェクトが前進し、見つかった、あるいは開発が終わったシステムコンポーネントの数が増えてくると、残された穴のリストを作ると役に立つだろうということになった。私たちは、そのリストを使って、足りない部分を開発してくれるプログラマを募った。このリストは、GNUタスクリストと呼ばれるようになった。私たちは、足りないUnixコンポーネントのほか、本当に完全なシステムが備えるべき役に立つソフトウェアとドキュメントをリストに加えた。

現在、GNUタスクリストに残っているUnixコンポーネントはほとんどない。あまり重要ではない一部を除き、それらの仕事は終わった。しかし、リストには、人によっては「アプリケーション」と呼ぶようなプロジェクトが満載されている。ごく狭い範囲のユーザー以外にも魅力的なプログラムは、オペレーティングシス

テムに追加すべき有益なプログラムである。

タスクリストには、ゲームさえ含まれている。それはリストを作った当初からのことである。Unixにはゲームが含まれているので、当然ながらGNUもそれらを含まなければならない。しかし、ゲームの場合、互換性が問題になることはないので、Unixゲームのリストに忠実に従っているわけではない。私たちのリストには、ユーザーが好みそうな別のゲームセットが含まれている。

GNU LGPL

GNU Cライブラリは、フリーライブラリに対する私有ソフトウェアのリンクを許可するGNUライブラリ一般公開使用許諾書（LGPL）という特別な種類のコピーレフトを使っている。このような例外を設けたのはなぜなのだろうか。

これは、原則の問題ではない。私有ソフトウェア製品が私たちのコードを組み込む権利を認める原則など存在しない（私たちとの共有を拒否することがわかっているプロジェクトに貢献する理由があろうか）。Cライブラリやその他のライブラリでLGPLを使っているのは、戦略の問題である。

Cライブラリは、一般的な仕事をする。すべての私有システムやコンパイラには、Cライブラリが付属している。そのため、フリーソフトウェアでなければ私たちのCライブラリを使えないようにしても、フリーソフトウェアには何の利益もない。単に、私たちのライブラリを使う気になれないものにするだけである。

これには例外となるシステムが1つある。GNUシステム（これにはGNU/Linuxも含まれる）では、GNU Cライブラリが唯一のCライブラリである。そのため、GNUシステムで私有プログラムをコンパイルできるかどうかは、GNU Cライブラリの頒布条件によって決まる。GNUシステムで私有アプリケーションの実行を認める倫理的な理由はないが、戦略的に考えて、それを禁止すると、フリーアプリケーションの開発を奨励するよりも、GNUシステムを使いたくない気分を助長することになる。

CライブラリではLGPLを使うことが戦略的に優れているというのは、そういうことである。他のライブラリについての戦略は、ケースバイケースで検討しなければならない。ある種のプログラムを書きやすくするような特別の仕事をこな

すライブラリは、GPLのもとでリリースし、使用をフリープログラムのみに制限すれば、他のフリーソフトウェア開発者を助け、私有ソフトウェアに対する優位性を与えることになる。

たとえば、BASHのコマンドライン編集機能のために開発されたGNU Readlineライブラリ[*4]について考えてみよう。Readlineは、LGPLではなく、通常のGNU GPLのもとでリリースされている。おそらく、このことによってReadlineの使用数は減るだろうが、私たちにとって、それはまったく損失ではない。その一方で、Readlineを使うために、少なくとも1つの役に立つアプリケーションがフリーソフトウェアとして作られた。これこそ、コミュニティにとって本物の利益である。

私有ソフトウェアの開発者は金銭的な面で優位に立つが、フリーソフトウェアの開発者は相互のために優位性を生まなければならない。いずれ、私有ソフトウェアに対応するもののないGPLによって保護されるライブラリの大規模なコレクションを作りたいものだ。これらのライブラリは、新しいフリーソフトウェアの部品として使える役に立つモジュールを提供し、フリーソフトウェアの更なる開発を促進する大きな力になるだろう。

痒いところを掻く？

エリック・レイモンドは、「ソフトウェアの優れた仕事は、どれもプログラマが自分で痒いと思ったところを掻くところから始まる」と言っている。そういうこともあるかもしれないが、GNUソフトウェアの多くの主要要素は、完全なフリーオペレーティングシステムを作るために開発されている。これらは、衝動ではなく、ビジョンとプランによって作られている。

たとえば、私たちがGNU Cライブラリを開発したのは、Unix風のシステムではCライブラリが必要だからであり、BASHを開発したのは、Unix風のシステムではシェルが必要だからである。GNU tarを開発したのは、Unix風システムがtarプログラムを必要とするからである。同じことが私自身のプログラム、GNU Cコンパイラ、GNU Emacs、GDB、GNU Makeにも当てはまる。

---

*4　GNU Readlineライブラリは、入力したコマンドラインを編集できるようにしたいアプリケーションのための関数群を提供する。

一部のGNUプログラムは、私たちの自由に対する個別の脅威に対処するために開発された。私たちがcompressプログラムの代わりにgzipを開発したのは、LZWの特許[5]のためにcompressがコミュニティから失われたためである。私たちはLessTifの開発者を見つけ、さらに最近ではGNOMEとHarmonyをスタートさせたが、それも特定の私有ライブラリに起因する問題に対処するためである(後述の「非フリーライブラリ」を参照)。また、ユーザーがプライバシーと自由のどちらかを選択しなければならないようなことのないよう、ポピュラーな非フリーの暗号ソフトウェアに代わるGNU Privacy Guardを開発している。

もちろん、これらのプログラムを書いている人々は、仕事に興味を持つようになったし、多くの人々がそれぞれのニーズや関心に基づいてプログラムにさまざまな機能を追加している。しかし、興味はプログラムが存在する理由ではないのである。

予想外の開発作業

GNUプロジェクトを立ち上げた頃、私はGNUシステム全体を開発し、全体としての形でリリースすることを想像していたが、実際にはそのようにはならなかった。

GNUシステムの個々のコンポーネントはUnixシステムで実装されたので、完全なGNUシステムが完成するはるか以前から、それらのコンポーネントはUnixシステムでも実行できた。これらのプログラムの一部はポピュラーになり、ユーザーはそれらの拡張や移植を始めた。移植先は、Unixのさまざまな互換性のないバージョンだったり、まったく別のシステムだったりした。

その過程でこれらのプログラムははるかに強力になり、GNUプロジェクトに資金や貢献者を集めることになった。しかし、GNU開発者の時間は、足りないコンポーネントを次々に書いていくことにではなく、これらの移植版の維持や既存コンポーネントへの機能の追加に割かれていったので、おそらくこの動きは最小限の稼動するシステムの完成を数年遅らせる効果も持っていたはずである。

---

*5　データの圧縮のためにLempel-Ziv-Welchアルゴリズムが使われている。

GNU Hurd

1990年までに、GNUシステムはほとんど完成していた。主要コンポーネントで欠けていたものは、カーネルだけだった。私たちは、Machの上で動作するサーバプロセスのコレクションという形でカーネルを実装することにしていた。Machは、カーネギーメロン大学、ついでユタ大学で開発されたマイクロカーネルである。GNU Hurdは、Machの上で動作するサーバコレクション（「GNUの群れ」）であり、Unixカーネルのさまざまな仕事を行う。開発の開始は遅れたが、それは約束通りにMachがフリーソフトウェアとしてリリースされるのを待っていたからである。

このような設計を選んだ理由の1つは、仕事の中でもっとも難しいと思われた部分を避けるためだった。それは、ソースレベルデバッガなしでカーネルプログラムをデバッグすることである。この部分の仕事はMachの中ですでに終わっているはずなので、Hurdサーバはユーザープログラムとして GDB でデバッグできると思っていた。しかし、それが可能になるまでには非常に長い時間がかかった。また、互いにメッセージを送り合うマルチスレッドサーバは、デバッグが非常に難しいことがわかった。Hurdを安定動作させる仕事は、何年も遅れている。

Alix

GNUカーネルは、もともとHurdという名前になるはずではなかった。最初の名前は、当時の私の恋人だった女性の名前を取ったAlixだった。彼女はUnixのシステム管理者で、自分の名前がUnixシステムの命名パターンに当てはまっていると言っていた。彼女は、「誰かがきっと私と同じ名前のカーネルを作るはずよ」と友人に冗談を言った。私は何も言わなかったが、Alixという名前のカーネルを作って彼女を驚かせることに心を決めていた。

しかし、事態はそのようには展開しなかった。カーネルのメインプログラマのマイケル・ブッシュネル（現在はトーマス）は、Hurdという名前を好み、Alixはカーネルの特定の部分（システムコールをトラップし、Hurdサーバにメッセージを送ってそれを処理する）を指すものとして再定義された。

最終的に、Alixと私は別れ、彼女は名前を変えた。それとは無関係に、Hurdの設計は、Cライブラリがサーバに直接メッセージを送るように変更された。そのため、Alixコンポーネントは設計から消えてしまった。

しかし、これらのことが起きる前に、彼女の友達の1人がHurdのソースコードにAlixという名前があることに気づき、彼女にそのことを教えた。だから、Alixという名前は機能を果たしたのである。

LinuxとGNU/Linux

GNU Hurdは、まだ稼動できる状態にはなっていないが、幸いにも別のカーネルが現れた。リーナス・トーバルズは、1991年にUnix互換カーネルを開発し、Linuxと名付けた。そして、1992年頃、Linuxとまだ完全には完成していないGNUシステムを結合することによって、完全なフリーオペレーティングシステムが完成した（もちろん、両者の結合は、それ自体大きな仕事だった）。現在、実際にGNUシステムの1バージョンを実行できるのは、Linuxのおかげである。

私たちは、GNUシステムとカーネルとしてのLinuxの結合という構成を表現するために、このバージョンのシステムをGNU/Linuxと呼んでいる。

将来の試練

私たちは、広い範囲のフリーソフトウェアを開発できるという自らの能力を証明した。だからといって、私たちが無敵で安泰だというわけではない。フリーソフトウェアの将来を不確実なものにするいくつかの試練が待ち構えている。それらの試練に遭ったときには、不屈の努力と忍耐が必要だろう。場合によっては、それが数年続くこともある。人々が自由を尊重し、自由を奪われることを拒否するときに示す決意が必要とされる。

以下の4つの節では、これらの試練について論じる。

非開示主義のハードウェア

ハードウェアメーカーは、次第にハードウェアの仕様を非開示にする傾向にある。これは、フリーのドライバを書いてLinuxやXFree86[*6]が新ハードウェアをサポートできるようにすることを困難にする。今日は、完全なフリーシステムがあるが、明日のコンピュータをサポートできなければ、明日にはフリーシステムがなくなるかもしれない。

この問題への対処方法は2つある。プログラマは、リバースエンジニアリングによって、ハードウェアのサポート方法を突き止めることができる。プログラマ以外の人々は、フリーソフトウェアがサポートしているハードウェアを選ぶことができる。フリーシステムが増えれば、仕様の非開示主義は自分の首を絞める結果になるだろう。

リバースエンジニアリングは大仕事である。そのような仕事に着手するだけの意志を持ったプログラマを集めることができるだろうか？　イエス。フリーソフトウェアは主義主張の問題で、非フリードライバは許容できないという強い意志を築き上げることができれば可能である。そして、フリードライバを使えるようにするために、プログラマ以外の人々は余分なお金や余分な時間を使ってくれるだろうか？　自由を確保するという意志が浸透すれば、それに対する答えもイエスである。

非フリーライブラリ

フリーオペレーティングシステム上で動作する非フリーライブラリは、フリーソフトウェア開発者を陥れる罠である。エサは魅力的な機能だが、そのライブラリを使ってしまうと、プログラムはフリーオペレーティングシステムの一部ではあり得なくなってしまうので、プログラマは罠に落ちたということになる（厳密に言えば、そのプログラムをフリーオペレーティングシステムの一部にすることはできるが、問題のライブラリがなければそのプログラムは動作しないだろう）。

---

*6　XFree86は、表示ハードウェア（マウス、キーボードなど）とのインターフェイスとなるデスクトップ環境を提供するプログラムである。多くの異なるプラットフォームで動作する。

さらに悪いことに、私有ライブラリを使っているプログラムがポピュラーになったら、疑いを知らない他のプログラマたちも罠に陥れることになる場合がある。

この問題の最初の事例は、80年代のMotif[*7]ツールキットだった。当時はまだ、フリーオペレーティングシステムはなかったが、Motifがその後どのような問題を起こすかは明らかだった。GNUプロジェクトは、Motifに2通りの反応を返した。個々のフリーソフトウェアプロジェクトにMotifだけではなく、フリーXツールキットもサポートするように呼びかけることと、Motifに代わるフリーシステムを書く人を募ることである。この仕事には長い時間がかかった。Hungry Programmersによって開発されたLessTifがほとんどのMotifアプリケーションをサポートできるだけの力を付けたのは、1997年のことである。

1996年から1998年にかけて、Qtという別の非フリーGUI（グラフィカルユーザーインターフェイス）ツールキットが、大規模なフリーソフトウェアコレクションのKDEデスクトップ環境で使われていた。

フリーGNU/Linuxシステムは、そのライブラリを使うわけにはいかなかったので、KDEを使うことができなかった。しかし、フリーソフトウェアに対する厳密なこだわりのない一部の市販GNU/LinuxディストリビューションがそれぞれのシステムにKDEを追加した。機能は強化されたが自由が弱体化されたシステムを作ったのである。KDEグループは、多くのプログラマに対してQtを使うように積極的に働きかけており、数百万の新しい「Linuxユーザー」は、そこに問題が潜んでいるということを教えられていなかった。この問題は深刻に感じられた。

フリーソフトウェアコミュニティは、GNOMEとHarmonyという2通りの方法でこの問題に対処した。

GNOME（GNU Network Object Model Environment）は、GNUのデスクトップ環境プロジェクトである。1997年にミゲル・デ・イカーザが開発に着手し、Red Hat Softwareの支援のもとで開発されたGNOMEは、KDEと同様のデスクトップ機能を提供するが、フリーソフトウェアだけを使っているところが異なっていた。C++だけではなく、さまざまな言語をサポートするなど、技術的な長所も持っていた。しかし、GNOMEの最大の目的は、自由——すなわち非フリーのソフト

*7　Motifは、X Windowの上で動作するグラフィカルユーザインターフェイスおよびウィンドウマネージャである。

ウェアを使うことを要求しないことである。

Harmonyは、Qtを使わなくてもKDEソフトウェアを実行できるようにするために設計された、互換性のある代替ライブラリである。

1998年11月、Qtの開発者たちは、実際に遂行されればQtをフリーソフトウェアにするはずのライセンス契約の変更を発表した。確かなことはわからないが、このような方針変更がなされた理由の一部は、Qtがフリーではなかった時代に起こした問題に対するコミュニティの断固とした反応にあったと思う（ただし、新ライセンスは不便で不公平なものなので、まだQtの使用は避けることが望ましい）[8]。

次の魅力的な非フリーライブラリに対しては、どのように対応すべきだろうか。コミュニティ全体が罠から身を遠ざけることの必要性を理解してくれるだろうか。それとも、多くのメンバーが利便性のために自由を放棄し、大きな問題を引き起こしてしまうのだろうか。私たちの将来は、私たちの哲学にかかっている。

ソフトウェア特許

私たちが直面している最悪の脅威は、ソフトウェア特許である。ソフトウェア特許は、20年もの間、フリーソフトウェアがアルゴリズムや機能を利用できないようにすることができる。LZW圧縮アルゴリズム特許は1983年に発効となり、私たちはまだ正しく圧縮されたGIFを作るフリーソフトウェアをリリースすることができないでいる。1998年には、特許がらみの訴訟が現実的な脅威となったので、ディストリビューションからMP3圧縮オーディオを作成するフリープログラムが取り除かれた。

特許には対処方法がある。特許が無効である証拠を探すこともできるし、同じ仕事をする別の方法を探すこともできる。しかし、これらの方法が機能するのは限られたときだけであり、両方とも失敗したときには、特許のためにすべてのフリーソフトウェアがユーザーの望む機能を失うことを余儀なくされる場合がある。このようなときにはどうしたらよいのだろうか。

---

[8]　2000年9月、QtはGNU GPLのもとで再リリースされ、この問題は全面的に解決された。

自由のためにフリーソフトウェアを評価する私たちは、いずれにしてもフリーソフトウェアの側に立つだろう。私たちは、特許の対象となっている機能なしで仕事ができるように工夫する。しかし、フリーソフトウェアに技術的な優位性を期待するユーザーは、特許によって優位性が失われたらそれを欠陥と呼ぶだろう。開発の「伽藍」[9]モデルの実践的な有効性や、一部のフリーソフトウェアの信頼性や能力について語ることは意味のあることだが、私たちはそこに留まっているわけにはいかない。私たちは、自由と原則について語らなければならないのである。

フリードキュメント

私たちのフリーオペレーティングシステムの最大の弱点は、ソフトウェアの中にはない。それは、システムに組み込める優れたフリーマニュアルがないことである。ドキュメントは、あらゆるソフトウェアパッケージに必要不可欠な部分である。重要なフリーソフトウェアパッケージに優れたフリーマニュアルが含まれていなければ、それは大きな欠陥と言わなければならない。現在の私たちはそのような穴を無数に抱え込んでいる。

フリードキュメントは、フリーソフトウェアと同様に、価格の問題ではなく、自由の問題である。フリーマニュアルの基準は、フリーソフトウェアとほぼ同じところにある。重要なのは、すべてのユーザーに自由を与えるかどうかだ。プログラムのすべてのコピーにマニュアルを同梱できるようにするために、オンラインと紙の両方の形態で再頒布（商業目的の販売を含む）が認められていなければならない。

変更に対する許可も重要である。一般原則として、私はあらゆる種類の論文や本を変更する権利が認められることが重要だとは考えていない。たとえば、このような論文に変更を加えることを許可する必要があるとは思わない。この論文には、私たちの活動と思想が書かれているのである。

しかし、フリーソフトウェアのドキュメントについては、書き換えの自由が重要な意味を持つ理由がある。人々がソフトウェアを書き換える権利を行使し、機

[9] おそらく、私は「バザール」モデルについて書こうとしていたのだろう。「バザール」モデルは、その頃としては新しく、最初は論争の的となっていた代替モデルである。

能を追加、変更したとき、彼らが仕事に忠実なら、マニュアルも書き換えるだろう。それにより、変更後のプログラムに合った正確で使えるドキュメントを提供できるようになるわけである。プログラマが職務を忠実に遂行し、仕事を完成させることを認めないマニュアルは、私たちのコミュニティのニーズを満たさない。

ここで変更方法に何らかの歯止めをかけても、問題は起きないだろう。たとえば、オリジナルの著者の著作権情報、頒布条件、著者リストを書き換えないことを要求することはかまわない。変更版に書き換えられているということの記載を求めることもかまわないし、技術以外のトピックを扱っている節については、節全体の削除や変更を禁止することまで認められるはずだ。これらの制限が問題とならないのは、職務に忠実なプログラマが変更後のプログラムに合わせてマニュアルを書き換えることを禁止していないからである。言い換えれば、それらの制限は、フリーソフトウェアコミュニティがマニュアルをフル活用することを妨げない。

しかし、マニュアルの「技術的な」内容はすべて書き換え可能でなければならないし、通常のすべてのメディア、すべてのチャネルを通じて結果を頒布できなければならない。そうでなければ、制限はコミュニティにとって有害であり、そのマニュアルはフリーではないということになる。私たちは、別のマニュアルを用意しなければならない。

フリーソフトウェアの開発者たちは、あらゆるタイプのフリーマニュアルを書くという意識と決意を持つようになるだろうか。繰り返すが、私たちの将来は、私たちの哲学にかかっている。

私たちは自由について話し合わなければならない

Debian GNU/Linux や Red Hat Linux などの GNU/Linux システムのユーザーは数千万人いると推計されている。フリーソフトウェアは、純粋に実践的な理由からそれだけのユーザーが追随するだけの現実的なメリットを生むところまで成長してきたのだ。

このことがもたらす良い結果は明らかである。フリーソフトウェアの開発に対する関心が高まり、フリーソフトウェアビジネスの顧客が増え、私有ソフトウェ

ア製品ではなく市販フリーソフトウェアを開発する企業の士気が高まる。

しかし、フリーソフトウェアに対する関心は、それが基礎としている哲学の認知度よりも速いペースで高まっている。今まで示してきた試練や脅威に私たちが対抗できるかどうかは、自由を求める固い意志にかかっている。コミュニティにその意志を確実に持たせるためには、コミュニティに入ってくる新しいユーザーにこの思想を普及させなければならない。

しかし、私たちはそれに失敗している。人々は、コミュニティの倫理を教えようとすることよりも、コミュニティに新しいユーザーを引き付けようとすることにはるかに熱心になっている。私たちは両方を必要としており、両方のバランスをとらなければならない。

「オープンソース」

コミュニティの一部が「フリーソフトウェア」という用語の使用を廃して「オープンソースソフトウェア」という言葉を使い出した1998年には、新しいユーザーに自由について教えることがさらに困難になった。

この用語を推した人々の一部は、「フリー」が「無料」と間違えられるのを防ごうと思っていた。これは正しい目標である。しかし、フリーソフトウェア運動とGNUプロジェクトの動因であった精神と原則を脇に置いて、経営者やビジネスユーザーにアピールすることを目指した人々もいた。経営者やビジネスユーザーの多くは、自由、コミュニティ、原則よりも利益を重視するイデオロギーを持っている。「オープンソース」というレトリックは、高品質で強力なソフトウェアを創造する可能性にスポットライトを当てるが、自由、コミュニティ、原則の思想を覆わせてしまう効果を持っている。

「Linux」関連の雑誌は、このことをはっきり示す例である。これらの雑誌には、GNU/Linuxと併用できる私有ソフトウェアの広告が満載されている。次のMotifやQtが現れたとき、これらの雑誌は、プログラマたちに離れよと警告を発するのだろうか、それともそれらの広告を掲載するのだろうか。

ビジネス界からの支援は、さまざまな面でコミュニティに貢献できる。そのことに関してはみな平等であり、意味がある。しかし、自由と正義についての言及

を減らすことによってビジネス界からの支援を勝ち取っても、惨めな結果を招く可能性がある。それは、ソフトウェアの普及と思想の浸透のアンバランスをさらに悪化させるだろう。

「フリーソフトウェア」と「オープンソース」は、多かれ少なかれ同じタイプのソフトウェアを指すが、ソフトウェアとその価値観についての思想は異なる。GNUプロジェクトは、単なるテクノロジーではなく、自由が重要なのだという思想を表現するために、「フリーソフトウェア」という用語を使い続ける。

トライしよう

「『トライ』というものはない」というヨーダの哲学はかっこよく聞こえるかもしれないが、私には無関係である。私は、自分にその仕事ができるのかどうか不安に思いながら、そしてできたとしても充分に目標を達したのか確信を持てないまま、ほとんどの仕事をしてきた。しかし、敵と私の町の間には私しかいなかったので、何はともあれトライしてきた。自分自身驚いているが、私はときどき成功を収めた。

逆に、ときどきは失敗し、私の町は敵に奪われた。そして、さらに脅威を受けている別の町を見つけ、新しい戦いに備えた。私は脅威を探し、敵と町の間に身を置き、他のハッカーを私の陣営に迎えてきた。

今は、多くの場合、私は、たった一人ではない。味方のハッカーたちが防衛線を築いているのを眺め、今のところは町が生き残れそうに感じられるのは、喜びであり、気持ちが休まることである。しかし、危険は年々増大している。そして、今やMicrosoftが、私たちのコミュニティを明確にターゲットに据えた。自由に未来があることを当たり前だと思うことはできない。思ってはならないのだ。自由を保ちたければ、自由を守る備えを持たなければならない。

初出: "*Open Sources: Voices from the Open Source Revolution*", O'Reilly, 1999。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第2章
GNU 宣言

GNU 宣言は、GNU プロジェクトの発足時に参加と支援を呼びかけるために執筆された。最初の数年間は、プロジェクトの進展に合わせて細かく更新されていたが、現在はすでにほとんどの人が見ているものなので、変更せずに残したほうがよいように思われる。その後、他の用語を使えば避けられる誤解があることを学んだので、この数年間は、それらの誤解を解くための脚注を追加してきた。

GNUとは何か、Gnu's Not Unixだ！

GNU は、Gnu's Not Unix の略語で、使えるすべての人々にフリーで提供できるように私が開発している Unix 互換ソフトウェアシステムの名前である[*1]。私以外にも、数人のボランティアが私の仕事を助けてくれている。時間、資金、プログラム、機器について、多方面からの貢献をぜひともお願いしたい。

今までの間に、Lisp でエディタコマンドを記述できる Emacs テキストエディタ、ソースレベルデバッガ、yacc 互換パーサージェネレータ、リンカ、その他 35

*1 この用語は軽率だった。GNU システムを使うための許可を得るためにかかる費用はないというつもりだったが、ここに書いた文章はそのことを明確にしておらず、GNU のコピーは無料あるいはわずかな料金で頒布しなければならないと言っているように誤解されることが多かったが、そういうつもりではなかった。ちなみに、「宣言」は、あとの部分で、営利のために頒布サービスを提供する企業の可能性について言及している。今の私は、自由という意味での「フリー」と無料という意味での「フリー」を区別するために注意を払わなければならないということを学んでいる。フリーソフトウェアとは、ユーザーが頒布と変更の自由を持つソフトウェアである。ユーザーは、コピーを無料で入手する場合と有料で入手する場合がある。そして、資金がソフトウェアの改良を助けるなら、それはとても良いことである。重要なことは、コピーを持つすべての人々が同様の他の人々と協力する自由を持っているということである。

種類ほどのユーティリティが完成している。シェル（コマンドインタープリタ）は、ほとんど完成している。新しい移植性の高い最適化 C コンパイラは、自身をコンパイルできており、今年中にはリリースできる。出発点となるカーネルは完成しているが、Unix をエミュレートするためにはまだ多くの機能が必要になる。カーネルとコンパイラが完成したら、プログラム開発に適した GNU システムを頒布することが可能になる。テキストフォーマッタとしては TEX を使うつもりだが、現在は nroff を開発している。また、フリーの移植性の高い X Window System を使う予定でいる。そのあとで、ポータブルな Common Lisp、Empire ゲーム、スプレッドシート、その他多くのプログラムとオンラインドキュメントを追加する。最終的には、Unix システムが通常備えている役に立つすべての機能はもちろん、それ以上のものを提供したい。

GNU は、Unix プログラムを実行できるものになるが、Unix と同じものではない。他のオペレーティングシステムを使った経験に基づいて、便宜性を高めるあらゆる改良を加えていくつもりである。特に、より長いファイル名、ファイルバージョン番号、クラッシュに耐えられるファイルシステム、ファイル名補完のサポートを予定しており、おそらく端末に依存しないディスプレイサポートも実現し、最終的には、複数の Lisp プログラムと通常の Unix プログラムが画面を共有できる Lisp ベースのウィンドウシステムをサポートすることを検討している。システムプログラミング言語としては、C と Lisp の両方が使える。また、通信のために、UUCP、MIT Chaosnet、Internet プロトコルをサポートしようと考えている。

GNU は、まず、仮想記憶をサポートする 68000/16000 クラスのマシンをターゲットとするが、それはこのマシンがもっともシステムを実行しやすいマシンだからである。より小さなマシンで実行できるようにするための追加作業は、そのようなマシンで GNU を使うことを望む他の方にお願いしたい。

なお、とんでもない誤解を避けるために、このプロジェクトの名前として「GNU」という単語を使うときには、最初の「G」の部分を発音するようにしていただきたい[*2]。

---

*2　【訳注】動物のほうの gnu は、「ヌー」と発音する。

なぜ私がGNUを書かなければならないのか

私は、自分があるプログラムを好きなら、それが好きなほかの人と共有しなければならないということを重要な原則だと考えている。ソフトウェアの商売人たちは、ソフトウェアの共有を認めないことによって、ユーザーを分断し、支配しようとしている。私は、このような方法で他のユーザーとの絆を断ち切られることを拒否する。私は、良心にかけて、NDA（情報非開示契約）やソフトウェアライセンス契約に署名することができない。私は、長年に渡り、AIラボにおいてそのような傾向やその他の妨害工作を拒否して働いてきたが、AIラボはそのような思いから遠く離れた存在になってしまった。私には、自分の意思に反してそのようなことが行われている組織に留まることはできなかった。

不名誉なことをせずにコンピュータを使い続けるために、私は充分な量のフリーソフトウェアを集め、フリーではないソフトウェアを使わずに過ごせるような環境を作ることにした。そして、私がGNUを放棄せざるを得なくなるような法的根拠をMITに与えないために、AIラボを辞職した。

なぜGNUはUnix互換か

Unixは私の理想のシステムではないが、それほど悪いものでもない。Unixの基本機能は優れたものだと思っており、私はそれらを損なわずにUnixに欠けているものを補うことができると思っている。また、Unix互換システムは便利なので、他の多くの人々が採用しやすいはずだ。

GNUはどんな条件で頒布されるか

GNUはパブリックドメインではない。GNUを書き換え、再頒布することはすべての人に認められるが、頒布者がそれ以上の再頒布を制限することは禁止される。つまり、変更版の私有化は禁止されるということである。私は、GNUのすべてのバージョンを確実にフリーにしたいと考えている。

他の多くのプログラマが支援を申し出るのはなぜか

私は、GNU に目を輝かせ、支援を申し出る他の多くのプログラマに出会った。

多くのプログラマは、システムソフトウェアの商業主義に不満を感じている。商業主義は彼らにより多くの儲けを与えるかもしれないが、他のプログラマ一般を仲間というよりも敵ととらえることを強いる。プログラマの友情を成立させるための基本行動は、プログラムを共有することだが、現在の販売メカニズムは、一般に、プログラマがほかのプログラマを友として扱うことを禁止するように働いている。ソフトウェアの購入者は、友情と法の遵守の二者択一を迫られる。友情のほうが大切だと考える人が多いのは当然である。しかし、法を尊重する人間は、どちらを選んでも落ち着かない気持ちにさせられることが多い。彼らはひねくれて、プログラミングは単なる金儲けの手段だと考えるようになる。

私有プログラムではなく、GNU を開発、利用すれば、すべての人に対する友情を保った上で、法に従うこともできる。GNU は、ほかのプログラマを刺激し、共有の輪につないでいくための旗印を作る格好の例を提供する。GNU は、フリーではないソフトウェアを使うときには絶対に得られない調和の感覚を与えてくれる。私が話をしたプログラマの約半分にとって、これはお金に換えられない大切な幸福である。

どうしたらGNUに貢献できるか

私は、コンピュータメーカーに対して、マシンと資金の寄付を呼びかけている。また、個人に対して、プログラムと作業の寄付を呼びかけている。

マシンを寄付したときに期待できる効果の 1 つは、比較的早い時期にそのマシンで GNU が動作するようになるということである。ただし、そのマシンは完全ですぐに使えるシステムになっており、住宅地で使うことが認められていて、大規模な冷却やパワーを必要としないものでなければならない。

GNU の作業のためにパートタイムで寄与することを望んでいるプログラマは非常にたくさん見つかった。ほとんどのプロジェクトでは、そのようなパートタイムでの分散作業は、非常に調整困難なものになるだろう。独立に開発された部

品は、うまくフィットしないものである。しかし、Unixに代わるシステムを作るというこの特定の仕事では、そのような問題はない。完全なUnixシステムには数百種のユーティリティプログラムが含まれており、それぞれは別個にドキュメントされている。ほとんどのインターフェイス仕様は、Unix互換ということによって固定されている。個々のボランティアが特定のUnixユーティリティの互換プログラムを書き、Unixシステムでオリジナルの代わりに正しく動作させられれば、それらのユーティリティは完成時にも正しく動作する。マーフィー[3]が予期せぬ問題を持ち出してくるのを認めたとしても、それらのコンポーネントの結合は実現可能な仕事である（カーネルの開発には密接なコミュニケーションが必要なので、小規模でタイトなグループが従事することになるだろう）。

資金という形の寄付を受け取ったときには、フルタイムまたはパートタイムで働く人を雇うことができる。彼らの給与はプログラマの標準からすれば高いものにはならないが、私は、コミュニティ精神の構築がお金を稼ぐことに劣らず大切なことだと思っている人々を探している。この形での貢献は、生計を立てるために別の仕事をする必要がなく、GNUの仕事に全精力を掛けられる専従職員の実現につながるはずだ。

GNUがすべてのコンピュータユーザーの利益になるのはなぜか

GNUが完成したら、すべての人が空気と同じように優れたシステムソフトウェアをフリー[4]で手に入れられるようになる。

このことは、すべての人がUnixのライセンス料を節約できるというよりもはるかに大きな意味を持っている。それは、システムプログラミングの無駄な重複を避けられるということである。その分、技術の最先端の開拓に労力を掛けられるのだ。

---

*3　この部分は、悪いことが起きる可能性があるときには、その悪いことは必ず起きるとするユーモラスな法則、マーフィーの法則を指している。

*4　ここも、「フリー」の2つの意味の区別ということでは軽率だった。この文自体は、間違いではない。友人やInternetから、GNUソフトウェアのコピーを無料で入手することはできる。しかし、この文は誤解を招きかねない。

完全なシステムソースコードがすべての人に公開される。そのため、システムに変更を加えたいユーザーは、いつでも自由に変更を加えることができる。あるいは、他のプログラマや企業にお金を払って変更を加えてもらうことができる。ユーザーは、ソースコードを所有し、変更を加えるための権利を独占している個人プログラマや企業の言いなりにならなくて済むようになるのだ。

学校は、すべての学生にシステムコードを研究、改良することを勧めることによって、はるかに優れた教育環境を提供できるようになるだろう。ハーバード大学のコンピュータ研究所は、かつてソースが公開されていないプログラムをシステムにインストールしないことを方針としており、実際にいくつかのプログラムのインストールを拒んだことがあった。私は、このことに非常に大きな刺激を受けた。

最後に、システムソフトウェアを所有するのが誰で、所有権の対象になっているのがどの部分までなのかということを考える手間がなくなる。

コピーライセンスを含め、プログラムの使用に対して料金を要求する契約は、個人がどの程度（つまり、どのプログラムに）料金を払わなければならないかを明確にするために必要な煩雑な手続きのために、社会に莫大なコストを押し付けることになる。そして、全構成員にその遵守を強制できるのは、警察国家だけだ。高いコストをかけて空気を製造しなければならない宇宙ステーションについて考えてみよう。1リットル単位で呼吸に課金するのは公正かもしれないが、たとえ、すべての人に空気使用料を支払う余裕があったとしても、昼も夜もメーター付きのガスマスクをつけて暮らすことは、耐えられないことだろう。さらに、そのマスクを取り外すような人間がいないかどうかをテレビカメラで監視するに至っては論外である。マスクなどは捨てて、人頭税で空気プラントを運営したほうがよほどましである。

プログラマにとって、プログラムの全部または一部をコピーするのは、呼吸をするのと同じくらい自然なことであり、生産的なことである。コピーは自由でなければならない。

GNUの目標に対する簡単に反論できる異論

「フリーだということは、サポートを期待できないということだから、
誰もそんなものは使わないだろう」
「サポートを提供するための資金を得るために、プログラムを有料に
しなければならない」

人々が、サービスなしのGNUを無料で入手することよりも、GNUとサービスの組み合わせに料金を払うことを選ぶのだとすれば、GNUを無料で入手した人々に対してサービスだけを提供する企業は収益を上げられるはずである。

私たちは、実際のプログラミングの仕事という形を取るサポートと単なるアフターケアとを区別しなければならない。前者は、ソフトウェアベンダーからは期待できないものである。ある問題が多くの人々によって共有されていなければ、ベンダーはその問題を抱える人に対して諦めろと言うだろう。

サポートを受けられる体制が必要なら、必要なソースとツールをすべて揃える以外の方法はない。そうすれば、問題を解決できる人物を雇うことができる。特定の個人に振り回されることはなくなる。Unixの場合、ソースが高価なために、ほとんどの企業にとってそのようなことを検討する余地はない。GNUなら、それは簡単である。それでも、有能な人物がいないという可能性はあるが、そのような問題は、頒布条件を非難する理由にはならないだろう。GNUは、世界のすべての問題を消滅させるわけではない。そのうちのいくつかを消そうとしているのである。

一方で、コンピュータについての知識を一切持たないユーザーは、自分でも簡単にできることだが、やり方がわからないのでできないことについて、アフターケアを必要とする。

そのようなサービスは、アフターケアと修理のサービスを販売する企業によって提供することができるだろう。ユーザーがお金を払ってサービス付きの製品を手に入れるほうがよいと思っているということが事実であれば、無料で入手した製品のためのサービスも喜んで買うだろう。そのようなサービス会社は、品質と

価格で競争することになる。ユーザーは、特定の企業に縛られない。そして、サービスを必要としないユーザーは、サービスにお金を使わずにプログラムを使うことができなければならないだろう。

「宣伝なしで多くのユーザーを掴むことはできないので、そのためにプログラムを有料にしなければならない」
「無料で入手できるプログラムを宣伝してもしょうがない」

コンピュータユーザーにGNUのような存在を知らせるために無料または非常に安いコストで使える媒体には、さまざまな形のものがある。しかし、宣伝を打てば、より多くのマイクロコンピュータユーザーを動かせることは、事実かもしれない。もしそうだとすれば、有料でGNUをコピー、郵送するサービスを提供すると宣伝する企業は、宣伝料以上の収益を上げられるくらいに成功するだろう。だとすれば、宣伝から利益を得ているユーザーだけがそのサービスに料金を支払うことになるだろう。

一方、多くの人々が友人からGNUを入手し、そのような企業が成功しないとすれば、GNUの普及のために本当は宣伝など必要なかったのだということになる。自由市場の推進者が自由市場に判断を委ねたがらないのはなぜだろうか[*5]。

「私の会社は、競争力を確保するために、私有オペレーティングシステムを必要とする」

GNUは、オペレーティングシステムソフトウェアを競争の分野から取り除くだろう。この分野でライバルを出し抜くことはできないが、ライバルがこの分野であなたを出し抜くこともできない。あなたとライバルは、この領域では相互に

*5　FSFは企業ではなく、慈善団体だが、その資金の大半を頒布サービスから得ている。FSFに発注してコピーを得る人がまったくいなくなれば、FSFは活動することができなくなるだろう。しかし、だからといって、すべてのユーザーに料金の支払いを強制する私有物としての制限を設けることは正当化されない。すべてのユーザーの中の一部がFSFにコピーを発注すれば、FSFの活動は続けられる。だから、私たちはこのような形での支援をユーザーにお願いしている。あなたは、ご自分の役割を果たされただろうか。

協力しながら、他の分野でしのぎを削ることになる。オペレーティングシステムの販売を業務とする人からすれば、GNUは好ましくない存在だろうが、これはやむを得ないことである。他のことを業務とする人にとっては、GNUはオペレーティングシステム販売という高価な業務に首を突っ込まなくて済むようにしてくれるありがたい存在である。

私は、GNUの開発が多くのメーカーやユーザーからの献金によって支えられ、互いのコストを下げるようになることを期待している*6。

「プログラマの創造は、報酬に値しないのか」

報酬に値するものがあるとすれば、それは社会的な貢献である。創造は社会的貢献になり得るが、それは、社会が創造の結果を自由に利用できる限りにおいてである。プログラマが革新的なプログラムを作ることが報酬を受けるに値することなら、同じ理由で、それらのプログラムの利用を制限することは懲罰を受けるに値する。

「プログラマは、創造に対して報酬を要求してはならないのか」

破壊的な手段を使わない限り、仕事に対して報酬を求めることや収入を増やそうとすることは決して悪いことではない。しかし、現在のソフトウェア業界の慣行は、破壊を基礎に置いている。

プログラムの利用を制限することによってプログラムのユーザーから料金を徴収することは、破壊的である。その制限によって、プログラムが利用される回数や形態は少なくなってしまうだろう。これは、人類がプログラムから得る利益を小さくする。制限することが意図的な選択であれば、制限がもたらす有害な結果は意図的な破壊である。

よき市民たちがそのような破壊的な手段を使わないのは、誰もがそのようなことをすれば、相互破壊によって全員がより貧しくなってしまうからである。これ

*6　最近、コンピュータメーカー数社のグループが、GNU C コンパイラのメンテナンスを支援するための基金を設立した。

はカントの倫理学であり、黄金律である。私は、すべての人が情報を死蔵したらどのような結果がもたらされるかがわかるため、誰かがそのようなことをすれば悪いと考えざるを得ない。具体的に言えば、創造に報いてほしいという気持ちは、世界一般から創造性の全部または一部を奪う理由にはならない。

「プログラマは餓えてしまうのではないか」

この問いには、プログラマになることを強制される人などないという答え方があるかもしれない。私たちの大半は、街頭に立っておかしな顔をするだけでは金を稼ぐことはできない。しかし、だからと言って、街頭でおかしな顔をしていて餓えるような人生を強制されることはない。何か他のことをするまでである。

しかし、この応え方には、質問者の暗黙の前提を受け入れているという点で問題がある。その前提とは、ソフトウェアの所有権がなければ、プログラマは1セントも報酬を得ることができないだろうというものである。オールオアナッシングの考え方である。

プログラマが餓えない本当の理由は、それでもプログラミングによって報酬を得ることは可能だからである。単に、今ほど高い報酬が得られなくなるだけである。

コピーの制限は、ソフトウェアビジネスの唯一の基礎ではないが、ソフトウェアが儲かる最大の理由である。コピーの制限が禁止されたり、顧客たちから拒否されたりすれば、ソフトウェアビジネスは、現在はまだあまり見られない組織形態に移っていくだろう。どのようなビジネスにも、組織形態はいつでも無数にある。

おそらく、新しい基礎に移れば、プログラミングは現在ほど儲かる仕事にはならないだろう。しかし、これは変更に反対する議論ではない。営業員が現在の方法で給与を受けることは、正義にもとるものとはみなされない。プログラマも同じようにすれば、不正だとはみなされなくなるだろう（実際には、プログラマはそれでも営業員よりもかなり儲かる仕事であり続けるだろう）。

「人間には、自らの創造がどのように使われるかをコントロールする
権利があるのではないか」

「発想の使い道をコントロールする」ということは、他の人々の生活をコントロールすることであり、通常、それらの人々をより暮らしにくくするために使われる。

知的財産権の問題をじっくりと研究した人々（たとえば、法律家）は、知的財産に固有の権利は存在しないと言っている。政府が容認しているようなタイプの知的財産権は、特定の目的のために特別の法律によって作られたものである。

たとえば、特許制度は、発明者がそれぞれの発明の詳細を公開することを促すために設けられた。その目的は、発明者ではなく、社会を助けることにあったのだ。その当時、特許の17年という有効期限は、技術の最先端の進歩と比べて短いものだった。特許は製造業者の間だけの問題であり、それらの企業にとって、ライセンス契約にかかる費用や労力は生産体制を整えることと比べれば少額だったので、特許が大きな害になることはあまりなかった。特許を受けた製品を使うほとんどの個人にとって、特許が障害になることはなかった。

著作権の考え方は、古代には存在しなかった。当時の著者は、ノンフィクションの分野では、他の著者が書いたものをひんぱんにかなりの長さでコピーしていた。この習慣は役に立っており、多くの著者の作品がたとえ部分的にであっても生き残るための唯一の方法だった。著作権制度は、創作意欲を促すという目的をはっきりと示して作られたものである。著作権制度が作られた対象となった領域（印刷機だけで経済的にコピーを作れる書籍という領域）では、著作権制度はほとんど害をもたらさず、本を読むほとんどの個人の障害にはならなかった。

知的財産権は、それを認めることによって社会全体が利益を得ると社会が考えたために（その考えが正しかったかどうかは別として）認められたライセンス（免許、鑑札）に過ぎない。しかし、個々の状況について、私たちは問わなければならない。私たちは、そのようなライセンスを認めることによって本当により良くなるのだろうか。私たちは、個人にどのような行為を認めようとしているのだろうか。

今日のプログラムは、百年前の書籍とは事情が大きく異なる。プログラムをコピーするもっとも簡単な方法は、隣人から隣人に伝えていくことだという事実、プログラムが別個の存在であるソースコードとオブジェクトコードの両方を持つという事実、プログラムが読んで楽しむためにではなく、使われるものであると

いう事実がある。これらの事実を結合すると、著作権を強制する人物が、物心両面で全体としての社会に害を流すという状況が生まれる。法律が認めているかどうかにかかわらず、人はそのようなことをしてはならない。

「競争は、進歩を生む」

競争のパラダイムは、かけっこである。勝者を褒め称えることによって、誰もがより速く走ることを奨励する。本当にそのように機能しているときの資本主義は、良い仕事をしている。しかし、資本主義の擁護者は、資本主義がいつもそのように機能していることを当然と考えているというところで誤っている。勝者がなぜ褒め称えられるかを忘れ、手段を選ばず勝つことにだけ夢中になれば、他のランナーへの攻撃などの新戦略に出るかもしれない。ランナーが互いに殴り合っていたら、彼ら全員のタイムが下がるだろう。

秘密の私有ソフトウェアは、モラルとしては、殴り合うランナーと同じである。悲しいことに、私たちの唯一の審判は、このような争いに反対していないようだ。彼は、争いを規制するのみである（「10ヤード走るごとに、1発撃ってよい」）。彼は殴り合いを止めさせ、殴り合おうとしただけでもランナーを懲罰すべきだったのである。

「金が儲かるという動機がなければ、誰もプログラミングしなくなるのではないか」

実際には、金儲けの動機がまったくなくても、多くの人々がプログラムを書く。プログラミングは抑えがたい魅力だと感じる人々が一部にはおり、通常それはもっともプログラミングに優れている人々である。音楽で食べていく望みがまったくなかったとしても、音楽を止めないプロのミュージシャンが足りなくなることはない。

しかし、この問いは、よく尋ねられる質問ではあるが、そもそも前提が間違っているのである。プログラマに対する報酬は、減りはしても、決してなくならない。正しい問いは、あまり儲からなくなってもプログラムを書く人間はいるだろ

うか、とすべきである。そして、私の経験から言えば、間違いなくいる。

世界でもっとも優れたプログラマの多くは、10年以上もの長きにわたってAIラボに集まっていたが、他の職場にいればずっと多くの金額を得ていたはずである。彼らは、名声や評価など、金とは別の種類のさまざまな報酬を得ていた。そして、創造は面白く、それ自身が報酬なのである。

その後、彼らの大半は、もっと多くの金で同じ面白い仕事をするチャンスを与えられて、AIラボを去っていった。

この事実は、人は豊かさ以外の理由でプログラムを書くことがあることを示すが、しかし、もっとも豊かになるチャンスが与えられると、豊かになることを望み、要求するようになるということを示している。給与の安い組織は、給与の高い組織との競争では不利である。しかし、給与の高い組織が社会の反発を受けていれば、不利な競争をする必要はなくなる。

> 「私たちには、どうしてもプログラマが必要である。プログラマが隣人との協力を拒めと言うなら、私たちは彼らに従わざるを得ない」

このような要求に従わなければならないほど、プログラマを必要とすることなど決してない。「守りに何百万ドルかけても、1セントたりとも貢ぐな」[*7]である。

> 「プログラマは、何らかの方法で生計を立てなくてはならない」

短期的には、これは正しい。しかし、プログラマは、プログラムの使用権を売らなくても、さまざまな方法で生計を立てられる。使用権の販売という方法は、生計を立てる唯一の方法だからではなく、プログラマとビジネスマンにもっとも多くの金をもたらす方法だから、現在の習慣になっているだけである。見つける気になれば、他の方法は簡単に見つかる。たとえば、次のようなものが考えられるだろう。

---

*7　【訳注】建国直後、1800年前後のアメリカで浸透していたスローガン。当時多くのアメリカ船が北アフリカで海賊の被害にあっており、海軍力増強が叫ばれた。

- 新しいコンピュータを製造するメーカーは、新ハードウェアにオペレーティングシステムを移植するために報酬を支払う。
- 教育、アフターケア、メンテナンスサービスも、プログラマを雇う。
- 新しい発想を得た人々は、満足を得たユーザーに寄付を求めたり、アフターケアサービスを販売することによって、プログラムをフリーウェアとして流通させることができる。私は、すでにこのようにして成功した人物に会ったことがある。
- 類似したニーズを持つユーザーがユーザーズグループを結成し、会費を集めることにした場合、そのグループは、メンバーが使いたいと思うようなプログラムを書くプログラミング会社と契約を結ぶ場合がある。

ソフトウェア税を設ければ、あらゆる種類の開発のために資金を調達できる。

- コンピュータを購入するすべての人が、ソフトウェア税として価格の *n* パーセントを払わなければならないものとする。政府は、集めた税を NSF[*8] のような機関に渡し、ソフトウェア開発に使わせる。
- ただし、コンピュータの購入者が、自分でソフトウェア開発に寄付を行う場合には、税からその分を控除できる。購入者は、自分が選んだプロジェクトに寄付できる。おそらく、完成したソフトウェアを使いたいと思うようなプロジェクトを選ぶことが多くなるだろう。納税者は、支払うべき税金の合計を上限として、任意の額の寄付を控除できる。
- 合計税額は、納税者の投票によって（課税額によって 1 人あたりの票数に重みを付ける）決める。

すると、次のような結果になるだろう。

- コンピュータユーザーのコミュニティがソフトウェア開発を支援する。

---

*8　【訳注】National Science Foundation（米国国立科学財団）の略。

- このコミュニティが、必要な支援の水準を決める。
- 支払った金額がどのプロジェクトのために使われるかが気になるユーザーは、自分でそれを決められる。

長期的には、プログラムをフリーとすることは、飢餓のない世界、つまりすべての人がただ生活するだけのために必死に働かなくても済む世界に向かっての一歩になるだろう。人々は、立法業務、家族カウンセリング、ロボットの修理、小惑星の調査などの1週間に10時間の義務労働をこなしたら、プログラミングなどの面白い活動に自由に従事できる。プログラミングから生活の資を得られなければ困るようなことはなくなるのだ。

私たちは、社会全体が実際の生産のためにしなければならない仕事の量をすでに大幅に削減してきたが、その大半は労働者の楽しみには還元されていない。生産的な活動を行うためには、大量の非生産的な活動が必要とされるからだが、そうなってしまう主な原因は、官僚主義と無駄な競争にある。フリーソフトウェアは、ソフトウェア製作の分野で、このような活力の浪費を大幅に削減するだろう。技術が生産性の向上のために獲得したものが労働時間の現象に還元されるようにするためには、フリーソフトウェアを実現しなければならないのである。

初出: 第1稿は1984年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第3章
フリーソフトウェアの定義

特定のソフトウェアプログラムをフリーソフトウェアとみなすためにはどのような条件が満たされていなければならないか？　私たちはそれを明確に示すために、フリーソフトウェアの定義に修正を加え続けている。

「フリーソフトウェア（Free Software）」は、価格ではなく、自由の問題である。この概念を理解するためには、「フリー（無料）ビール」ではなく「フリースピーチ（言論の自由）」の「フリー」を考えていただかなければならない。

フリーソフトウェアは、ソフトウェアを実行、コピー、頒布、研究、変更、改良するユーザーの自由の問題である。より正確に言えば、ソフトウェアユーザーの４種類の自由を指している。

- **自由０：プログラムを任意の目的のために実行する自由。**
- **自由１：プログラムが動作する仕組みを研究し、自分のニーズに合わせて書き換える自由（そのためには、ソースコードへのアクセスが前提条件となる）。**
- **自由２：隣人を助けるためにコピーを再頒布する自由。**
- **自由３：プログラムを改良し、改良点を公開して、コミュニティ全体の利益を図る自由（そのためには、ソースコードへのアクセスが前提条件となる）。**

ユーザーがこれらすべての自由を持つとき、そのプログラムはフリーソフトウェアである。つまり、変更を加えたか否かにかかわらず、また頒布料金を徴収する

か否かにかかわらず、いつでもどこでも誰にでもコピーを再頒布する自由がなければならない。これらのことを自由にできるということは、許可を求めたり、許可料を払ったりする必要がないということである（それだけではないが）。

また、変更を加えた上で、そのようなものが存在することをいちいち断らずに、自分の仕事や遊びのためにプライベートにそれを使う自由も持っていなければならない。自分が加えた変更を公表する場合には、特定の誰かに、あるいは特定の方法で通知することなく、公開できなければならない。

プログラムを使う自由とは、開発者などの特定の主体とあとで交渉することを要求されずに、あらゆる用途のために任意のコンピュータシステム上であらゆる個人または組織がプログラムを利用する自由のことである。

コピーを再頒布する自由には、変更、未変更の両方のバージョンについて、バイナリ（実行可能形式）のプログラムとソースコードの両方が含まれていなければならない（実行可能形式のプログラムの頒布は、フリーオペレーティングシステムに簡便にインストールできるようにするために必要である）。バイナリ、実行可能形式を作るための手段がなければ実行可能形式を含まなくてもかまわないが、バイナリ形式を作る手段があれば、バイナリ形式を再頒布する自由は保証されなければならない。

自由1と3（変更を加える自由と、改良版を公開する自由）が意味を持つためには、プログラムのソースコードにアクセスできなければならない。そのため、ソースコードの公開は、フリーソフトウェアになるための必要条件である。

これらの自由を現実のものにするためには、誤ったことをしていない限り、これらの自由を取り消せないようにする必要がある。ソフトウェアの開発者が、問答無用で使用許可を取り消す権力を持つ場合、そのソフトウェアはフリーではない。

しかし、自由の核心に対して矛盾を来たさない限り、フリーソフトウェアの頒布方法にはある種の規制を設けてよい。たとえば、コピーレフトは、（非常に簡単に言えば）他者の自由を否定するような制限を加えてはならないというプログラム再頒布時の規制である。この規制は、自由と矛盾せず、自由を守る。

フリーソフトウェアのコピーを入手するためにお金を払ったかもしれないし、払わなかったかもしれないが、どのような手段でコピーを取得したのかにかかわらず、そのソフトウェアをコピー、変更する自由、さらにはコピーを販売する自

由さえ必ず保証される。

「フリーソフトウェア」は、「非営利」を意味しない。フリープログラムは、営利目的の利用、開発、頒布のために利用できなければならない。営利目的でのフリーソフトウェアの開発は、もう特別なことではなくなっている。そのような市販フリーソフトウェアは、非常に重要な意味を持っている。

変更版のパッケージ方法についても、変更版をリリースする自由を妨げる効果を持たない限り、規則を設けてよい。「プログラムをこのように入手したら、同じように入手できるようにしなければならない」という規則は、同じ条件から認められる（このような規則が、プログラムを公表するか否かの選択の余地を残していることに注意していただきたい）。変更版を頒布したときに、以前のバージョンの開発者がコピーを要求したら、それを送らなければならないというライセンスも認められる。

GNU プロジェクトは、すべての人々に対してこれらの自由を法的に保護するために「コピーレフト」を使っている。しかし、コピーレフトの規制外のフリーソフトウェアも存在する。私たちは、コピーレフトを使ったほうがよいという重要な理由があると考えているが、あるプログラムがコピーレフトの規制外であってもフリーソフトウェアなら、私たちはそれを使うことができる。

政府による貿易管理法規や経済制裁によって、プログラムのコピーを国際的に頒布する自由が制限されることがある。ソフトウェア開発者は、これらの制限を削減したり無視したりする権限はないが、プログラムの使用条件としてそれらを強制することを拒否できるし、しなければならない。こうすれば、ある政府の支配を受けていない人々や活動は、その政府が設けた制限の影響を受けない。

フリーソフトウェアについて論じるときには、「無償提供する」「無料で」のような用語を使うことを避けたほうがよい。これらの用語は、問題が自由ではなく、価格にあるかのような意味を含んでいる。「海賊版」のような用語は、よく使われるが、私たちの立場からは支持してほしくない考え方が体現されている。この種の用語については、本書の「避けるべき用語」の議論を参照していただきたい。私たちは、「フリーソフトウェア」のさまざまな言語への翻訳をまとめたリストも用意している。

最後に、このフリーソフトウェアの定義で記述されているような基準を解釈す

るためには、細心の注意が必要だということに注意していただきたい。私たちは、特定のソフトウェアライセンスがフリーソフトウェアライセンスの条件を満たすかどうかを判断するとき、用語の正確さとともに精神がこれらの基準に合っているかどうかも考慮している。ライセンスに不当な制限が含まれていれば、たとえそれがこれらの基準で予想された問題ではなくても、私たちはそのライセンスをフリーソフトウェアライセンスだとは認めない。また、ライセンスの要求内容の中には、私たちがそれを受け入れられるかを判断するために、コストの高い思考（弁護士との議論など）を必要とする問題を提起するものが含まれる場合がある。新しい問題に対して結論に達することがあれば、私たちはこの基準をひんぱんに更新し、特定のライセンスがフリーソフトウェアライセンスと認められるか否かの理由を簡単にわかるようにする。

特定のライセンスがフリーソフトウェアライセンスと認められるかどうかに関心のある読者は、私たちのライセンスリスト（http://www.gnu.org/licenses/license-list.html）を参照していただきたい。問題のライセンスがこのリストに含まれていない場合には、licensing@gnu.org に電子メールで問い合わせることができる。

初出: 第 1 稿は 1996 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。
"*Open Sources: Voices from the Open Source Revolution*", O'Reilly, 1999。

第4章
ソフトウェアが所有権者を持ってはならない理由

デジタル情報技術は、情報のコピーや変更を楽にすることを通じて世界に貢献している。コンピュータは、あらゆる人のためにコピーや変更を楽にする。

しかし、誰もが楽になることを望んでいるわけではない。著作権制度は、プログラムに「所有権者」を与え、その大半は自分以外の人々がソフトウェアの潜在利益を引き出すことを阻もうとする。彼らは、私たちが使っているソフトウェアをコピー、変更できる唯一の存在であろうとする。

著作権制度は、コピーの大量製造技術である印刷とともに成長してきた。著作権は、コピーの大量製造者のみに制限を加えてきたので、この技術にうまく適合していた。著作権は、本の読者から自由を奪いはしなかった。印刷機を持たない平均的な読者であれば、ペンとインクで本をコピー（書写）することは可能だったし、そのために告訴された読者はほとんどいない。

デジタル技術は、印刷機よりも融通が利く。デジタルな形態を持つ情報は、簡単にコピーして他者と共有できる。著作権のような制度がうまく適合しないのは、まさにこの融通性である。ソフトウェアの著作権を強制するために次第に陰険で過酷な基準が使われるようになりつつある理由は、ここにある。SPA（ソフトウェア出版協会）が繰り広げている次の４つの行為について考えてみよう。

- 所有権者に従わず、友人を助けるのは悪いことだという大規模なプロパガンダ。
- 同僚や仲間を通報する密告者の勧誘。

- 不正コピーを行っていないことの証明を強要されるようなオフィスや学校への抜き打ち検査（警察力の助けを借りる）。
- ソフトウェアをコピーしたことではなく、コピー機能を無防備な状態に放置して使用状況を検閲しなかったことに対して、MITのデビッド・ラマチーア[*1]のような人物を訴追すること（SPAの要求に基づいて、合衆国政府が行ったが、彼の訴追理由はコピーではない）。

これら4つの行為は、すべてかつてのソ連を思い起こさせるものである。ソ連では、すべてのコピー機は違法コピーを阻止するために見張り付きになっており、人々は秘密裏に情報をコピーして地下出版物として手渡ししなければならなかった。もちろん、違いはある。ソ連の情報統制の動機は政治的なものだったのに対し、アメリカの動機は利潤にある。しかし、私たちに影響を与えるのは、動機ではなく、行為である。情報の共有を阻もうとするあらゆる試みは、理由のいかんにかかわらず、同じ手法を取り、同じ過酷さを示す。

所有権者たちは、私たちの情報の使い方をコントロールする権力が自分たちに与えられていることについて、いくつかの論点を提出している。

用語

所有権者たちは、「海賊版」、「窃盗」などの誹謗中傷と「知的財産権」や「侵害」などの専門用語を使って、プログラムと物理的なものとの単純な同一視という思考様式を一般人に押し付けてくる。

物理的なものの所有ということについての私たちの考えや直感は、誰かからものを奪い取ることが正しいかどうかということである。これと何かのコピーを作ることとの間に直接的なつながりはない。しかし、所有権者たちは、いずれにしても所有権の適用を私たちに要求する。

---

*1　1995年1月27日、デビッド・ラマチーア事件は無罪となり、まだ控訴はされていない。

誇張

所有権者たちは、ユーザーが自分でプログラムのコピーを作ると、「危害」、「経済的損失」を蒙ると主張する。しかし、コピーを作ることは所有権者に直接的な影響を与えないし、誰も傷つけない。コピーを作った人物が、コピーを作る代わりに所有権者からもう1つ同じ物を買うようなことがない限り、所有権者は損失を蒙ることができない。

少し考えれば、ほとんどの人々がそのようなコピーを買ったりしないことはわかるだろう。にもかかわらず、所有権者たちは、すべてのユーザーがコピーを買っていたはずだとして「損失」を計算している。これは、よく言ったとしても誇張だろう。

法律

所有権者たちは、よく法律の現状を示し、苛酷な刑罰が存在すると私たちを脅す。このやり方には、現在の法律が、倫理についての疑問の余地のない考え方を反映しているという思考が透けて見える。しかし、それと同時に、私たちはこれらの刑罰が誰からも非難できない自然の摂理であるとみなすことを迫られているのだ。

この種の説得方法は、批判的思考に耐えることを目指していない。惰性的な思考回路を補強する役割を果たすだけである。

法律が正邪を決めるわけではないことは基本中の基本である。すべてのアメリカ人は、40年前、黒人がバスの前の方に座ることが多くの州で違法とされていたことを知っていなければならない。しかし、それが違法だったと声高に言うのは、人種差別主義者だけだろう。

自然権

プログラムの作者たちは、自分には自作プログラムとの間に特別なつながりがあることを主張し、さらに踏み込んで、プログラムに対する自分の希望や利権が

他の誰かのそれよりも（あるいは、自分以外のすべての人々のそれよりも）単純に重要だと言うことが多い（一般に、ソフトウェアの著作権を持っているのは作者ではなく、企業だが、私たちはこの違いを無視するように仕向けられている）。

作者は誰よりも重要であるということを倫理綱領として提案する人々に対しては、著名なソフトウェア作者である私自身がそれはベテンだと言うしかない。

しかし、一般の人々は、2つの理由から、自然権の主張に対して同情を感じるらしい。

1つは、物理的なものへの誇張された類推である。スパゲッティを作ったときに誰かがそれを食べてしまったら、私は自分が食べられなくなってしまうのでその人を非難するだろう。彼の行為は、彼にもたらした利益と同じだけ私に損害を与えている。スパゲッティを食べられるのは私たちの中のどちらかであり、問題はどちらかということである。倫理的なバランスは、私たちの間の最小限の区別で充分に覆る。

しかし、私が書いたプログラムにあなたが変更を加えても、直接的な影響を受けるのはあなたであり、私には間接的な影響しかない。あなたがあなたの友人にコピーを渡すかどうかは、私よりもあなたとあなたの友人に大きな影響を与える。私は、あなたにそのようなことをするなと命令する権力を持つべきではない。誰もがそんな権力を持ってはならない。

第2の理由は、人々が作者の自然権は私たちの社会に受け入れられている疑問の余地のない伝統だと教え込まれていることにある。

歴史的に言えば、真実は逆である。合衆国憲法が立案されたとき、作者の自然権という思想は、提案されたものの断固として退けられている。憲法が著作権制度を容認しているだけで、要求していないのはそのためであり、著作権が一時的なものでなければならないとしているのもそのためである。合衆国憲法は、著作権の目的は、作者への報酬ではなく、進歩の促進だとも述べている。確かに、著作権は作者にある程度の報酬を与え、出版業者にそれ以上の報酬を与えるが、彼らの行動を変えることを意図して作られている。

私たちの社会の本当の伝統のもとでは、著作権は自然権に割り込んできたものであり、公共の福祉に役立たない限り正当化できない存在なのである。

経済

ソフトウェアの所有権者たちに残された最後の論拠は、所有権がより多くのソフトウェアの生産を促すというものである。

他の論拠とは異なり、これは少なくとも問題に対して正当なアプローチを取っている。つまり、ソフトウェアのユーザーを満足させるという正当な目標に基づいている。そして、報酬が増えれば生産量が増えることは、経験的に明らかである。

しかし、この経済論には欠陥がある。まず、違いは、払わされる金額の多寡だけだという暗黙の前提を持ち込んでいる。また、私たちの望みが「ソフトウェアの生産」であって、ソフトウェアが所有権者を持つかどうかは無関係だという前提にも立っている。

これらの前提条件は物理的なものの生産における私たちの経験とうまく調和するので、人々はすぐにこれらを受け入れてしまう。たとえば、サンドイッチについて考えてみよう。同じサンドイッチは、無料でも有料でも手に入れることができる場合がある。その場合、違いは支払う金額だけである。サンドイッチは、買うかどうかにかかわらず、同じ味を持ち、同じ栄養価を持ち、一度しか食べられない。サンドイッチを所有権者から手に入れるかどうかは、そのあとであなたの手元に残る金額以外には直接的な影響を与えない。

物理的なものについては、いつもこれが当てはまる。所有権者を持つかどうかは、それが何かということに直接的な影響を与えず、手に入れてしまえば、それを使って何ができるかということにも影響を及ぼさない。

しかし、プログラムが所有権者を持つかどうかは、それが何かということ、購入したときにそのコピーで何ができるかということに非常に大きな影響を与える。違いは、金額の多寡だけではないのだ。ソフトウェア所有権制度は、ソフトウェアの所有権者が何かを生産することを促進するが、その何かは、社会が本当に必要としているものではない。そして、ソフトウェア所有権制度は、私たち全員に影響を与える目に見えない倫理的な汚染を垂れ流す。

社会は何を必要としているのだろうか。それは、市民が本当の意味で自由に扱える情報である。たとえば、単に実行するだけではなく、人々が読み、書き直し、修正し、改良できるプログラムがそれである。しかし、ソフトウェアの所有権者た

ちが流通させているものは、一般に研究、変更できないブラックボックスである。

社会は自由も必要としている。プログラムが所有権者を持つ場合、ユーザーは自分自身の生活の一部をコントロールする自由を失ってしまう。

そして何よりも、社会は、市民の自発的な協力精神を奨励する必要がある。ソフトウェア所有権者たちが隣人を自然な形で助けることを「海賊行為」だと言うなら、彼らは社会に属する市民の精神を汚しているのである。

私たちがフリーソフトウェアとは価格の問題ではなく、自由の問題だと言っているのは、そのためである。

所有権者たちの経済理論は誤っているが、経済は現実の問題である。書くことの楽しみや、尊敬、愛のために役に立つソフトウェアを書く人はいるが、彼らが書く以上の数のソフトウェアがほしいなら、資金が必要だ。

フリーソフトウェアの開発者たちは、10年前から資金を獲得するためのさまざまな方法を試してきて、一定の成果を収めている。誰かを金持ちにする必要はない。アメリカの平均世帯収入が35000ドルということは、その程度の収入がプログラミングよりも面白くないさまざまな仕事をするための動機として充分であることを証明している。

ある財団からの賞金によって不要になるまでの何年間か、私は自分が書いたフリーソフトウェアのカスタム拡張で生計を立てていた。それらの拡張は、順次標準リリースバージョンに追加され、最終的には一般人が使えるものになった。クライアントは、私がもっとも高い優先順位を持つと考えていた機能よりも、自分が必要とする機能のために私が働くことに対して報酬を支払ったのである。

フリーソフトウェア財団（FSF）は、フリーソフトウェア開発のための非課税慈善団体で、寄付以外に、GNU CD-ROM、Tシャツ、マニュアル、デラックスディストリビューションの販売（これらはすべてユーザーが自由にコピー、変更できる）によって基金を維持している。現在は、5人のプログラマとメールオーダー担当の3人のスタッフを抱えている。

一部のソフトウェア開発企業は、サポートサービスの販売によって資金を得ている。［1994年の本稿執筆時点で］50人前後の従業員を抱えるCygnus Support*2

は、活動の約15%がフリーソフトウェア開発だと見ている。ソフトウェア企業として、尊敬すべき数字である。

C言語用のフリーGNUコンパイラの開発では、企業数社が基金を設立、維持している。一方、Ada言語用のGNUコンパイラは、合衆国空軍の資金提供を受けた。空軍は、高品質コンパイラを手に入れるためのもっともコストのかからない方法としてこれを選んだのである［空軍の資金提供は数年前に止まったが、GNU Adaコンパイラは動作する状態になっており、メンテナンスは商業的に行われている］。

これらの例はどれもごく小さなものである。フリーソフトウェア運動自体、まだ小さいし、まだ若い。しかし、個々のユーザーに購入を強制せずに大規模な活動をサポートすることが可能だということは、アメリカのリスナーサポートラジオ*3の例が示している。

今日のコンピュータユーザーとして、あなたは私有プログラムを使っているかもしれない。あなたの友人がコピーを作ってくれと頼んだとき、断るのは誤りである。協力は、著作権よりも重要である。しかし、秘密の地下活動的な協力は、よき社会のためにはならない。人間たるもの、誇りを持ち、背筋を延ばして生きることを望むべきである。そしてそのためには、私有ソフトウェアに「ノー」を突きつけなければならない。

あなたは、ソフトウェアを使う他の人々とオープンかつ自由に協力する権利を持つ。あなたは、ソフトウェアの仕組みを研究し、学生たちにそれを教える権利を持つ。あなたは、ソフトウェアが壊れたときに、お気に入りのプログラマを雇って修正する権利を持っている。

あなたは、フリーソフトウェアの権利を持つ。

---

*2　Cygnus Supportは、その後も成功を続けたが、外部からの投資を受け入れてから貪欲になり、非フリーソフトウェアの開発を始めた。その後、Red Hatに買収され、Red Hatはそれらの非フリーソフトウェアの大半をフリーソフトウェアとして再リリースした。

*3　【訳注】listener-supported radio（聴取者からのお金によって成り立っているラジオ局）。

初出: 第 1 稿は 1994 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。
"*Open Sources: Voices from the Open Source Revolution*", O'Reilly, 1999。

第5章
名前にどういう意味があるのか

名前は意味を持つ。私たちがどのような名前を選ぶかは、私たちが言おうとしていることの意味を決める。不適切な名前は、人々に誤った考えを植え付ける。薔薇は、どんな名前で呼ぼうが、甘い香りがするかもしれない。しかし、薔薇をペンと呼ぶなら、それを使って書こうと思っていた人々はがっかりするだろう。ペンを「薔薇」と呼ぶなら、人々はそれが何の役に立つのかわからないだろう。私たちのオペレーティングシステムを「Linux」と呼ぶなら、システムの起源、歴史、目的について誤った考え方をもたらすことになるだろう。しかし、それを「GNU/Linux」と呼ぶなら、(詳しくはないものの) 正しい考え方が伝わるようになる。

しかし、私たちのコミュニティにとって、そんなことに意味があるのだろうか。人々がこのシステムの起源、歴史、目的を知っているかどうかは重要なのだろうか。重要なのだ。歴史を忘れる人間は、同じ歴史を繰り返して非難を受けるものである。GNU/Linux を中心として開発された自由な世界は、安全ではない。私たちをして GNU の開発に導いた問題は、完全に根こそぎにされたわけではなく、再び襲来する恐れがある。

私がこのオペレーティングシステムを「Linux」ではなく、「GNU/Linux」と呼んだほうがよい理由を説明すると、人々は、次のような反応をすることがある。

> GNU プロジェクトがこの仕事でクレジットを受けるだけの権利を持っていることは認めるが、人々がクレジットをつけるのを忘れたときに大騒ぎする意味はあるのだろうか。重要なのは、仕事が完成したということで、誰が仕事をしたかではないのではないか。肩の力を抜

き、仕事が成功したことに誇りを持つべきで、クレジットのことなどで騒ぐべきではないだろう。

おっしゃる通り、仕事が終わって、肩の力を抜いてもよい時期になっているなら、これは賢明な忠告と言えるだろう。本当にそうであれば、どれだけ良いことか。しかし、試練はまだたくさん残っており、未来を当然のこととして期待できる時期にはまだ達していない。私たちのコミュニティの強さは、自由と協力に立脚点を置くことにある。GNU/Linuxという名前を使うことは、人々にこの目標を思い出させ、他の人々にこの目標を伝えてもらうための手段である。

GNUのことを思い起こさなくても、優れたフリーソフトウェアを書くことは可能である。Linuxという名のもとでも、多くの優れた仕事がなされてきた。しかし、「Linux」という用語は、初めて登場して以来、協力の自由を支持するわけではない思想と結び付けられてきた。業界がこの名前を使う頻度が高まるにつれ、私たちはこの言葉とコミュニティ精神を結び付けるのが難しくなっていくだろう。

将来のフリーソフトウェアは、「Linux」ディストリビューション企業が便利さとパワーの名のもとにGNU/Linuxに非フリーソフトウェアを追加していく傾向から大きな試練を受けることになるだろう。すべての大手商用ディストリビューション企業がこれを行っており、完全にフリーなディストリビューションを作っている企業は1つもない。ほとんどのディストリビューション企業は、ディストリビューションの中の非フリーパッケージを明確に特定していない。それどころか、多くのディストリビューション企業は、非フリーソフトウェアを開発し、それぞれのシステムに追加している。一部のディストリビュータは、不当にもユーザーに対してMicrosoft Windowsと同様の自由しか与えない「per seatライセンス」[1]に基づく「Linux」システムを販売している。

人々は「Linuxの人気」の名のもとに非フリーソフトウェアを追加することを正当化する。つまり、自由よりも人気に高い価値を置いているのだ。これがオープンに認められている場合もある。たとえば、*Wired*誌によれば、*Linux Magazine*の編集者、ロバート・マクミランは、「政治的な判断からではなく、技術的な判断

・1　【訳注】接続クライアントによるライセンス。

からオープンソースソフトウェアに対する動きに火がつくはずだと感じている」。Caldera の CEO などは、ユーザーに対し、自由の目標を捨て、代わりに「Linux の人気」のために働くことを大っぴらに促している。

GNU/Linux システムに非フリーソフトウェアを追加すれば人気は上がるが、その人気とは、GNU/Linux と非フリーソフトウェアの組み合わせを使っている人々の数を意味する。しかし同時に、人気主義はコミュニティが非フリーソフトウェアを良きものとして受け入れ、自由の目標を忘れることを暗黙のうちに促すことである。道に留まっていられないほど速く運転しても意味がない。

非フリーの「アドオン」がライブラリやプログラミングツールだと、それがフリーソフトウェア開発者を陥れる罠になることがある。彼らが非フリーパッケージに依存するフリーソフトウェアを書いても、彼らのソフトウェアは完全な形でフリーシステムの一部になることはできない*2。

私たちのコミュニティが、この方向に進み続けるなら、GNU/Linux は、フリーコンポーネントと非フリーコンポーネントの寄せ集めに方向転換することを迫られるかもしれない。今から5年後、私たちがまだ多くのフリーソフトウェアを確保していることは間違いない。しかし、私たちが注意しなければ、ユーザーたちがほしがっている非フリーソフトウェアがなければほとんど使いものにならないシステムになっているかもしれない。そのような場合、私たちの自由を求める運動は失敗に帰する。

フリーの代替物をリリースすることが単純にプログラミングの問題なら、私たちのコミュニティの開発資源が増えれば、将来の問題を解決するのは簡単になるかもしれない。しかし、私たちはこれが難しくなるような障害に直面している。それは、フリーソフトウェアを禁止する法律である。ソフトウェア特許が加熱し、

*2　以前、Motif と Qt GUI ライブラリがこの手口で多くのフリーソフトウェアを罠にかけ、解決に何年もかかるような問題を生み出した。Qt 問題は、Qt がフリーになったために解決した。Motif 問題は、フリーの代替システム、LessTif にまだ磨きをかけなければならないので（ボランティアよ来たれ！）、完全に解決されたとは言えない。Sun の非フリー Java 実装と標準 Java ライブラリが、今同様の問題の原因となりつつあり、これらに代わるフリーソフトウェアを作ることが現在の GNU の大きな仕事の1つになっている。

DMCA[3]などの法律が、DVDの表示やRealAudioストリームの聴取などの重要な仕事のためのフリーソフトウェア開発を禁止するために行使されると、それらを使う非フリープログラムを拒否する以外に、特許で守られた秘密のデータ形式と闘う明確な手段はなくなってしまう。

これらの試練を乗り越えるためには、さまざまな形の努力が必要とされる。しかし、あらゆるタイプの試練に直面したときに何よりも大切なことは、協力の自由という目標を思い出すことである。私たちは、強力で信頼性の高いソフトウェアに対する単なる期待では、人々に大きな力を発揮させる動機としては弱いと考えている。私たちが必要としているのは、人々が自由やコミュニティのために闘うときに示すのと同じ種類の決意、何年も持続し、諦めない決意である。

私たちのコミュニティでは、この目標と決意は、主としてGNUプロジェクトから発している。FSFは、自由とコミュニティを闘い取る目標として掲げている組織である。「Linux」という表現を使う組織は、一般にこのようなことを言わない。「Linux」を標榜する雑誌には、一般に非フリーソフトウェアの広告が満載されている。非フリーアプリケーションで「Linuxをサポート」する企業もあれば、非フリーアプリケーションのセールスマンを招待する「Linux」ユーザーグループもある。私たちのコミュニティの人々が自由と決意についての考えを見つけられる主要な場は、GNUプロジェクトである。

しかし、その場にやってきた人々は、それが自分に関係のある問題だと思うのだろうか。

自分がGNUプロジェクトから生まれたシステムを使っていることを知っている人々なら、自分たちとGNUとの直接的な関係を理解できるだろう。彼らは、私たちの哲学に自動的に同意したりはしないだろうが、少なくとも、GNUの哲学には真剣に考えてみるだけの意味があると考えてくれるに違いない。それに対し、自分は「Linuxユーザー」で、GNUプロジェクトは「Linuxで役に立つツールを開発した」と考えている人々は、一般にGNUと自分との間に間接的な関係しか

•3　1998年に成立したデジタルミレニアム著作権法（DMCA）は、合衆国の著作権法を一新しようとするものである。DMCAには、著作権保護制度、公正利用、オンラインサービスプロバイダの義務に関連する五点に対処する条項が含まれている。DMCAの詳細については、第12章「著作権の誤解――一連の誤り」を参照していただきたい。

認めていない。彼らは、GNUの哲学に出くわしても、単純に無視してしまうかもしれない。

GNUプロジェクトは理想主義的であり、今日理想主義の側に立つ人々は大きな障害に直面している。支配的なイデオロギーは理想主義を「非現実的」と片付ける傾向を助長しているが、私たちの理想主義は極端なまでに現実的であり続けた。だからこそ、フリーのGNU/Linuxオペレーティングシステムが存在しているのである。このシステムを愛する人々は、このシステムが私たちの理想主義を現実化したものだということを知る必要がある。

「仕事」が本当に終わったら、そしてクレジット以外に失うものがなくなったら、クレジットなどなくなるに任せたほうが賢明だろう。しかし、私たちはまだその段階には達していない。しなければならない仕事をする人々を鼓舞するために、すでに成し遂げられたことを思い出してもらう必要がある。オペレーティングシステムをGNU/Linuxと呼ぶことによって、GNUを支援していただきたい。

初出: 第1稿は2000年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1の一部である。

第6章
「オープンソース」ではなく「フリーソフトウェア」と呼ぶべき理由

フリーソフトウェアは、どんな名前で呼ぼうが、同じ自由を与えるはずだが、どの名前を使うかは大きな違いを生む。異なる言葉は、**異なる思想を意味する。**

1998年に、フリーソフトウェアコミュニティの一部の人々が、自分のしていることを表すために、「フリーソフトウェア」という用語の代わりに「オープンソースソフトウェア」[*1]という用語を使い始めた。「オープンソース」という用語は、ただちに異なるアプローチ、異なる哲学、異なる価値と結び付けられ、ライセンスの許容基準さえ異なるものになってしまった。いくつかの実際のプロジェクトでは協力することができるだろうし、現にそうしているが、今日のフリーソフトウェア運動とオープンソース運動は、異なる思想と目標を持つ別個の運動となっている。

2つの運動の間の根本的な違いは、価値観、世界観にある。オープンソース運動にとって、ソフトウェアがオープンソースであるべきか否かという問題は、倫理的な問題ではなく、実用的な問題である。ある人物が語ったように、「オープンソースは開発の方法論である。フリーソフトウェアは社会運動である」　オープンソース運動では、非フリーソフトウェアは劣位の選択肢だが、フリーソフトウェア運動では、非フリーソフトウェアは社会問題であり、受け入れられるのはフリーソフトウェアだけである。

---

*1　http://www.opensource.org/

フリーソフトウェア運動とオープンソース運動の関係

フリーソフトウェア運動とオープンソース運動は、フリーソフトウェアコミュニティの中の2つの政治潮流のようになっている。

1960年代の急進派は、党派主義のそしりを受けた。組織は戦略の細部での不一致を理由に分裂し、互いに相手を敵として扱った。実際にそうだったかどうかは別として、少なくとも人々が彼らに対して持っていたイメージではそうだった。

フリーソフトウェア運動とオープンソース運動の関係は、その図式のちょうど逆である。私たちは基本原則では一致していないが、現実的な場面で推奨する方向は、多かれ少なかれ一致している。そのため、私たちは多くの個別のプロジェクトで共同作業をすることができるし、現にしている。私たちはオープンソース運動を敵だとは考えていない。敵は私有ソフトウェアである。

私たちはオープンソース運動に反対はしないが、彼らとひとまとめにされるのは困る。私たちは、彼らが私たちのコミュニティに貢献してきたことを認めるが、このコミュニティを作ったのは私たちであり、人々にはそのことを知っていただきたい。私たちの達成点は、彼らの価値や哲学ではなく、私たちの価値や哲学と結び付けて考えていただきたい。私たちは、異なる価値観を持つグループの陰に隠れるのではなく、自分の主張を聞いていただきたいと考えている。

だから、私たちが成し遂げた仕事や私たちが開発したソフトウェア（たとえば、GNU/Linuxオペレーティングシステム）を取り上げるときには、フリーソフトウェア運動に言及していただきたいのである。

2つの用語の比較

以下では、「フリーソフトウェア」という用語と「オープンソース」という用語を比較する。「オープンソース」という用語ではどんな問題も解決されず、実際には新しい問題さえ作ってしまうのはなぜかが明らかになるだろう。

曖昧さ

「フリーソフトウェア」という用語には曖昧さという問題がある。「価格ゼロで入手できるソフトウェア」という意図せぬ意味と「ユーザーに自由を与えるソフトウェア」という意図した意味が同じ程度に伝わる。私たちは、フリーソフトウェアのより正確な定義を発表してこの問題に対処した（第3章「フリーソフトウェアの定義」を参照）が、これは完全な解決方法ではない。定義によってこの問題を完全に取り除けるわけではない。曖昧さのない正しい用語があれば、そしてその用語に他の問題がなければ、そのほうがよいだろう。

残念ながら、英語の代替用語候補は、どれもそれぞれの問題を抱えている。私たちは、人々が提案してくれたさまざまな新用語を検討したが、そちらに切り替えたほうがよいとはっきり言える「正しさ」を持ったものはなかった。「フリーソフトウェア」に代わる用語として提案されたすべての用語は、同様の、あるいはより大きい意味上の問題を持っていた。「オープンソースソフトウェア」もそのような用語の1つである。

オープンソースイニシアティブが公表している「オープンソースソフトウェア」の公式定義は、私たちのフリーソフトウェアの定義に非常に近い。しかし、いくつかの点で少し緩やかになっており、ユーザーに許容できない制限を加えていると私たちなら考えるようなライセンスも彼らのもとでは受け入れられている。しかし、「オープンソースソフトウェア」という表現が明らかに持つ意味は、「ソースコードを見ることができる」ということである。これは、フリーソフトウェアよりもはるかに弱い基準である。オープンソースソフトウェアにフリーソフトウェアは含まれるが、Xvのような半フリープログラムやオリジナルライセンスのもとでのQt（QPL以前）のようなより私有性の強いプログラムも含まれる。

「オープンソース」という用語があからさまに持つ意味は、その推進者が意図している意味とは異なる。そのため、ほとんどの人々は彼ら推進者たちが推進しようとしていることを誤解している。作家のニール・スティーブンソンは、「オープンソース」を次のように定義している。

> Linuxは、誰もがソースコードファイルのコピーを入手できるという

> 単純な意味で「オープンソース」ソフトウェアである。

私は、彼がわざと「公式」の定義を拒絶したり、異論を差し挟んだりしたわけではないと思っている。彼は、英語の慣行に従って、用語の意味を考えたのだろう。カンザス州も、同様の定義を公表している。

> OSS（オープンソースソフトウェア）を利用すること。OSSとは、ソースコードが無料で公開されているソフトウェアである。ただし、そのコードで何をすることが認められているかについては、個々のライセンス契約によってまちまちである[2]。

もちろん、オープンソース陣営の人々は、私たちが「フリーソフトウェア」の定義について行ったのと同じように、用語の正確な定義を公表して、この問題に対処しようしてきた。

しかし、「フリーソフトウェア」の説明は単純である。「フリービールではなく、フリースピーチだ」という考え方を理解した人は、もう二度と間違えないだろう。「オープンソース」の正しい意味を説明し、自然な定義が誤っている理由を明確に示す簡明な方法はない。

自由に対する恐れ

「オープンソースソフトウェア」という用語を使う最大の論拠は、「フリーソフトウェア」では一部の人々を不安にさせるというものである。それは事実である。便利さとともに自由について、倫理的な問題について、責任について語ることは、無視したくなるようなことについて考えることを人々に求めることである。これは不安を招き、一部の人々は考えたくないと言うだろう。しかし、私たちがこれらのことについて語るのを止めれば、社会が住みやすくなるというわけではない。

フリーソフトウェアの開発者たちは、数年前にこの不安げな反応に気づき、それ

---

[2] 【訳注】http://www.kshs.org/government/records/electronic/electronicrecordsguidelines.htm

を避ける方策を探り始めた。彼らは、倫理や自由について沈黙し、特定のフリーソフトウェアが目の前に持つ実際的な効果だけを語れば、特定のユーザー、特に企業に対してより効率よくソフトウェアを「販売」できることに気づいた。「オープンソース」という用語は、これをより大々的に行うために、「企業が受け入れやすいものにするために」考え出されたものである。オープンソース運動の考え方と価値観は、この判断に由来している。

このアプローチは、文字通り効率的であることを証明した。今日、多くの人々は、純粋に実際的な理由から、フリーソフトウェアへの切り替えを行っている。その範囲内では、それは良いことだが、私たちがしなければならないことはそれだけではないのだ！　ユーザーをフリーソフトウェアに誘うことがすべてではない。それは第一歩に過ぎないのだ。

これらのユーザーは、遅かれ早かれ、何らかの実際的なメリットのために私有ソフトウェアに戻っていく。そのような誘惑を提供しようと躍起になっている企業が無数にあるのに、ユーザーがそれを拒否する理由があるだろうか。フリーソフトウェアが文字通り与えてくれる**自由の価値**を学んでいない限り、そのような理由はない。この思想を普及させるのは、私たちの義務である。そして、そのためには自由について語らなければならない。企業に対してある程度「沈黙を守る」アプローチはコミュニティにとって有益に働く可能性があるが、私たちは自由についての議論もたくさんしなければならない。

現在のところ「沈黙を守る」戦術はあふれているが、自由についての議論は足りない。フリーソフトウェアにかかわるほとんどの人々が自由についてあまり語らない。通常、その理由は「企業に受け入れられやすくするため」である。ソフトウェアディストリビュータは、特にこのパターンを顕著に示している。GNU/Linux オペレーティングシステムの一部のディストリビューションは、基本フリーシステムに私有パッケージを追加し、このことを自由からの一歩後退ではなく、メリットと考えるようにユーザーを誘惑している。

私たちは、フリーソフトウェアユーザーの流入に追いついておらず、コミュニティに参加すると同時に、自由と私たちのコミュニティについての理解も得られるような体制を作れないでいる。非フリーソフトウェア（最初に人気を掴んだとき

のQtのようなもの）や部分的にフリーではないオペレーティングシステムディストリビューションが肥沃な土壌を見出せている理由は、そこにある。今「フリー」という単語を使うのを止めるのは誤りである。私たちは自由についてもっと少なくではなく、もっとたくさん話す必要がある。

「オープンソース」という用語を使っている人々が私たちのコミュニティにより多くのユーザーを引き込むなら、それは貢献だと言えるが、私たちフリーソフトウェア陣営の人間は、それらのユーザーの視界に自由の問題が入ってくるようによりいっそう働かなければならない。私たちは、今まで以上に大きな声で、「それは、フリーソフトウェアで、あなたに自由を与えてくれるんです」と言わなければならない。

商標は役に立つか

「オープンソースソフトウェア」陣営の人々は、この用語が誤って使われないようにするためにと称して用語の商標化を試みたが、1999年に却下されてこの目論見は失敗に帰した。つまり、「オープンソース」の法的地位は、「フリーソフトウェア」の法的地位と同じである。フリーソフトウェアという用語の使用については、法的な規制はない。私は、公式定義に一致していないにもかかわらず、「オープンソース」を称しているさまざまな企業のソフトウェアパッケージについての報告を聞いている。いくつかの例については、私自身が直接目にしている。

しかし、オープンソースが晴れて商標になっていれば、大きな違いはあっただろうか。必ずしもそうとは言えないだろう。

企業は、明示的に「オープンソースソフトウェア」という表現を使わなくても、プログラムがそうなんだという印象を与えるような発表をしている。たとえばIBMは、公式定義に合致しないあるプログラムについての発表で次のように表現している。

> 「オープンソースコミュニティでは一般的なことですが、……テクノロジーのユーザーもIBMと協力していくことができます……」

この表現は、そのプログラムが「オープンソース」だと言っているわけではないが、読者の多くはそこまで細かいところには気づかない（IBMがこのプログラムをフリーソフトウェアにしようとして真摯に努力していたこと、その後、プログラムがフリーソフトウェアにも「オープンソース」にもなるような新しいライセンスを採用したことを付記しておくべきだろう。しかし、発表がなされた時点では、このプログラムはどちらの定義も満足させられないものだった）。

そして、Cygnus Solutionsのような例もある。Cygnusは、フリーソフトウェア企業になることを目的として設立されたが、その後私有ソフトウェアの方向にも（いわゆる）枝分かれしていった会社で、ある私有ソフトウェア製品について次のように宣伝していた。

> 「Cygnus Solutionsは、オープンソース市場のリーダーであり、
> [GNU/]Linux市場に2つの新製品を送り出したところです」

IBMとは異なり、Cygnusはこれらのパッケージをフリーソフトウェアにしようとはしていなかったし、これらのパッケージは、フリーソフトウェアの基準に近づいてきてさえいない。しかし、Cygnusは実際にこれらが「オープンソースソフトウェア」だと言っているわけではない。疑い深くない読者にそのような印象を与えるようにこの用語を使っているだけである。

このような事例から見ても、商標化によって「オープンソース」という用語が与える誤解を完全に避けることはできなかっただろうと考えられる。

「オープンソース」の誤解（？）

オープンソースの定義は充分に明解であり、ごく普通の非フリープログラムがオープンソースの基準を満たさないのは明らかである。だから、「オープンソース企業」は、フリーソフトウェア（あるいはそれに近いもの）を製品としている企業というように考えているのではないだろうか。残念ながら、多くの企業は、この言葉に別の意味を与えようとしている。

1998年8月に開催された「Open Source Developers Day」（オープンソース開

発者の日）という会議では、招待されたいくつかの市販プログラム開発企業は、自社製品のごく一部だけをフリーソフトウェア（あるいは「オープンソース」）にしようとしていると語った。それらの企業の重点は、このフリーソフトウェアのユーザーに販売する私有アドオン（ソフトウェアやマニュアル）の開発にある。彼らは、一部の資金をフリーソフトウェア開発に提供しているのだから、このような活動を私たちのコミュニティの一部の仕事として正当なものとして見るよう求めてきた。

実際には、これらの企業は、フリーソフトウェアと何らかの関係を持っているから、あるいは何らかのフリーソフトウェアのメンテナンスも行っているからという理由で、それぞれの私有ソフトウェア製品（つまり、それらは「オープンソースソフトウェア」ではないにもかかわらず）に「オープンソース」の見栄えのよいマークを付けようとしているのである（ある企業の創設者は、その会社がサポートしているフリーパッケージに対してコミュニティが許容できる範囲内で最小限の労力しか割かないときわめてはっきり明言した）。

多くの企業が以前からフリーソフトウェアの開発に貢献してきた。そのうちの一部の企業は主として非フリーソフトウェアを開発していたが、2つの活動は別個のものだった。そこで、私たちは彼らの非フリー製品を無視し、フリーソフトウェアプロジェクトの部分で協力することができた。そして、彼らのその他の活動について云々せずに、彼らのフリーソフトウェアへの貢献に対して率直に感謝することができた。

しかし、これらの新しい企業は私たちの活動の妨げになるので、彼らに対して同じ態度を取ることはできない。これらの企業は、一般に対してすべての活動が一貫したものであるかのように見せかけようと企んでおり、私たちに対して本当の貢献に対する敬意と同等の敬意を非フリーソフトウェアに対して示すよう望んでいる。もちろん、非フリーソフトウェアは、本当の貢献ではない。彼らは、私たちが自分たちに対して温かいファジーな感情を持つことを期待し、私たちがファジーな気分でフリーソフトウェアの認定を下すことを期待しているのである。

このような欺瞞的な行為でも、「フリーソフトウェア」の名のもとに行われていれば、それほど有害ではないだろう。しかし、企業はそのようなときに「フリーソフトウェア」という用語を使おうとはしない。おそらく、理想主義と結び付い

ているこの言葉が不適切に感じられるのだろう。「オープンソース」という用語がこのような行為の扉を開いたのである。

1998年末に開催された「Linux」と呼ばれるオペレーティングシステム専門のあるトレードショウで、講演者として立ったのは、著名なソフトウェア企業の役員だった。おそらく彼は、その企業がそのシステムを「サポート」することを決めたために招待されたのだろう。残念ながら、彼らの「サポート」の形態は、そのシステムのもとで動作する非フリーソフトウェアのリリースだった。つまり、私たちのコミュニティに貢献するのではなく、私たちのコミュニティを市場として利用しようとしたのである。

彼はこう言った。「私たちが製品をオープンソースにすることは考えられないが、おそらく『社内』オープンソースにすることになるだろう。顧客サポートのスタッフにソースコードへのアクセスを認めれば、彼らは顧客のためにバグをフィックスできる。そして、私たちはより良い製品とより良いサービスを提供できる」(彼の言葉を書き留めておいたわけではないので、これは正確な引用ではないが、要点としては誤っていないはずである)。

聞いていた人々はあとで私に「彼は、勘違いしているだけだ」と言ったが、本当にそうだろうか。彼は何を勘違いしていたのだろうか。

彼は、オープンソース運動のポイントを見誤っていない。この運動は、ユーザーは自由を持つべきだとは言わず、より多くの人々がソースコードを見られるようにして改良を助けられるようにすれば、開発の高速化や向上に役立つと言っているだけである。この役員氏は、このポイントを完全に掴んでいた。ユーザーを含む形でそのアプローチを完全に遂行するのは気乗りがしないので、部分的に社内だけで採用しようとしているのである。

彼が掴んでいないのは、「オープンソース」が奨励しようとしているポイントではない。ユーザーは自由という権利を持つというポイントである。

自由の思想の普及は大きな仕事である。この仕事はあなたの助けを必要としている。私たちがその仕事の遂行に役立つように、GNUプロジェクト内で「フリーソフトウェア」という用語にこだわっているのはそのためである。自由とコミュニティが自分にとって重要だと（これらがもたらしてくれる便宜性だけが重要な

のではなく）考えているのなら、私たちとともに「フリーソフトウェア」[3]という用語を使う立場に立っていただきたい。

初出: 第 1 稿は 1998 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

[3] ジョー・バーは、この問題についての自分の考え方を述べた「Live and let license」という論文を書いている。

第7章
大学勤務のプログラマがフリーソフトウェアをリリースする方法

フリーソフトウェア運動に携わる私たちは、コンピュータユーザーが自分で使うソフトウェアの変更、再頒布の自由を持つべきだと考えている。フリーソフトウェアの中のフリーは、自由である。ユーザーが、ソフトウェアを実行、変更、再頒布する自由を持っていることを意味している。フリーソフトウェアは学問の進歩に貢献するが、非フリーソフトウェアは貢献しない。そのため、大学は、学者たちに仕事の成果を出版することを奨励するのと同じように、学問の進歩のためにフリーソフトウェアを奨励すべきである。

残念ながら、多くの大学の管理者たちは、ソフトウェア（および科学）に対して、金儲け主義を示している。彼らは、人類の学問の進歩に貢献するチャンスとしてではなく、収入のチャンスとしてプログラムを見ているのである。フリーソフトウェアの開発者たちは、ほとんど20年近くもこの傾向と闘ってきた。

1984年にGNUオペレーティングシステムの開発を始めたとき、私がまずしたことは、MITの職を辞することだった。それは、MITのライセンス担当がGNUのフリーソフトウェアとしてのリリースを妨害できないようにすることを意図した行動だった。私は、GNUプログラムのすべての変更バージョンも必ずフリーソフトウェアになるようなライセンスを課するというアプローチを計画しており、これはその後GNU一般公開使用許諾書（GNU GPL）に発展したが、それはMITの管理者に使用許可を乞い願うようなことをしたくなかったからである。

その後、大学勤務の支持者たちが、ソフトウェアを売り物としか考えていないような管理者との付き合い方をアドバイスするためにフリーソフトウェア財団（FSF）をたびたび訪れるようになった。その中でも特に優れていて、専用の資金

を得ているプロジェクトにも応用できる方法は、GNU GPL のもとでリリースされている既存のプログラムを基礎にして仕事を組み立てていくというものである。そして、しばらくしてから管理者にこう言うのである。「私たちは、GNU GPL 以外の方法ではこの変更バージョンをリリースすることができません。他の方法を使えば、著作権侵害になります」　目の前のドルマークが消えてしまった以上、通常彼らはそれをフリーソフトウェアとしてリリースすることに同意する。

資金提供者に助けを頼むこともできる。NYU のグループが合衆国空軍の資金で GNU Ada コンパイラを開発したとき、その契約は、開発結果として得られたコードを FSF に寄付することを明示的に要求していた。先に資金提供者との契約をまとめた上で、大学の管理者たちに礼儀正しく再交渉が不可能な契約を示すのである。大学の管理者たちは、契約がないくらいならフリーソフトウェア開発の契約を結ぼうとするので、その契約に賛成する可能性が高い。

何をするにしても、問題は早めに提出しよう。プログラムが半分完成する前がよい。この時点では、大学はまだあなたを必要としているので、あなたは厳しい態度に出られる。プログラムをフリーソフトウェアにする条件で書くことに同意したら（そして、あなたによるフリーソフトウェアライセンスの選択に同意したら）、プログラムを完成させて使えるものにすると管理者に告知するのである。そうでなければ、論文を書ける程度までプログラムを開発し、リリースできるだけのバージョンは作らない。大学にクレジットをもたらすフリーソフトウェアパッケージを作るか、何も手にしないかの選択を迫られれば、大学の管理者とて普通なら前者を選ぶものである。

すべての大学が金儲け主義に走っているわけではない。テキサス大学は、開発されたすべてのソフトウェアがデフォルトで GNU 一般公開使用許諾書管理下のフリーソフトウェアとしてリリースされるようにするという方針を持っている。ブラジルの UNIVATES[*1] とインドのハイデラバードのインド情報科学研究所[*2] は、ともに GPL のもとでソフトウェアをリリースする方針を採用している。まず学部の支援体制を発展させれば、大学全体にその方針を採用させることができるか

*1　【訳注】http://www.univates.br/sections.php?op=viewarticle&artid=600
*2　【訳注】http://www.iiit.net/ltrc/

もしれない。原則の１つとして問題提起をしよう。大学は、学問を進歩させる使命を持つのか、自己保存だけを目的とするのか。

どのようなアプローチを取ろうが、それはフリーソフトウェア運動と同様に、倫理的な視点からの方針とその決定を助ける。公共の倫理を論ずるには、ソフトウェアはすべての人々に対してフリー（自由という意味で）でなければならない。

多くのフリーソフトウェア開発者は、ソフトウェアをフリーにする狭い意味で現実的な理由を公言している。彼らは、ソフトウェアを強力で信頼性の高いものにする都合のよい手段として、他者にソフトウェアの共有、変更を許容することを擁護しているのである。もしそのような価値がフリーソフトウェアを開発するための動機となっているのなら、それは結構なことであり、あなたの貢献に感謝したい。しかし、大学の管理者たちがプログラムを非フリーにしようとあなたを誘惑したときに、そのような価値では揺らぎのないしっかりとした足場にはならないだろう。

たとえば、管理者側が「われわれが集めてきたお金を使えばもっと強力で信頼性の高いものになるはずだ」と言ったとする。この主張は結果的に真実になるかもしれないし、そうでないかもしれないが、前もって反証することは難しい。管理者側は、「学術目的に限り無料」でコピーを提供するライセンスを提案するかもしれないが、これは一般人に対して自由を与えないと宣告する内容である。そして、管理者側は、「こうすれば学術機関の協力が得られる、それこそ君が必要としているものではないのか」と言うだろう。

「実利的」な価値観を出発点とするなら、袋小路に追い込まれてしまって、この提案を拒絶するのはほとんど不可能だろう。しかし、倫理的、政治的な価値に立脚すれば、提案を拒絶するのは簡単である。ユーザーの自由を犠牲にしてプログラムを強力で信頼性の高いものにしたからといって何の良いことがあろうか。自由は、学術機関内とともに、学術機関の外にも適用されるべきではないだろうか。あなたの目標の中に自由とコミュニティというものがあれば、答えは明白である。フリーソフトウェアがユーザーの自由を尊重するのに対し、非フリーソフトウェアはユーザーの自由を否定する。

コミュニティの自由がその一員である自分に懸かっているということを自覚する以上にあなたの決意を強めるものはない。

初出: 第 1 稿は 2002 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第8章
フリーソフトウェアの販売

多くの人々は、GNUプロジェクトの精神では、ソフトウェアのコピーを頒布するために料金を徴収してはならない、あるいは徴収する額はできる限り少なくしなければならない（経費を回収できる分だけ）ものだと考えている。

しかし実際には、私たちは希望するだけの額で、あるいは販売可能な最高額でフリーソフトウェアを再頒布することを奨励している。そう聞いて驚いた方は、続きを読んでいただきたい。

「フリー」という単語は、一般的に妥当とされる意味を2つ持っている。自由と無料の両方を指すことができるのである。私たちが「フリーソフトウェア」と言うとき、私たちは価格のことではなく、自由のことを指している[*1]。具体的に言えば、フリーソフトウェアとは、有料であれ無料であれ、ユーザーがプログラムを実行、変更、再頒布する自由を持つプログラムのことである。

フリーソフトウェアは、無料で頒布される場合もあれば、かなり高額で販売される場合もある。多くの場合、同じプログラムが異なる場所から両方の形で頒布される。しかし、フリーソフトウェアは、ユーザーが使う自由を持っているので、価格にかかわらずフリーである。

非フリープログラムは通常高い価格で販売されているが、店が無料でコピーをくれる場合もある。だからと言って、そのプログラムはフリーソフトウェアではない。有料であれ無料であれ、そのプログラムはユーザーに自由がないので非フリーである。

---

*1 「フリービール（無料ビール）」のフリー（無料）ではなく、「フリースピーチ（言論の自由）」の「フリー」（自由）だと覚えていただきたい。

フリーソフトウェアは価格の問題ではないので、価格が安くても、よりフリーであるとか、フリーに近いということにはならない。フリーソフトウェアのコピーを再頒布するとき、かなり高額の料金を徴収して、ちょっとした金儲けをしたほうがむしろよい。フリーソフトウェアの再頒布は、正当で良い行為である。再頒布してくれるなら、そこから利益を得たほうがなおよい。

フリーソフトウェアはコミュニティのプロジェクトであり、フリーソフトウェアの恩恵を蒙るすべての人は、コミュニティの構築に貢献する方法を探さなければならない。ディストリビュータの場合、そのための方法は、収益の一部をフリーソフトウェア財団などのフリーソフトウェア開発プロジェクトに寄付することである。開発資金を提供すれば、フリーソフトウェアの世界を前進させることができる。

フリーソフトウェアの頒布は、開発資金を調達するチャンスなのである。チャンスを見逃してはならない。

資金調達に貢献するためには、プラスアルファが必要である。価格が低すぎれば、開発を支援するための分は残らない。

頒布価格を上げると一部のユーザーを傷つけることになるか

頒布価格を高くすると、フリーソフトウェアがあまりお金を持っていないユーザーからは手の届かないものになると心配する人がいる。私有ソフトウェアで価格を高くすればその通りになるだろう。しかし、フリーソフトウェアは違う。

その違いとは、フリーソフトウェアには自然に普及する傾向があるということと、入手方法がいくつもあるということである。

ソフトウェアの世界の死蔵家たちは、標準価格を支払わずにユーザーが私有プログラムを実行することを阻止するために汚い手を尽くす。標準価格が高ければ、一部のユーザーはそのプログラムを使いにくくなるだろう。

フリーソフトウェアでは、ユーザーはソフトウェアを使うために頒布価格を支払う必要はない。コピーを持っている友人からプログラムをコピーすることができるし、ネットワークアクセスを持っている友人の助けを借りることもできる。ある

いは、数人のユーザーが共同で1枚のCD-ROMを購入し、それぞれがソフトウェアをインストールすることもできる。ソフトウェアがフリーであれば、CD-ROMの価格が高くても、大きな障害にはならない。

頒布価格を上げるとフリーソフトウェアを使いたい気持ちの足を引っ張ることになるか

よく示されるもう1つの懸念は、フリーソフトウェアの人気である。人々は、頒布価格が高くなるとユーザー数が減るだろうとか、逆に価格が低ければユーザーが使う気になるだろうと考えるのである。

私有ソフトウェアではその通りだが、フリーソフトウェアは違う。コピーの入手方法がたくさんあるので、頒布サービスの価格が人気に与える影響はそれほど大きくはない。

長期的には、フリーソフトウェアを使う人がどこまで多くなるかは、フリーソフトウェアで何ができるかとフリーソフトウェアがどのくらい簡単に使えるかによって決まる。フリーソフトウェアではやりたい仕事全部をこなせないというのなら、多くのユーザーが私有ソフトウェアを使い続けるだろう。つまり、長期的にユーザー数を増やしたいのなら、何よりも先にフリーソフトウェアをもっと開発しなければならないということである。

そのためのもっとも直接的な方法は、必要なフリーソフトウェアやマニュアルを自分で書くということである。しかし、書くよりも頒布をしようという場合には、フリーソフトウェアを書くほかのプログラマのために、資金を提供して支援することが最良の方法となる。

「ソフトウェアの販売」という用語も誤解を招く

厳密に言えば、「販売」とは商品をお金と交換することである。フリープログラムの販売は正当な行為であり、私たちはこれを奨励している。

しかし、人々が「ソフトウェアの販売」ということを考えるときに普通に想像するのは、ほとんどの企業が行っていることだろう。つまり、フリーソフトウェ

アではなく私有ソフトウェアを開発するということである。

そこで、本稿のように両者の境界線をていねいに引くのでなければ、「ソフトウェアの販売」という用語を避けて代わりに他の用語を選んだほうがよいかもしれない。たとえば、「フリーソフトウェアの有料頒布」と言えば、曖昧さはなくなる。

料金の高低とGNU GPL

1つの特別な状況を除き、GNU一般公開使用許諾書（GPL）は、フリーソフトウェアのコピーを頒布する際にどの程度の料金を設定できるかについて何も規定していない。無料、1ペニー、1ドル、10億ドルのどれでもかまわない。価格を決めるのは、あなたと市場なので、コピーの取得のために10億ドルも払いたくないからと言って私たちに文句を言わないでいただきたい。

例外というのは、対応するソースコードを完全な形で含まずにバイナリを頒布するときである。GNU GPLは、このようなことをする場合には、あとで要求されたときにソースコードを提供しなければならないことを義務付けている。ソースコードの料金に制限を設けなければ、誰も払えない料金（たとえば10億ドル）を設定し、実際にはソースコード隠しを行っているのに、ソースコードをリリースしているようなふりをすることができるようになってしまう。そのため、このような場合には、ソースコードの料金に枠をはめて、ユーザーの自由を確保しなければならない。しかし、通常の場合は頒布料金に制限を加えることに正当な理由はないので、私たちは制限を設けていない。

GNU GPLが認めている範囲を逸脱するような活動をしている一部の企業が、「GNUソフトウェアからは料金を徴収しません」などと言って活動に目をつぶってもらおうと躍起になっているが、そのようなことをしても無駄である。フリーソフトウェアのフリーは自由であり、GPLを強制するのは自由を守るためである。ユーザーの自由を守ろうとしているときに、頒布料金としていくらの額を徴収するかというような瑣末な問題に気を散らしているわけにはいかない。問題は自由であり、自由がすべてで唯一の問題なのだ。

初出: 第 1 稿は 1996 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第9章
フリーソフトウェアはフリードキュメントを必要とする

フリーオペレーティングシステムの最大の弱点は、ソフトウェア自体にはない。それは、OSに同梱できる優れたフリーマニュアルがないことである。私たちのもっとも重要なプログラムの大半が、完全なマニュアルを持っていない。ドキュメントは、あらゆるソフトウェアパッケージの必要不可欠な一部である。重要なフリーソフトウェアパッケージに優れたフリーマニュアルが含まれていなければ、それは大きな欠陥と言わなければならない。現在の私たちはそのような欠陥を無数に抱え込んでいる。

以前、何年も前のことだが、私はPerlを学ぼうと思ったことがある。フリーマニュアルのコピーを入手したが、それはとても読みにくいものだった。Perlユーザーたちにもっと良いものはないか尋ねてみると、入門用の優れたマニュアルがあるけれども、それはフリーではないというのである。

なぜ、このようなことになったのだろうか。優れたマニュアルの著者たちは、それをO'Reilly Associatesのために書いていたが、O'Reillyは制限付きの条件（コピー、変更禁止、ソースファイルなし）でそれを出版していた。そのため、そのマニュアルは、フリーソフトウェアコミュニティが受け入れられないものになってしまったのである。

この種のことが起きたのは、それが初めてではなかった。そして、私たちのコミュニティにとっては大きな損失だったが、それが最後とはとても言えない状況が続いている。私有マニュアルの出版社たちは、それ以来、非常に多くの著者に対して、マニュアルに制限を加えることをそそのかしてきた。私は、GNUユーザーが執筆中のマニュアルについて熱烈に語るのを何度も聞いたことがある。そ

れによって、GNUプロジェクトを助けたいというのである。そして、制限付きで私たちが自由に使えない条件の出版社と契約を結んだという説明を聞き、私の希望は打ち砕かれるのである。

優れた英語を書くという技能はプログラマたちの間ではまれなので、このようにしてマニュアルを失うのは大きな損失だ。

フリードキュメントは、フリーソフトウェアと同様に、価格の問題ではなく、自由の問題である。O'Reilly Associatesが印刷されたコピーに対して価格を設定する、そのこと自体に問題はない（フリーソフトウェア財団も、フリーGNUマニュアルの印刷済みコピーを販売している）。しかし、GNUマニュアルはソースコード形式でも入手できるのに、O'Reillyのマニュアルは紙版しかない。GNUマニュアルはコピーや変更が許可されているが、Perlのマニュアルはそうではない。問題は、制限である。

フリーマニュアルの基準は、フリーソフトウェアの基準とほぼ同様である。すべてのユーザーに特定の自由を与えるかどうかの問題だ。プログラムのすべてのコピーにオンラインであれ、紙版であれ、標準で同梱できるようにするために、再頒布（商業目的の販売を含む）は許可されていなければならない。変更許可も、非常に重要である。

一般原則として、私はあらゆる種類の論文や本を書き換える権利が絶対必要だとは考えていない。文章とソフトウェアでは、問題は必ずしも一致しないだろう。一般原則として、私は人間があらゆるタイプの論文や書籍を書き換える権利を持つことが重要だとは考えていない。たとえば、このような論文に変更を加えることを許可しなければならないとは思わない。この論文には、私たちの行動と思想が書かれているのである。

しかし、フリーソフトウェアのドキュメントについては、書き換えの自由が重要な意味を持つ理由がある。人々がソフトウェアを書き換える権利を行使し、機能を追加、変更したとき、彼らが仕事に忠実なら、マニュアルも書き換えるだろう。それにより、変更後のプログラムに合った正確で使えるドキュメントを提供できるようになるわけである。プログラマに職務を忠実に遂行し、仕事を完成させることを認めないマニュアルや、プログラムを書き換えたときには0から新しいマニュアルを書くことを強制するようなマニュアルは、私たちのコミュニティ

のニーズを満足させない。

もっとも、変更の包括的な禁止は認められないことだが、変更方法に何らかの歯止めをかけても、問題は起きないだろう。たとえば、オリジナルの著者の著作権情報、頒布条件、著者リストを書き換えないことを要求することはかまわない。変更版に書き換えられているということの記載を求めることもかまわないし、技術以外のトピックを扱っている節については、節全体の削除、変更を禁止することまで認められるはずだ（一部のGNUマニュアルにはそれが含まれている）。

これらの制限が問題とされないのは、職務に忠実なプログラマが変更後のプログラムに合わせてマニュアルを書き換えることを禁止していないからである。言い換えれば、そのような制限は、フリーソフトウェアコミュニティにおいてマニュアルを最大限に活用することを妨げない。

しかし、マニュアルの「技術的な」内容はすべて書き換え可能でなければならないし、通常のすべてのメディア、すべてのチャネルを通じて結果を頒布できなければならない。そうでなければ、制限はコミュニティにとって有害であり、そのマニュアルはフリーではないということになる。私たちは、別のマニュアルを用意しなければならない。

残念ながら、私有マニュアルが存在するときに代替マニュアルを書く人材を見つけるのは難しいことが多い。多くのユーザーが私有マニュアルでも充分に良いと思っていて、フリーマニュアルを書く必要性を感じないのである。彼らは、フリーオペレーティングシステムが埋めなければならない穴を抱えているとは思っていない。

ユーザーが私有マニュアルを充分に良いものだと考えるのはなぜだろうか。一部のユーザーは、それが問題だとは考えたこともないようだ。本稿でこの傾向を変えられればと私は願っている。

他のユーザーは、私有ソフトウェアを認める多くのユーザーと同じ理由で、私有マニュアルを許容できると考えている。彼らは、純粋に実用的な条件で判断し、自由を基準としていない。彼らでも思想を持たないわけではないが、彼らの思想は自由を含まない価値観の延長なので、自由を価値と認める私たちを導くことはできない。

この問題を広く伝えるようにしていただきたい。私たちは、私有出版のために

マニュアルを失い続けているのだ。私有マニュアルでは不充分だという考え方を普及させれば、ドキュメントの執筆によって GNU を支えたいと考えている次の人が、何よりもまずそのドキュメントをフリーにしなければならないということを考えてくれるはずだ。まだ遅くはない。

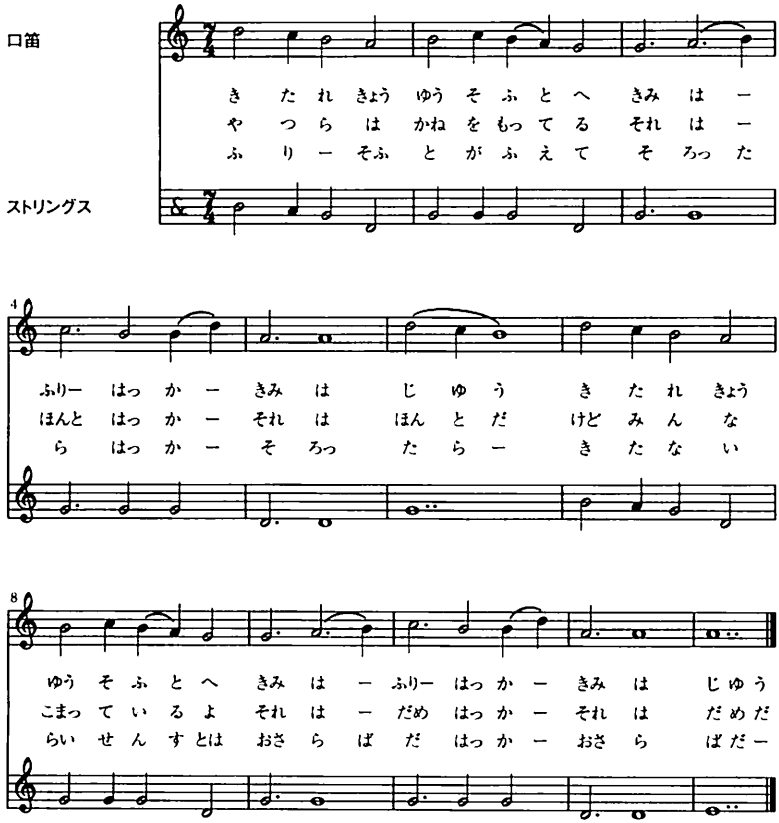
また、商業出版に対しても、私有マニュアルではなく、フリーのコピーレフトマニュアルを販売することをお勧めしたい。読者は、マニュアルを買う前に、頒布条件をチェックし、コピーレフトになっていないマニュアルではなく、コピーレフトになっているマニュアルを買うことによって、フリーマニュアル出版を支援することができる。

[注: フリーソフトウェア財団は、他の出版者が発売しているフリー書籍のリストをまとめた Web ページ（http://www.gnu.org/doc/other-free-books.html）を管理している。]

初出: 第 1 稿は 2000 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

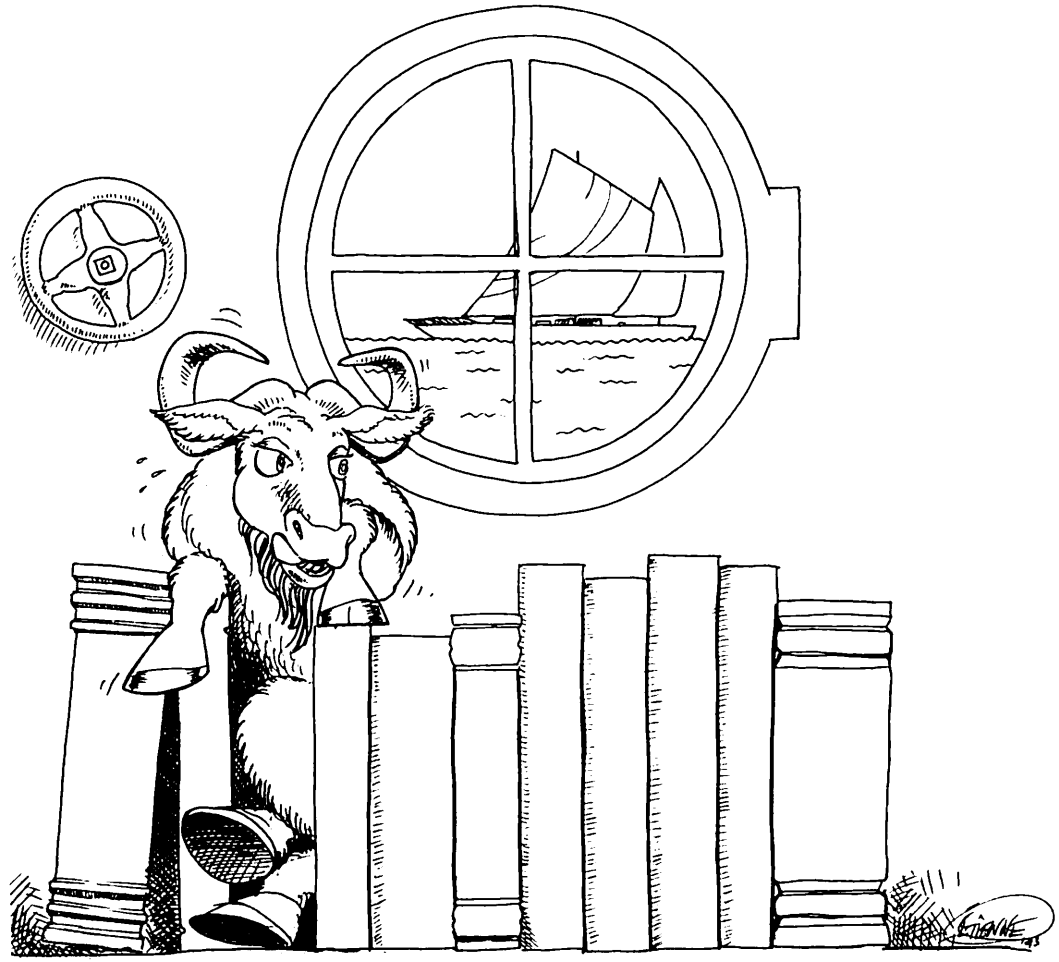
第10章
フリーソフトウェアの歌

ブルガリア民謡「Sodi Moma」のメロディで



初出: 第 1 稿は 1993 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第2部
コピーライト、コピーレフト、特許



第11章
読む権利

月面革命の先人たちの論文を集めた「ティコへの道」(2096年、ルナシティ刊)より。

ダン・ハルバートにとって、ティコへの道は大学でリッサ・レンツがコンピュータを貸してくれと言ったときに始まった。彼女のコンピュータは壊れてしまっており、誰かのものを借りなければ、彼女は中間試験のプロジェクトを提出できなくなってしまう。彼女があえて頼める相手はダンだけだった。

この依頼は、ダンをジレンマに陥らせた。彼は彼女を助けなければならないが、彼女に自分のコンピュータを貸したら、彼女は彼の本を読んでしまう。他人に本を読ませるようなことをすれば何年も牢屋に放り込まれるということは別として、何よりもまず、彼は読まれるという考えにぎょっとしたのである。彼は、他のすべての人々と同様に、小学校以来、本を共有することは危険で正しくないことだと教え込まれてきた。そのようなことをするのは、海賊だけだというのである。

そして、SPA(ソフトウェア保護局)が彼を逮捕しそびれる可能性はほとんどなかった。ダンは、ソフトウェアの授業で、個々の本には著作権モニターが付けられており、いつどこで誰が読んだかは中央ライセンス局に報告されているということを教わっていた(中央ライセンス局は、この情報を使って読書海賊を摘発するだけではなく、関心についての個人情報を書店に販売していた)。彼のコンピュータが次にネットワークに接続されたときに、中央ライセンス局は事態を把握するだろう。コンピュータの所有者である彼は、犯罪防止の労を惜しんだため

に、もっとも過酷な処罰を受けることになる。

もちろん、リッサは必ずしも彼の本を読もうとはしないかもしれない。彼女は、中間試験のプロジェクトを書くためにだけコンピュータを必要としているのかもしれない。しかし、ダンは彼女が中産階級出身で、授業料を払うのもやっとだということを知っていた。まして、読書料金など払えるものだろうか。彼の本の盗み読みは、彼女が卒業するための唯一の手段になるかもしれなかった。彼自身、状況は身にしみてよくわかっていたのである。彼もまた、自分が読んだすべての研究論文の読書料金を借りなければならなかったからである（読書料金の10%は、論文を執筆した研究者に支払われることになっていた。ダンは学者になることを目指していたので、自分自身の研究論文がひんぱんに参照されれば、この借金を充分返済できるだろうと計算していた）。

その後、ダンは、図書館に行けば、雑誌論文だけではなく書籍でさえ誰でも無料で読めた時代があったことを知った。政府の読書助成金なしで、数千ページの文献を読み漁った在野の学者たちもいた。しかし、1990年代になって、営利、非営利を問わず、雑誌出版社はアクセス料を徴収し始めた。2047年までには、一般人が学術文献に自由にアクセスできる図書館は、遠い過去の思い出になっていた。

もちろん、SPAと中央ライセンス局を出し抜く方法はあったが、そうすること自体も違法だった。ダンのソフトウェアのクラスには、違法のデバッグツールを入手し、それを使って本を読むときに著作権モニターコードをスキップしていたフランク・マートゥッチがいた。しかし、彼はそのことを多くの友達に話しすぎた。その中の1人が賞金目当てに彼のことをSPAに通報した（借金で首が回らなくなっている学生たちは、簡単に裏切りの誘惑に負けた）。2047年にフランクは投獄されたが、それは読書海賊行為のためではなく、デバッガを所有していたためだった。

その後、ダンは、誰もがデバッグツールを持つことができた時代があったことを知った。その頃は、CDやネットダウンロードで入手できるフリーデバッグツールさえあった。しかし、ごく普通のユーザーが著作権モニターを読み飛ばすためにそれを使い始めるようになり、実際の主用途はモニター逃れだという判決が下った。つまり、デバッガを持つことが違法になったのである。デバッガの開発者たちは、牢獄に送られた。

もちろん、プログラマはそうなったあともデバッグツールを必要としていたが、2047 年のデバッガベンダーは番号付きのコピーしか販売しておらず、それを入手できるのは公式ライセンスと保証人を持つプログラマだけだった。ダンがソフトウェアの授業で使っていたデバッガは、特殊なファイアウォールの向こう側で管理されており、授業の実習以外では使えないようになっていた。

著作権モニターは、書き換え済みのシステムカーネルをインストールするという方法でもバイパスできた。ダンは、フリーカーネル、それどころか全体としてフリーなオペレーティングシステムについての情報も見つけた。世紀の変わり目までは、そういうものがあったのである。しかし、それらはデバッガ同様違法なだけではなく、コンピュータの管理者パスワードを知らなければインストールすることさえできなかった。そして、FBI も Microsoft も、パスワードを教えてくれるはずはなかった。

ダンはリッサに単純にコンピュータを貸すことはできないという結論に達した。しかし、彼は彼女のことを愛していたので、彼女を助けることを拒むことはできなかった。彼女と話をするたびに、彼は喜びに満たされていたのである。そして、彼女が彼にだけ助けを求めたということは、彼女も彼を愛しているということかもしれなかった。

ダンは、さらに信じられないような方法でこのジレンマを解決した。彼女にコンピュータを貸した上で、自分のパスワードを教えたのである。こうすれば、彼女が彼の本を読んだとしても中央ライセンス局は彼が本を読んでいると思うだろう。これも犯罪であることに違いはなかったが、SPA は自動的にこれを見つけることはできない。リッサが密告しなければ、事は漏れないのである。

もちろん、彼がリッサに自分のパスワードを教えたことが大学当局にばれたら、彼女がそれを何のために使ったかにかかわらず、2 人とも学籍を失うことになる。彼らの大学では、学生のコンピュータ利用を監視するさまざまな手段に対するあらゆる妨害行為が懲戒処分の対象になるという制度を採用していた。それは、何か害になることをしたかどうかとは無関係である。そのような攻撃は、学生管理を非常に困難にする。ということは、何か禁止されていることをしようとしているのだと管理者側は考えるのである。管理者にとっては、学生たちが何をしようとしているのかを知る必要はないのだ。

通常、学生はそのために退学させられることはない。もっとも、それは直接的には、ということだが。彼らは、大学のコンピュータシステムから締め出され、すべての単位を落とすことを余儀なくされる。

その後、ダンは、この種の指針が制定されたのは学生の多数がコンピュータを使い始めた1980年代だということを知った。それまでの指針では、懲戒処分の考え方は異なるものだった。有害な行為は処罰されたが、単に疑惑が発生しただけでは処罰されなかったのである。

リッサは、SPAにダンを密告しなかった。ダンの決意は、2人を結婚に導き、子供の頃から海賊行為について教えられてきたことに対する疑問を呼んだ。2人は著作権の歴史について、ソ連とそのコピー規制について、また最初の形の合衆国憲法について、学習を始めた。そして、彼らは月に移住した。彼らはそこでSPAの長い腕から逃れ出てきた人たちと出会った。2062年にティコの反乱が始まったとき、普遍的な読む権利は、反乱軍の中心目標の1つになった。

作者のコメント

読む権利は、現在戦われている戦闘である。私たちの現在の生活様式が忘却の闇に沈むまで、50年ぐらいかかるかもしれないが、物語の中で取り上げた具体的な法律や慣習の大半はすでに提案されており、その中の多くのものが米国やその他の国々で立法化されている。米国では、1998年のデジタルミレニアム著作権法（DMCA）が、コンピュータ化された書籍（およびその他のデータ）の読み出しや貸借に制限を加える法律的基礎を確立した。EUも、2001年の著作権指令で同様の制限を加えた。

例外が1つある。FBIとMicrosoftがパーソナルコンピュータの管理者キーワードを管理し、個人にそれを知らせないというアイディアは、まだ提案されていない。これは、Clipperチップと合衆国政府のキーエスクロー提案からイメージされたものだが、これらはともに長期的な傾向である。コンピュータシステムは、次第に目に見えないオペレータが実際の利用者を制御するようなものになりつつある。

私たちは次第にここに描かれている世界に近づいてきている。2001年にはディズニーの献金を受けているホーリングス上院議員がSSSCAという法案（現在は

CBDTPA と改名されている）を提出した。これは、すべての新しいコンピュータに対してユーザーがバイパスできないコピー制限機能を搭載することを義務付けるというものである。

SPA は、実際には Software Publisher's Association（ソフトウェア出版業協会）の略語だが、ここで触れた警察的な役割を BSA（ビジネスソフトウェア同盟）に譲った。今のところ、BSA は公式の警察権力ではなく、警察的に機能する非公式の組織だが、以前のソ連と同様の手法を使い、同僚や友人の密告をけしかけている。BSA が 2001 年にアルゼンチンで繰り広げたテロキャンペーンは、ソフトウェアを共有する人々が牢獄に連れ去られるという表からは見えない脅威を生み出した。

この物語が書かれた頃、SPA は小規模な ISP（インターネットサービスプロバイダ）に対し、全ユーザーを SPA が監視できるようにせよと圧力をかけていた。ほとんどの ISP は、法廷で SPA と闘うだけの体力を持たないので、脅されると同時に降参してしまったが、少なくとも 1 つの ISP（カリフォルニア州オークランドの Community ConneXion）は要求を拒絶し、実際に告訴された。その後 SPA は告訴を取り下げたが、求めようとしていた力を与えてくれる DMCA を手に入れた。

物語の中のセキュリティポリシーは、空想の産物ではない。たとえば、シカゴ地区のある大学のコンピュータは、ログイン時に次のようなメッセージを表示する。

> 「このシステムは、権限のあるユーザーしか使えません。権限なく、あるいは与えられた権限を越えてこのコンピュータシステムを利用している個人のこのシステムにおけるすべての活動は、システム管理者による監視、記録の対象とされます。このシステムを不正利用している個人を監視している過程、あるいはシステムメンテナンスの過程では、権限を持つユーザーの活動も監視されます。このシステムを使うすべての人は、そのような監視に明示的に同意するものとします。また、監視活動によって、不法行為や学則違反の証拠となり得るものが明らかになった場合には、システム管理者が大学当局あるいは司法当局またはその両方に監視から得られた証拠を提出することをご承知おきください」

これは、前もって権利放棄に同意するよう、ほぼすべての人に圧力をかけるという手法であり、憲法修正第4条に対する重大な挑戦である。

参考文献

- 政府の白書: "Information Infrastructure Task Force, Intellectual Property and the National Information Infrastructure: The Report of the Working Group on Intellectual Property Rights" (1995). (「情報インフラストラクチャタスクフォース、知的財産権と国家情報インフラストラクチャ: 知的財産権作業グループ報告)
- 白書の解説: "The Copyright Grab," Pamela Samuelson, Wired, Jan. 1996 (http://www.wired.com/wired/archive/4.01/white.paper_pr.html)
- "Sold Out," James Boyle, The New York Times, 31 March 1996
- "Public Data or Private Data," The Washington Post, 4 Nov 1996. (以前、私たちはこのWebサイトにリンクを張っていたが、ワシントン・ポストがWebサイト上で記事を読むことを希望するユーザーに対して料金徴収を開始することを決定したので、リンクを取り除くことにした)。
- Union for the Public Domain——著作権と特許が持つ過度の力に抵抗し、逆転することを目指す組織 (http://www.public-domain.org/)。

初出: "*Communications of the ACM*", volume 40, number2, February 1997。「作者のコメント」は、2002年に更新されている。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第12章 著作権の誤解 —— 一連の誤り

著作権法に奇妙で危険なことが起きようとしている。合衆国憲法のもとでは、著作権はユーザー（本を読み、音楽を聴き、映画を見、ソフトウェアを実行する人々）を守るために存在するもので、出版社や著作者のために存在するわけではない。しかも、人々は「自らの利益のために」押し付けられた著作権の制約を拒絶し、従わない傾向を強くしている。にもかかわらず、合衆国政府は制約を追加し、新しい過酷な刑罰に服するよう人々を脅迫している。

著作権政策は、どのような経緯でもともと述べられていた目的と正反対のものになってきたのだろうか。そして、当初の目的に沿ったものに著作権政策を引き戻すためにはどうしたらよいのだろうか。それを理解するためには、合衆国著作権法のルーツである合衆国憲法を見てみなければならない。

合衆国憲法における著作権

合衆国憲法が立案されたとき、著作者が著作権を独占するという考え方が提案され、そして否決された。私たちの国家の創設者たちは、著作権は著作者の自然権ではなく、進歩のために著作者に対してなされた譲歩という異なる前提を採用した。憲法は、以下の条文（第1条第8節）で著作権制度を認めている。

> ［議会は］著作者と発明者に対し、それぞれの作品および発明についての独占的な権利を一定期間保証することによって、科学および有用な技芸の進歩を促進する［権限を有する］。

最高裁は、進歩の促進とは、著作権が付与されている仕事の利用者（ユーザー）の利益を意味することを繰り返し確認している。たとえば、Fox Film-Doyal 裁判の判決は、次のようになっている。

> 合衆国の唯一の関心と［著作権］独占を認めている主要な目的は、作者の労力から国民が引き出せる利益全般にある。

このきわめて重要な判決は、著作権が憲法によって**義務付けられている**のではなく、選択肢として**許可されている**だけだということの理由、そして「限定された期間」で消滅する理由を説明している。著作権が自然権で、著作者が著作者としての資格のために持つ権利だとすれば、著作権が一定期間経過後に消滅することを説明できない。建設後一定期間を経過した住宅を公有財産にすべきだと言うのとは違うのである。

「著作権の廉売」

著作権制度は、特権を提供し、出版者と著作者に利益を供与することによって機能するが、それを目的としているわけではない。出版者や著作者の行動を変える、つまり、より多くのものを執筆、公刊する動機を著作者に与えるために、そのようなことをしているのである。実質的に、政府は、公刊され、国民に供される仕事を増やすための買物として、国民に代わって国民の自然権を売っていることになる。法学者たちは、この概念を「著作権の廉売（copyright bargain）」と呼んでいる。これは、納税者のお金を使って高速道路や航空機を購入しているのと同じようなものである。違うのは、政府が国民のお金の代わりに国民の自由を使っていることである。

しかし、今ある取引制度は、国民にとって良い買物になっているだろうか。取引の方法はいくつもある。どれがベストだろうか。著作権制度のすべての争点は、この問の一部である。この問の性質を誤解すれば、争点の解決方法がまずいものになる。

憲法は著作者に著作権を認めているが、実際には著作者は出版者（社）に著作

権を譲ることが多い。確かに著作者もわずかな利益を得るが、通常この権利を行使して利益の大半を得るのは、著作者ではなく出版社なのである。つまり、通常著作権を増強するために議会に圧力をかけてくるのは、出版社である。本稿では、著作権の神話よりも現実を反映させた議論のために、著作権の保持者を著作者ではなく出版社と呼ぶことにしよう。また、著作権の保護下にある仕事を使うことは必ずしも「読む」ということではないが、「ユーザー」では抽象的でぴんとこないので、利用者のことを「読者」と呼ぶことにしよう。

最初の誤り：「利益衡量」

著作権の廉売は、国民を第 1 に位置付けている。読者たる国民の利益はそれ自体で目的だが、出版社のための利益（あるなら）はその目的のための手段に過ぎない。読者の利益と出版社の利益の優先順位は、質的に異なるのである。著作権の目的を誤解する第一歩は、出版社に読者と同水準の重要性を認めてしまうことである。

合衆国著作権法は、出版社と読者の間で利益の「衡平を保つ」ことを目的とするものだとよく言われている。この解釈を口にする人々は、憲法に書かれている基本的な立場の言い換えとしてこれを提出する。つまり、著作権の廉売は利益衡量と同じ意味だとされるのである。

しかし、これら 2 つの解釈は同じだとはとても言えない。概念的にも、内包する意味にも違いがある。衡平概念は、読者と出版社の利益の差とは量的なものだけだという前提に立っている。彼らに「どの程度の重さ」を与えるか、両者がどのような行為をするかの違いだけだとみなしている。「係争物受寄者」という用語は、このような形で問題の枠組みを作るために使われることが多い。この用語は、政策決定においてあらゆる種類の利益が等しく重要だということを前提としており、この考え方は、著作権の廉売に政府が参加する根拠となっている読者と出版社の利益の質的な違いというものを無視している。

このすり替えの結果は非常に重大である。と言うのも、著作権の廉売では国民のために与えられていた大きな保護（著作権という特権が正当化されるのは、出版社の名ではなく、読者の名のもとによってのみであるという思想）が、「利益衡

量」解釈によって捨て去られてしまうからである。出版社の利益がそれ自体で目的とみなされるので、それが著作権特権を正当化してしまう。つまり、「利益衡量」概念は、国民以外の誰かの名のもとにこの特権が正当化されると言っているのだ。

現実問題としては、「利益衡量」概念は、著作権法の改訂理由の挙証責任を転換する結果を生む。著作権の廉売は、読者に特定の自由を放棄させるために、出版社に負担を負わせる。現実的に言って、利益衡量概念はこの負担を逆転させる。特権の付加によって利益を得るのは間違いなく出版社なのだ。出版社の利益よりも「重い」読者側の害が立証できない限り、出版社は要求するほぼすべての特権を得ることができるという結論に導かれてしまうのである。

出版社と読者の間の「利益衡量」概念は、読者に与えられるべき１番の地位を否定するものなので、私たちはこれを拒否しなければならない。

何に対する衡平か

政府が国民のために何かを買うとき、政府は国民の代理として機能している。政府の責任は、契約相手のためにではなく、国民のために最良の買物をすることである。

たとえば、高速道路の建設のために建設会社と契約を結ぶとき、政府は国民の税金をできる限り節約しようとする。そこで、政府機関は価格引き下げのために競争入札を行う。

現実問題として、契約相手が０ドルで入札してくることはないので、価格が０になることはありえない。深く考えるまでもなく、彼らは自由社会の市民としての通常の権利を持っており、その中には不利な契約を拒否する権利が含まれている。しかし、最低入札額でも、契約相手によっては充分に儲けられる仕事になるかもしれない。そのため、一種の利益衡量は確かに必要である。しかし、それは、各々特別に考慮すべき主張を持った２つの利益の衡平を慎重に取るということではない。国民の目標と市場の力の間で衡平を取るということである。政府は、自動車を持つ納税者のために、自由社会と自由市場の枠内で入手できる最良の買物をするように努力する。

著作権の廉売の場合、政府は、国民のお金の代わりに自由を支払う。自由はお

金よりも貴重なものなので、賢く、できる限り節約して支払うという政府の責任は、お金のときよりもさらに重いものになる。政府が国民の自由と同じ基準で出版者の利益を後押しするようなことがあってはならない。

「衡量」ではなく「調整」

読者の利益と出版者の利益を秤にかける考え方は、著作権政策の判断方法として誤っているが、本当に 2 つの利益を秤にかけなければならない場合もある。それは、**読者の間の** 2 つの利益である。読者は、出版された仕事を利用する自由という利益を持つ一方で、状況によっては、何らかの動機誘発制度によって出版を奨励する関心を持つ場合がある。

著作権の議論における「衡量」は、読者と出版者の間の「利益衡量」の考え方の略称となってしまった。そのため、読者の間の 2 種類の利益について「衡量」という単語を使うと、混乱を招いてしまう。別の用語が必要なのだ。

一般に、共同当事者が部分的に矛盾する 2 つの目標を持ち、両者を完全に実現することができないとき、私たちはこれを「調整」と呼ぶ。そこで私たちは、双方当事者の間で「利益衡量」を行うとは言わず、「私たちの自由の放棄と確保の間で正しい調整方法を探す」と言うことにする。

第 2 の誤り：生産の最大化

著作権政策の第 2 の誤りは、出版された仕事の数を単に増やすのではなく、最大限に増やそうとすることを目標に掲げることである。「利益衡量」の誤った考え方は、読者を背後に持つ当事者というところまで、出版社の地位を引き上げた。第 2 の誤りは、出版社を読者よりもはるかに上の位置に置いてしまう。

私たちは、買物をするときに、一般にすべての在庫を押さえたり、もっとも高価なモデルを買ったりはしないものである。必要な分量だけ買ったり、最高品質ではなく自分にとって充分なモデルを選択して、他のものを買うための資金を残しておくものである。収穫逓減の法則によれば、特定の商品にすべての資金を投入すると、資源を効率的に配分できなくなる。一般に、私たちは他の用途のため

にお金をいくらか残しておくものである。

著作権の場合でも、収穫逓減の法則は他の買物と同じように働く。私たちが最初に自由を明け渡すとき、私たちが失う自由は最小で、出版社が得る援助は最大になる。その後、自由をさらに明け渡していくと、私たちの犠牲は以前よりも大きくなり、著作活動に与えられる援助は小さくなっていく。著作活動の増分がゼロになる前に、価格増分には意味がないと言ってもよいはずだ。出版量の増加につながる取引には同意できるが、最大限の増加を目的とする取引には同意できない。

出版物を最大限に増やすという目標を受け入れることは、より賢明でより有利な取引をすることをあらかじめ封じることになる。最大化政策は、出版物のほんのわずかな増加のために、すべての自由を出版物に明け渡せと国民に命令するものである。

最大化のレトリック

実際には、犠牲となる自由を省みず、出版物を最大限まで増やすという目標は、「コピーは、違法、不正で本質的に悪い」と断言する広く行き渡ったレトリックによって支えられている。たとえば、コピーをする人間を「海賊」と呼ぶが、これは隣人と情報を共有しようとすることを船に攻撃を仕掛けることと同じだとみなす誹謗中傷である（この中傷語は、もともと無許可版を出版する合法的な方法を見つけた出版社に対して著作者が使っていたものだが、出版社による現在の使い方は、ほとんど逆の意味になっている）。このレトリックは、著作権に対する憲法的な基礎を直接否定するものだが、アメリカの法制度の当然の伝統を表すものとしてそびえ立っている。

「海賊」のレトリックが受け入れられているのは、それがメディア全体を覆い尽くし、その過激な意味を理解している人がほとんどいないからである。このレトリックは非常に効果的であり、国民がコピーすることが基本的に違法なのだとすれば、出版社が国民の自由を代償とする要求を掲げてきたときに国民は反論できなくなってしまう。つまり、出版社に新たな権力を与えるべきではない理由を国民が述べなければならなくなったときに、何よりも重要な「コピーしたい」という理由があらかじめ排除されてしまうのである。

そのため、著作権の強化への反論には、副次的な争点しか持ち出せなくなる。今日の著作権強化に対する反論が、副次的な争点ばかりを持ち出し、正当で公的な価値としてのコピー頒布の自由をほとんど言い出せないでいるのは、そのためである。

現実問題として、「特定の行為が出版社の営業成績を引き下げる、あるいはそういう結果をもたらす可能性があると考えられるがゆえに、その行為は、何らかの量の出版物を減らすことになると思われるので、その行為は禁止されるべきである」といった類の出版社側の論拠を成立させているのは、最大化目標にある。私たちは、国民の利益が出版社の営業成績によって計られるという粗暴な結論に導かれているのである。メディア全般にとって良いものがアメリカ全体にとって良いものになってしまうのだ。

第３の誤り：出版社の権力の最大化

出版社は、あらゆる犠牲を払ってでも出版物を最大限まで増やすという政策目標への同意を勝ち取ると、次の段階では、その目標を実現するためには自分たちに最大限の権力を与えることが必要だという推論を引き出そうとする。つまり、想像できるあらゆる用途に著作権が及ぶようにしたり、「シュリンクラップ」ライセンスなどの法的な小道具を使って同等の効力を獲得しようとする。この目標は、「公正利用」や「消尽の法理」などの全面破棄などを含むもので、州政府から国際機関に至るあらゆるレベルの政府が支持している。

この段階の誤りは、厳格すぎる著作権法が、役に立つ新しいものの創造を阻むところにある。たとえば、シェークスピアは、一部の自作台本の中で、数十年前に出版された他の台本から筋書きを借用している。今日の著作権法が当時有効なら、彼の台本は違法ということになってしまうだろう。

たとえ、国民が支払う代償の高低にかかわらず出版物の量を最大限に引き上げたいのだとしても、出版社の権力を最大限に引き上げるのは誤りである。進歩を促進する手段として、この政策は自滅的である。

３つの誤りがもたらしたもの

昨今の著作権法の流れは、出版社により広範な権力をより長期にわたって差し出す方向にある。発生時の著作権の概念的な基礎は一連の誤りによって歪められ、ノーと言うための基礎をほとんど提供してくれない。立法者たちは、国民のための著作権という思想にリップサービスを使いながら、実際には出版社に求められたすべてのものを与えている。

たとえば、ハッチ上院議員は、S.483（著作権の有効期限を 20 年延長する 1995 年の法案）を提出したときに次のように語った。

> 私は、著作権の現在の有効期限が著作者の利益を充分に保護できるか、またそれに付随して、著作権の現在の有効期限が新しい作品を創作する充分な動機を提供し続けられるかという疑問を検討すべき時期に差し掛かっていると考える。

この法案は、1920 年代以降に書かれた出版済みの作品の著作権を延長した。すでに出版されている本の数を今から遡って増やすことはできないので、この改正は、国民にまったく利益を与えずに出版社に一方的に利益を与える不公正なものになった。しかも、国民の自由、すなわち当時の本を再頒布する自由は犠牲になったまま今日に至っている。

同法案は、まだ書かれていない作品の著作権の有効期限も延長した。無名著作物の著作権は、現行の 75 年から 95 年に延長される。理論的には、これで新しい作品を書く動機は増すだろう。しかし、この新たな動機の必要性を主張した出版社は、2075 年の貸借対照表を添えて、自らの主張を実証しなければならなかったはずである。

言うまでもなく、議会は出版社の主張に疑問を差し挟みはしなかった。著作権を延長する法律は 1998 年に成立した。この法律は、その年の初めに亡くなった資金提供者の 1 人の名前を取って、ソニーボノ著作権保護期間延長法と呼ばれている。彼の著作権を継承した未亡人は、次の声明を発表した。

> 実際には、ソニーは著作権が永遠に継続することを望んでいました。しかし、スタッフから、そのような法改正は憲法に抵触するだろうという知らせを受けました。私たちの著作権法を可能なあらゆる方法で強化するために、すべての皆さんで闘いましょう。ご存知のように、永遠マイナス 1 日の存続というジャック・バレンティの提案もあります。おそらく、委員会は次の議会でこの提案を検討するでしょう。

最高裁は、遡及的な延長は憲法が掲げる進歩の促進という目標に反するという論拠に基づき、同法の違憲判決を目指す訴訟で原告勝訴の判決を下した。

1996 年に成立した別の法律では、すべての出版済みの作品について、たとえ友達のために贈る場合でも、充分多数のコピーを作ることが重罪とされることになった。合衆国では、このようなことが犯罪とされることはそれまで決してなかったのである。

さらにひどい悪法、デジタルミレニアム著作権法（DMCA）は、コピープロテクトを破ること、さらには破り方を公表することさえ犯罪とすることによって、コピープロテクト（コンピュータユーザーが憎んでいるもの）を復活させることを目的として作られたものである。この法律は、実質的に出版社に対して独自の著作権法を制定するチャンスを与えるものであり、「メディア企業による支配法（Domination by Media Corporations Act）」とでも呼ぶべきものである。DMCA には、出版社は作品の用途について任意の制約を課することができると書かれており、その制約を強制するための暗号/ライセンスマネージャが作品に埋め込まれている場合には、制約が法律としての効力を持つとも書かれている。

この法案の論拠の 1 つは、著作権強化を目指す最近の条約を履行できるようにするというものだった。その条約なるものは、著作権・特許権保持者に支配された機関である世界知的財産権機関（WIPO）がクリントン政権の協力を得て制定したものである。この条約は著作権を強化するだけなので、どこの国でも国民の利益に奉仕するかどうかは疑問である。いずれにせよ、DMCA は、条約が要求している線よりもはるかに先に進んでいる。

この法案、特に「公正利用」とみなされるコピーを禁止する部分についての反論の主な論拠は図書館だった。出版社は、この議論にどのように反論したのだろ

うか。米国出版者協会の前代表で、今はロビイストとして活躍しているバット・シュレーダーは、出版社は「[図書館から] 求められていることを認めたのでは生きていけない」と言った。図書館は、現状の一部を残すことを求めていただけである。このような議論に対しては、出版社が今までどうやって生き残ってきたのか不思議だと反論することができるだろう。

バーニー・フランク下院議員は、私を含む法案反対者たちとの会合の場で、合衆国憲法の著作権思想がどこまで軽視されているかを身をもって示した。彼は、「"映画産業"、"音楽産業" などの各種 "産業" の懸念を払拭するために、罰則を伴う新しい権力が緊急に必要とされている」と語った。私は、彼に尋ねた。「でも、それは国民の利益なのですか」　彼の返答はこうだった。「なぜ、国民の利益などを持ち出すのだ。これらの創造的な人々が国民の利益などのために自らの権利を放棄する必要はない！」　「産業」がそこに雇われている「創造的な人々」と同一視され、著作権が独自の資格をもって扱われ、憲法が逆立ちさせられている。

DMCA は、1998 年に成立した。立法時に、公正利用は名目的な合法性を保ったが、出版社の圧力に負けて、公正利用を行使するためのソフトウェアやハードウェアの禁止を認めてしまった。公正利用は、実質的に禁止されてしまったのである。

映画産業は、DMCA に基づき、DVD を読み、再生するフリーソフトウェア、さらには読み方についての情報にさえも、検閲をかけてきた。2001 年 4 月には、プリンストン大学のエドワード・フェルテン教授が、全米レコード産業協会（RIAA）からの告訴の脅迫に屈し、録音済み音楽へのアクセスを制限する暗号化システム案についての研究内容をまとめた科学論文を取り下げた。

読者の伝統的な自由の多くが取り除かれた e-book（電子ブック）も見かけるようになった。たとえば、友人に本を貸す自由、古本屋に売る自由、企業のデータバンクに自分の名前を記入せずに買う自由、さらには 2 度読む自由である。暗号化された e-book は、一般にこれらすべての行為を制限する。読者の自由を制限するために作られた特別な秘密のソフトウェアがなければ、それらは読めない。

私はこのような暗号化され、制限の付けられた e-book を決して買うつもりはないし、あなたも拒否してくれたらと思っている。e-book が従来の紙の本と同じ自由を与えてくれないのなら、そのようなものを受け入れてはならない！

制限付きの e-book を読めるソフトウェアを自主リリースした人は、訴追のリスクを負うことになる。2001 年、ロシアのプログラマ、ディミトリー・スクリヤロフは、あるコンファレンスで講演するために米国滞在中に逮捕されたが、それは彼がそのようなプログラムを書くことが合法的に認められているロシアでそのようなプログラムを書いたからである。現在、ロシアも同様の行為を禁止する法律を準備しており、EU は最近同様の法律を成立させた。

今まで、一般市場向け e-book は、商業的に失敗してきたが、それは読者がそれぞれの自由を守ることを選んだからではない。コンピュータの画面が読みにくいなどの別の理由で e-book が魅力的ではなかったからである。私たちは、この幸運な偶然が長期にわたって私たちを守ってくれると考えてはならない。次世代の e-book は、「電子ペーパー」を使うようになるだろう。これは、暗号化され、制限が付けられた e-book をダウンロードできる本型のオブジェクトである。この紙風の表示が現在の画面よりも魅力的なら、私たちは自由の擁護のために発言しなければならないだろう。一方で、e-book は隙間から侵入しつつある。NYU を始めとするいくつかの歯科大学では、学生たちは制限付きの e-book 形式の教科書を買わなければならない。

メディア企業は、それでもまだ満足していない。2001 年には、ディズニーの献金を受けているホーリングス上院議員が SSSCA（セキュリティシステム標準および証明書発行法）という法案[1]を提出した。これは、すべてのコンピュータ（およびその他のデジタル記録再生装置）に強制的に政府指定のコピー制限システムを搭載させるというものである。これは彼らの最終的な目標だが、そこに至る日程表の第 1 項目は、国民が「みだりに変更」できないように（つまり、自分の目的で変更できないように）設計されていない限り、デジタルハイビジョンテレビチューナーを販売できないようにすることにある。フリーソフトウェアはユーザーが書き換えられるソフトウェアなので、私たちは特定の仕事を対象とするフリーソフトウェアの開発を明示的に禁止する法案に初めて直面することになったわけであ

*1　発音不能な CBDTPA に改名されて以来、"Consume, But Don't Try Programming Anything（消費してもプログラミングしようとしてはならない）"という言葉の記憶法として便利になった。実際には、"Consumer Broadband and Digital Television Promotion Act（消費者向けブロードバンドデジタルテレビ促進法案）" の略語である。

る。おそらく、他の仕事の禁止もあとに続くだろう。連邦通信委員会（FCC）がこの規則を採用したら、GNU Radio などの既存のフリーソフトウェアは禁止される。

これらの法案や規則の成立を阻止するためには、政治行動が必要だ[2]。

正しい廉売方法を見つける

著作権政策の適正な決め方はどのようなものだろうか。著作権が国民のための廉売なのであれば、何よりもまず国民の利益に奉仕しなければならない。政府が国民の自由を売りに出すときの義務は、売らなければならないものだけを売ることと、できるだけ高く売ることである。最低でも、同程度の出版部数を確保しつつ、著作権の範囲を最小限に切り詰めなければならない。

建設プロジェクトの場合とは異なり、最低価格は競争入札ではわからない。では、どのようにして最低価格を見つけたらよいのだろうか。

1つの可能な方法としては、段階的に著作権を縮小し、結果を観察するというものがある。出版点数が目に見えて減少したら、国民の目的を達成するために必要な著作権の程度というものがわかる。私たちは、出版社が起きるだろうと言っていることからではなく、実際の観察からこれを判断しなければならない。出版社は、どのような形であれ、自らの権力が弱体化されるとなれば、予想を誇張して述べる動機を十二分に持っている。

著作権政策には、いくつかの独立した軸が含まれている。それらの軸は、別個に調整できるはずだ。独占的な複製作成権を発行日以降10年に短縮するのが第一歩としては適切だと思う。派生的な作品の製作など、著作権のその他の側面については、より長期にわたって継続してもよいと思う。

なぜ発行日から起算するのか。それは、出版されていない作品の著作権は、読者の自由を直接的に制限しないからである。コピーがないのに作品をコピーする自由があるかどうかを論じても無意味である。だから、著作者に作品を出版するまでの時間をより長く与えても、害はない。著作者（一般に、出版に先立って著

•2　支援希望者には、digitalspeech.org と www.eff.org の2つの Web サイトをお勧めする。

作権を保持している）は、著作権が切れるのを先延ばしにするために発行日を遅らせるようなことをまずするものではない。

なぜ 10 年なのか。それは、提案として安全だからである。私たちは、この数字まで期間を短縮しても、今日の出版物の寿命にはほとんど影響を与えないという現実的な根拠を提出できる。ほとんどのメディア、ジャンルにおいて、成功した作品が特に大きな利益をもたらすのは数年であり、成功した作品でさえ、発行後 10 年以内に品切れになることが多い。数十年にわたって実用的な生命を失わない参考図書でも、著作権は 10 年で充分である。ほぼ定期的に改訂版が発行されれば、多くの読者は 10 年以上経過したパブリックドメイン版ではなく、著作権付きの最新版を買うものだ。

あるいは、10 年でも必要以上に長すぎるかもしれない。全体が落ち着いたら、制度の最適化のためにさらに期間を削減してみるとよいだろう。文学関係のある大会で開かれた著作権についてのパネルディスカッションで私が 10 年論を提案したところ、私の隣に座っていたある著名なファンタジー作家が私に激しく反論した。5 年以上は問題外だというのである。

しかし、あらゆるタイプの仕事に同じ期間を適用する必要はない。著作権政策において極端な画一性を維持することは、国民の利益にとってどうしても必要なことではないし、現行の著作権法自体、すでに特定の用途やメディアに対してはさまざまな例外を持っている。国内のもっとも高価な地域のもっとも困難なプロジェクトで必要とされる経費をすべての高速道路プロジェクトにかけるのは馬鹿げている。同様に、最高値の自由を差し出す必要性が感じられるような特定の分野の作品と同じだけの自由をすべての作品に「支払う」のも馬鹿げている。

おそらく、小説、辞書、コンピュータプログラム、歌、交響曲、映画は、それぞれ有効期限の異なる著作権を持つことになるだろう。そうすれば、それぞれの種類の仕事について、多くの作品が出版されるために必要なところまで有効期限を短縮していくことができるだろう。1 時間以上の映画は、製作コストから考えて、著作権を 20 年としてもよいかもしれない。私自身の分野であるコンピュータプログラムでは、3 年もあれば充分である。実際の製品サイクルは、それよりも短い。

公正利用の範囲も、著作権政策の中の独立した軸の 1 つである。著作権法の保護下にある出版された作品の全部または一部の複製の作成を合法的に認める方法

は、いくつか考えられる。著作権のこの軸における緩和のための自然な第一歩は、個人の間で少量非営利臨時の私的なコピー、頒布を認めることだろう。こうすれば、個人の私生活に著作権警察が足を踏み込んでくるのを防ぐことができるし、出版された作品の営業成績に大きな影響を与えることはあまりないだろう（また、このようなコピーの制限のために、著作権に代えてシュリンクラップライセンスを使うことを禁止するための法的な措置も必要になるはずだ）。Napsterの経験からも、一般人に非営利の本文に一切の変更を加えない複製の再頒布を認める必要があることは明らかである。非常に多くの国民がコピー、共有を望んでおり、それが非常に有益なことがわかっている場合、それを止められるのは厳罰主義だけだろう。そして、国民は必要としているものを手に入れる権利を持っている。

小説を始めとして一般に娯楽用に使われる作品の読者の自由としては、非営利の本文に一切の変更を加えない再頒布を保証すれば充分だろう。しかし、機能的な目的で（仕事を終わらせるために）使われるコンピュータプログラムについては、改良版の出版の自由を含むより踏み込んだ自由が必要である。ソフトウェアユーザーが持つべき自由については、本書の第3章「フリーソフトウェアの定義」を参照していただきたい。しかし、これらの自由を広く認めるのは、プログラムの発売から2、3年後に限るというような妥協は許容範囲内かもしれない。

このような変更を加えれば、国民のデジタルテクノロジーをコピーしたいという希望に著作権も追い付くようになる。出版社は、間違いなく、これらの提案を「衡平を欠く」とみなすだろう。彼らは、石を捨てて家に帰ると脅すかもしれない。しかし、このゲームはまだ収益を生むし、街で行われている唯一のゲームになるので、彼らが本当にそうすることはないだろう。

私たちは、著作権の弱体化を検討するのと同時に、メディア会社が単純に著作権の代わりにエンドユーザーライセンス契約を使えるような状態を放置してはならない。著作権法が認めている範囲を越えてコピーに制限を加える約款の使用を禁止する必要がある。大市場における交渉抜きの約款に対するその種の規制は、アメリカの法制度の標準的な構成要素である。

個人的なメモ

私はソフトウェアデザイナーであって、法学者ではない。私が著作権問題に懸念を持つようになったのは、コンピュータネットワーク[*3]の世界ではこの問題を避けられないからである。コンピュータとネットワークを 30 年以上使ってきた 1 人のユーザーとして、私は、私たちが失ってきた自由、次に失いそうな自由の価値の高さを知っている。また、1 人の著作者として、出版社が著作権強化を正当化するときによく口にする準創造主としての著作者というロマンティックな神話を拒否することができる。著作者は、どうせ契約によってそれを出版社に譲り渡すことになるのだ。

本稿の大半は、読者がチェックできる事実と根拠、および読者が自分自身の意見を持てる提案から構成されている。しかし、私が言っていることのうち、たった 1 つだけはどうか受け入れていただきたい。それは、私のような著作者には、あなたに対して特別な権力を持つ権利はないということである。私が書いたソフトウェアや書籍のために私にさらに報酬を提供したいというのであれば、私は喜んでそれを頂戴することにしよう。しかし、あなたの自由を放棄して私の名義にするようなことだけは決してしないようにしていただきたい。

初出: 本稿は従来未発表だったもので、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

*3　インターネットは、世界最大のコンピュータネットワークである。

第13章
科学は著作権を離れなければならない

科学文献が科学的知識の普及のために存在し、科学雑誌がその作業を促進するために存在することは、言うまでもないことである。そのため、科学文献の使用規則は、この目標の達成を助けるようなものでなければならない。

現在存在する著作権と呼ばれる法制は、コピーの一元的な大量製造方法という本質を持つ印刷機の時代に確立されたものである。印刷機の時代には、雑誌記事の著作権は、雑誌出版社のみに制限を加えるものだった。出版社は、論文を出版する許可を得なければならない。許可を得ずに出版すれば、剽窃者になってしまう。この制度は、論文の書き手であり、読み手でもある科学者や学生の有用な仕事の障害になることなく、雑誌の運営と知識の普及を助けるものだった。規則と体制がうまくフィットしていたのである。

しかし、現代の科学出版テクノロジーはWebである。Web上で科学論文や知識を最大限に普及させられるようにするには、どのような規則が良いのだろうか？

論文は、非私有フォーマットで、誰からもアクセスできる形で頒布されるようにすべきである。そして、すべての人が論文の「ミラーリング」、すなわち適切な帰属情報を付加した上で、本文に一切の変更を加えずに再出版する権利を持たなければならない。

これらの規則は、過去の論文だけではなく、最初から電子形式で流通する将来の論文にも適用すべきである。しかし、紙の形式の雑誌に適用される現行の著作権制度に特に変更を加える必要はない。その分野には問題はないからである。

残念ながら、誰もが本稿の冒頭で述べた当然のことを当然のこととは考えないものらしい。多くの雑誌出版社は、科学文献の目的は、科学者や学生からの購読

予約を集めて科学雑誌を出版できるようにすることにあると思っているようである。そのような考え方は、「手段と目的の転倒」と呼ぶべきものである。

出版社の手法は、雑誌を購入でき、料金を支払う人間以外に対しては、科学文献を読むことさえ制限するというものである。出版社は、コンピュータネットワークに対しては不適切な面を持ちながらまだ法的な効力を失っていない現行の著作権法を言い訳として、科学者が新しい法制を選ぶのを押し留めようとしている。

科学者の協力と人類の未来のために、私たちはそのような手法を根本から拒否しなければならない。つまり、単に今までに制定されてきた悪法に反対するだけではなく、その根拠となった優先順位の誤りも正していかなければならないということである。

雑誌出版社は、オンラインアクセスには強力なサーバマシンが必要であり、サーバの運営のためにはアクセス料を徴収しなければならないと主張することがある。この「問題」は、出版社の「解決方法」が必然的に招いた結果である。需要に応じてすべての人にミラーリングの自由を与え、世界中の図書館にミラーサイトのセットアップを認めればよい。このような中央主権的ではない解決方法は、ネットワークの帯域幅にかかる負担を削減し、高速アクセスを提供する上に、学術記録が事故によって失われることも防いでくれる。

出版社は、編集者に給与を支払うためにアクセス料が必要だと主張することもある。編集者に給与を与えなければならないという前提条件には同意しよう。しかし、本末転倒である。一般的な論文の編集コストは、論文を書くための研究プロジェクトの資金の1%から3%までの間である。そのようなごくわずかのコストを、研究成果の自由な利用を妨げる理由とするのはほとんど無理というものだろう。

編集コストは、たとえば、著者が負担するページ料として取り戻せばよいだろう。著者は、ページ料をそのまま資金提供者に転嫁する。資金提供者は、現在でも大学図書館の雑誌購読料という間接費を介するというより煩雑な方法で出版社にお金を払っているのだから、これを新しい出費だと考えてはならない。研究資金提供者に編集コストを負担させるように経済モデルを変更すれば、アクセス制限の明白な理由はなくなる。組織や企業に所属しておらず、資金提供者を持たない著者のページ料は免除し、その分は組織に属する著者が負担すればよい。

オンライン出版のアクセス料の言い訳としては、紙版の雑誌のバックナンバーをオンライン形式に変換するために資金が必要だというものもある。この仕事はぜひとも行わなければならないものだが、作業成果へのアクセスを妨げない別の資金調達方法を追求すべきである。作業自体は、通常の作業以上に難しいものではなく、それ以上にコストがかかるものではない。バックナンバーをデジタル化しながら、アクセスを制限して成果を無駄にするなら、本末転倒である。

合衆国憲法は、著作権は「科学の進歩を促進するため」に存在すると述べている。著作権が科学の進歩を妨げるなら、科学は著作権を離れなければならない。

初出: http://www.nature.com の Web Debates セクション、1991 年。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第14章
コピーレフトとは何か

コピーレフトは、プログラムをフリーソフトウェアにするとともに、そのプログラムのすべての変更、拡張バージョンもフリーソフトウェアにするための一般的な方法である。

プログラムをフリーにするためのもっとも簡単な方法は、それを著作権なしのパブリックドメイン（権利消滅状態）で差し出すことである。こうすれば、人々がそのつもりになれば、プログラムとその改良点を共有することができる。しかし、この方法では、プログラムを私有ソフトウェアに転換してしまうような協調性のない人間の存在も許してしまう。彼らは、多少の変更を加えて、その結果を私有製品として流通させる。修正が加えられた形のプログラムを手にした人々は、オリジナル版の作者が与えた自由を手にすることができない。仲介者が、自由を抜き取ってしまったのである。

GNU プロジェクトの目的は、すべてのユーザーに GNU ソフトウェアを再頒布、変更する自由を与えることである。仲介者が自由を抜き取ることができるのなら、私たちは多くのユーザーを得るかもしれないが、それらのユーザーは自由を持たなくなってしまう。そこで、私たちは GNU ソフトウェアをパブリックドメインにするのではなく、「コピーレフト」にすることにした。コピーレフトは、すべてのユーザーが自由を持つことを保証してくれる。

コピーレフトは、他のプログラマが何かフリーソフトウェアを追加しようという動機も与える。GNU C++コンパイラなどの重要なフリープログラムが存在する理由は、コピーレフト以外にはない。

コピーレフトは、フリーソフトウェアの改良に貢献したいと考えているプログラ

マが、実際にそれを行う許可を得るためにも役に立つ。これらのプログラマは、ほぼすべての業務の目的が金儲けになっている企業や大学のために働いていることが多い。プログラマが自分で書き換えた内容をコミュニティに提供したいと思っても、雇用主は変更後のプログラムを私有ソフトウェア製品に転換したがるものである。

しかし、私たちが改良版をフリーソフトウェアとして流通させなければ違法行為になるということを雇用主に説明すれば、通常、彼らは改良版を放棄するよりも、フリーソフトウェアとしてリリースすることを選ぶものである。

プログラムをコピーレフトにするために、私たちはまず、プログラムが著作権の管理下にあることを述べ、頒布条件を付加する。その頒布条件とは、頒布条件を変更しないという条件を満たす場合に限り、プログラムおよびそれから派生したプログラムのコードを利用、変更、再頒布する権利をすべてのユーザーに与える法的な装置である。このようにして、コードと自由は法的に分離不能になる。

私有ソフトウェア開発企業は著作権を使ってユーザーの自由を取り除くが、私たちは著作権を使ってユーザーの自由を保証する。私たちが右を左と言い換えて、「コピーライト」を「コピーレフト」と呼んでいるのは、そのためである。

コピーレフトは、一般的な概念である。細部の詰め方には多くの方法が考えられる。GNU プロジェクトが使っている具体的な頒布条件は、GNU 一般公開使用許諾書（General Public License）に含まれている。GNU 一般公開使用許諾書は、GNU GPL という略称で呼ばれることが多い。GNU GPL には、FAQ ページ（http://www.gnu.org/liccenses/gpl-faq.html）も用意してある。また、FSF が貢献者に著作権の譲渡を求めている理由も説明されている（http://www.gnu.org/copyleft/why-assign.html）。

コピーレフトのもう 1 つの形態である GNU 準一般公開使用許諾書（Lesser General Public License: LGPL）は、GNU ライブラリの多く（しかしすべてではない）に適用される。以前、このライセンスはライブラリ GPL と呼ばれていたが、この名前では、本来使うべきライセンスではなくこのライセンスを使うことを開発者に奨励することになってしまうので、名前を変更した。この変更が必要だった理由については、「次のライブラリでライブラリ GPL を使うべきではない

理由」(http://www.gnu.org/philosophy/why-not-lgpl.html) という論文を読んでいただきたい。

GNU ライブラリ一般公開使用許諾書は、準 GPL によって廃止になったが、HTML とテキストフォーマットではまだアクセスできる。

GNU 自由公開文書使用許諾書 (Free Documentation License: FDL) は、すべてのユーザーが営利、非営利を問わず、未変更または変更済みの文書をコピー、再頒布する自由を保証するマニュアル、教科書などのドキュメント用に作られたコピーレフトの一形態である。

多くのマニュアル、およびすべての GNU ソースコードディストリビューションには、適切なライセンス (使用許諾書) が含まれている。

GNU GPL は、作者が著作権保持者なら、自らのプログラムに簡単に適用できるように作られている。適用のために GNU GPL を変更する必要はない。プログラムに GNU GPL を参照せよという断り書きを追加すればよい。

GNU GPL か GNU LGPL でプログラムをコピーレフトにしたければ、GPL 学習ページの忠告を参照していただきたい (http://www.gnu.org/copyleft/gpl-howto.html)。GPL を使う場合には、GPL の全文を使わなければならないことに注意していただきたい。GPL は密接不可分の全体であり、部分的なコピーは認められていない (LGPL も同様)。

多くの異なるプログラムで同じ頒布条件を使えば、さまざまなプログラムの間でコードをコピーすることが容易になる。これらはみな同じ頒布条件を持っているので、頒布条件のずれを気にする必要はない。LGPL には、GPL の対象となっているほかのプログラムにコードをコピーできるようにするために、頒布条件を通常の GPL に変更できるようにする条項が含まれている。

GNU FDL でマニュアルをコピーレフトにしたい場合には、FDL のテキストの末尾にある指示と GFDL 学習ページ (http://www.gnu.org/copyleft/fdl-howto.html) を参照していただきたい。GNU GPL と同様に、ライセンス全文を使わなければならない。部分的なコピーは認められていない。

初出: 第1稿は1996年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第15章
コピーレフト：プラグマティックな理想主義

一人の人間が下す判断は、その人の価値観や目標に基づいている。人はさまざまな目標、価値を持つことができる。良き人が持つことのできる目標のほんの一部を挙げてみると、名声、利益、愛、生き残り、面白さ、自由などになるだろう。目標が自分自身だけではなく他者をも助けようというものなら、私たちはそれを理想主義と呼ぶ。

私のフリーソフトウェアの仕事は、理想主義的な目標を動機としていた。それは、自由と協力を広めることである。私は協力を禁ずる私有ソフトウェアを駆逐し、フリーソフトウェアの普及を奨励することによって、社会がより良いものになることを望んだ。

GNU一般公開使用許諾書（GNU GPL）がコピーレフトとして現在あるような形で書かれた根本的な理由はここにある。GPL対象プログラムに追加されたすべてのコードは、たとえ別のファイルに入れられていても、フリーソフトウェアでなければならない。私は、自分のコードをフリーソフトウェアの中では使えるが、私有ソフトウェアの中では使えないように変えたが、それは、ソフトウェアを書く他の人々が同じようにそれをフリーにするのを奨励するためである。私有ソフトウェア開発企業は著作権を使って私たちがコードを共有できないようにするが、私たち協同作業者は著作権を使って他の協同作業者に特別な利益を提供することができる。彼らは私たちのコードを使えるのである。

GNU GPLを使っているすべての人々がこのことを目標としているわけではない。何年も前のことだが、私たちのある友人がコピーレフト管理下のプログラムをコピーレフト以外の条件で再リリースすることを求められたとき、彼は概ね次

のように返答した。

> 私はフリーソフトウェアの仕事をすることもあるし、私有ソフトウェアの仕事をすることもある。しかし、私有ソフトウェアの仕事をするときには、報酬がほしい。

彼は、ソフトウェア共有コミュニティと自分の仕事を共有する意思を持っていたが、私たちのコミュニティからは無縁な存在になる製品を作る企業のためにただ働きする気もなかった。彼の目標は私たちの目標とは異なっていたわけだが、彼はGNU GPLを使うことが自分の目標のためにも役に立つと判断したのである。

世界で何事かを達成したいと思うなら、理想主義だけでは足りない。目標を実現するために役に立つ手法を選択する必要がある。つまり、「プラグマティック」でなければならない。GPLはプラグマティックだろうか。その成果を検証していきたい。

GNU C++について考えてみよう。私たちがフリーのC++コンパイラを持っているのはなぜか。それは、GNU GPLがフリーでなければならないと規定していたからであり、それ以外の理由はない。GNU C++は、GNU Cコンパイラを出発点としてMCCという業界内のコンソーシアムによって開発された。MCCは、通常なら自分の成果をできる限り私有ソフトウェアにしようとするところだが、C++フロントエンドはフリーソフトウェアにした。それは、GNU GPLがそれ以外にリリース方法はないことを明言していたからである。C++フロントエンドには多くの新しいファイルが含まれているが、GCCとリンクすることを考えて作られていたので、GPLが適用されたのである。私たちのコミュニティが得た利益は明らかだろう。

GNU Objective Cについて考えてみよう。NeXT[1]は、もともと、このフロントエンドを私有ソフトウェアにすることを望んでいた。彼らは、その部分を.oファイルとしてリリースし、GCCの残りの部分とのリンクはユーザーに委ねることを提案していた。こうすれば、GPLの要件を逃れられると思ったのである。しか

---

[1] スティーブ・ジョブズによって作られたオペレーティングシステム。最終的にAppleによって買収された。

し、私たちの弁護士は、それでは要件を満たさず、そのようなリリース形式は認められないことを通告した。そこで、彼らは Objective C フロントエンドをフリーソフトウェアにした。

これらの例は何年も前に発生したことだが、GNU GPL は依然として多くのフリーソフトウェアをコミュニティにもたらしてきている。

多くの GNU ライブラリは、GNU LGPL の対象となっているが、すべてのライブラリが該当するわけではない。たとえば、Readline ライブラリは、通常の GNU GPL の対象となっている。以前、私は Readline を使うように設計された非フリープログラムを見つけ、開発者に対し、それは禁じられていることを通告した。彼は、コマンドライン編集機能をプログラムから取り外してもよかったところだが、GPL のもとで再リリースすることにした。現在、それはフリーソフトウェアになっている。

GCC（あるいは Emacs、Bash、Linux など、GPL 管理下プログラム）の改良コードを書くプログラマは、企業や大学に雇われていることが多い。プログラマが改良点をコミュニティに返し、次のリリースで自分のコードを見たいと思うと、上司は次のように言うかもしれない。

> ちょっと待て。お前のコードは会社のものだ。共有など認めない。会社はお前の改良バージョンを私有ソフトウェア製品にすることに決めているのだ。

GNU GPL は、このようなときにプログラマを助ける。プログラマは、この私有ソフトウェア製品が著作権侵害になるだろうということを上司に示す。上司は、新しいコードをフリーソフトウェアとしてリリースするか、何もリリースしないかの二者択一を迫られる。ほとんどの場合、上司はプログラマが最初から考えていた通りにすることを黙認し、コードは次のリリースに組み込まれることになるだろう。

GNU GPL は、お人好しではない。GPL は、人々がしたいと思うことの一部に対して「ノー」と言うことができる。ユーザーの中には、これを悪いことだと主張する人々がいる。GPL は、「フリーソフトウェアコミュニティに招くべき」私

有ソフトウェア開発者を「排除」しているというのである。

しかし、私たちは彼らを私たちのコミュニティから排除しているわけではない。彼らのほうが、コミュニティに加わらないことを選択しているのである。ソフトウェアを私有のものにするという判断は、私たちのコミュニティから距離をおくという判断である。私たちのコミュニティの一員であるということは、私たちとの協同作業に加わるということである。彼らが加わることを望まないのなら、私たちは「彼らを私たちのコミュニティに招く」ことはできない。

私たちができるのは、彼らがコミュニティに参加するための誘因を作ることだけである。GNU GPL は、私たちの既存のソフトウェアから誘因を作り出せるように設計されている。「あなたが自分のソフトウェアをフリーにするなら、あなたはこのコードを使える」　もちろん、それですべてのプログラマを獲得できるものではないだろう。しかし、実際にプログラマを獲得できるケースはある。

私有ソフトウェア開発は私たちのコミュニティに寄与しないが、その開発者たちは私たちからの助けを望むことがよくある。フリーソフトウェアユーザーは表彰とか感謝といった形でフリーソフトウェアプログラマの自我をくすぐることができるが、企業が次のように言うと、プログラマは非常に魅力的に感じるのである。

> 君のパッケージをうちの私有プログラムに追加してくれないか。そうすれば、数千人の人々が君のプログラムを使うことになる。

この誘惑は強力だが、長期的に見れば、拒否したほうがずっと良い結果が得られる。誘惑や圧力は、間接的にやってきたときには認識しにくくなる。たとえば、私有ソフトウェアに便宜を図る方針を採用したフリーソフトウェア組織を介してやってきたときである。X Consortium（およびそれを継承する Open Group）が良い例を提供している。X Consortium は、私有ソフトウェアを作ってきた企業から資金を得ており、10 年間にわたってコピーレフトを使わないようにプログラマに説得活動を続けてきた。Open Group が X11R6.4 を非フリーソフトウェアにした今、X Consortium からの圧力を拒否してきた私たちの仲間は、私たちの抵抗に

感謝している[2]。

プラグマティックに言って、非常に長期にわたる目標について考えれば、このような圧力をはねのける意思は強化される。あなたがじっと我慢していれば、構築できる自由とコミュニティのことに気持ちを集中させていれば、そのために必要な強さが見つかる。「何かのために闘え。そうでなければ、何でもないものに負けるだろう」

そして、皮肉屋が自由を嘲笑い、コミュニティを嘲笑い、「不細工なリアリスト」が利益が唯一の理想だと言っても、彼らのことは無視すればよい。そしていつもコピーレフトを使うのである。

初出: 第 1 稿は 1998 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

[2] X11R6.4 が非フリーの頒布条件でリリースされてから数か月後の 1998 年 9 月、Open Group は決定を覆し、X11R6.3 のときに使っていたのと同じ非コピーレフトのフリーソフトウェアライセンスのもとで、X11R6.4 は再リリースされた。ありがとう、Open Group。しかし、あとから決定を覆しても、制約の追加が可能だったという事実から私たちが導き出した結論は無効にはならない。

第16章
ソフトウェア特許の危険

皆さんは、私のフリーソフトウェアの仕事をよくご存知でしょう。でも、今回はフリーソフトウェアの話ではありません。ソフトウェア開発を危険に晒す法律の濫用について話します。つまり、ソフトウェアの分野に特許法が適用されたときに発生することです。

しかし、ソフトウェアに特許を与えることについての話ではありません。そういう問題ではないので、ソフトウェアに特許を与えることが危険だと言うと、誤解を招いてまずいことになります。個々のプログラムに特許を与えても、それは今までと特に変わりはなく、基本的に無害です。私が言いたいのは、アイディアの特許です。すべての特許は、何らかのアイディアを含んでいます。ソフトウェア特許とは、ソフトウェアのアイディア、皆さんがソフトウェアを開発するときに使うアイディアを対象とする特許です。これがあらゆるソフトウェア開発の仕事を危機に晒すものなのです。

皆さんは、「知的財産権」という誤った用語を使っている人を見かけたことがあるでしょう。ご存知のように、この用語にはバイアスがかかっています。ほかに選択肢はいくらもあるのに、この用語は、対象になるあれこれを一種の財産として扱うという前提条件を持ち込んでいます。「知的財産権」という用語は、どの対象について使ったとしても、基本的な疑問の大半をあらかじめ封じてしまいます。明晰で開かれた精神による思考を導くものとは到底言えません。

この用語には、バイアスを広げることとは無関係なところでもう1つ別の問題があります。事実の理解さえ阻むのです。「知的財産権」という用語は、何にでも対応できます。この用語は、著作権と特許のようにまったく異なる分野の法律を

ひとまとめにします。この2つは、細部の隅々に至るまで、まったく異なる存在ですが、さらにまったく異なる分野である商標や、その他のあまりお目にかからないものまですべてひとまとめにしてしまいます。これらの中で、他のものと共通点を持っているものはありません。歴史的な起源だって違うし、対応する法律はまったく別々に作られています。対象となる生活領域だって違います。要請する公共政策問題だってまったく無関係ですから、十把一絡げにして考えたところで、得られる結論は馬鹿げたものになること間違いなしです。「知的財産権」なるものに対しては、まともな意味のある意見の持ちようがありませんから、明晰な思考を目指すなら、これらを混ぜこぜにしてはなりません。著作権について考えてから、特許について考える、著作権法について学習してから、別個に特許法について学習するのです。

著作権と特許の間の大きな違いをいくつか示しておきましょう。

- 著作権は作品の表現の細部を対象とし、アイディアを対象としません。特許はアイディアとアイディアの用途だけを対象とします。
- 著作権は自動的に発生しますが、特許は特許庁が出願に対して設定を行います。
- 特許には多額の経費がかかります。実際の出願にかかるコストよりも、出願書を書いてもらうために弁理士に支払うコストのほうが高いくらいコストがかかります。一般に、出願された特許が審査されるまで数年がかかりますが、特許庁の審査は恐ろしく杜撰です。
- 著作権はとてつもなく長い間存続します。150年も継続する場合があります。特許の期限は20年で、皆さんのほうが特許よりも長生きできるかもしれませんが、ソフトウェアのような分野の尺度からすればあまりにも長いと言えましょう。20年前のことを思い起こしてください。当時、PCはまったく新しいものだったのです。1982年にわかっていたアイディアだけでソフトウェアを開発するよう制限されていたらどうなるか、どうなるでしょうか。

- **著作権は、複製の作成だけを対象とします。『風と共に去りぬ』と一言一句まったく同じ小説を書いたとしても、『風と共に去りぬ』を読んだことがないことを証明できれば、著作権侵害で告訴されても、自分を守ることができます。**

  特許は、アイディアの絶対的な独占権です。たとえ、自分で独自に発明したアイディアだということを証明できたとしても、誰か他人がそのアイディアに対して特許権を持っていれば、その証明は何の役にも立ちません。

今回は特許についての話ですから、ここからは著作権のことは忘れるようにしてください。そして、これからは著作権と特許を混ぜこぜにしないようにしてください。これらの法律問題を明晰に理解するためには、そうしなければならないのです。

水とエタノールの区別がつかなければ、実験科学（あるいは料理）の理解はどうなってしまうか考えてみてください。

皆さんが誰かから特許制度の説明を聞くとき、通常、その人は特許権をほしがっている人の視点から説明します。まるであなたが特許を取ることがあるかのように、あなたが街を歩き回るときにポケットに特許状を持っていて、ひんぱんにそれを取り出しては、誰かに示し、「私にお金を支払いなさい！」と言えるかのような説明をします。

このようなバイアスがかかるのには理由があります。特許制度について説明するたいていの人は、この分野に利害関係を持っており、あなたがこの制度に好感を持ってくれるとよいなあと思っているのです。理由はほかにもあります。特許制度は宝くじのようなもので、特許保持者に実際に利益を運んでくるものは、全体のうちのごくわずかなのです。*The Economist* 誌は、以前特許を「時間のかかる宝くじ」に喩えたことがあります。宝くじの広告というのは、見ればわかるように、いつでも勝つことを考えるように誘惑します。負ける確率のほうがはるかに高いのですが、負けたらどうなるだろうなどと考えさせるような宣伝はありません。特許制度の宣伝にも同じことが言えます。いつでも勝者になることを考えるように誘いをかけているのです。

バランスを取るために、犠牲者の視点から特許制度について説明してみましょう。つまり、ソフトウェアを開発しようとしたものの、ソフトウェア特許制度の壁にぶち当たった人の視点から見てみようというのです。この人は下手をしたら告訴されてしまいます。

では、自分が書こうとしているプログラムについてアイディアが浮かんだら、まず最初に何をするでしょうか。

特許制度とうまくやっていくために最初にやっておきたいことは、あなたが書きたいプログラムに関わりのある特許がどれかを見つけ出すことです。でも、不可能ですね。

出願された特許の一部が保留状態になっている理由は、秘密にされています。特許が設定されるのは、18か月くらいの時間が経過してからです。しかし、それだけの時間があれば、プログラムを書き、リリースしても、それがいずれ特許の対象になり、あなたが告訴されることに気づかないということは充分にあり得ます。

これは机上の空論ではありません。1984年にデータ圧縮を行うcompressというプログラムが書かれました。当時、LZW圧縮アルゴリズムに特許は設定されていませんでした。その後、合衆国はこのアルゴリズムに特許を設定し、それから数年間、compressプログラムを頒布した人々は、びくびくさせられることになりました。

compressの作者は、まさか告訴されるかもしれなくなるとはとても予想できませんでした。彼は、プログラマがいつもするように、雑誌で見つけたアイディアを使っただけです。雑誌でアイディアを見つけたとき、そのアイディアはもう安全に使えないかもしれないなどとは思いもしませんでした。

この問題については忘れておくことにしましょう。設計された特許は、特許庁によって公表されます。ですから、その長いリストを探し、実際に何が書かれているかを見ることはできます。

もちろん、その長大なリストはあまりにもたくさんありますから、実際にリスト全体を読むことはできません。合衆国には、数百あるいは数千のソフトウェア特許があります。特許の状態がどうなっているかを把握する方法などありません。関係者を探さなければならないのです。

たとえば、スプレッドシートの自然順序再計算にはソフトウェア特許がかけら

れていました（今はもう消滅しているかもしれません）。これはどういうことかというと、他のセルに依存するセルを作ったとして、依存元が変わったら、関係するすべてのセルを再計算し、再計算が終了したときにはすべてのセルが最新の状態になるということです。最初期のスプレッドシートは、トップダウン方式で再計算を行っていましたので、下のほうのセルに依存するセルを作るとか、その手のことをすると、新しい値を上のセルに反映させるために数回にわたって再計算しなければならなかったのです（依存するセルは上にあるものだとされていたのです）。

その後、ある人が、なあんだ、依存しているものに合わせてすべてのセルを再計算すればいいじゃないかということに気が付きました。このアルゴリズムは、トポロジカルソートと呼ばれています。私が見つけられた最初の参考文献は1963年のものです。この特許は、トポロジカルソートを実装する数十種類の方法を対象としていました。

しかし、この特許は、「スプレッドシート」という単語で検索しても見つかりません。「自然順序」とか「トポロジカルソート」という単語で検索してもだめです。特許の中には、これらの用語は一切含まれていないのです。実際、この特許は、「式をオブジェクトコードにコンパイルする」方法と説明されていました。初めて見たとき、その特許は間違いかと思いました。

特許のリストを入手し、自分がしてはならないことを調べようと思ったとします。特許を調べてみるとすぐにわかりますが、これらは非常にひねくれたわかりにくい法律用語で書かれており、そう簡単には理解できません。特許庁が言っていることは、それらが意味しているように見えることを意味していないことがよくあります。

1980年代にオーストラリア政府が特許制度について研究したものがあります。その研究は、国際的な圧力がなければ、特許制度を持つ理由は見当たらない（国民にとっての利益は何もない）と結論付けており、国際的な圧力さえなければ制度を廃止することを推奨しています。論拠の1つとしては、特許が難しすぎて理解不能であるために、技術者たちが特許を読んで学習しようとしないことが挙げられています。ある技術者が「特許状に書かれている自分の発明を理解できなかった」と言ったことが引用されています。

これは、ただの理屈ではありません。1990年頃、ポール・ヘッケルというプログラマが、Hypercardによって彼の2つの特許が侵害されたと主張してAppleを訴えました。初めてHypercardを見たとき、彼は自分の特許だの「発明」だのとの間に関係があるとは思いませんでした。似たところなどなかったんですから。特許状を読むとHypercardの一部に特許に抵触する部分があるよと弁護士に吹き込まれたから、彼はAppleを攻撃する気になったのです。スタンフォードで私がこの話をしたとき、聴衆の中に彼がいました。彼は、「それは正しくない。私は自分の権利がどこまで保護されているのかを理解していなかっただけだ」と言いました。私は、答えました。「だからそう言ったでしょ」

というわけで、あれやこれやの特許が何を禁じているのかを知るためには、何時間もかけて弁護士と相談しなければなりません。最終的に、彼らが言うのはこんなことです。「ここでこういうことをすれば、あなたは確実に負けます。ここ（RMSが広いスペースを払いのける仕草をする）で何かをすれば、負ける可能性はかなり高いです。本当に安全でいたければ、この領域（さらに広いスペースを払いのける仕草）には、近づかないことです。ちなみに、訴訟の結果は、かなりの部分偶然に左右されます」

仕事をするための予想可能な領域が用意されたところで（！）、皆さんはどうしたらよいでしょうか。トライできるアプローチは3種類あり、どれも場合によっては使えます。

1. 特許を避ける。
2. 特許の実施権を得る。
3. 法廷で特許を無効にする。

特許を避ける

「特許を避ける」とは、特許の対象となっているアイディアを使わないということです。これは、アイディアが何かによって簡単にも難しくもなります。

場合によっては、ある機能が特許を受けていることがあります。その場合、その機能を実装しなければ特許を避けることができます。あとは、その機能がどれだけ重要かという問題になります。

その機能なしでやっていける場合はあります。かなり以前になりますが、XyWriteというワードプロセッサのユーザーたちは、略語をあらかじめ定義できる機能を取り除くダウングレードをメールで受け取りました。略語に続いて句読点を打つと、その略語を展開した内容に置き換わるという機能です。長いフレーズに対して略語を定義しておき、略語を入力すれば、そのフレーズが文書内に残るというわけです。開発元は、Emacs エディタに同様の機能があることを知っていたので、私にこのことを知らせてきました。Emacs にこの機能がついたのは 1970 年代からですね。何しろ、特許にできるアイディアを少なくとも 1 つ発明したことがわかったわけで、へーそうかと思いました。誰か他人があとで特許を取るまで、それで特許が取れるなんてわかりませんでしたよ。

実際のところ、XyWrite の開発元は、3 つのアプローチをすべて検討しました。まず、特許権者と交渉しようとしましたが、まともに話のできる相手ではありませんでした。次に、特許を無効にするチャンスを窺うこともしました。しかし、結局最終的には、その機能を諦めたのです。

この機能がなくても生きていくことはできます。失った機能がこれだけなら、人々はまだそのワードプロセッサを使い続けるでしょう。しかし、さまざまな機能が特許に該当し始めたら、人々はこんなプログラムは余り良くないと思い、プログラムを使うのをやめるでしょう。

今話したものは、非常に限定された機能に対するごく狭い範囲の特許です。しかし、ダイアルアップアクセスでハイパーリンクをたどることについての British Telecom の特許はどうしたらよいでしょうか。ハイパーリンクをたどることは、今日のコンピュータにとって欠くことのできない重要な機能です。ダイアルアップアクセスも非常に重要です。この機能なしでどうやっていったらよいのでしょ

うか。ついでに、これは1つの機能ですらありません。適当に並べた2つの機能の組み合わせです。これは、同じ部屋にソファとテレビを置くことに特許をかけるようなものです。

非常に広範で根本的なアイディアに特許がかかってしまって、ある分野全体が金縛りにかかってしまうことがあります。たとえば、合衆国で特許を取った公開鍵暗号化のアイディアなんかがそうです。この特許は、1997年に切れましたが、それまでは合衆国内で公開鍵暗号化が使われるのを大きく阻んできました。いくつものプログラムが開発途上で頓挫しました。特許権者に脅されたために、どうしてもリリースできなかったのです。その後、PGPという最初はフリーソフトウェアだったプログラムがリリースにこぎつけました。特許権者は、攻撃の態勢を整える前に、このままではえらく評判が悪くなるぞと思ったのでしょう。非営利目的なら利用できるように制限を緩めました。こうすれば、そんなに騒がれることはありません。彼らは、10年以上にわたって公開鍵暗号化の使用を大きく制限しました。この特許は避けようがなく、公開鍵暗号化に代わるものは他にありませんでした。

特定のアルゴリズムが特許化される場合もあります。たとえば、高速フーリエ変換（FFT）の最適化バージョンにかけられている特許です。速さがだいたい2倍になりますが、普通のFFTを使えばこの特許は避けられます。プログラムのその部分は2倍の時間がかかることになります。ひょっとすると、引っかかるのはプログラムの実行時間のごくわずかの部分で、大した問題にはならないかもしれません。2倍もかかっているのに、ちっとも気が付かないかもしれません。しかし、その仕事に2倍もの時間がかかるので、プログラムは全然動かなくなっちゃうかもしれません。効果はまちまちです。

場合によっては、もっといいアルゴリズムを見つけられることもあります。うまくいく場合もそうでない場合もあります。GNUプロジェクトではcompressが使えなかったので、別のデータ圧縮アルゴリズムを探し始めました。すると、いいアルゴリズムがあるよと知らせてきた人がいました。彼はすでにプログラムを書いており、それを寄付することにしたというのです。私たちは、それをリリースするつもりでした。その頃、偶然*New York Times*を見たところ、たまたま週に1度の特許のコラムが目に入りました（私が*Times*を見るのは、数か月に1回

だけです)。それで、読んでみると、誰かが「新しいデータ圧縮方式の発明」の特許を取ったというのです。この特許は読んでおいたほうがよかろうと思い、コピーを取り寄せると、リリースまであと1週間というところまでこぎつけたプログラムがそれに引っかかることがわかりました。そのプログラムは、ボツになりました。

その後、特許と関係のない別のアルゴリズムが見つかり、それがgzipのもとになりました。今、gzipはデータ圧縮の分野では、事実上の標準になっています。データ圧縮プログラムで使うアルゴリズムとして、それは優れていました。データ圧縮を実行したい人は、誰でもcompressの代わりにgzipを使うことができます。

GIFなどのイメージフォーマットでも、同じ特許が引っかかるLZW圧縮が使われています。しかし、この分野でユーザーが本当にやりたいことは、単純にデータを圧縮することではなく、それぞれのソフトウェアで表示できるイメージを作ることだったので、異なるアルゴリズムへの切り替えは、とても大変でした。10年たってまだ実現できていないのです！　確かに、GIFファイルを使っているために訴えられる脅威が現実のものとなってから、gzipのアルゴリズムを使った新しいイメージフォーマットが定義されました。私たちがGIFファイルを使うのを止めて、こちらのフォーマットに切り替えるように呼びかけを始めたとき、人は、「切り替えはできないよ、ブラウザがまだ新しいフォーマットをサポートしていないもの」と言いました。ブラウザの開発元は、「その仕事については急ぐ予定はありません。何しろ新フォーマットは誰も使っていないのですから」と言いました。

このようにGIFを使うという社会内の慣性が強く働いたため、私たちはまだ新フォーマットへの切り替えに成功していません。基本的に、コミュニティがGIFを使っているということが、GIFの使用を後押ししており、コミュニティを脅威に晒す結果を招いています。

実は、この問題には、さらに変な話がくっついています。LZW特許は、実際には2つあるのです。特許庁は、同じものに2つの特許を設定していることにさえ気づかず、自分がしたことさえ把握できない。しかし、これには理由があります。2つの特許を見て、両者が実際には同じものを対象としていることに気づくまでには、かなり時間がかかるのです。

何らかの化学処理のようなものに対する特許なら、話はもっと簡単だったはず

です。どのような物質が使われており、入力が何で、出力が何で、どのような物理操作を加えたのかはすぐにわかるでしょう。どのような書き方になっていても、それが何か、2つのものが同じかどうかはわかります。しかし、純粋に数学的なものの場合、記述方法はいくつもありますし、その中にはかなりかけ離れたものも含まれます。それらは、一見同じものには見えません。本当は同じことを言っているんだと気づくためには、本当に理解しなければなりません。特許庁には、それだけの時間がありません。数年前の時点で、合衆国特許庁は、1つの特許について17時間ずつを使っていました。じっくり考えるにはあまりにも短いので、当然、このような誤りも犯します。先ほど、ボツになったプログラムの話をしましたが、そのアルゴリズムも米国内で2つの特許に引っかかっていたのです。これは決して特別なことではありません。

特許を逃れるのは簡単な場合もあれば、不可能な場合もあります。簡単でもプログラムが使い物にならなくなることもあります。どうなるかは、状況次第です。

私が言いたいことはもう1つあります。企業やコンソーシアムは、あるフォーマットやプロトコルを事実上の標準に押し上げることができます。その後で、そのフォーマットやプロトコルに特許がかけられたら、本当に深刻な打撃になります。世の中には、特許によって制限が加えられた公式標準さえあります。2001年にW3C（World Wide Web Consortium）が特許がかかっている技術を標準として採用しようと提案したときには、大騒ぎになりました。コミュニティの反対のために、W3Cは提案を取り下げ、すべての特許は誰もが自由に実装できなければならない、そして標準は誰もが自由に実装できなければならない、という主張に引き戻されました。これは、面白い意味のある勝利です。標準策定団体がこのような決定を下したのは、初めてなんじゃないかなあ。普通なら、標準策定団体は、標準の中に特許で制限されているものを突っ込み、人々が先に進んで自由に実装することを禁止する方向に走るものです。私たちは他の標準策定団体にも働きかけ、規則の変更を要求しなければなりません。

特許の実施権を得る

特許を避けるというのではない第 2 の可能性は、特許の実施権の取得です。もっとも、うまくいくとは限りません。特許権者は、実施権を提供しなくてもよいのです。実施権提供は義務ではありません。10 年前、プログラミング自由連盟（LPF: League for Programming Freedom）は、ある人物から助けを求める手紙を受け取りました。その人物は、家業でカジノ用ギャンブルマシンを作っており、そのマシンは当時すでにコンピュータを組み込んでいました。彼は、別の会社から、「うちは特許を持っている。お前にはそんなものを作る権利はない。今すぐ止めろ」と脅しをかけられたのです。

私はその特許を見ました。ネットワーク上にいくつかのコンピュータを置き、個々のコンピュータが複数のゲームをサポートして、同時に複数のゲームを楽しめるようにすることを対象としていました。

特許庁は複数のものをすることが何かすばらしいことだと本気で考えている節があります。彼らは、コンピュータ科学においては、こうすることが一般化のためのもっとも自明な方法だということを知らないのです。1 度やったことを複数回やりたければ、サブルーチンを作ればよい。しかし、特許庁は何かを複数回行ったら、それをした人は優れているのだから、その人には誰も争えず、その人がみんなを支配していいんだと思っているのです。

いずれにせよ、彼は実施権を得られず、手を引かざるを得ませんでした。彼には、裁判に訴える資力さえなかったのです。その特許は特別な発明ではないと主張できたはずですし、判事もその通りだと思う可能性もありましたが、彼が諦めたので、どういう結果になっていたかはわかりません。

もっとも、多くの特許権者は、実施権を提供しています。ただし、そのためにかなり高額の料金を要求することがよくあります。例の自然順序再計算特許の会社は、米国のすべてのスプレッドシートの総売上の 5%を要求していましたが、訴訟になる前の価格としては安いほうだと言われました。実際に訴訟になって相手方が勝てば、もっと高額の料金を要求されることになるでしょう。

この 1 つの特許の実施権のために売上の 5%を支払うことは可能だとしても、プログラムを作るために 20 種類の特許の実施権が必要だとしたらどうなるでしょ

うか。入ってきた売上は、すべて特許のために消えちゃいますねえ。21種類の特許の実施権が必要ならどうなるでしょうか。ビジネス界の人から聞いたところによれば、現実的に言って、そのような特許実施権が2、3件もあれば、ビジネスは成り立たなくなるそうです。

特許の実施権を得ることが非常にうまく機能する場合があります。それは、大規模な多国籍企業に勤務している場合です。その手の企業は無数の特許を持っていますので、互いに特許実施権を与え合うクロスライセンスを結ぶのです。大企業は、このようにして特許制度のほとんどの害を逃れ、利点だけを手にします。

IBMは、*Think*誌の、確か1990年の第5号だったと思いますが、それに手持ちの特許についての記事を公表しました。それによれば、IBMは、9000件の米国特許（現在はもっと大きな数字になっているはずです）から、2種類の利益を引き出していると言います。1つはロイヤリティ収入、もう1つは「他社の特許へのアクセス」です。IBMによれば、第2の利益のほうが桁違いに大きいということです。つまり、他社が特許を持っているアイディアをIBMでも使えるようになる利益は、IBMが特許の実施権を販売して得る直接的な利益の10倍だったということです。

これはどういう意味でしょうか。IBMが「他社の特許へのアクセス」から得ている利益とは何なのか。それは、基本的に、特許制度が引き起こす障害に巻き込まれないという利益です。特許制度は、宝くじのようなものです。特定の1つの特許は、何の意味もないかもしれないし、特許権者に棚ぼたの大もうけをさせるかもしれないし、他のすべての人々にとって大きな不幸の元になるかもしれない。しかし、IBMは非常に巨大ですから、これらの効果が平均化されます。だから、彼らは特許制度の利害を平均化して計算します。実際には、IBMにとって、特許制度の害は、利益の10倍だったはずなのです。

今「だったはず」と言ったのは、IBMがクロスライセンスによって実害を受けるのを避けているからです。その実害は潜在的なものであり、実際に降りかかってくるわけではありません。しかし、その実害を避けられた利益を計算するときに、IBMは特許から集めたお金の10倍だという数字をはじき出しているのです。

クロスライセンスのこの現象は、「飢えた天才」の神話や特許が「小さな発明家を保護する」という神話など、広く受け入れられている特許神話を打ち壊すもの

です（この手の話はプロパガンダですから、乗っからないように）。

シナリオはこんなところです。何かの分野の「優れた」設計者がいたとします。彼は、「屋根裏で極貧のうちに数年を」過ごし、新しくすばらしい種類の何やらを設計し、それを製造したいと考えます。そこに、大企業がやってきて彼と競争し、すべてのシェアを持っていってしまい、彼が「飢える」ことになったら、ひどい話でしょ？

まず、ハイテク分野の人々は、一般に自営では働いておらず、アイディアが真空からやってくるわけではない、つまり他人のアイディアをもとにしている、そして、仕事が必要なら、そういう人が職に着くチャンスはいくらでもあるということを言っておかなければなりません。ですから、一人で働いているこの優れた個人に優れたアイディアが浮かぶというこのシナリオは非現実的であり、彼が餓死する危険があるという考えもリアリティがない。

しかし、あのアイディア、このアイディアというように 100 件から 200 件くらいのアイディアを持っている誰かが、何らかの製品を作るための核となり、大企業が彼に競争を仕掛けてくるというシナリオは考えられます。彼が特許を使って大企業を排除しようとしたらどうなるか考えてみましょう。彼は言います。「だめだよ、IBM、ぼくと競争することはできない。この特許を持っているのはぼくなんだから」　IBM は答えるでしょう。「ちょっと、お前んとこの製品を見せてみろ。ふむふむ、うちはこの特許とこの特許とこれとこれとこれとこれだけ特許を持っているぞ。お前の製品のこれだけの部分が特許侵害だ。法廷で戦うつもりだと言うのなら、会社に戻ってうちの特許をもっと探してくるぞ。それでも、うちとクロスライセンスを結ばないつもりか」　すると、優れた小発明家は、「わかりました、クロスライセンスを結ばさせていただきます」と答えることになるだろう。これで彼は自分の会社に戻り、すばらしい何とやらを作れるようになりますが、IBM だって同じものを作れるのです。IBM は、彼の特許への「アクセス」を手に入れ、彼と競争する権利を手に入れます。この特許は、彼を全然「保護」していませんよね。特許制度は、神話通りの機能など果たしていないのです。

大企業は、特許制度の害の大半を避けられますので、自分の得になる方をメインに見ています。大企業がソフトウェア特許を持ちたがるのはそのためです。大企業は、ソフトウェア特許から利益を受けるのです。しかし、一人の小発明家で

あったり、中小企業で働いていたりしたらどうなるのか。中小企業は、大企業のようなことはできません。やってみても、(すべての相手とクロスライセンスを結ぶためには) 手持ちの特許が少ないのです。

特許はどれも、特定の方向を指差しています。中小企業がそことそことそこを指している特許を持っているときに、向こうの人 (別の場所を指して) がそっちを指す特許を持っていて、お金をくれと言ったら、中小企業はもうお手上げです。IBM は、すべての方向を指す 9000 件の特許を持っているので、それでも大丈夫です。どこに行っても、IBM はあなたの方を指す特許を持っていることでしょう。というわけで、IBM はほとんど必ずクロスライセンスを結べます。中小企業が誰かとクロスライセンスを結べるのは、ごくたまにです。中小企業が防衛のために特許がほしいと言ったとしても、自分を守れるだけの特許を取ることはできません。

しかし、IBM でさえ、クロスライセンスを結べない場合はあるかもしれません。それは、特許権を取り、他人からお金を搾り取ることだけを目的としている会社があった場合です。自然順序再計算特許の会社は、まさにそういう会社でした。この企業の唯一の業務は、人々に訴えるぞと圧力をかけ、実際に何かを開発している人々からお金を巻き上げることでした。

ついでに言えば、訴訟手続きには特許はありません。法律の専門家は、特許制度そのものを相手にするのがどれだけしんどいことかを知っているんでしょうね。結局のところ、その会社とクロスライセンスを結んで特許を手に入れることはできません。その会社はすべての相手からお金を搾り取ります。しかし、IBM のような会社は、それも業務遂行のためのコストの一部として充分に計算に入れているのでしょう。

特許実施権を得るという可能性については以上です。可能な場合もそうでない場合もあり、払える場合もそうでない場合もある。そこで、第 3 の可能性に目を向けることになります。

法廷で特許を無効にする

特許を得るためには、おそらく新規性、有用性、非自明性が必要です（おそらくと言ったのは、これがアメリカでの用語だからで、他の国にはほぼ同様の他の表現があると思います）。もちろん、特許庁がゲームに参加した時点から、「新規性」と「非自明性」の解釈は始まっています。「新規性」とは「うちの原簿には含まれていない」、「非自明性」とは「IQ が 50 の誰かさんには自明ではない」という意味のようです。

合衆国内のソフトウェア特許案件について研究しているある人物によれば（少なくとも、以前は研究していましたが、今も追いかけているかどうかは知りません）、特許庁の 90%の人間は、「クリスタルシティテスト」をパスしないだろうということです。つまり、特許庁の人たちが外に出てニューススタンドに寄り、何かのコンピュータ雑誌を買えば、それらのアイディアがすでによく知られたものだということがわかるはずだというのです。

特許庁は、自明に馬鹿げたことをしており、彼らが馬鹿だということを理解するには、別にこの分野の最先端を知っている必要さえありません。これはソフトウェアに限ったことではありません。私は有名なハーバードマウスの特許を見たことがあります。この特許は、ハーバード大学が発癌性の遺伝子を持つマウスを遺伝学的に作り出したあとに設定されたものです。その発癌性遺伝子はすでに知られており、すでに知られている種類のマウスにすでに知られているテクニックを使って導入されています。その特許は、任意の方法で任意の種類の哺乳類に任意の発癌性遺伝子を導入することを対象としています。これが馬鹿げていることを理解するために、遺伝子工学の知識はいらないでしょう。このような「請求のし過ぎ」はよくあることであり、合衆国特許庁は出願者に対象を広げることを勧誘することがあるそうです。基本的に、出願対象は、明らかに他者の先行研究にぶち当たりそうだと思うくらいのところまで広げるものだそうです。頭の中のスペースをどのくらい持ち逃げできるかイメージしてください。

プログラマは、さまざまなソフトウェア特許を見ると、「これは、滑稽なくらい自明だ！」と言います。しかし、特許庁の役人たちの手元には、プログラマが考えるようなことを無視することを正当化するありとあらゆる言い訳が用意されて

います。「でも、10年前とか20年前とかにどうだったかということを基準として考えなければいけませんよ」　彼らは、その後も何かをうんざりするほど言っていれば、あなたのほうがおかしくなってくることを知っています。どんなものでも、充分ばらばらにして充分に分析すれば、自明ではないように見えてきます。あなたは、自明性のすべての標準を見失うことでしょう、少なくとも、自明と非自明を分けるあらゆる標準を正当化できなくなります。するともちろん、彼らは一人残らず特許所持者のことを優秀な発明家というわけです。ですから、私たちは、彼らが権力を振りかざす資格を問うことはできないのです。

法廷に行くと、判事たちは、何が自明で何がそうではないかということについて、ほんの少しだけ厳格になるようです。しかし、問題は、そのために何百万ドルもかかるということです。

私が聞いたある特許裁判では、被告はQualcommだったと記憶していますが、判決は確か1300万ドルで、その大半が双方の弁護士報酬に化けたそうです。原告に残されたのは（Qualcommは負けたわけですから）、数百万ドルだったといいます。

特許が正当なものかどうかという問題のかなりの部分は、歴史的な偶然に左右されます。いつ正確なところ何が出版され、誰かがうまく見つけ出したのがその中のどれで、失われなかったのがそのうちのどれで、正確な日付がどうで、といった偶然です。多くの歴史的な偶然が、特許の有効/無効を決めてきました。

実際、British Telecomの「電話アクセスでハイパーリンクをたどる」件についての特許が1975年に出願されているのは、変なことです。私が初めてInfoパッケージを開発したのは、1974年のことだったと思います。Infoパッケージは、ハイパーリンクをたどれるようにするもので、ユーザーは電話を使ってダイアルアップし、システムにアクセスしていました。つまり、実際には私がこの特許の先行業績を作っていたことになります。これは、私の人生で2番目の特許にできるアイディアということになります。

しかし、その証拠は見つからないでしょう。私は、まさかこれが論文にして公表するほど面白いことだとは思いませんでした。だって、ハイパーリンクをたどるというアイディアは、エンゲルバートのエディタのデモから得たものだったわけで、公表すべき面白いアイディアを生んだのは彼です。私がしたことは、自分

では「貧乏人のハイパーテキスト」と呼んでいたもので、それはTECOのコンテキストで実装しなければならなかったからです。これは彼のハイパーテキストほど強力ではないものの、少なくともドキュメントのブラウジングには役立つもので、目的はまさにそれだったのです。そして、システムへのダイアルアップアクセスの方について言えば、確かに当時からありましたが、私には片方ともう片方に特別な関係があるなどとはとても思いつきませんでした。「見てください。私は貧乏人のハイパーテキストを実装しました。それに、聞いてください。コンピュータにダイアルアップ回線も接続したんです」などというタイトルの論文を出版しようとはとても思いませんでした。

私がこれを実現した正確な日付を調べる方法はないんじゃないかと思います。何らかの意味でこれを公表したことはあったでしょうか。ARPANET越しに来客を招き、私たちのマシンにログインしてもらったので、彼らはInfoを使ってドキュメントをブラウズしたはずです。彼らが私たちに尋ねてくれれば、彼らは私たちがダイアルアップアクセスしていたことを知っていたでしょう。ここからもわかるように、先行業績と認められるかどうかは、歴史的偶然によって決まるのです。

そして、エンゲルバートがハイパーテキストについての発表をしていますから、彼ら被告はもちろんそれを提出するでしょう。しかし、その発表には、コンピュータへのダイアルアップアクセスについては何も書かれていないので、それで充分な証拠になるかどうかははっきりしません。

法廷で特許を無効にするという可能性は選択肢の1つではありますが、特許を無効にできるだけの明確な先行研究を見つけられる場合でも、費用のことを考えれば問題外となることが多くなります。そのため、無効な特許、文字通り存在すべきではない特許（しかし、実際にはそのようなものが無数にありますが）は、大変危険な武器なのです。誰かが無効な特許であなたを攻撃したら、それは、あなたにとって非常に大きなトラブルになる可能性があります。先行研究を示して彼らを追い返せる場合もあるかもしれませんが、それは彼らがそのような形で怖がるかどうかにかかっています。「どうせ、はったりをかましているだけだろう。お前が実際に法廷に行けないことは百も承知さ。お前にはそんな余裕はない。だから、こっちはお前のことを訴えるまでだ」と思うかもしれないのです。

これら3種類の選択肢は、どれもあえて使う可能性のある手段ではあるものの、使えないケースもよくあります。次から次から襲ってくる特許に対処しなければなりません。これら3つの中のどれかが使えると思う場合、必ず次の特許、その次の特許、さらにその次の特許が現われます。地雷原を横切るようなものです。一歩前に進み、1つ設計上の決定を下すたびに特許を踏ん付けることはたぶんないでしょうから、数歩進むだけなら地雷が爆発しないこともあるでしょう。しかし、地雷原を完全に横切り、プログラムが大きくなっていくと、特許を踏ん付けないで目的のプログラムを開発できる可能性は下がっていきます。

さて、私はよくこう言われます。「他の分野にも特許はあるでしょう。なぜ、ソフトウェアだけが例外なのですか」　この言い方には、私たちみなが特許制度の犠牲にならなければならないという奇妙な仮定が紛れ込んでいることに注意してください。これは、「一部の人は癌になる。なぜあなたは例外でなければならないのか」と言っているのと同じです。当然ながら、癌にならない人はみんなよかったということなのです。

しかし、それは別としても、あまりバイアスのかかっていない、良い質問が含まれています。それは、ソフトウェアと他の分野に違いはあるのか、特許政策は分野によって変えるべきなのか、もしそうだとしたら、それはなぜか、ということです。

私の答えは、分野が異なれば特許と製品の関係も変わるので、特許政策は分野によって変えるべきだ、ということになります。

一方の極には、1つの化学式が特許になり、1つの特許が1つの製品だけを対象とする薬学があります。新薬が既存の特許に引っかかることはありません。この新しい製品のために特許が必要だとすれば、特許権者は新製品を開発した人になるはずです。

このような形態は、私たちの特許制度に対する素朴なイメージ、つまり、新製品を設計したら、「特許」を獲得するというイメージにぴったり合っています。これは、1つの製品について1つの特許があり、その特許は製品のアイディアを対象としているという考えです。このイメージは、分野によっては真実に近く、分野によっては真実から遠くかけ離れています。

ソフトウェアという分野は、真実からかけ離れた分野の最たるものです。1つ

のプログラムが無数の特許と交錯します。これは、通常ソフトウェアパッケージというものが非常に大規模だからです。ソフトウェアパッケージは、多くの異なるアイディアを組み合わせて作られています。プログラムが真新しいもので、ただコピーしただけのものではなくても、それはいくつかの異なるアイディアを組み合わせて使っています。もちろん、アイディアの名前を呼んだらそれらが魔法のように動き出すわけではないので、それらのアイディアは新しく書かれたコードによって具体化されているのです。プログラマは、アイディアをすべて実装しなければなりません。それらすべてのアイディアをその組み合わせで実装しなければならないのです。

1つのプログラムを書くときでさえ、無数の異なるアイディアを使うわけですから、その中の1つが誰かの特許の対象になっている可能性はあります。2つのアイディアが誰かによって組み合わせの形で特許化されている場合もあります。1つのアイディアの記述方法にはいくつかの異なる方法がありますから、1つのアイディアがさまざまな人々の特許の対象になることさえあります。プログラムに数千のアイディアが含まれていれば、他人がすでに特許を取っている危険のある場所が数千箇所含まれていることになります。

ソフトウェア特許がソフトウェアの進歩、ソフトウェア開発の仕事の障害になるのは、このためです。「一特許、一製品」の分野なら、特許が製品開発の障害になることはないでしょう。そういうことであれば、新製品を開発しても、誰かがすでに特許を取っているということはありません。しかし、1つの製品が無数の異なるアイディアの結合体になっている場合、新製品の一部または全部に、誰かがもう取った特許が含まれている可能性は高くなります。

実際、段階的に発展する分野に特許制度を押し付けるといかに進歩が遅れるかを示す経済調査があります。ソフトウェア特許の推進者たちは、「ええ、確かに問題はあるでしょう。しかし、どの問題よりも重要なのは、特許が発明を確実に促進するということです。このことは非常に重要ですから、特許によってほかにどんな問題が起きようがかまわないということになるのです」などと言います。もちろん、彼らとて、こんな馬鹿げたことを声高に言うことはしませんが、暗黙のうちに、特許が進歩を促進するなら、他のあらゆるコストよりもその価値を高く評価してもらいたいと思ってはいるのです。しかし、特許が進歩を促すなんてい

うことは決してありません。現在は、特許が進歩をどのようにして遅らせるかをきっちり示すモデルがあります。このモデル、段階的発展は、ソフトウェアの分野を非常によく説明してくれます。

ソフトウェアが、もう一方の極に位置するのはなぜでしょうか。それは、ソフトウェアの世界では、観念化された数学的なもの、オブジェクトを開発しているからです。この分野では、複雑な城を構築しながら、それを細い線の上に立たせることができますが、それは城に重さがないからです。他の分野では、現実のもの、物質が持つ厄介な癖と格闘しなければなりません。物質には、変えられない性質があります。何かを物質のモデルとして試してみることはできますが、モデルが実際の物質に合わなければ、その作業は困難になります。本当の問題は、物質を本当に働かせることだからです。

ソフトウェアの世界では、while 文の中に if 文を挿入したいと思ったとき、if 文が特定の周期で振動し、while 文に揺さぶりをかけ、最終的に while 文を壊してしまうのではないかなどと考える必要はありません。また、if 文を特定の高周波数で振動させると、他の変数に値が格納されるような信号が生まれるのではないかなどと考える必要もありません。if 文がどれだけの電流を流すかとか、while 文内に熱を放散するかどうかとか、while 文全体に電圧低下が発生して if 文が機能しなくなるのではないかといったことを心配する必要もありません。塩水の中でプログラムを実行したら、if 文と while 文の間に塩水が入り込み、腐食が起きるのではないかという心配はないのです［聴衆は、この間大笑い］。

変数の値を参照しているときに、20 回も参照したのでファンアウトを越えてしまうのではないかとか、静電容量がどのくらいかとか、値をチャージアップするだけの時間があるかといった心配はどこにもありません。

プログラムを書くときに、個々のコピーを物理的に組み立てる方法とか、while 文の中に if 文を挿入することが本当に可能かどうかとか、if 文が壊れたときにそれを取り除き、新しいものと交換するためにはどうしたらよいかとかいったことに煩わされる必要はありません。ソフトウェアには、心配しなくてもよい問題が非常にたくさんあります。その分、プログラムを書くということは、動作する現実のものを設計することと比べて非常に簡単なのです。

今の話は、奇妙に感じられたかもしれません。皆さんはおそらく、ソフトウェ

アの設計がいかに難しいかとか、これがいかに大きな問題で、それをどう解いたらよいか考えているところだといった話を聞いたことがあるでしょう。こういう話は、私が今言ったことと同じ問題について話しているわけではありません。私が比較したのは、複雑さが同程度で、部品数が同じ物理システムとソフトウェアシステムです。私が言ったのは、ソフトウェアシステムは、物理システムよりもはるかに設計が簡単だということです。しかし、さまざまな分野に取り組む人々の知性は同程度です。では、易しい分野に直面した人々は、どのような行動を取るでしょうか。先に押していくのです！　私たちの能力を極限まで押し進めていくのです。同じ規模のシステムが簡単なら、10 倍大きいシステムを作ろう。そうすれば、難しくなります。私たちがしているのは、そういうことです。私たちは、物理システムよりも部品数がはるかに多いソフトウェアシステムを作っています。

100 万個もの異なる部品を使う設計の物理システムは、巨大プロジェクトです。それに対し、100 万個の部品を持つコンピュータプログラムは、30 万行くらいのものでしょう。それくらいなら、数人で 2 年もかければ書くことができます。特に巨大なプログラムだとは言えません。現在の GNU Emacs の設計には数百万個の部品が含まれており、コードの行数は 100 万行になっています。これが資金と言えるような資金なしで、主として余暇を活用して成し遂げられたプロジェクトなのです。

ソフトウェアには、もう 1 つ楽なところがあります。物理製品を設計したら、次に、それを製造するための工場を設計しなければなりません。工場を建設するためには、数百、数千万ドルがかかります。それに対し、プログラムのコピーを作るためにしなければならないことは、「copy」と入力することだけです。この同じ copy コマンドで任意のプログラムをコピーできます。CD の形のコピーがほしい？　マスター CD を焼いて、CD 工場に送るだけです。CD 工場は、CD にどんなコンテンツが収められていても同じ装置でコピーすることができます。個々の製品のために専門の工場を建設する必要はないのです。これは、ものを設計する上で、非常に大きな単純化であり、非常に大きなコスト削減です。

自動車の新モデルの製造工場を建設するために 5000 万ドルを費やす自動車メーカーは、特許実施権の交渉のために数人の弁護士を雇うことができます。その気になれば、訴訟にも耐えられます。複雑さが同程度のプログラムの設計には、5

万ドルから10万ドルしかかかりません。比較すれば、特許制度に対応するためのコストは圧倒的です。自動車の機械的な設計と同程度の複雑さを持つプログラムの設計は、実際、1か月ほどの仕事でしょう。自動車にはいくつの部品があるでしょうか。つまり、コンピュータを内蔵していなければという話ですが[*1]。私が言いたいのは、優れた自動車を設計することが簡単だということではありません。自動車の中には、それほどたくさんの部品が含まれているわけではないということです。

数学的なオブジェクトを操作する場合、設計ははるかに簡単になります。ですから、ソフトウェアは他の分野とは本当に違うのです。私たちは日常的に、他の分野よりもずっとずっと大規模なシステムをごく数人で作っています。私たちの世界では、1製品1特許からはほど遠く、すでに特許取得済みの無数のアイディアが1つの製品に詰め込まれています。

これは、交響曲に喩えて説明すれば、もっともわかりやすいでしょう。交響曲も長大で、多くの音符を含んでおり、おそらく無数の音楽的アイディアを使っています。1700年代のヨーロッパ政府が、交響曲の進歩を促進するために、言葉で表現できる音楽的アイディアに特許を与えるヨーロッパ音楽特許庁を設立することにしていたらどうなっていたでしょうか。

時代は1800年前後だとします。皆さんはベートーベンで、交響曲を書きたいと思っています。すると、優れた交響曲を書くことよりも、音楽特許を侵害せずに交響曲を書くほうが難しいことに気づくでしょう。

あなたは不平を言いますが、特許権者はこう言うでしょう。「ベートーベン君、君が文句を言っているのは、自分のアイディアがないからだよ。君がしたいと言っていることは、我々の発明を盗むということじゃないか」　ベートーベンは、実際にそうだったように、新しい音楽的アイディアをたくさん持っていましたが、認識できる音楽を作るためには、つまり、聴衆が好み、音楽として認識できる音楽を作るためには、多くの既存の音楽的アイディアを使わなければならなかったの

*1　自動変速機には、おおよそ300から400種の部品が含まれており、変速機は、一般に自動車のコンポーネントの中でもっとも複雑なものである。変速機の設計には6か月から1年かかり、実際に構築、機能させるためにはさらに長い期間がかかることがある。しかし、500から800個の関数部品を持つプログラムは、実際のコードでは200から300行であり、優れたプログラマなら、おそらく1日から1週間でコーディング、テスト、デバッグできる。

です。音楽をまったく異なるものに発明し直し、人々が聞きたくなるようなものを作れるようなすごい人はいません。ピエール・ブレーズはそれをやってみようとしたと言いましたが、誰がピエール・ブレーズを聴くでしょうか。

コンピュータ科学全体を発明し直してまったく新しいものにすることができるようなすごい人はいません。もしそんなことをしても、ユーザーからはとても奇妙で使えないようなものを作ることになるでしょう。今日のワードプロセッサを見れば、数百もの異なる機能が含まれていることがわかるはずです。すばらしい画期的な新ワードプロセッサを開発したとして、つまり、いくつかの新しいアイディアが含まれているということですが、そのワードプロセッサには、数百の古いアイディアも含まれることになるでしょう。古いアイディアを使うことが許されなければ、革新的なワードプロセッサを作ることはできません。ソフトウェア開発の仕事は非常に大規模なので、新しいアイディアが生まれるのを促進するために人為的な制度を用意しておく必要はないのです。人々にソフトウェアを書かせておけば、彼らはきっと新しいアイディアを生み出します。皆さんがプログラムを書きたいと思い、それを優れたものにしたいと思うなら、きっといくつかのアイディアが浮かんでくるはずですし、その中のいくつかについては使い道が見つかるはずです。

私はソフトウェア特許が現れる前からソフトウェアの世界にいるのではっきり言えますが、以前は、ほとんどのプログラマは、自分でなかなかのものだと思えるような、尊敬や名誉に値すると思えるような新しいアイディアが生まれたら、必ず公表していたものです。あまりにも小さい、あるいはぱっとしないアイディアなら、ばかばかしいからいちいち公表しません。現在、特許制度は、アイディアの開示を奨励するためのものだとされています。実際には、昔でも、アイディアを秘密にしていた人はいません。コードを秘密にしていた人はいました。これは事実です。結局、仕事の大部分は、コードによって表現されるわけですから。雇われプログラマたちは、コードを秘密にしてアイディアを公表することにより、名誉を勝ち取るとともに、いい気分を味わっていました。

ソフトウェア特許が登場してから、彼らはコードを秘密にするとともに、アイディアに特許をかけました。ですから、まともな意味での情報開示は奨励されなくなったのです。以前秘密だったものは今でも秘密で、以前公表されていたアイ

ディアは、今では特許の対象となり、20年もの間手の届かないものになってしまいます。

国はこのような状況を変えるために何ができるでしょうか。私たちはこの問題を解決するために政策をどのように変えたらよいのでしょうか。

手をつけられる場所は2つあります。1つは特許が発行される場所、すなわち特許庁で、もう1つは特許が適用される分野、つまり特許の対象となるのは何かという問題です。

たとえば、特許の発行基準を正しく保つ方法があります。今までソフトウェア特許を認めていない国、たとえばヨーロッパのほとんどの国がこれに当たります。欧州特許機構（EPO）の規則に、ソフトウェアは特許の対象とならないと明確に断りを入れておけば、ヨーロッパの場合には良い解決方法になります。ヨーロッパは今、ソフトウェア特許のEU指令を検討しています（指令の対象範囲はもっと広いものかもしれませんが、その中の柱の1つがソフトウェア特許であることは間違いありません）。単純にこの指令に変更を加え、ソフトウェアのアイディアには特許を与えられないと規定すれば、ヨーロッパのほとんどの地域では問題をシャットアウトできます。ただし、いくつかの国はすでにソフトウェア特許を認めているので例外になります。皆さんには残念なことですが、英国もその1つです。

アメリカではこのアプローチは通用しません。というのは、アメリカにはすでに大量のソフトウェア特許があるからです。ですから、特許の発行基準に変更を加えても、既存の特許を取り除くことはできません[2]。したがって、アメリカでの問題解決方法は、特許の適用範囲を変更することでなければなりません。ソフトウェアによる純粋な実装、汎用コンピュータハードウェア上での実行は、それ自体では特許の侵害とはならず、特許の対象にはならないと規定すれば、ソフトウェアを開発したからといって訴えられるようなことはなくなるでしょう。これ

---

[2] 私は「ソフトウェア特許」という言葉を使っているが、この言葉は何を意味するのだろうか。米国特許庁は、公式には特許をソフトウェア特許とその他の特許というようには分類していない。しかし、実際には、ソフトウェアに適用できる特許は、ソフトウェアを開発したことについて訴えを起こすために使えるはずである。だから、ソフトウェア特許とは、ソフトウェアに適用される可能性のある特許、ソフトウェアを開発したことについて訴えを起こすために使える特許と言うことができる。

がもう 1 つの解決方法です。

第 1 の解決方法、どのような種類の特許が有効になるかについての解決方法は、ヨーロッパでうまく機能するはずです。

アメリカがソフトウェア特許を認め始めた頃、政策論争はまったくありませんでした。実際のところ、誰も問題に気づかなかったのです。ソフトウェアの世界の大半が、問題に気づきもしませんでした。1981 年、最高裁はゴムの加硫処理の特許案件について判決を下しました。この判決は、加硫処理装置の一部にコンピュータとプログラムが含まれているからといって特許の対象にならないわけではないというものでした。次の年、すべての特許案件を扱う上訴裁判所が、限定語を逆転させました。つまり、コンピュータとプログラムが含まれていれば、特許の対象になるとしたのです。何であれ、コンピュータとプログラムが使われていれば特許の対象になるのです。アメリカで業務手続に特許を認め始めたのはそのためです。業務手続はコンピュータで遂行されますから、特許対象内になったのです。

この判決がこのときのものですから、自然順序再計算特許は、ソフトウェア特許の中でも最初のほうのもの、あるいは最初のものだったのではないかと思います。

80 年代を通じて、私たちはこの問題を知りませんでした。アメリカのプログラマがソフトウェア特許の危険性に直面して意識的になり始めたのは、1990 年前後でした。私は、ソフトウェア特許以前の現場と以後の現場の両方を知っていますが、1990 年以降、作業が特に高速化されたということはありません。

アメリカには政策論争はありませんでしたが、ヨーロッパには大きな政策論争がありました。数年前、欧州特許機構を設立したミュンヘン条約の改正問題がありました。ミュンヘン条約には、ソフトウェアは特許不能という条項が含まれていましたが、改正派は、ソフトウェア特許を認めるよう圧力をかけてきたのです。しかし、コミュニティはこの動きに気づきました。実際に、闘いを導いたのは、フリーソフトウェア開発者とフリーソフトウェアユーザーでした。しかし、ソフトウェア特許によって脅かされるのは、私たちだけではありません。すべてのソフトウェア開発者がソフトウェア特許によって脅かされますし、ソフトウェアユーザーでさえ脅かされるのです。

たとえば、ポール・ヘッケルは、Appleが彼の脅迫にそれほど脅威を感じていなかった時期に、Appleユーザーを訴えると脅しをかけました。Appleは、これには大きな脅威を感じました。最終的に勝訴できたとしても、このようにしてユーザーが訴えられたのでは、会社がもたないと考えたのです。ここからもわかるように、開発者を攻撃するための手段として、純粋にユーザーからお金を搾り取るため、さらには騒ぎを起こすために、ユーザーが訴えられる可能性はあります。すべてのソフトウェア開発者とユーザーは、特許侵害の訴えを起こされる危険性を持っているのです。

しかし、ヨーロッパで反対運動をリードしたのは、フリーソフトウェアコミュニティでした。実際、現在のところ、欧州特許機構加盟国のうち、条約の修正に反対している国は賛成している国の2倍あります。その後、EUが乗り出しましたが、この問題を扱う欧州委員会の担当総局は分かれています。ソフトウェアの奨励を仕事としている担当総局は、ソフトウェア特許に反対しているように見えますが、この問題にあまり熱心ではありませんでした。熱心だったのは域内市場担当総局で、ソフトウェア特許に好意的な人々によって牛耳られていました。彼らは、基本的に自分たちに向けて表明された世論を無視しました。彼らは、ソフトウェア特許を認める方向を提案したのです[3]。

フランス政府は、すでにソフトウェア特許への反対を表明しています。人々は、ヨーロッパの他の各国政府に対してソフトウェア特許に反対するよう働きかけていますが、その運動がはじまったのがここ、イギリスであることは非常に重要なことです。ヨーロッパにおけるソフトウェア特許反対運動のリーダーの1人であるハートマット・ピルチによれば、彼らの運動の最大の動因になっているのは、英国特許庁です。英国特許庁は、単純にソフトウェア特許に肩入れをしています。英国特許庁は公開協議を行いましたが、回答の大半はソフトウェア特許に反対でした。その後、英国特許庁は、これらの回答を完全に無視し、国民はソフトウェア特許に賛成しているとする報告書を書きました。フリーソフトウェアコミュニティは、「回答は特許庁と私たちの両方に送ってください」と呼びかけています。

---

[3] 【訳注】RMSはOpen Market Directorateと言っているが、http://lwn.net/2001/0315/a/eurolinux-consult.php3などを参照する限りでは、Internal Market Directorate-Generalのことを言っているものと思われる。

コミュニティで回答を公表すれば、特許庁の見解の反対になります。英国特許庁が発表している報告書から、世論の動向を知ろうとしてはなりません。

彼らは、「技術的効果」という用語を使います。この用語は、どこまでも引き伸ばせる便利な言葉です。技術的効果と言われれば、プログラムのアイディアに特許を与えられるのは、具体的な物理的動作に対応している場合に限られると思うのが普通でしょう。そういう解釈であれば、問題はほとんど解決できます。特許を与えられるソフトウェア上のアイディアというものが、プログラムを使わなくても特許を与えられるような特定の技術的物理的結果と結び付いていることであれば、問題はありません。問題は、用語を引き伸ばして使うことです。任意のプログラムを実行して得られる結果を物理的な結果と称することができてしまうことです。この物理的結果なるものは、他の結果とどこが異なるのでしょうか。コンピュータ処理の結果だということに過ぎません。このように、英国特許庁は、問題をほとんど解決するかに見えるようなものを提案することによって、ほとんど何にでも特許を与えられる白紙委任状を与えようとしているのです。

同じ省庁の人々が著作権問題にも携わっていますが、著作権とソフトウェア特許は、同じ人が処理しているということを除けば、まったく無関係です（おそらく、「知的財産権」という用語によって、2つの問題をひとまとめにしてしまったのでしょう）。EUは、米国のデジタルミレニアム著作権法（DMCA）によく似た恐ろしい指令を出しましたが、それの解釈が問題です。国によってそれを具体化する方法には、ある程度の自由があります。英国は、この指令を具体化するための方法としてはもっとも厳格なものを提案していますが、具体化の方法次第では、指令の害を大幅に緩和できます。英国は、この指令の専制的な効果を最大限に引き出したがっています。特定のグループ——通商産業省でしょうか——については、歩調を緩めさせる必要があるようです。彼らの活動にチェックを入れ、権力の新しい形を作り出すのを止めさせなければなりません。

ソフトウェア特許はすべてのソフトウェア開発者とすべてのコンピュータユーザーを新しい形の規制に縛り付けます。コンピュータを使っている産業が、これによってどれだけ大きな障害が生まれるかを認識すれば、彼らも立ち上がるでしょうし、この動きを阻止できると思います。産業は、規制に縛られるのを好むものではありません。もちろん、官僚たちには重要な役割があります。英国政府には、

動物の運搬など[*4]、産業の規制に関してよりていねいな仕事をしてほしかった分野がいくつかあります。しかし、人工的な独占を作り出して、ソフトウェア開発を妨害できる人物を作り出すような、その人物によって開発者やユーザーがお金を搾り取られるような効果しか見込めないような規制は、断固として拒否すべきです。私たちは、ソフトウェア特許が企業に及ぼす影響を経営者たちに意識させ、ヨーロッパにおけるソフトウェア特許反対運動を支援するよう経営者たちに仕向ける必要があります。

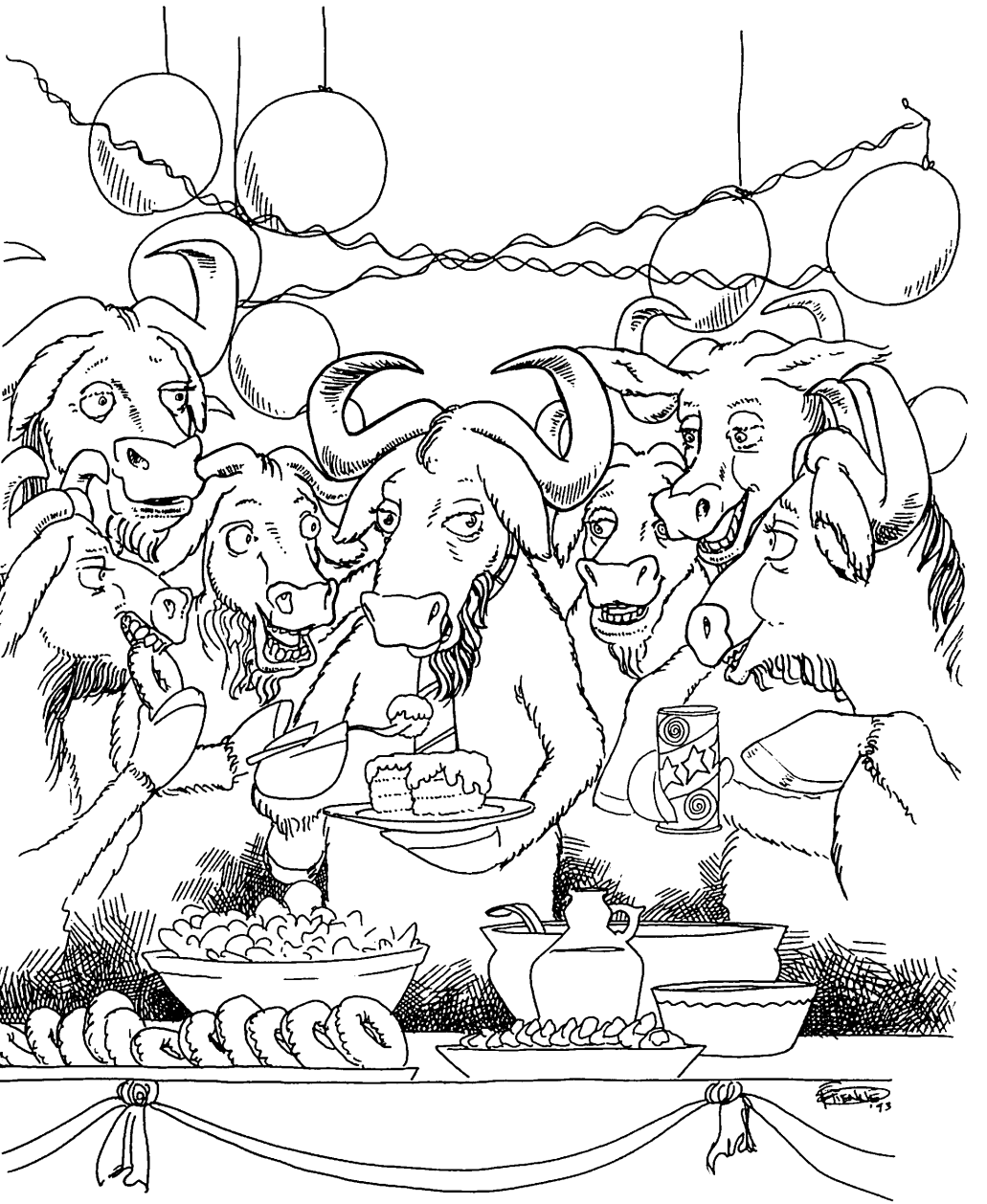
闘いは終わっていません。まだ勝つことはできます。

初出: 2002年3月25日にロンドンのケンブリッジ大学で行われた講演。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1の一部である。

*4　厳しい規制があれば、口蹄疫の蔓延は防げたはずである。

第3部

自由、社会、ソフトウェア



第17章
自分のコンピュータを信用できるか

コンピュータは誰の命令を受けるべきだろうか。ほとんどの人は、誰か他人ではなく、自分の命令を受けるべきだと考えるだろう。しかし、「信託コンピューティング（trusted computing）」と呼ばれる計画では、大手メディア企業（映画会社とレコード会社を含む）とMicrosoftやIntelなどのコンピュータ企業は、あなたではなく、彼らの指示を受けるようにコンピュータを改造することを計画している。私有プログラムは、以前から悪意のある機能を含んでいたが、この計画はそれを普遍化しようというものである。

私有ソフトウェアとは、基本的にユーザーがプログラムの動作をコントロールできないソフトウェアを意味する。ソースコードを研究したり、変更したりできないプログラムのことである。ずる賢いビジネスマンが自分の力を利用してあなたを不利な立場に追い込む手段を探すのは驚くべきことではない。Microsoftは、そのようなことを数回行っている。Windowsのあるバージョンは、ハードディスク上のすべてのソフトウェアをMicrosoftに報告するように作られていた。Windows Media Playerの最近の「秘密の」アップグレードでは、ユーザーは新しい制限に従わざるを得なくさせられている。しかし、問題はMicrosoftだけではない。KaZaa音楽共有ソフトウェアは、KaZaaのビジネスパートナーがあなたのコンピュータのCPU時間を顧客に貸し出せるように作られている。これら悪意に満ちた機能は秘密にされていることが多いが、たとえ存在に気づいたとしても、手元にソースコードがないので、取り除くことは難しい。

しかし、従来、これらはそれぞれ別個の問題だった。「信託コンピューティング」は、これを全面的な問題にする。むしろ、「背信コンピューティング（treacherous

computing)」とでも呼ぶべきところだ。なぜなら、この計画は、コンピュータがシステマティックにユーザーに背くようにしようというものだからである。実際、この計画は、コンピュータが汎用コンピュータとして機能しないようにする。すべての操作が明示的な許可を必要とするのである。

背信コンピューティングの技術的なアイディアは、コンピュータにデジタル暗号化、署名装置を埋め込み、ユーザーからは鍵がわからないようにするというものである（Microsoftのこれに対応しているバージョンは、「Palladium」と呼ばれている）。私有プログラムは、この装置を使って、ユーザーが実行できる他のプログラム、アクセスできるドキュメントやデータ、データを渡せるプログラムをコントロールする。これらのプログラムは、Internetを介して新しい権限付与規則を絶えずダウンロードし、あなたの作業に自動的に適用する。コンピュータがInternetから定期的に新規則を取得しないようにしようとすると、何らかの機能が自動的に働かなくなる。

もちろん、ハリウッドやレコード会社は、「DRM（Digital Restrictions Management：デジタル規制管理）」のために背信コンピューティングを利用し、ダウンロードされたビデオや音楽が特定の1台のコンピュータだけで再生できるようにすることを考えている。少なくとも、これらの企業から得た権限付与ファイルを使わなければ、共有はまったく不可能である。私たち一般人は、これらのものを共有する自由も能力もともに持つべきところである（誰かが暗号化されていないバージョンを作る方法を見つけ、それをアップロード、共有すれば、DRMは頓挫するが、だからと言ってシステムが免罪されるわけではない）。

共有を不可能にするだけでも充分悪いことだが、この計画にはさらに問題点がある。電子メールやドキュメントにも同じ機能を使う計画があるのだ。そのため、電子メールが2週間で消えたり、ある企業のコンピュータ上でなければドキュメントが読めなくなったりするのである。

たとえば、あなたが危険だと思うような仕事をするように指示する電子メールが上司から送られてきたとする。1か月後、その仕事が裏目に出たとき、電子メールを使って決定者が自分ではないことを証明することができなくなる。命令が消えるインクで書かれていたら、「書面を取っておく」方法では自分を守れない。

あるいは、会社の監査記録を破棄するとか、チェックなしで国を動かすような

危険な脅しをかけるとか、違法な、あるいは倫理的に許されない方針を指示する電子メールを上司から受け取ったとする。今日なら、それを記者に送れば、悪事は露見する。しかし、背信コンピューティングが実現すれば、記者はドキュメントを読むことができない。彼女のコンピュータは、彼女に従うのを拒否するようになる。背信コンピューティングは、腐敗の楽園を招く。

Microsoft Word などのワープロが背信コンピューティングを使えば、文書を保存するときにライバルのワープロではそれを読み出せないようにすることができる。現在、私たちは、Word 文書を読み出せるフリーのワープロを作るために、骨の折れる実験を重ねて Word フォーマットの秘密を暴かなければならないと考えている。しかし、Word が文書を保存するときに背信コンピューティングを使って文書を暗号化するなら、フリーソフトウェアコミュニティは、それを読み出すソフトウェアを開発することはできない。もし開発できたとしても、そのようなプログラムはデジタルミレニアム著作権法によって禁止されてしまうだろう。

背信コンピューティングを利用するプログラムは、Internet を介して絶えず新しい権限付与規則をダウンロードし、あなたの作業に自動的にそれらの規則を押し付ける。あなたが自分の文書に書いたことが Microsoft や合衆国政府のお気に召さないものなら、彼らはすべてのコンピュータにその文書の読み出しを拒否せよという新しい命令をポストできる。各コンピュータは、新しい指示をダウンロードすると同時にその指示に従うようになるだろう。あなたの作品は、『1984』風の遡及的な消去の危険に晒されるのである。自分で自分が書いたものを読めなくなるかもしれない。

あなたは、背信コンピューティングアプリケーションができることはわかるし、それがいかに苦痛かも調べられるし、背信コンピューティングを受け入れるかどうかも自分で決められると考えるかもしれない。受け入れるのは近視眼的で馬鹿げたことになるだろう。しかし、重要なのは、自分が結んだと思った契約がその場でじっとしていないことである。プログラムを使わずに済ませられなくなった頃には、プログラムにがんじがらめに縛られてしまう。彼らはそれを知っていて、そのときになって契約を変える。一部のアプリケーションは、契約と異なることをするアップグレードを自動的にダウンロードする。そして、アップグレードするかどうかについて選択の余地を与えない。

今なら、使わないという方法によって私有ソフトウェアの制約を避けることができる。GNU/Linux などのフリーオペレーティングシステムを実行し、その上に私有アプリケーションをインストールしないようにすれば、コンピュータがすることに関与できる。フリープログラムに悪意のある機能が含まれていたら、コミュニティの他のプログラマがそれを取り除き、訂正されたバージョンを使うことができる。フリーではないオペレーティングシステムの上でフリーアプリケーションやツールを実行することもできる。これは、自由を完全に得られる方法ではないが、多くのユーザーがそうしている。

背信コンピューティングは、フリーオペレーティングシステムとフリーアプリケーションの存続を危機に晒す。と言うのも、これらはまったく実行できなくなってしまうかもしれないからだ。背信コンピューティングの一部のバージョンは、特定の企業が個別に承認したオペレーティングシステムを要求するようになるだろう。フリーオペレーティングシステムはインストールできなくなる。背信コンピューティングの一部のバージョンは、オペレーティングシステム開発者が個別に承認したプログラムしか実行できなくなるだろう。そのようなシステムでは、フリーアプリケーションを実行することはできない。その仕組みを探り出し、誰かに言えば、犯罪になる可能性がある。

合衆国には、すでに、すべてのコンピュータが背信コンピューティングをサポートすることを義務付け、古いコンピュータを Internet に接続することを禁止する法案が提出されている。CBDTPA（私たちは、この法案を Consume, But Don't Try Programming Anything [消費してもプログラミングしようとしてはならない] と呼んでいる）は、それらの中の1つである。しかし、たとえ背信コンピューティングへの切り替えが法的に強制されなくても、そうせよという圧力は圧倒的なものになるだろう。現在、Word フォーマットは、いくつかの問題の原因になるにもかかわらず、通信手段としてよく使われている（http://www.gnu.org/no-word-attachments.html 参照）。背信コンピューティングマシンが最新の Word 文書しか読めなければ、ユーザーが最新版への切り替えを個々の判断（取るか取らないか）だと思うなら、多くの人々が切り替えに走るだろう。背信コンピューティングに反対するには、団結して、共同の選択として、状況に対処しなければならない。

背信コンピューティングの詳細については、http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html を参照していただきたい。

背信コンピューティングを阻止するためには、非常に大勢の市民を組織しなければならない。私たちはあなたの支援を必要としているのである。Electronic Frontier Foundation（www.eff.org）と Public Knowledge（www.publicknowledge.org）は背信コンピューティングに反対するキャンペーンを進めており、フリーソフトウェア財団が資金提供している Digital Speech Project（www.digitalspeech.org）も同様の運動を進めている。これらの Web サイトを訪れ、作業を支援するために登録をしていただきたい。Intel、IBM、HP/Compaq など、それぞれのコンピュータを買ったメーカーの消費者対策部門に、「信託コンピューティングシステムを買えという圧力を受けるのはいやであり、メーカーがそのようなものを生産するのを望まない」という手紙を書くという支援方法もある。こうすれば、消費者の力が実を結ぶはずだ。独自にこれらの活動を行うときには、上記の組織に手紙のコピーを送っていただきたい。

おわりに

GNU プロジェクトは、公開鍵暗号化と電子署名を実装する GNU Privacy Guard（GPG）を頒布している。このプログラムは、機密電子メールの送信に使える。GPG と信託コンピューティングの違いを探り、片方が役に立つのに対してもう片方が危険な理由を理解するのは有益かもしれない。

誰かが GPG を使って暗号化された文書を送ってきたら、あなたは GPG を使ってそれを復号化する。得られたものは、あなたが読み、転送し、コピーし、再び暗号化して誰か他人に安全に送ることさえできる暗号化されていない文書である。背信コンピューティングアプリケーションは、画面に言葉を表示しても、他の方法で使える暗号化されていない文書を作れないようにする。フリーソフトウェアパッケージである GPG は、ユーザーにセキュリティ機能を提供する。ユーザーがそれを使う。それに対し、背信コンピューティングは、ユーザーに制限を加えることを目的として設計されている。それがユーザーを使う。

初出: 本稿は従来未発表だったもので、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第18章
ソフトウェアがフリーであるべき理由

ソフトウェアが存在すると、その利用形態をどのように決めるべきかという問題が必然的に発生する。たとえば、プログラムのコピーを持っているある個人が、そのコピーをほしいと思っている別の誰かと会ったとする。彼らはプログラムをコピーする能力を持っているが、そうしてよいかどうかは誰が決めるべきだろうか。出会った2人の個人か、「所有権者」と呼ばれる別の主体だろうか。

ソフトウェア開発企業は、一般に、答の基準は自分の利益を最大限に確保することにあるという前提でこれらの問題を考える。政府は、ビジネスの政治的な力関係に基づき、メーカーが提案した基準と答の両方を受け入れている。プログラムは所有権者を持ち、それは開発に携わった企業だというのである。

私は、異なる基準を使って同じ問題を考えてみたいと思う。それは、一般人の自由と繁栄である。

現在の法では、答を決めることはできない。法律は、倫理に従うべきで、逆になってはならない。また、現在の慣行は、この問題に対してある答を誘い出すかもしれないが、だからといってそれを答にしてはならない。唯一の判断方法は、ソフトウェアの所有権者を認めることによって、誰が得して誰が損をするのか、それはなぜでどの程度なのかということである。言い換えれば、私たちは、商品の生産とともに個人の自由を考慮に入れながら、全体としての社会のために、費用便益分析を行わなければならない。

本稿では、所有権者を持つことがどのような影響を及ぼすかを詳論し、有害な結果がもたらされることを示す。私の結論は、プログラマには自分が書いたソフトウェアの共有、再頒布、研究、改良を他者に奨励する義務があるというもので

ある。つまり、フリーソフトウェアを書かなければならないということだ*1。

所有権者が自らの権力を正当化する手口

プログラムが財産となる現行制度の恩恵を被っている人々は、プログラムの所有権を主張するために、感情的なものと経済的なものの計2つの論拠を提示する。

感情的な論拠とは、このようなものである。「私は、このプログラムに汗と気持ちと魂を込めている。このプログラムは私が作ったものだから、私のものだ」

この議論は、たやすく論破できる。プログラマがそれにふさわしい仕事をしたのなら、プログラマはプログラムに愛着を感じるだろう。それは避けられないことだ。しかし、たとえば同じプログラマが、報酬を得るためなら、大企業にすべての権利を喜んで譲り渡してしまうことを考えてみよう。愛着の感情は不思議に消えてしまっている。それに対し、作品に署名すら残さなかった中世の優れた芸術家や職人たちのことを考えてみよう。彼らにとって、自分の名前は重要ではなかった。重要なのは、仕事を完成させることであり、完成した仕事が果たすはずの目的だった。数百年にわたって優位を保ってきたのは、この考え方である。

経済的な論拠のほうは、このようなものである。「私は金持ちになりたい（通常、「生計を立てる」という不正確な表現が使われるが）。プログラミングで金持ちになることが許されないなら、私はプログラムを書かない。誰もが私と同じであれば、プログラムを書く人間はいなくなるだろう。そうしたら、皆さんはプログラムがなくなって困るはずだ！！」　通常この脅迫は、賢人による心からの忠告という装いで行われる。

この脅迫がはったりに過ぎないことは、あとで説明することにしよう。先に、この論拠に含まれているもう1つの論点のほうのよりわかりやすい前提条件を明らかにしたい。

その論点は、まず、私有プログラムの社会的有用性とプログラムが存在しないことを比較した上で、私有ソフトウェア開発が全体として有益であり、奨励され

*1　「フリーソフトウェア」の中の「フリー」は価格のことではなく、自由のことである。フリープログラムのコピーに支払われる価格は0でも、少額でも、（まれだが）非常に大きな額でもかまわない。

るべきだという結論を導いている。この議論の欺瞞は、比較の俎上に私有ソフトウェアかソフトウェアなしかの2つの結論しか載せず、他の可能性をあらかじめ排除していることにある。

ソフトウェア著作権制度のもとでは、通常、ソフトウェア開発はソフトウェアの利用形態をコントロールする所有権者の存在と結び付けて考えられてしまう。この結び付きが存在する限り、私有ソフトウェアかソフトウェアなしかの二者択一に迫られることになりがちである。しかし、この結び付きは本質的なものでも不可避のものでもない。これは、私たちが疑問を投げかけているある特定の社会政策・立法上の決定、すなわちソフトウェアは所有権者を持つことという決定がもたらした結果に過ぎない。私有ソフトウェアかソフトウェアなしかという二者択一を定式化することに疑問の眼を向けなければならないのである。

所有権者を持つことに反対する論拠

今ある疑問点は、「ソフトウェア開発とソフトウェアの利用形態を制約する所有権者を持つことを結び付けてよいのか」ということである。

この問題に答えるためには、2つの行為が社会に対して与える影響をそれぞれ独立に見極めなければならない。つまり、ソフトウェア開発が社会に及ぼす影響（頒布方法とは無関係に）とソフトウェアの利用形態を制限することが社会に及ぼす影響（ソフトウェアが開発されていることを前提として）を考えるのである。これらの中の片方が有益でもう片方が有害なら、両者の結び付きを切り離し、有益なほうだけを残せばよい。

言い方を変えるなら、すでに開発されているプログラムの頒布を妨げるのが社会全体にとって有害なら、倫理的なソフトウェア開発者はそうすることを拒否するだろう。

共有を制限することの社会的影響を考えるには、制限付き（すなわち私有）ソフトウェアと誰もが自由に使える同じプログラムの社会的価値を比較しなければならない。これは、2つの可能な世界を比較するということである。

この分析は、「所有権者に害を与えることになるので、隣人にコピーを与えることによって得られる利益は帳消しになる」という私たちへの単純な反論にも応答

することになる。この反論は、害と利益が同程度だという前提のもとに成り立っている。これから示す分析は、両者の規模を比較し、利益のほうがはるかに大きいことを明らかにする。

議論をわかりやすくするために、道路建設という別の分野に同じ問題を当てはめてみよう。

道路建設にかかるすべての資金を通行料金によって調達することは可能だろう。その場合、すべての街角に料金所を設けなければならない。このような制度は、道路を改善しようという意欲を膨らませるだろうし、すべての道路のユーザーに道路使用料を支払わせる効果を持つ。しかし、料金所はスムーズな通行に対する人為的な障害になる。人為的だと言うのは、道路と車の仕組みが自然にもたらした結果ではないからである。

無料の道路と有料道路を有益かどうかによって比較するなら、（他の条件がみな同じなら）料金所のない道路のほうが建設コストや運営コストがかからず、安全で、使用効率が良いということになる[*2]。貧しい国では、料金によって多くの人々は道路を使えなくなる場合がある。このように、料金所のない道路は、少ないコストでより大きな利益を社会にもたらす。料金所のない道路のほうが社会にとっては望ましい。建設済みの道路は、無料で利用できるようにすべきである。

料金所の推進者たちが単なる資金調達方法として料金所を提案するなら、彼らは選択肢を歪めているのである。料金所は資金を調達するが、他のこともする。実際、料金所は道路の価値を下げる。有料道路は、無料の道路よりも劣っている。道路を増やしたり技術的に優れたものにすることが無料の道路を有料道路に置き換える理由なのだとすれば、これらは改良ではないかもしれない。

もちろん、無料の道路を建設するためには費用が必要であり、国民は何らかの形でその費用を負担しなければならない。しかし、だからといって、料金所が避けられないことにはならない。いずれにしても費用を負担しなければならない私たちは、無料の道路にかけたほうが同じお金からより大きな価値を得ることがで

*2　大気汚染と交通混雑の問題を導入しても、結論は変わらない。運転一般を抑制するために運転コストを高くしたいのなら、大気汚染と交通混雑の両方の原因となる料金所を使うのは逆効果である。ガソリンに税金をかけたほうがはるかに良い。同様に、最高速度を制限することによって安全性を高めたいという考えも料金所とは無関係である。制限速度をどこに設定しようが、無料の道路は、停止と遅れを避けることによって平均運転速度を上げる。

きる。

私は、有料道路があるほうが道路がまったくないよりも良いというような話をしているわけではない。料金が高過ぎてその道路を使う人がほとんどいないなら、ないほうがましかもしれないが、料金徴収サイドがそのような政策を取るはずはないだろう。しかし、料金所は大きな浪費であり、不便なので、より邪魔にならない方法で資金を調達したほうがよいということである。

ソフトウェア開発に同じ議論を当てはめるなら、有益なソフトウェアに対して「料金所」を設けることは、社会にとって非常に大きな出費になる。そのようなことをすれば、プログラムの「建設」コスト、頒布コストは上がり、利用時の満足度や効率は下がる。だから、プログラムの開発は、何か別の手段で奨励すべきなのである。では、ソフトウェア開発を奨励し、(実際に必要な程度の) 資金を調達するための別の方法というものを説明しよう。

ソフトウェアに制限を設けることによる害

プログラムの開発が終わり、開発に必要だったすべてのコストを支払ったとする。社会は、ソフトウェアを私有にするか、自由に共有、利用できるようにするかを決めなければならない。プログラムが存在し利用できることは、望ましいことだという前提で話を進めよう[*3]。

プログラムの頒布と変更に制限を加えることは、その利用を促進しない。害になるだけである。そのため、その社会的な影響は否定的なものでしかあり得ない。しかし、その害はどの程度でどのような種類のものなのだろうか。

このような制限が実際に引き起こす有形の害には、3 つの異なる段階がある。

---

*3　特定のコンピュータプログラムについては、有害であり、まったく利用できないようにすべきだという見解はあり得る。たとえば、一般社会が拒否したために販売停止に追い込まれた個人情報データベースの Lotus Marketplace である。私の本稿での論点の大半はこのような場合には適用されないが、所有権者がプログラムを利用できる人々の範囲を狭めるから所有権者を設けることにしようという議論はほとんど無意味である。利用することが有害だと考えられるプログラムに対して人々が期待するように、所有権者がそのプログラムを完全に利用できない状態にすることなどあり得ない。

1. プログラムの利用者が減る。
2. ユーザーがプログラムに修正、訂正を加えられない。
3. 他の開発者がプログラムから学習することができず、既存のプログラムを新しい仕事の基礎にすることができない。

これらの有形の障害は、それぞれ心理的社会的な害悪を付随させる。心理的社会的な害悪とは、人々の決定が彼らのその後の感覚、態度、傾向に与える影響である。このような人々の思考形態の変化は、彼らの仲間である市民たちとの関係に影響を与え、さらには目に見える形の結果を生むのである。

3つの有形の障害の段階は、プログラムが寄与できるはずだった価値の一部を無駄にする。それらがプログラムのほとんどすべての価値を無駄にするなら、プログラム開発は、高々プログラム開発にかかった労力の分だけ社会に害を与えることになる。確かに、販売すれば利益になるようなプログラムは、何らかの直接的な有形の純益を生んでいるに違いない。

しかし、社会的心理的害悪を考慮に入れるなら、私有ソフトウェア開発の有害性には限度がない。

プログラムの利用に対する制限

最初の段階は、プログラムの単純な利用を妨げるというものである。プログラムのコピーにはほとんどコストはかからない（自分で作業をすれば、自分でコストを支払える）ので、自由市場ではほとんど無料になる。ライセンス料は、プログラムを使おうという意欲を大きく挫く。広く使えるはずのプログラムが私有なら、そのプログラムを使う人は大幅に減少するだろう。

プログラムに所有権者を与えることによって、プログラムの社会に対する寄与が全体として下がることは簡単に示せる。プログラムの潜在ユーザーは、使うために料金が必要だということに直面すると、料金を支払うか、プログラムを使わずに済ませるだろう。ユーザーが料金を支払うことにした場合、2つの主体の間でゼロサム転移が発生することになる。しかし、プログラムを使わないことにし

た人が発生するたびに、この人は誰にも利益を与えずに害を受けることになる。負数と0の合計は、必ず負数になる。

しかし、これによってプログラムを開発するために必要な作業量が減るわけではない。そのため、作業効率、すなわち1時間の作業あたりのユーザーの満足度は下がる。

このことは、プログラムのコピーを作ることと、車、椅子、サンドイッチのコピーを作ることとの決定的な違いを反映している。SF以外の世界には、物体のコピー機はない。しかし、プログラムは簡単にコピーできる。誰もが、ごくわずかの労力で必要なだけのコピーを作れる。物体はそういうわけにはいかない。新しいコピーは、最初のコピーを作ったときとまったく同じように原材料から作らなければならない。

物体の場合、それを使いたくないという人が出ても、販売数が減るということは製品を作るために必要な原材料と労力が減るということであり、無意味ではない。通常、起業コスト、開発コストがかかるのは事実だが、それは生産物全体に分散させることができる。しかし、コピーを作るための限界費用が高ければ、開発コストの部分を追加しても質的な違いはない。そして、限界費用が高い分、通常のユーザーの自由に制限を加えなければならなくなったりはしない。

しかし、無料にできるものに価格を押し付けることは、質的な変化をもたらす。ソフトウェアの頒布に中央集権的に押し付けられた料金は、購買意欲を大きく削ぐ効果を持つ。

さらに、現在行われているような中央集権的な生産は、ソフトウェアのコピーを配る手段としても非効率的である。この方式は、無意味に大きいパッケージに物理的なディスクやテープを収め、大量のパッケージを世界中に出荷し、店舗で販売するというものである。このコストは、業務遂行のための経費になる。実際には、これは所有権者を持つことに起因する無駄の一部である。

社会的な団結力の阻害

あなたとあなたの隣人がともにあるプログラムを実行すると役に立つと思っていたとする。隣人に対する倫理的な配慮からすれば、２人がともにそれを使えるようにしたいと考えるのが当然である。２人のうちの１人だけにプログラムを使うことを認め、もう１人には制限を加えるような提案は、２人の間に仲違いをもたらす。あなたも隣人も、そんなのには耐えられないと思うだろう。

しかし、典型的なソフトウェアライセンス契約に署名するということは、隣人を裏切ることである。「私は、コピーを自分のものにするために、隣人からこのプログラムを奪うことを約束します」と言っているのである。そのような選択を迫られた人々は、自らを正当化する内的心理的な圧力を受け、隣人を助けることの重要性を軽視するようになる。このようにして公共心が失われるのである。これは、プログラムの利用を損なう有形の害悪に対応する心理的社会的害悪である。

多くのユーザーは、共有を拒むのは悪いことだということを無意識のうちに認識しているので、ライセンスや法律を無視して、いずれにしてもプログラムを共有しようとする。しかし、彼らはそうすることに罪悪感を感じることが多い。彼らは、良き隣人であるためには法を破らなければならないことを知っているが、まだ法律の権威を信じているので、良き隣人であること（彼らはまさしくそうなのだが）は、良くないこと、あるいは恥ずかしいことなのだと結論付けてしまう。これもまた一種の心理的社会的害悪だが、ライセンスや悪法に倫理的な力を与えなければ、この害悪からは逃れられる。

プログラマたちも、多くのユーザーが自分の作品の使用を認められていないことを認識することによって、心理的社会的害悪を受ける。彼らは、ひねくれるか問題を否定する方向に走る。プログラマは、技術的に手応えのある仕事について熱烈に語るかもしれないが、「私もそれを使っていいの？」と尋ねられると、途端に顔が曇り、答がノーであることを認める。がっかりする気持ちを避けたければ、普段はそのような問題があることを無視するか、問題の重要性を最小限に見積もるひねくれたスタンスを取るしかない。

レーガン[4]時代以来、合衆国でもっとも不足しているのは技術革新ではなく、公共の福祉のために協力して働く意思である。後者を犠牲にして前者を奨励したところで意味はない。

プログラムのカスタム修正の阻害

有形の障害の第 2 の段階は、プログラムを修正できないことである。修正しやすさは、ソフトウェアが古くからの技術に対して持つ大きな利点の 1 つである。しかし、ほとんどの市販ソフトウェアは、購入後でも変更できないようになっている。購入者は、ブラックボックスとして使うか、手離すかしかない。それですべてである。

実行できるプログラムは、意味不明の数字の連続から構成されている。プログラムが他の動作をするようにこの数値を簡単に書き換えられる人は、優れたプログラマを含めて決していない。

通常、プログラマは Fortran や C などのプログラミング言語で書かれたプログラムの「ソースコード」を操作する。ソースコードは、使われているデータやプログラムの部分を区別するために名前を使い、加算の「+」、減算の「-」のような記号で演算を表現する。プログラミング言語は、プログラマが簡単にプログラムを読んだり書き換えたりできるように作られている。例を示すことにしよう。これは、平面上の 2 点の距離を計算するプログラムである[5]。

```
float
distance (p0, p1)
     struct point p0, p1;
{
  float xdist = p1.x - p0.x;
  float ydist = p1.y - p0.y;
```

---

[4] アメリカ合衆国第 40 代大統領、ロナルド・レーガンは、多くの社会保証制度を削減したことで有名である。また、多くの人々に失敗とみなされている「トリクルダウン経済」と呼ばれる経済政策を作ったことでも知られる。

[5] このソースコードがどのように動作するかを理解することには、重要な意味はない。ここで重要なのは、ソースコードが簡単に理解できる抽象レベルで書かれていることに気づくことである。

```
  return sqrt (xdist * xdist + ydist * ydist);
}
```

次に示すのは、私が通常使っているコンピュータの実行可能形式[6]で書かれた同じプログラムである。

```
1314258944  -232267772  -231844864  1634862
1411907592  -231844736  2159150     1420296208
-234880989  -234879837  -234879966  -232295424
1644167167  -3214848    1090581031  1962942495
572518958   -803143692  1314803317
```

ソースコードは、プログラムのすべてのユーザーにとって（少なくとも潜在的に）役に立つ。しかし、ほとんどのユーザーは、ソースコードのコピーを持つことを許されない。通常、私有プログラムのソースコードは、所有権者によって秘密に保たれているが、それは他人がそのソースコードから何かを学ぶのを防ぐためである。ユーザーは、コンピュータが実行する理解不能な数値のファイルだけを受け取る。これは、プログラムの所有権者だけがプログラムを変更できるということである。

以前、ある友人が銀行でプログラマとして6か月働いていたときのことを話してくれた。それは、市販されているものとよく似たプログラムを書く仕事だった。彼女は、市販プログラムのソースコードが手元にあれば、会社のニーズに合わせて修正を加えるのは簡単なことだと思っていた。銀行は、そのためなら喜んでお金を払っただろうが、それは認められていなかった。ソースコードは秘密にされていたのである。そこで、彼女は7か月の無駄な仕事をしなければならなかった。この仕事は、国民総生産の中には数えられるが、実際にはただの浪費である。

MIT 人工知能研究所（AI ラボ）は、1977年ごろ、Xerox から贈り物としてグラフィックスプリンタを受け取った。このプリンタは、私たちが多くの便利な機能を追加したフリーソフトウェアによって動作していた。たとえば、このソフトウェアは、印刷ジョブが終了すると同時にユーザーに通知を送った。紙詰まりや用紙切れなどのトラブルが発生すると、このソフトウェアは、印刷ジョブをキュー

---

[6] 実行可能コードが理解不能だということに注意していただきたい。上のソースコードよりも意味がわからないことは明らかだろう。

イングしたすべてのユーザーに直ちに通知を送った。これらの機能は、スムーズな運用の役に立っていた。

その後、Xerox は AI ラボに新しいより高速なプリンタを送ってきた。最初のレーザープリンタの 1 つである。このプリンタは、別個の専用コンピュータで実行される私有ソフトウェアによって駆動されていたので、私たちはお気に入りの機能を何 1 つ追加できなかった。印刷ジョブが専用コンピュータに送られたときに通知を送るように設定することはできたが、印刷ジョブが実際に印刷されたときには通知を送れなかった（そして、通常遅れはかなりなものだった）。ジョブが本当に印刷されたタイミングを知る方法はなかった。推測できただけである。紙詰まりが発生しても誰にも通知は送られなかったので、プリンタはトラブルを起こしたまま 1 時間も放置されていることがよくあった。

AI ラボのシステムプログラマたちは、おそらくプログラムのオリジナルの作者たちと同じくらいの能力は持っていたので、そのような問題を解決することは可能だったはずだが、Xerox はこれらの問題を解決することに関心を持たず、私たちの邪魔をしたので、私たちは問題を受け入れざるを得なかった。これらの問題は、最後まで解決されなかったのである。

ほとんどの優れたプログラマは、この種の欲求不満を経験している。銀行ならゼロから新しいプログラムを書いて問題を解決するだけの余裕があるが、一般的なユーザーは能力の有無にかかわらずあきらめるしかない。

あきらめることは、心理的社会的害悪の原因となる。独立独行の精神が損なわれる。自分のニーズに合うように改造できない家に住むのは、気持ちよくないものである。服従し、落胆することは、人生の他の側面にも影響を与える。このような感じ方をする人々は不幸であり、良い仕事をしない。

レシピがソフトウェアと同じように死蔵されていたらどうなるか想像してみていただきたい。あなたが、「このレシピから塩を取り除くにはどうしたらよいのでしょうか？」と尋ねると、グランシェフが「俺の頭脳と舌の申し子である俺のレシピに土足で踏み込んで侮辱するとはどういう了見だ。お前に俺のレシピを訂正しろなどと言われる筋合いはねえ！」と答える。

「しかし、医者に塩を取るなと言われたんです。私はどうしたらいいんですか。私のために塩を取り除いてくれませんか」

「ああ、喜んでやってやろう。料金はたったの5万ドルだ（所有権者は変更に対する独占権を持っているので、料金は高額になりがちである)。しかし、今は時間がない。俺は海軍省から注文のあった艦船用ビスケットの新レシピ作成委員会で忙しいんだ。お前の注文を聞いてやれるのは、だいたい2年後だな」

ソフトウェア開発の阻害

有形の害悪の第3の段階は、ソフトウェア開発に影響を与える。かつてのソフトウェア開発は、発展的な作業だった。つまり、ある人が既存のプログラムを取り出して新機能のために一部を書き換え、他の人が別の機能のために別の部分を書き換えるというようなことを20年以上も続けてきたのである。プログラムの部分部分は、他のプログラムのスタート地点となるべく、「盗用」された。

所有権者の登場によって、このような発展は阻害され、プログラムを書くときにはいちいちゼロから書かなければならなくなった。所有権者の存在は、新人が既存のプログラムから役に立つテクニックや大規模なプログラムの構築方法を学ぶことも不可能にしてしまった。

所有権者は、教育の阻害要因にもなっている。私は、大規模プログラムのソースコードを見たことのないコンピュータ科学科の優等生に会ったことがある。彼らは小さなプログラムを書くのは得意だが、他のプログラマがどのようにして大規模プログラムを書いたかを見ることができないので、大規模プログラムを書くために必要な別の種類の技能を学ぶことができないでいる。

知的な分野では、他人の肩に乗ることによってより高い地点に到達できるようになる。しかし、ソフトウェアの分野ではこれがもう一般的な形では認められない。自分と同じ会社の他人の肩に乗れるだけである。

これに伴う心理的社会的害悪は、科学における協力精神への影響である。かつては、科学者たちの協力精神は、それぞれの祖国が戦争をしていても協力を忘れないほど強固なものだった。太平洋の孤島の研究所を捨ててきた日本の海洋学者たちが、侵入してきたアメリカ海軍のために、苦心して自らの仕事を保存し、大切に使ってくれるようにというメモを残してきたのは、まさにこの協力精神があったからである。

利益を求めるための紛争は、国際紛争によっても失われなかったものを破壊してしまった。最近のさまざまな分野の科学者は、論文の中で他者が実験を再現できるだけの情報を開示しない。自分が成し遂げたことに読者が驚嘆できる程度にしか情報を開示しないのである。このことは、コンピュータ科学では紛れもない事実である。報告されるプログラムのソースコードは、通常秘密にされているのである。

共有がどのように制限されるかの問題ではない

ここまで、プログラムのコピー、変更、構築を妨げることが及ぼす影響を論じてきた。破壊がどのように行われるかについては具体的に触れてこなかったが、それは結論に影響を与えないからである。コピープロテクション、著作権、ライセンス、暗号化、ROMカード、ハードウェアシリアル番号などのどの方法で行われたとしても、使用の妨害に成功したら、それは害悪を生んでいる。

ユーザーたちは、これらの方法のうち、一部のものは他のものよりも悪質だと考えているようだが、私からすれば、もっとも憎むべき方法は、その目的を達成できる方法だと言いたい。

ソフトウェアはフリーでなければならない

プログラムの所有権、すなわちプログラムの変更やコピーを制限する権力がいかに大きな障害になるかを示してきた。その反動的な効果は、広く行き渡っており、深刻である。それゆえ、社会はプログラムの所有権者を持つべきではない。

同じことは、社会が必要としているのはフリーソフトウェアで、私有ソフトウェアはその粗悪な代用品に過ぎないという形で理解することもできる。代用品を奨励することは、必要なものを手に入れるための合理的な方法ではない。

ヴァーツラフ・ハベルは、「成功のチャンスだからではなく、良いからという理由を持つもののために働け」と言っている。私有ソフトウェアの製造というビジネスは、狭い意味での成功のチャンスを提供するが、社会にとって良いものではない。

人がソフトウェアを開発する理由

ソフトウェア開発を奨励する手段として、著作権を捨てたとすると、最初は開発されるソフトウェアの数が減るかもしれないが、そのソフトウェアはより有益なものになるだろう。ユーザーに与えられる満足度全体が少なくなるかどうかについては何とも言えないが、もしそのようなことがあるなら、またいずれにしてもユーザーの満足度を上げたいと思うなら、料金所以外にも道路建設の資金調達方法があるのと同じように、開発を奨励する別の方法はあるはずだ。しかし、その方法について論じる前に、人為的な奨励策が本当に必要なのかを問うてみたい。

プログラミングは面白い

お金を手にするということ以外にほとんどの人が参入する理由を見つけられないような仕事がある。たとえば、道路建設がそうである。その一方で、金持ちになるチャンスはほとんどないが、仕事に魅力があるからとか、社会的に価値が認められているからという理由で、人々が喜んで飛びつく学問や芸術の仕事もある。たとえば、数学的論理学、クラシック音楽、考古学、働く人々の政治的オーガナイザーなどがそうである。人々は、お金が得られるごくわずかの地位を求めて熾烈な競争を繰り広げるが、それらの地位さえ、それほどのお金が得られるわけではない。そのような競争は、苦しいというよりも寂しい色合いを帯びる。彼らは、その分野で仕事をするチャンスを得るために、余裕があれば自らお金を払うことさえある。

そのような分野は、豊かになる可能性を示し始めると、一夜のうちに変身してしまうことがある。ある人物が豊かになると、他の人々も同じチャンスを要求するようになる。早晩、誰もがかつては楽しみのためにしていたことのために大量のお金を要求するようになる。さらに数年がたつと、その分野に関わりを持つすべての人々は、大金が見込めないような仕事がその分野で成り立つはずがないと自明的に考えるようになる。彼らは、特別な恩恵、権力、独占権が必要だというように言いながら、そのような見返りを可能にすることをソーシャルプランナーたちに吹き込む。

この変化は、この 10 年間にコンピュータプログラミングの世界で起きたことである。15 年前[7]には、「コンピュータ耽溺症」についての論文があったくらいだ。「オンライン」で 1 週間に 100 ドル費やすユーザーが話題になっていた。それは、一般にプログラミングが好きで結婚生活を壊してしまう人々がいるというように理解されていた。今日では、一般に高い報酬が与えられなければ、誰もプログラムを書かないと理解されている。人々は、15 年前に知っていたことを忘れてしまったのである。

特定の時点で、ある分野のほとんどの人々が高給でなければ働かないということが事実であっても、いつまでもそれが続くわけではない。社会が刺激を与えれば、変化の波によって事態が逆転する可能性はある。大儲けの可能性を取り除き、それからしばらくして人々が態度を改めるなら、彼らは再び達成の喜びのために熱心に仕事をするようになるだろう。

「プログラマに報酬を与えるためにはどうしたらよいか」という問題は、彼らに大金を与えるかどうかという問題ではないということがわかれば、簡単な問題になる。ただ生活できるようにするための資金なら、簡単に用意できる。

フリーソフトウェアのための基金

プログラマに報酬を与える組織は、別にソフトウェアメーカーでなくてもよい。同じことができる他の組織はすでにたくさんある。

ハードウェアメーカーは、ソフトウェアの利用形態をコントロールできなくても、ソフトウェア開発を支援することが重要なことだと考えている。1970 年には、ハードウェアメーカーのソフトウェアの大半はフリーだったが、それは彼らがソフトウェアに制限を加えることを考えなかったからである。今日、ハードウェアメーカーがコンソーシアム（協力組織）を結成する意欲を高めつつあることは、彼らにとってソフトウェアの所有権を持つことがそれほど重要ではないことを示している。

大学は、さまざまなプログラミングプロジェクトを組織している。今日の大学

---

[7] 本稿執筆時の 15 年前は 1977 年である。

はプロジェクトの成果を売ることが多いが、1970年代にはそのようなことはなかった。大学がソフトウェアを売ることを認められていなければ、大学がフリーソフトウェアを開発するだろうということに疑問の余地があるだろうか。これらのプロジェクトは、現在私有ソフトウェア開発を支援している政府の契約や助成金で支援できるはずのものである。

今日では、大学の研究者が助成金を受け取ってシステムを開発し、ほとんど完成というところまで作業を進めたところで、それを「完成した」と称し、企業を起こしてそこで本当にプロジェクトを完成させて、使えるものにするということがよくある。彼らの中には、未完成バージョンを「フリー」と宣言する者もいるが、徹底的に堕落した分子は、大学から排他的なライセンスを手に入れる。これは秘密ではない。関係者全員が公然と認めていることである。しかし、このような誘惑にさらされていなければ、研究者たちもそれぞれの研究をしていたはずである。

フリーソフトウェアを開発しているプログラマたちは、ソフトウェアに関連するサービスを販売することによって、生活をしていける。私は、新しいハードウェアにGNU Cコンパイラを移植するため、また、GNU Emacsのユーザーインターフェイスに拡張を加えるために雇われたことがある（私は、完成後には、これらの改良点を公開している）。私は、教室で授業を行うことによっても給与を得ている。

このようにして働いているのは、私だけではない。現在、他のタイプの仕事を一切せずに、成功し、成長している企業が1つ存在する。ほかにも、GNUシステムのフリーソフトウェアにサポートを提供している営利企業がいくつかある。これは、独立したソフトウェアサポート産業の始まりである。この産業は、フリーソフトウェアが主流になれば、大きく成長するだろう。この産業は、私有ソフトウェアでは一般に得られないような選択肢を裕福でないユーザーにも提供する。

フリーソフトウェア財団（FSF）のような新しい[8]組織も、プログラマに資金を提供できる。財団の基金の大半は、メールでディスクやテープを購入したユーザーから集められたものである。テープ上のソフトウェアはフリーだが、それはすべてのユーザーがそれをコピー、変更する自由を持っているということで、多く

---

*8　本稿は、1992年4月24日に書かれた。

の人々はコピーを入手するためにお金を払っている（「フリーソフトウェア」のフリーは、自由のことで価格のことではないということを思い出していただきたい）。すでにコピーを持っているユーザーが、FSFに貢献するための手段としてテープを注文している例もある。FSFは、コンピュータメーカーからかなり高額の寄付も受けている。

FSFは慈善団体であり、収入はできる限り多くのプログラマを雇うために使われる。企業として設立され、一般人に対して同じフリーソフトウェアを同じ料金で頒布していれば、創立者は今ごろ莫大な財産を作っていただろう。

FSFは慈善団体なので、プログラマは他の職場で得られる報酬の半分で仕事をすることが多い。彼らがそれでも仕事をするのは、私たちが官僚主義に縛られておらず、自分の仕事が自由に使われることに満足を感じているからである。何よりもまず、彼らはプログラミングが楽しいから仕事をする。さらに、私たちのために多くの役立つプログラムを書いたボランティアがたくさんいる（テクニカルライターのボランティアさえいる）。

このことは、プログラミングが音楽や芸術と並んであらゆる分野の中でもっとも魅力的な仕事だということを証明している。私たちは、プログラムを書く人間がいなくなることを恐れる必要はない。

ユーザーが開発者に対して持っている借りとは何か

ソフトウェアのユーザーがその支援のために貢献しなければならないという義務感を感じるのは不思議なことではない。フリーソフトウェアの開発はユーザーの活動に貢献しているのであり、ユーザーが開発を継続できるように資金を提供するのは公平なことでもあり、長期的な利益をもたらすことでもある。

しかし、このことは私有ソフトウェア開発者には当てはまらない。阻害は、報酬ではなく懲罰に値する。

ここに矛盾がある。役に立つソフトウェアの開発者はユーザーの支援を受ける資格を持つが、この倫理的な義務感を要求に転化させると、義務感の基礎が失われてしまうのである。開発者は報酬に値するか報酬を要求するかで、両方を兼ねることはできない。

この矛盾に直面した倫理的な開発者は、報酬に値するように行動するに違いないと私は思うが、ユーザーに寄付を依頼することも考えてよいと思う。ユーザーは、市民ラジオや市民テレビを支援したときと同じように、いずれ強制されなくても開発者を支援することを学ぶはずである。

ソフトウェアの生産性とは何か

ソフトウェアがフリーでも、プログラマは残るはずだが、その数はおそらく減るだろう。これは、社会にとって悪いことだろうか。

必ずしもそうとは言えない。今日の先進国では、1900年よりも農民の数は減ったが、その少ない農民が以前よりも多くの食料を消費者に提供できるようになったので、農民の減少が社会にとって悪いことだとは考えていない。私たちは、これを生産性の向上と呼ぶ。フリーソフトウェアは、あらゆる段階でソフトウェアの生産性を高めるので、プログラマの数を大幅に減らした上で、需要を満足させるはずである。

- **開発されたプログラムをより広範囲で利用できるようにする。**
- **プログラムをゼロから作るのではなく、カスタマイズのために既存のプログラムを修正できるようにする。**
- **プログラマの教育環境を改善する。**
- **重複する開発作業を減らす。**

プログラマの雇用を減らすと主張して協同作業に反対している人々は、実際には生産性の向上に反対しているのである。にもかかわらず、それらの人々でさえ、ソフトウェア産業は生産性の向上を必要とするという広範に支持されている考え方を受け入れるのである。これは一体どういうことだろうか[*9]。

---

*9　エリック・レイモンドによれば、ソフトウェア産業の95%の仕事は、公開をまったく意図していないカスタムソフトウェアの生産に費やされている。そのため、理論上の最悪、すなわちフリーソフトウェア開発の仕事がまったくない（私たちはすでにいくらかの仕事は確実にあ

「ソフトウェアの生産性」は、2つの意味を持つことができる。すべてのソフトウェア開発全体の生産性と個々のプロジェクトの生産性である。全体的な生産性は社会が向上させたいと思っているものであり、そのためのもっとも直截的な方法は、生産性を低下させる協力に対する人為的な障害を取り除くことである。しかし、「ソフトウェアの生産性」の研究者たちは、第2の狭い意味での用語しか相手にしていない。そのため、生産性の向上のためには、難しい技術的な前進が必要となるのである。

競争は避けられないものなのか

人々が社会でライバルを出し抜くために競争しようとすることは避けられないことなのだろうか。おそらくそうだろう。しかし、競争自体は有害ではない。有害なのは、**闘争**である。

競争には無数の方法がある。競争は、他人の成果を乗り越え、頂上を目指すことを意味することがある。たとえば、昔はプログラムの達人たちの間で競争があった。コンピュータにもっとも驚くべき仕事をさせられるのは誰かとか、特定の仕事のためにもっとも短くもっとも高速なプログラムを書けるのは誰かを競ったのである。この種の競争は、良きスポーツマン精神が保たれる限り、すべての人々の利益になる。

建設的な競争は、人々に大きな力を発揮させる動機として充分である。地球上のすべての国を訪問した最初の人物になるために競争している人々がいる。中にはそのために大金を使うような人さえいる。しかし、彼らは船長に賄賂を贈って砂漠の島にライバルを置き去りにさせるようなことはしていない。もっとも優れた人物が勝つことに満足している。

競争者が自分の前進ではなく、互いに相手の足を引っ張り合おうとすると、競

ることを知っているが）という状態を想定しても、フリーソフトウェアへの切り替えは、ソフトウェアの仕事全体に対する影響は小さくならざるを得ない。プログラマがカスタムソフトウェアを書く仕事を持ちながら、余暇にフリーソフトウェアを開発する余地は無数にある。フリーソフトウェアへの完全な切り替えがソフトウェアの仕事の総数を増やすか減らすかは、まったく予想できない。

争は闘争になる。それは、「もっとも優れた人物が勝つようにする」という態度が「もっとも優れているかどうかにかかわらず、私を勝たせてくれ」という態度に変わったときである。私有ソフトウェアが有害なのは、それが競争の1つの形態だからではなく、社会を構成する市民の間での闘争の形態だからである。

ビジネス上の競争が必ず闘争になるわけではない。たとえば、2つの文房具店が競争するとき、それぞれの店の全精力は、ライバルを妨害することではなく、それぞれの仕事を向上させることに注がれる。しかし、これは企業倫理に特に力を入れているわけではない。物理的な暴力を持たないこの種の業務には、闘争を招くような領域がほとんどないのである。しかし、ビジネスのすべての分野が、このような性質を共有しているわけではない。すべての人々が前進することを助けられるような情報を隠匿することは、一種の闘争である。

ビジネスのイデオロギーは、人々が闘争の誘惑に抵抗するのを助けるわけではない。反トラスト法、誇大広告防止法などは、闘争のいくつかの形態を禁止しているが、それを一般化して闘争全般の禁止という原則を導いているわけではない。経営者たちは、明示的に禁止されていない他の闘争形態を発明してくる。社会資源は、経済的な内戦とも呼ぶべきものによって浪費されている。

「なぜロシアに移住しないのか」

アメリカでは、もっとも極端な形の自由放任の利己主義者以外の人間は、このような非難を受けることが多い。たとえば、自由世界の他の先進国と同様の国民健康保険制度を支持すると、このような非難を受ける。同じく先進国では当たり前になっている芸術の国家補助を支持すると、やはりこの非難を受ける。アメリカでは、市民が公共の福祉を向上させる義務を持つという思想は共産主義とみなされる。しかし、これらの思想に類似点があるだろうか。

ソ連で実践されている共産主義は、すべての活動が統制下にある中央集権的な制度である。その目的は、全体の福祉のためとされているが、実際には共産党の党員のためである。そして、ソ連では、コピー装置は違法コピーを禁止するために厳重にガードされている。

アメリカのソフトウェア著作権制度は、プログラムの頒布を中央集権的な管理

統制のもとに置き、違法コピーを防ぐために、自動コピー防止機能でコピー装置をガードしている。

それに対し、私は、人々が自由に自分の行為を決められるような制度を築くために働いている。特に、隣人を助ける自由、日常生活で使う道具を修正、改良する自由を守ろうとしている。自発的な協力と分権を基礎とする制度である。

ロシア共産主義との類似から判断するなら、共産主義者はソフトウェアの所有権者たちのほうだ。

前提条件への疑問

本稿では、ソフトウェアのユーザーが作者、ましてや作者の雇用主などよりも重要でないということはありえないという前提で話を進めてきた。つまり、どの方向に進むのがもっとも良いかを決めるときに、ユーザーの利益やニーズも同等の重さを持つということである。

この前提条件は、普遍的に受け入れられているわけではない。多数派は、作者の雇用主が他の誰よりも根本的に重要だと考えている。たとえば、ソフトウェアが所有権者を持つことの目的は、作者の雇用主に、地位に見合うだけの利益を与えることだと言っている。一般人にどのような影響を与えるかには無頓着なのだ。

これらの前提を証明したり反証したりすることに意味はない。証明を成立させるためには、前提の共有が必要である。そのため、私が言おうとしていることの大半は、私と同じ前提を共有している人々か、少なくともこれらの前提がどのような結果を生み出すかに関心を持っている人々以外には届かない。所有権者は他の誰よりも重要だと信じている人々にとって、この論文は無意味だろう。

しかし、特定の人々が他の誰よりも重要だという前提条件を多数のアメリカ人が受け入れるとしたら、それはなぜなのだろうか。その理由の一部は、このような前提がアメリカ法の伝統の一部だと信じられていることにあるだろう。この前提を疑うことは、社会の基盤に挑戦することだと感じる人がいるかもしれない。

そのような人々にとって重要なことは、その前提条件がアメリカ法の伝統ではないことを学ぶことである。そんな伝統は決して存在しない。

憲法は、著作権の目的が「科学と工芸の進歩を促進する」ためにあると言って

おり、最高裁は、Fox Film-Doyal 裁判の判決でこの内容を拡張して「合衆国の唯一の関心と［著作権］独占が認められている主要な目的は、作者の労力から国民が引き出せる利益全般にある」と述べている。

私たちは、憲法や最高裁に同意しなければならないわけではない（以前、両者はともに奴隷制を容認していた）。だから、これらの見解が所有権者至上主義の前提条件の反証になるわけではない。しかし、所有権者至上主義が、伝統的に認められた見解ではなく、極右的な前提だということがわかれば、訴求力も弱まるのではないだろうか。

結論

社会は、隣人を助けることを奨励するものであってほしい。しかし、他人の邪魔をしたことに対して誰かに報酬を与えるたびに、あるいはそのようにして富を得た人物に敬意を表するたびに、私たちは逆のメッセージを送ることになるのである。

ソフトウェアの死蔵は、個人の利得のために社会の福祉を軽視する私たちの一般的な意思の1つの現れである。このような社会軽視の系譜は、ロナルド・レーガンからジム・バッカー[*10]、イワン・ボイスキー[*11]、エクソン[*12]に、失敗した銀行から失敗した学校にたどることができる。この傾向の規模は、ホームレス人口と獄中人口によって測ることができる。私たちは他人が自分を助けてくれないのを見れば見るほど、他人を助けるのはばかばかしいと考えるようになる。そのため、反社会的精神は、自力で成長するのである。社会はこのようにして腐敗、崩

*10 ジム・バッカーは、1980年代にHeritage USA、PTL、Inspirational Networkという宗教団体のためのテレビを通じて数百万ドルの資金を集めたが、PTLの資金集めのための郵便と電話を使った詐欺のために有罪とされ、懲役45年の刑を受けている。

*11 イワン・ボイスキーは、1980年代にインサイダー取引のために罰金1億ドルを課せられた上に刑務所に送られた。彼は、「欲望は善である。私が欲望を健全だと考えていることを知ってもらいたい。貪欲でありながら、自分を罪悪視しないでいることは可能だ」と言ったことで知られる。

*12 1980年、エクソン・ヴァルデスは、アラスカ海岸で世界最大の石油流出事故を起こし、計り知れない損害を及ぼした。エクソンは、今までに清掃費用と罰金のために10億ドルを費やしている。

壊していく。

ジャングルのような過酷な生存競争の場で生きるのがいやなら、私たちは自らの態度を改めなければならない。私たちはまず、良き市民とは、適切なときに協力を惜しまない者で、他人からうまく物を奪い取る者のことではないというメッセージを送らなければならない。私は、フリーソフトウェア運動が、この流れに寄与することを期待している。少なくとも 1 つの分野では、ジャングルを、自発的な協力を奨励して実現する効率的な体制に変えていきたい。

初出: 第 1 稿は 1992 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第19章
コンピュータネットワーク時代の著作権とグローバリゼーション

イントロダクション

デビッド・ソバーン（司会者）：今日の講演者であるリチャード・ストールマン氏は、コンピューティングの世界では伝説的な人物であり、私にとっては、彼と同じ演壇に立つ回答者を探すだけでも非常にいい経験になりました。MITのある著名な教授は、旧約聖書の預言者を引き合いに出して、ストールマン氏はカリスマ的人物として理解する必要があると言いました。「モーゼやエレミヤを想像するとよい。どちらかというとエレミヤだな」　私は答えました。「それはすごい賞賛ですね。すばらしいことです。世界に対する彼の貢献について私が感じていることにもぴったり合います。では、なぜ彼と同じ演壇に立とうとなさらないのですか」

彼の答えはこうです。「単純に、エレミヤやモーゼと同様に、彼は私よりも圧倒的に優れているからだ。彼と同じ演壇に立つつもりはないが、すべての人々のために真の意味で貢献した世界中の5人の現存する人物を挙げよと言われたら、リチャード・ストールマンは間違いなくその中に含まれる」

講演

まず、問題をはっきりさせるために、このフォーラムのWeb中継を拒否した理由から説明しなければならないでしょう。主催者がWeb中継に使っているソフトウェアは、中継を見るために特定のソフトウェアをダウンロードすることを

ユーザーに要求しますが、このソフトウェアはフリーソフトウェアではありません。無料ですが、不可解な数値の束である実行可能形式しかないのです。

ソフトウェアが行っていることは秘密です。このソフトウェアは、研究、変更することができませんし、当然ながら修正を加えた独自バージョンを公開することもできません。そして、これらは、「フリーソフトウェア」の定義の中で本質的な意味を持つ自由の一部です。

そこで、もし私がフリーソフトウェアの誠実な擁護者であるなら、このような講演をする一方で、人々にフリーではないソフトウェアを使えと圧力をかけるわけにはいきません。それでは、自分自身の主張を掘り崩すことになってしまいます。自分の主張を真剣に扱っていないことを示すなら、他人にその主張を真剣に考えてもらうことはできないでしょう。

しかし、この講演はフリーソフトウェアについてのものではありません。何年間かフリーソフトウェア運動に携わり、人々がGNUオペレーティングシステムを一部でも使うようになってから、私はこのような講演を行うように誘われるようになりました。そして、「ソフトウェアユーザーの自由という思想は、他の分野にも一般化できるものなのでしょうか」と尋ねられるようになりました。

もちろん、「ハードウェアはフリーであるべきでしょうか」とか、「このマイクはフリーであるべきでしょうか」というような馬鹿げた質問を受けることもあります。

この質問は一体どういう意味なのでしょうか。このマイクを自由にコピー、変更できなければならない？　変更ということについて言えば、あなたがマイクを買ったら、あなたがそれを改造することを誰も止められません。コピーということについて言えば、マイクのコピー機を持っている人はいません。スタートレックの外の世界には、そんなものは存在しないのです。ひょっとすると、いずれ微小工学分析機と組立機が作られ、物体のコピーを作ることが本当に可能になるかもしれません。すると、自由にそれをできるかどうかが本当に重要な問題になり始めるかもしれません。そのような技術的能力が存在するようになれば、農業関連産業が食料のコピーを作成しようとする人々を妨害しようとして、それが大きな政治問題になるかもしれません。しかし、私には、どうなるかはわかりません。現時点では、空想の世界です。

しかし、他の種類の情報についてはこの問題が起きる可能性があります。コンピュータに格納できるあらゆる種類の情報は、コピー、変更できるからです。そのため、フリーソフトウェアの倫理的な問題や、ソフトウェアをコピー、変更できるユーザーの権利の問題は、他の種類の公開された情報についてのそれらの問題と同じです。ここで、プライベートな情報、たとえば個人情報は扱わないことにします。これらの情報は、決して公開されるべきものではありません。私が取り上げるのは、秘密を保つ意図のない公開された情報のコピーを取得したときに、人が持つべき権利についての問題です。

著作権の歴史

このテーマについての私の考えを説明する前に、情報の頒布と著作権の歴史をおさらいしておきたいと思います。古代の世界では、書籍はペンで手によって書かれていました。そして、読み書きを知っている人間は、他の人々と同じくらい効率的に本のコピー（写本）を作ることができました。一日中同じことをしていれば少しはうまくなるかもしれませんが、大差はありません。同時に作れるコピーは 1 冊だけで、規模の経済性というものはありません。10 冊のコピーを作るためには、1 冊のコピーを作る時間の 10 倍の時間がかかります。また、中央集権を強制するものもありませんでした。本はどこでもコピーできたのです。

このような技術的理由から、またコピーが必ずしも同じにはならないということから、古代の世界では、本を写すことと本を書くことに現在と同様のはっきりとした境界線はありませんでした。両者の間にも意味のあるものが存在していました。古代人も、作者という観念は知っていました。たとえば、この劇がソフォクレスによって書かれたものであることはわかっていました。しかし、本を書くことと本をコピーすることの中間にも、意味のある仕事がほかにありました。たとえば、本の一部だけをコピーしてから、新しい言葉を書き足し、さらにコピーを続けて、また新たな言葉を書き足すというようなことができました。これは「注釈（コンメンタール）を書く」と呼ばれる作業です。そして、これらの注釈は高く評価されていました。

また、1 冊の本から一節をコピーし、別の言葉を書いてから、別の本の一節を

コピーし、さらに言葉を書き足すこともできました。これは、要約集を作るという作業で、やはり有益な仕事でした。作品の中にはそれ自体としては失われてしまったものの、オリジナルよりも人気を集めた他の本に引用されることによって、部分的に生き残ったものがあります。おそらく、もっとも面白い部分がコピーされたのでしょう。人々はこのようなコピーを無数に作っていましたが、それほど面白くないオリジナルをいちいちコピーしたりはしませんでした。

私が知る限り、古代の世界には、著作権のようなものはありませんでした。本をコピーしたい人は、誰でも本をコピーすることができました。その後、印刷機が開発され、本は印刷機によってコピーされるようになりました。現在、印刷機は、コピーをしやすくした量的な改善以上の意味を持っています。印刷機は規模の経済性を導入したので、コピーのさまざまな方法に一様でない影響を与えました。活字をセットするのは大変な労力でしたが、ページのまったく同じコピーを大量に作るのはずっと楽でした。そのため、本のコピーを作る仕事は、中央集権化された大量生産行為になったわけです。特定の本のコピーは、おそらくごくわずかの場所で作られるようになりました。

印刷機の登場は、通常の読者が効率的に本のコピーを作れなくなったということでもありました。印刷機がなければ、効率的にコピーを作ることはできません。そのため、本のコピーの作成は、産業になりました。

しかし、印刷機が登場してから数世紀の間は、印刷本は写本を完全に駆逐したわけではありませんでした。写本は、金持ちによって、あるいは貧乏人によって、依然として作られていました。金持ちは、自分がいかに金持ちかを誇示する非常に美しいコピーを得るために手写本を作り、貧乏人は印刷されたコピーを買うだけのお金はないものの、手で本を書き写す時間はあるために手写本を作りました。歌にもあるように、「持っているのが時間だけなら、時は金なりとは言わないものさ」

そこで、ある程度までは、書写の仕事も行われていました。印刷のコストが充分安くなり、貧乏人でも文字が読めれば印刷された本を手に入れられるようになったのは、1800年頃ではないかと思います。

著作権は、印刷機とともに生まれ、成長しました。そして、印刷機という技術の性格上、著作権は産業に対する規制という効果を持ちました。著作権は、読者

ができることに対しては制限を加えませんでした。著作権が制限したのは、出版社と著作者ができる行為です。イギリスにおける著作権は、最初は一種の検閲でした。本を出版するためには、政府の許可を得なければなりませんでした。しかし、考え方は変わってきました。合衆国憲法が制定される頃には、人々は著作権の目的として別のことを考えていました。そして、同じ考えがイギリスでも受け入れられたのだと思います。

合衆国憲法の策定過程では、著書のコピーを独占的に作成する権利である著作権を著作者に与えるべきだと提案されましたが、この提案は却下されました。代わりに、まったく異なる提案が採用されたのです。それは、進歩を促進するために、国会はこのような独占権を生み出す著作権制度を確立する選択肢を持つというものでした。ですから、合衆国憲法に従えば、このような独占権は、権利所有者のために存在するわけではありません。科学の進歩を促進するために存在するのです。著作者に独占権を与えるのは、一般人のために奉仕するように著作者の行動を変える手段なのです。

目的は、他の人々が読める本がより多く書かれ、出版されるようにすることです。この制度［著作権］は、著作活動の活性化に貢献し、科学やその他の分野の著作を増やして、社会に学習の機会を与えるはずだと考えられています。著作権が奉仕すべき目的は、ここにあるのです。個人的な独占権の創設は、目的のための手段に過ぎず、その目的とは一般国民にあります。

印刷機時代の著作権は、産業規制でしたので、苦痛を生むようなものではありませんでした。著作権が制限していたのは、出版社と著作者の活動だけでした。厳密に言えば、手作業で写本を作っていた貧乏人も、著作権を侵害していたと言えるかもしれません。しかし、著作権は産業規制と理解されていましたので、彼らに対して著作権を強制しようとする人はいませんでした[1]。

印刷機時代の著作権は、出版社が存在する場所に限って実施すればよく、出版社はその性質上どこに存在するかが明らかですから、非常に簡単に実施できました。本を売ろうとするなら、どこに行けば買えるかを人々に知らせなければなりません。すべての人の家に押しかけて、著作権を強制する必要はなかったのです。

---

*1　最初の著作権法が言及していたのは、出版と印刷だけである。手作業による複製作成はまったく規制されていなかった。おそらく、この規制は産業を目的としていたからだろう。

著作権は、その文脈のもとでは有益な制度になっていたかもしれません。アメリカの著作権は、法学者からは取引、すなわち国民の著作者に対する廉売と考えられています。国民は、コピーを作るという自然権の一部を売り渡し、その見返りとしてより多くの本が執筆、交換されるという利益を受け取るのです。

では、これは有利な取引でしょうか。印刷機以外ではコピーを効率的に作成できず、印刷機の所有者が限られていたために、一般人がコピーを作れなかった時代には、国民は行使できない、実際的な価値のない自由を売り渡していただけです。生きていることの副産物として役に立たないものを手に入れたとき、それを価値のあるものと交換できるなら、得をしたと言うことができるでしょう。当時の国民にとって、著作権が有利な取引だったのはそのためです。

しかし、文脈は変わり、著作権の倫理的な評価も変わらざるを得なくなりました。しかし、倫理の基本原則は技術の進歩によって変わるようなものではありません。それは非常に根の深いものであり、技術の進歩というような偶然によって変わるようなものではないのです。しかし、私たちが特定の問題に対して下す判断は、どのような選択肢があるかによって決まってしまいますし、選択肢は文脈の変化によって変わります。著作権法の分野で現在起きていることは、印刷機が次第にコンピュータネットワークにその座を明け渡し、自らの時代を終えようとしているということです。

コンピュータネットワークとデジタル情報テクノロジーは、情報を読み、利用できる人間なら誰もがコピーでき、コピーの難度が誰にとっても同じだった古代と同じような世界に私たちを連れ戻そうとしています。しかも、それらは完全なコピーであり、今までに作成できたどのコピーにも匹敵する品質を持っています。そのため、印刷機や類似のテクノロジーが導入した中央集権化と規模の経済性の時代は、終わろうとしているのです。

このような文脈の変化は、著作権法の作用の仕方も変えました。現在の著作権法は、産業規制ではありません。一般人に対する苛酷な規制になっています。以前の著作権は、作者のために出版社に課せられた制約でした。現在は、実質的に、出版社のための国民に課せられた制約になっています。以前の著作権は苦痛にならないもので、論争の種にはなりませんでした。著作権が一般人の足枷になることはありませんでした。今［現在］は違います。出版社にとって優先順位のもっ

とも高い仕事は、コンピュータを持っている一般人に規制をかけることです。以前の著作権は、居場所が簡単にわかり、何を出版しているかが簡単にわかる出版社だけを対象とした規制でしたから、簡単に実施できました。現在は、著作権を守らせるためには、監視、強制、厳罰が必要です。そして、私たちは米国を始めとする各国でこれらの内容が立法化されるのを見ています。

以前の著作権は、国民にとって、行使できない自由を取引するものであり、有利な取引でした。しかし、現在の国民は、これらの自由を行使できます。役に立たず、売りに出す習慣になっていた人生の副産物が、突然役に立つものになったらどうするでしょうか。実際に、消費したり利用したりすることができるようになったのです。そういうものを全部売りに出したりはしないでしょう。一部は手元に残すはずです。そして、一般人が自然に望むことはこれなのです。自分の希望を声にする機会があったとして、一般人が当然行うのは、このようにこの自由の一部を手元に残して行使することでしょう。Napster は、このことをよく示す大きな例です。一般人はコピーする自由を放棄せず、行使することを選んだのです。今日の環境に合わせて私たちが著作権法を自然な形に改正するとすれば、著作権者が持つ権力を削減することです。著作権者が一般人に対して課している制約を削減し、一般人の当然の自由を拡大することです。

しかし、出版社はそのようなことを望んでいません。出版社が望んでいるのは、正反対のことです。出版社は、情報のあらゆる利用をしっかりとコントロールできるところまで著作権者の権力を高めることを望んでいます。そして、この圧力は、著作権者にいまだかつてない高い権力を与えるところまで、著作権法を変えてきました。印刷機の時代に一般人が持っていた自由が今や奪い去られようとしています。

たとえば、e-book を見てみましょう。e-book には、非常に大がかりな誇大広告があり、ほとんど騙されないではいられないほどになっています。私が飛行機でブラジルに行ったときに機内で読んだ雑誌には、10 年から 20 年で e-book への完全な移行が完了するだろうとする記事が掲載されていました。この種のキャンペーンは、明らかに、e-book に投資している人々が流しているものです。彼らはなぜそのようなことをするのでしょうか。たぶん、e-book は、印刷された本の読者が常に持っていて、今でもまだ失われていない自由を奪い去るチャンスなの

です。たとえば、友人に本を貸す自由、公共図書館から借りてくる自由、コピーを古本屋に売る自由、誰がどの本を買ったかを管理するデータベースに記録を残さずに匿名でコピーを買う自由です。あるいは、本を2度読む自由も含まれるかもしれません。

これらは出版社が奪いたいと思っている自由ですが、印刷された本の場合、紙を奪うのではあからさまに過ぎて大騒ぎになるので手を出すことができません。そこで、出版社は間接的な戦略を見つけたのです。まず、e-bookが存在しない時代にe-bookからこれらの自由を取り除く法制を手に入れ、論争を封じ込めます。自らの自由に慣れ、それを守ろうとするe-bookの既存のユーザーはいません。出版社が1998年のデジタルミレニアム著作権法（DMCA）で手に入れたのは、それです。次に、出版社はe-bookを導入し、人々を次第に印刷された本からe-bookに移行させます。結果として、読者はこれらの自由が取り除かれた瞬間、自由を取り戻すために闘う瞬間を持たずに、これらの自由を失ってしまうのです。

同時に、他のメディアからユーザーの自由を奪う動きも顕在化しています。たとえば、DVD化された映画は、以前は秘密だった（秘密になるのを意図して作られた）暗号化されたフォーマットで発売され、プレーヤーにある制限を組み込む契約にユーザーがサインしない限り、映画会社はそのフォーマットについての情報を与えず、ユーザーはDVDを再生できません。国民は、それぞれの法的権利を完全に行使することさえ禁止されてしまうのです。その後、ヨーロッパの数人のプログラマがDVDのフォーマットを明らかにし、DVDを読み出すフリーソフトウェアパッケージを書きました[*2]。これで、GNU/Linuxオペレーティングシステムの上では、買ってきたDVDを鑑賞できるフリーソフトウェアを使えるようになったということですが、これは完全に合法的な行為です。フリーソフトウェアでこのようなことをすることは、当然認められなければならなりません。

しかし、映画会社はこの動きに反対して法廷に打って出ました。映画会社は、誰かがマッドサイエンティストに「でも博士、人類には知ってはならないことがあります」というような説教をたれる映画を無数に作ってきましたが、自分の映画を見過ぎた余り、DVDのフォーマットは人類が知ってはならないことだと思

*2　現在は、そのようなパッケージが無数にある。最初のものは、「DeCSS」という名前だった。

い込むようになったらしい。しかも、映画会社はDVD再生ソフトウェアを全面的に検閲できるような判決を勝ち取りました。米国外のこの種の情報が法的に認められているサイトにリンクを張ることさえ禁止されてしまったのです。この判決に対しては、控訴がなされています。法廷助言者摘要書（friend-of-the-court brief）の署名者の末席に私も名前を連ねているのは、誇ってよいことでしょう。もっとも、この闘いで私が果たしている役割はごく小さなものに過ぎませんが。

合衆国政府は、直接向こう側に肩入れしています。でも、DMCAがそのまま通過した理由を考えれば、これは驚くに足らないことです。それは、アメリカの政治献金制度です。候補者は、選挙の前に企業に買収され、言わば合法的に賄賂を受け取っています。もちろん、候補者は主人が誰か、誰のために働いているかを知っています。ですから、企業により強大な権力を与える法案を通過させるのです。

この個別の闘いが今後どうなるかはわかりません。その一方で、オーストラリアは同様の法律を成立させ、ヨーロッパもほとんど立法化するところまで来ています。地球上でDVDのフォーマット情報が公開されている場所がなくなるようにしようというのです。もっとも、アメリカは公開情報の頒布を国民に禁止するという分野では、依然として世界のトップを独走しています。

しかし、アメリカは情報統制を優先事項に置いた最初の国ではありません。ソ連もまた、情報統制を非常に重視しました。ソ連では、違法コピーとその頒布は地下出版と呼ばれ、その撲滅のために一連の方法が編み出されました。まず、違法コピーを阻止するために、個々のコピー装置に何がコピーされているかを監視するガードを付けました。次に、違法コピーによって逮捕された人々に厳罰を下しました。シベリア送りになったのです。第3に、隣人や同僚を情報警察に密告するよう唆しました。第4は連帯責任。「お前らは、あのグループを監視しろ。俺があの中の誰かを違法コピー実行犯として検挙したら、お前らは全員監獄行きだ。だからしっかり監視するんだぞ」　第5はプロパガンダ。子供の頃から、違法コピーなどするのは恐るべき人民の敵だけだと信じ込ませるのです。

アメリカは、今やこれらすべての方法を使うようになりつつあります。まず、コピー装置の監視ですが、コピー店には、客が何をコピーしているのかをチェックする人間の護衛がついています。しかし、コンピュータで何がコピーされてい

るのかを人間の護衛に監視させるのでは、高くつきます。人間の労働コストは高いのです。そこで、ロボットの護衛を用意しました。DMCAの目的はこれです。このソフトウェアは、コンピュータに入り込みます。データにアクセスするにはそのソフトウェアを使うしかなく、コピーはそのソフトウェアによって禁止されます。

現在、すべてのハードディスクにこのソフトウェアを導入し、何らかのネットワークサーバからアクセス許可を得ない限り、ハードディスク上のファイルにさえアクセスできなくしようとする計画が進められています。このソフトウェアをバイパスしたり、バイパスの仕方を教えたりするだけで罪になるというのです。

第2の厳罰主義。数年前、役に立つだろうということで何かのコピーを作って友達に手渡したからといって罪に問われることはありませんでした。アメリカでそのようなことが犯罪になったことはなかったのです。しかし、今日では、これが重罪になります。隣人と共有したために何年も監獄に閉じ込められるのです。

第3の密告については、テレビやボストンの地下鉄の広告で、同僚を情報警察に密告するよう呼びかけているものがあります。情報警察の正式名称は、ソフトウェア出版協会と言います。

第4の連帯責任。アメリカでは、ISPに対して顧客がポストした情報への法的責任を負担させるという手法を取っています。クレームがついてから2週間以内に情報を切り離すか取り除く体制を取らない限り、ISPは責任を問われることを避けられません。数日前のことですが、Citibankの汚い手口を批判する抗議用サイトがこのようにして切り離されたという話を聞きました。今では、運が悪ければ裁判で争うことさえできません。単純にサイトが切り離されてしまうのです。

最後のプロパガンダは子供時代から始まっています。「海賊行為」という単語は、そのために使われているものです。数年前を振り返るなら、「海賊行為」という単語は、著作者に報酬を支払わない出版社に対して使われていました。しかし、今はまったく正反対です。出版社の支配下から逃れた一般人に対して使われているのです。禁止されているコピーを行うのは、人民の仇敵以外の何者でもないと思い込ませようというのです。これは、「隣人と情報を共有するのは、船を攻撃するのと倫理的には同じだ」と言っているのと同じです。皆さんはそうだと思わないでいただきたいし、そう思わないなら、この意味でこの言葉を使わないように

してください。

出版社は、自らの権力を強化するために法を買収しています。また、出版社は著作権の存続期間の延長も進めています。合衆国憲法では、著作権の存続期間は限定されていなければならないとされていますが、出版社は著作権を永遠に抱え込んでいたいのです。しかし、憲法の修正は難しいので、同じ結果が得られるより簡単な方法を見つけてきました。20年ごとに著作権を遡及的に20年ずつ延長するのです。ある特定の時点を取れば、著作権は名目上ある期間で失効し、ある特定の日に消えます。しかし、すべての著作権が20年ごとに20年ずつ延長されるので、失効日は永遠にやってきません。つまり、すべての仕事が再び共有に戻ることは永遠にないのです。この方式は、「分割払い式永久著作権」と呼ばれています。

著作権を20年延長した1998年の法律は、メインスポンサーの1つがディズニーだったので、「ミッキーマウス著作権延長法[*3]」と言われています。ディズニーは、ミッキーマウスの著作権が切れそうなことに気づいており、その著作権から巨額の利益を得ていたので、何としても失効を避けたかったのです。

グローバリゼーション

ところで、この講演のもともとのタイトルは、「著作権とグローバリゼーション」でした。グローバリゼーションとは、経済的効率性とかいわゆる自由貿易条約の名のもとに進められている政策のことですが、実際には企業の法律や政策に対する影響力を高めようという動きです。本当は、自由貿易の問題ではなく、権力の移行の問題です。それぞれの利益を追求してよいはずのすべての国の国民から立法権を奪い、それら国民の利益を考慮しない企業に権力を与えようとしているのです。

企業から見て、民主主義は問題であり、これらの条約はこの問題に止めを刺すように作られています。たとえば、NAFTA[*4]は、企業が外国で利益を上げるのを

---

*3　正式名称は「ソニーボノ著作権保護期間延長法」。

*4　North American Free Trade Agreement: 北米自由貿易協定

妨害する法律を取り除けるようにその国を告訴できるような条項を持っています。つまり、外国企業のほうが自国の国民よりも強い権力を持つのです。

これをNAFTAよりも広い範囲に拡張しようとする動きがあります。たとえば、いわゆるアメリカ自由貿易地域の目標の1つはそれで、この原則を南米、カリブ海のすべての国々に広げようというのです。さらにそれを全世界に広げる、投資に関する多国間合意の形成も計画されています。

1990年代に明るみに出た動きの1つは、この種の条約がより強力でより制限の厳しい形で世界中に著作権を押し付けようとし始めたことです。これらの条約は、自由貿易条約とはとても言えません。実際には、自由貿易を廃絶し、大企業が世界経済を支配できるようにする、企業保護貿易条約です。

アメリカは、発展途上国だった1800年代、外国の著作権を認めませんでした。これは慎重に考えられた賢明な決定でした。アメリカが外国の著作権を認めても、資金の海外流出を招く一方で、良いことはほとんどなく、単純に不利だと考えたのです。

今日の発展途上国にも同じ論理が当てはまるはずですが、アメリカはこれらの国が自らの利益に反する方向に進むよう強制するだけの強大な権力を持っています。実際、この文脈で国の利益を言うことは誤りです。ほとんどの皆さんは、一人一人の富を加算することによって国民の利益を判断するのは誤りだという話を聞いたことがあるでしょう。働くアメリカ人が10億ドルを失い、ビル・ゲイツが20億ドルを儲けたら、アメリカ人全般の生活がよくなったと言えるでしょうか。それはアメリカにとって良いことでしょうか。合計だけを見れば、良くなっているように見えます。しかし、ビル・ゲイツが本当にもう20億ドル必要としているはずはありませんし、最初からそれほど持っていなかった他の人々は、10億ドルを失ってさらに苦しくなっているはずです。ですから、合計で判断するのは誤っているということです。これらの貿易条約についての議論で、この国あの国の利益という話が飛び交っているとき、実際に議論されているのは、それぞれの国の全国民の収入の合計です。裕福な人々と貧しい人々が合計されているのです。ですから、国の利益という言葉は、実際には国の中での富の分配の効果を無視させ、条約がさらに格差を広げることを隠すためのペテンなのです。そして、アメリカで行われているのは、そういうことです。

世界中に著作権を強制することは、実際にはアメリカの利益に寄与するわけではありません。得られるのは特定の企業のオーナーたちの利益です。彼らの多くはアメリカにおり、一部は他の国にいますが、いずれにしても、一般人の利益にはなりません。

著作権を見直そう

しかし、どうすればよいのでしょうか。たとえば、合衆国憲法に述べられている著作権の目標（進歩を促進するという目標）を信じるなら、コンピュータネットワーク時代の賢明な政策とはどのようなものでしょうか。当然ながら、著作権を強化するのではなく、デジタルテクノロジーやコンピュータネットワークの長所を利用できるような自由の領域を一般人のために確保するために、著作権を後退させなければなりません。しかし、どの程度後退させたらよいのでしょうか。これは面白い質問です。私だって、著作権を完全に葬り去るべきだと思っているわけではありません。確かに、従来の著作権の考え方では自由を放棄しすぎになりますが、進歩のために自由の一部を取引するという考え方は、今でもある程度までは有効です。しかし、賢明な思考のためにまず認識しなければならないことは、完全に統一的な制度を作る必要はないということです。あらゆる種類の仕事に対して、同額の取引をすることにこだわることはありません。

実際、すでに音楽にはさまざまな例外が設けられていて、統一は崩れています。音楽は、現行著作権法のもとでかなり異なる扱いを受けています。しかし、出版社がご都合主義的に統一性を強調するのは、あるずるい方法を使うためです。出版社は、特殊で特異な事例を引っ張り出してきて議論を進め、その特殊ケースでは、強力な著作権があるのは良いことだということを示します。そして、統一を守るために、すべてに対してその強大な著作権が必要だと結論付けるのです。ですから出版社は、比較的まれで特殊な上に、実際には大して重要な意味がないくせに、彼らにとって最強の議論を進められるような例を拾ってきます。

しかし、その特殊で特異な事例では、言われる通りの強大な著作権を容認すべきかもしれません。私たちは、すべての買物に同じ額だけ支払う必要はないのです。1000 ドルで新車が買えれば、いい買物かもしれません。しかし、牛乳の容器

のために1000ドルも支払うのだとすれば、とんでもないことです。生活のほかの場面では、買おうとしているあらゆるものに対して、特別な値段を支払うことはないでしょう。なぜ、著作権では同じ額を払おうとするのでしょうか。

私たちは、仕事の種類を区別する必要があります。そして、私はそのための方法を提案したいと思います。

まず最初に分類できるのは、機能的な作品です。つまり、仕事をするために使われる作品です。

これには、レシピ、コンピュータプログラム、マニュアル、教科書、辞書や百科事典などの参考図書が含まれます。これら機能的な作品については、どれも問題は基本的にソフトウェアと同じで、同じ結論が適用できると思います。機能的な作品に変更を加えるのは非常に役に立つことですから、変更済みのバージョンを公表する自由さえ与えられるべきです。人のニーズは、みな同じだとは限りません。私が自分で必要だと思う仕事をするためにこの作品を書いたとしても、皆さんがしたいと思う仕事のイメージは少し異なるかもしれません。ですから、あなたはこの作品を書き換えて、自分向きのものにしたいと思うでしょう。ここに、皆さんと同様のニーズを持つさらに別の人が現れたとします。この人は、皆さんが書き換えたバージョンをほしがるでしょう。料理をする人なら誰でもこのことは知っており、それは数百年前からずっと変わりません。レシピのコピーを作り、他人に渡すのはごく一般的なことであり、そのレシピを書き換えるのもごく当たり前のことです。皆さんがレシピを書き換えて友達のために料理を作ったとします。おいしいと思ったら、友達は皆さんに言うでしょう。「レシピをもらえる？」

すると、皆さんは自分のレシピを書き下ろして、そのコピーを友達に上げるでしょう。私たちがずっとあとの時代になってフリーソフトウェアコミュニティで始めたのは、これと同じことです。

というわけで、これが作品の第1分類です。

作品の第2分類は、思想を述べることを目的としたものです。これらの作品の目的は、人を語ることにあります。たとえば、紀要、評論集、学術論文、売買申込書、商品カタログなどです。これらの作品の目的は、誰かが考えたこと、見たこと、信じていることを読者に伝えることにあるので、書き換えてしまうと著作者を誤解させることになります。ですから、この分類に属する作品の書き換えは、

社会的に有益な行為ではありません。この種の作品について、読者に認められるべき自由は、本文に一切の変更を加えない複製だけでしょう。

しかし、次に考えなければならないことがあります。それは、営利目的で本文に一切の変更を加えない複製を作る権利を国民に与えるべきかどうかです。非営利目的だけで充分でしょうか。これらは2つの明確に異なる行為ですから、これらの問いについては別個に考えていくことにしましょう。つまり、非営利目的で本文に一切の変更を加えない複製を作る権利と、営利目的で本文に一切の変更を加えない複製を作る権利です。営利目的の本文に一切の変更を加えない複製作成を著作権の対象とし、国民に非営利目的の本文に一切の変更を加えない複製作成の権利を与えるのは、政策的な妥協として良い線引きかもしれません。営利目的の本文に一切の変更を加えない複製作成とすべての改訂版（これは、著者だけが認められるものです）を著作権の対象とすれば、現在と同じ程度の範囲内の執筆資金が同じような流れで著作者に渡るようになります。

非営利目的の本文に一切の変更を加えない複製を認めるということは、著作権がすべての人の家庭まで侵入してくることがなくなるということです。著作権は、簡単に実施できて苦痛のない産業規制に再び戻ります。厳格な懲罰や密告者がなくても実施できるわけです。こうすれば、現行システムの利点の大半を得た上で、恐怖の大半を取り除けます。

作品の第3分類は、美的、あるいは娯楽的な作品です。これらの作品でもっとも重視されるのは、作品を見たときに引き起こされる感覚です。これらの作品については、書き換えの問題は非常に難しくなります。一方では、芸術家のビジョンを反映する思想というものがあり、作品の書き換えはそれらのビジョンを混乱させることになります。しかし、もう一方では、一連の人々が作品に変更を加えていく過程で非常に豊かな結果を生み出すことがあるという伝承の効果があります。作品を作り出すのは芸術家ですが、既存の作品からの借用が良い結果を生み出すこともよくあります。シェークスピアの一部の作品は、他の芝居からストーリーを借りてきています。現在と同じような著作権法が当時も有効だったら、それらの作品は違法になっていたでしょう。美的、芸術的作品の変更版の発表をどのように扱うかは非常に難しい問題です。この問題を解決するためには、さらに小さな分類をいくつか作らなければならないのかもしれません。たとえば、コン

ピュータゲームのシナリオについては、変更版を公開する自由が与えられるべきでしょう。しかし、小説については、別の扱いが必要になると思います。おそらく、営利目的で出版するためには、オリジナルの作者の了承を必要とすべきでしょう。

これら美的作品の営利目的の出版を著作権の対象とする場合、著作者やミュージシャンに対する収益の流れの大半は現行通り、限界を含んだ形で残されることになります。こんなことを言うのは、現行制度が全然機能していないからです。作品が特定の人々を表現する場合に限り、適切な妥協を認めるようにすべきでしょう。

現在のような移行期を完全に通過し、コンピュータネットワーク時代が本格的に始まったときを先取りして考えるなら、著作者が仕事の対価を得るための新しい方法を構想できます。Internet越しに他人に送金できる、逆にいえば仕事の対価を得ることができるデジタル振替システムを想像してみてください。たとえば、暗号化技術を使えば、これはさまざまな形で実現できます。そして、これら美的作品の本文に一切の変更を加えない複製は全面的に認めることとしましょう。ただし、それらの作品を見たり読んだりしているときに、画面の横に「これをクリックして、作者（あるいはミュージシャン、その他）に1ドルを送ろう」と書かれたボックスが現れるものとします。ボックスはそこに留まります。邪魔になるような形では表示されません。脇のところにちょこっと表示されます。邪魔にはなりませんが、作家や音楽家を支援するのは良いことだということを思い出せるように、必ず表示されます。

そして、読んだり聴いたりしている作品が気に入ったとします。皆さんはきっと、「作者に1ドル払わないって法はないな、たった1ドルだぞ、それが何だってんだ？　なくしちゃうことだってあるくらいだぞ」というように考えることでしょう。こうして人々が1ドルずつ送り始めるようになります。この方法の良いところは、コピーを作ることが作家やミュージシャンのためになることです。誰かが友達にコピーを電子メールすれば、その友達も1ドルを送るでしょう。本当に気に入って1ドル送金を1度ならずするなら、現在皆さんが本やCDを買ったときに作者が手にする額よりも多くの額を作者に贈れます。現在、作者が手にしているのは、売上のほんのわずかなのです。作家と音楽家の名のもとに一般人に対する権力の強化を要求している同じ出版社が、実は作家や音楽家をいつも冷遇しているのです。

*Salon*誌に掲載されたコートニー・ラブの記事をぜひ読んでみてください。彼女が取り上げているのは、報酬を支払わずにミュージシャンの仕事を使おうとする海賊行為の計画です。その海賊とは、平均で売上の4%をミュージシャンに支払うレコード会社です。もちろん、大成功を遂げているミュージシャンは、もっと大きな割合を得ます。彼らは、もともと大きい売上から4%よりも大きい額を得ます。ということは、大多数のミュージシャンは、もともと小さい売上の4%に満たない額で契約しているということです。

仕組みはこうです。レコード会社は宣伝にお金をかけており、この出費をミュージシャンに対する前払い金と考えていますが、ミュージシャンのほうは間違ってもそうは思っていません。そこで、皆さんがCDを買ったとき、名目上はその売上の何%かはミュージシャンに渡るはずですが、実際はそうではなくて宣伝費の回収に使われます。ですから、大成功を遂げたミュージシャンでなければ、CDの売上からの収益など見たこともないということになります。

もちろん、ミュージシャンがレコード契約を結ぶのは、ごくわずかの金持ちの1人になりたいからです。ですから、ミュージシャンに与えられているものは、餌としてのくじ引きの抽選機のようなものです。ミュージシャンは、音楽に優れていても用心深いとは限りませんので、この罠に落ちたとしても不思議ではありません。そこで、契約はしたものの、手にしたものは宣伝だけということがあり得ます。だったら、一般人に制限を課すことを基礎とし、売りやすいうるさい音楽を押し付けている複合企業に奉仕しているシステムではなく、別の宣伝方法をミュージシャンに提供してみてはどうでしょうか。好きな音楽を共有したいというリスナーの自然な衝動をミュージシャンのために役立てない手はないでしょう。ミュージシャンに1ドル送金をするためのボックスをプレーヤーに表示すれば、コンピュータネットワークは、ミュージシャンに現行のレコード契約から得られる宣伝費とまったく同じだけの報酬を提供するメカニズムになるでしょう。

私たちは、現行の著作権制度がミュージシャンを支えるということでは実にお粗末な機能しか果たしていないことを認識するとともに、貿易がフィリピンや中国の生活水準の向上のためにほとんど機能していないことも認識しなければなりません。これらの「産業振興地域」ではひどい搾取がまかり通り、すべての製品がそのような搾取の産物だと言っても過言ではありません。グローバリゼーショ

ンは、これら海外の人々の生活水準を向上させるということでは、非常に非効率的です。たとえば、アメリカ人が時給20ドルで何かを作っていたとします。その仕事を日給6ドルのメキシコ人に与えたらどうなるでしょうか。アメリカ人労働者から大金を取り上げ、そのごく数%をメキシコ人労働者に支払い、残りが会社に入るわけです。目標がメキシコ人労働者の生活水準向上なのだとすれば、これはお粗末なやり方です。

著作権が関わる産業と同じ現象が現れ、同じ発想が通用していることは、注目すべきことです。間違いなく報酬に値するこれら労働者の名のもとに、皆さんは彼らにごくわずかのお金を与えるだけで、実際には私たちの生命を支配する企業の権力強化のために大部分を費やしているのです。

非常に優れた制度を取り替える場合には、より良い制度を準備するために必死に働かなければなりません。しかし、現行の制度がお粗末なものだということがわかっていれば、より良い制度を見つけるのはそれほど難しいことではないでしょう。現在の比較基準は非常に低いところにあります。著作権政策について考えるときには、いつもこのことを忘れないようにしなければなりません。

以上で、私は言いたいことをほとんど言い尽くしました。最後に、明日[*5]はカナダのPhone-In Sick Day[*6]だということに触れておきたいと思います。明日は、アメリカ以外の国々への企業権益の拡張を目論む米州自由貿易圏の最終交渉として位置付けられているサミットの初日で、ケベックで大規模な抗議行動が予定されています。私たちは、今回の抗議行動を弾圧するために過激な方法が使われてきたことを知っています。米加国境は本来いつでも自由に出入りできるはずですが、多くのアメリカ人がカナダへの入国を拒否されています。もっとも言い訳の効かない蛮行は、抗議行動参加者たちをブロックアウトする要塞化の手段としてケベックセンターのまわりに壁が築かれたことでしょう。これらの条約に対する広範な抗議に対してさまざまな汚い手段が無数に使われてきたことを私たちは知っています。民主的に選出された統治者たちから権力を奪い、企業や選挙を経ていない国際組織に与えたあとに残された民主主義なるものは、どのようなものであ

---

*5　2001年4月20日

*6　【訳注】特定の日に「病気」で休む比較的新しい示威行動。過去にブリティッシュ航空、アイルランド警察などの業務に大きな影響を与えたことがある。

れ、大衆抗議行動を弾圧してまで守るべきものではないのです。

私は、フリーソフトウェアとそれに関連した問題のために人生の中の 17 年間を捧げてきました。世界でもっとも重要な政治問題だと思ったから、そうしてきたわけではありません。良いことをするためには、この分野で自分の能力を使わなければならないと思ったからです。しかし、そのうちに政治全般の問題が大きくなってきて、今や、世界最大の政治問題の 1 つは、企業権力を一般国民や政府の上に位置付ける傾向への抵抗となっています。今日論じてきたフリーソフトウェアや情報に関連するその他のさまざまな問題は、その大問題の一部だと思っています。ですから、私は自分がこの問題に関わっていることに間接的に気づくことになりました。問題解決のために、自分が何か貢献できればと考えているところです。

質疑応答

司会者: それでは、しばらく皆さんから質問やコメントを受け付ける時間を持つことにしましょう。しかし、まず私に全般的な感想を言わせてください。ストールマン氏が私たちに示されたもっとも強力でもっとも重要な指針には、2 つの主要素が含まれていると思います。1 つは、著作権の古い前提条件、古い使い方が、不適切なものになっているという認識です。これらは、コンピュータとコンピュータネットワークの登場により蝕まれ、否定されています。わかり切ったことかもしれませんが、本質的なことです。

もう 1 つは、デジタル時代の知的創造的労働が取るさまざまな形態を見極め、評価する方法について、見直しが必要になってくるということです。ストールマン氏は、特定の分野の情報産業については他の分野よりも著作権保護を厚くすることを認めておられますが、これは間違いなく正しいことでしょう。著作権保護のさまざまな種類あるいはレベルを系統的に明らかにしていく作業は、コンピュータの登場によって知的労働

に課せられた問題に取り組む上で、非常に価値のあることだと思います。

しかし、コンピュータとは直接的には無関係ながら、より広範に民主的な権限の問題についてストールマン氏が述べられたこと、つまり政府や企業が一般人に対して行使している権力が強化されているという問題について述べられたことの背後には、別のテーマが潜んでいるように思います。ストールマン氏が話された内容のこのようなポピュリスト的で反企業的な側面は、示唆に富むものの、還元主義的で、あるいは単純化し過ぎなのではないかと思います。理想主義的に過ぎるのではないかとも思います。たとえば、著作者にお金を支払うことが奨励されていても義務とはなっていないこのすばらしい新世界で、小説家や詩人や作曲家やミュージシャンや教科書の著者は、果たして生き残ることができるのでしょうか。言い換えれば、現在実践されていることと、ストールマン氏が夢想する可能性との間には、まだ非常に大きな隔たりがあるのではないかということです。

というわけで、ストールマン氏に、この部分について話された内容をもう少し展開していただけるよう、特に「伝統的なクリエーター」と呼ぶべきものがストールマン氏の著作権制度のもとでちゃんと守られるのかどうかを説明していただけるようお願いして、私の発言を締めくくらせていただきたいと思います。

リチャード・M・ストールマン（RMS）:まず第１に、著作権の機能として「保護」という用語を使うべきではないということを申し上げなければなりません。著作権は、人々を制限するものです。「保護」は、著作権所有ビジネスのプロパガンダ用語です。「保護」という言葉は、何かが何らかの形で破壊されるのを防ぐことを意味します。演奏されるコピーが増えたからといって歌が破壊されるとは思いませんし、コピーを読む人が増えたからといって小説が破壊されるとも思いません。ですから、私はこの言葉を使いません。この言葉は、人々を誤った立場に導くものだと思います。

また、私は2つの理由から「知的財産権」について考えるのは良くないと思っています。まず、この分野におけるもっとも根本的な問いについて先入観を与えます。それは、これらのものをどのように扱うべきか、またこれらを一種の財産として取り扱ってよいものかどうかということです。「知的財産権」という用語を使うことは、この問いに対する答えが「イエス」であることを最初に前提とすることであり、他の考え方はないと決め付けることになります。

第2に、知的財産権という用語は、過度の一般化を招きます。知的財産権という用語は、著作権、特許権、商標権、企業秘密、その他独立した起源を持ち、異なる法制が敷かれているさまざまなものをひとまとめにしてしまいますが、これらはまったく異なる概念であり、共通点はどこにもありません。しかし、「知的財産権」という用語を聞いた人々は、一定の領域に適用される知的財産権の一般原則というものがあるかのような誤ったイメージに導かれ、これらさまざまな法律に類似点があるように勘違いします。著作権法と特許法と商標法はまったく違うものなのに、同じようなものだと考えるわけですから、このような誤解は、何をすべきかについて考えるときに混乱を招くだけではなく、法律が実際に主張していることを理解する上でも障害になります。

ですから、精密に思考を進め、法律の主張をはっきりと理解したければ、「知的財産権」という用語は避けるべきです。著作権について話し、特許権について話し、商標権について話し、そのほか話そうとしているテーマについて話すべきですが、知的財産権について話そうとすべきではありません。知的財産権についての見解は、馬鹿げたものにならざるを得ません。私には、知的財産権についての意見はありません。著作権、特許権、商標権についての意見はありますが、それらは別のものです。これらの法制はまったく別のものですから、それらの意見は、まったく異なる思考プロセスをたどって獲得したものです。

脇道に逸れてしまいましたが、これは非常に重要なことです。

では、ご質問の点についてお答えしましょう。もちろん、今はそれがどの程度うまく機能するかはわかりません。好きな作家やミュージシャンに自発的にお金を払うように人々に要請する方法がうまく機能するかどうかはわかりません。しかし、はっきりしていることは、そのような制度がうまく機能するかどうかは、ネットワークに参加している人々の数に比例するということです。そして、その数は、数年のうちに桁が変わるくらいの勢いで増えるでしょう。今試みても失敗するかもしれませんが、だからといって何かが証明されるわけではありません。10倍の人々が参加すれば成功するかもしれません。

まだデジタル振替システムは存在しないという問題もあります。ですから、今試してみたくても試せないのです。少し似たことを試すことはできます。今でも、Pay Palのように誰かに料金を支払うためのサービスに加入することはできます。しかし、Pay Palで誰かに送金しようと思ったら、煩雑な手続きを踏んだ上で、自分の個人情報をPay Palに差し出さなければなりません。しかも、Pay Palは誰が誰にお金を振り込んだかを記録しています。Pay Palがその情報を悪用しないと保証できるでしょうか。

1ドルを支払うのはいやでなくても、支払いに関連したトラブルに嫌気がさすということはあるかもしれません。ポイントは、その気になったときにできるだけ簡単に支払えるようにして、実際の金額以外にブレーキをかけるものがないようにすることです。そして、金額がごく小さなものであれば、躊躇する理由があるでしょうか。ファンというものは本当にミュージシャンを愛しているものですし、ザ・グレイトフルデッドのように成功を収めた（そして現に成功し続けている）バンドが、ファンに音楽のコピー、再頒布を勧めているという例もあります。テープを作ってそれをコピーすることを奨励しているからといって、彼らが音楽では生活できなくなったなどということはありませんし、レコードの売上が落ちたということさえありません。

私たちは、印刷機の時代からコンピュータネットワークの時代に緩やか

に移行しつつありますが、1日で時代が切り替わるということはありません。人々はまだ大量のレコードを買っていますし、それはまだ当分続くでしょう。あるいは永遠に続くかもしれません。レコードが残る限り、営利目的でのレコードの販売に未だに適用されている著作権保持という制度は、ミュージシャンを支えるということについて、現在と同程度の仕事をする必要があります。もちろん、この制度は余り良いものではありませんが、少なくともこれ以上悪くしてはなりません。

質問者（Q）：[フリーダウンロードとWeb上で連載小説を販売するというスティーブン・キングの試み[7]についてのコメントと質問]

RMS: はい、彼がしたことと実際に起きたことは、面白いテーマです。初めてその話を聞いたとき、私は勇気付けられました。彼は、一般人を縛り付けて逃がさないようにするのとは違う新しい世界に一歩踏み出そうとしているのだと思ったのです。彼が実際に読者に支払いを要請するために書いたものを見たのは、そのあとです。彼がしたことを説明するなら、分割払いの連載小説を刊行しつつ、「充分な入金があったら、続きを書く」と言ったわけです。しかし、彼が書いた要請は、ほとんど要請ではなく、読者に対する威嚇です。「お金を払わない読者は悪いやつだ。悪いやつが多過ぎるようだったら、俺は書くのを止めるぞ」

一般人が送金しようという気になるのは、このような形でないことは明らかでしょう。恐れさせるのではなく、好きにさせるのでなければ。

同じQ: 詳しく言えば、彼は読者の中の一定の割合（正確な数字はわかりませんが、90%くらいだと思います）が、一定額（これも1ドルか2ドルか、その程度です）を送ることを要求したのです。ダウンロードするときには、名前と電子メールアドレスとあといくらかの情報を入力しなければなりませんでした。そして、彼は、第1章をリリースしたあと、読者が規定の割合に達しなければ、次の章はリリースしないと言いました。ダ

---

*7　スティーブン・キングは、ホラー小説を中心として多くの著書を持ち、*New York Times* のベストセラー作家にも選ばれている。1冊の本をオンライン分割払い方式で販売しようとしたが（1回に1章分ずつ購入する）、作品を完結させる前にサービスを停止した。

ウンロードした読者からすれば、ずいぶん反感をそそられる内容だったと思います。

Q:　著作権がなく、自発的に寄付してほしいと言われるだけの制度は、盗作者に悪用されるのではないでしょうか。

RMS:　いいえ、私が提案したのはそういうことではありません。私の提案は、営利目的の頒布には著作権を適用し、営利目的ではない本文に一切の変更を加えない再頒布だけを認めるというものです。ですから、本当の作者のWebサイトではなく、自分のWebサイトにポインタを書き換えた人物は、著作権を侵害したということになり、今日と同じように告発されます。

Q:　わかりました。つまり、あなたはまだ著作権がある世界をイメージしているのですね。

RMS:　はい。さっき言ったように、そういう種類の仕事に対しては著作権はあります。私は、すべてのことを認めるべきだとは言っていません。私が提案しているのは、著作権の権力を弱めることで、廃止することではありません。

司会者:　リチャード、あなたの講演中に思い浮かび、今あなたがこの質問に答えているときにまた浮かんだ疑問なのですが、あなたはなぜ、コンピュータ自体が仲介者を取り除き、個人的な関係を築く方法（スティーブン・キングが拒んだ方法ですが）を検討されないのですか。

RMS:　はい、それは可能ですし、自発的な寄付方式もその1つです。

司会者:　出版社をまったく介さない方法を考えているわけですか。

RMS:　もちろんそうです。出版社は、作家を散々食いものにしていますからね。出版社の代表にこのことについて尋ねると、彼らは、「作家やバンドが出版社を通したくないと言われるのなら、それで結構、法的に義務付けられているわけではありませんよ」と答えるでしょう。しかし、実際には、出版社は、自分抜きで仕事をすることが不可能になるように、全力を尽く

しています。たとえば、出版社はコピー制限付きのメディアフォーマットを提案していますが、このフォーマットで作品を発表したければ、大手出版社を通さなければなりません。大手出版社は、こういうメディアの作り方を誰にも教えませんから。つまり、出版社は、すべてのプレーヤーがこのフォーマットで再生を行い、プレーヤーで再生できるものを作りたければ、出版社を介在させなければならないような世界を望んでいるのです。実際、作家やミュージシャンが直接出版することを禁止する法律はありませんが、直接出版には現実性がないのです。それに、ヒットすれば金持ちになれるという誘惑があります。出版社は、「うちで君のことを宣伝すれば、君はきっとビートルズ（ここのところには、大成功を収めた任意のグループの名前を入れてください）のような金持ちになれるよ」と言いますが、もちろん、本当にそうなるミュージシャンはごくわずかです。しかし、多くのグループは、その言葉につられて契約書にサインし、永遠に縛り付けられるのです。

出版社は、一般に作者との契約を余り守ろうとしません。たとえば、一般に書籍の出版契約には、本が品切れになったら権利は作者に返還されるという条項が含まれていますが、一般に出版社はこの条項を守ろうとはしてきませんでした。強制しなければ守ろうとしないことが多いのです。そして、最近は電子出版を永遠に品切れにならないという言い訳に使おうとしています。品切れにならなければ、権利を返還する必要もないというわけです。出版社は、作者が無力なうちに契約させれば、そのまま無権利状態が続くと考えているわけです。権力を持っているのは、出版社だけです。

Q:　さまざまな種類の仕事について、すべてのユーザーに対して適切な形でコピーを作る自由を保証するようなフリーライセンスを用意しておいてはいかがでしょうか。

RMS:　はい、その仕事は始まっています。しかし、機能的な作品以外の分野では、1 つのものが別のものの代わりになるということはありません。機能的な作品、たとえばワープロについて考えてみましょう。誰かがフリーの

ワープロを作ったら、それを使えばいいわけです。フリーではないワープロは不要になります。しかし、1曲のフリーソングでフリーではないすべての曲が不要になるわけではありませんし、1篇のフリー小説でフリーではないすべての小説が不要になるわけではありません。この種の作品では、事情が異なるのです。そこで、そんな法律など守るに値しないということを私たちが認識すればよいのではないでしょうか。隣人と共有するのは悪いことではありません。誰かが隣人との共有は認められていないと言ったとしても、聞く耳を持たなければよいのです。

Q:　機能的な作品に関してですが、著作権廃止のニーズとこれら機能的作品の開発を促進するための経済的な動機のニーズとの間で、どのようにバランスを取っていったらよいとお考えですか。

RMS:　まず最初に確認しておかなければならないことは、その経済的動機というものは、人が考えているほど必要ではない、人が考えているよりずっと少なくてよいということです。フリーソフトウェア運動を見てください。10万人ものパートタイムのボランティアがフリーソフトウェアを開発しているんですよ。また、一般の人々にこれらの仕事のコピー、変更を禁止することなく、仕事のための資金を獲得するための方法がほかにもあることを示しています。これは、フリーソフトウェア運動が残したすばらしい教訓だと思います。コンピュータを使いながら、他人と協力し、共有する自由を確保する手段を提供しているということは別として、フリーソフトウェア運動は、支払いを強制する特別な権力がなければ誰もお金を出さないという否定的な想定を覆したのです。多くの人々は、自発的にそうします。たとえば、モノグラフの執筆について考えてみてください。これは科学のさまざまな分野で教科書として役に立つものですが、非常に基本的なものを除けば、著者たちはモノグラフで金儲けしようなどとは思っていません。現在は、フリー百科事典のプロジェクトがあります。これは、実際には商用フリー百科事典プロジェクトですが、これが進行しつつあります。私たちにもGNU百科事典のプロジェクトがありましたが、このプロジェクトが私たちのライセンスを受け入れてくれたので、

一緒に行うことにしました。1 月に、商用プロジェクトは、百科事典のすべての項目について GNU 自由公開文書使用許諾書を適用することになりました。ですから私たちは、「彼らと力を合わせて、このプロジェクトに貢献するよう、人々に呼びかけていこう」と言っています。このプロジェクトは NUPEDIA というもので、http://www.gnu.org/encyclopedia/にはここへのリンクが含まれています。このように、私たちはフリーな知識のコミュニティによる開発の試みを、ソフトウェアから百科事典に広げたわけです。今、私には、これら機能的作品のすべての分野で、作品が使い辛くなってしまうほどの経済的動機を設ける必要はないという確信があります。

司会者: 他の 2 つの分野（[思想、娯楽]）はどうでしょうか。

RMS: 他の 2 分類の作品については、わかりません。作品がお金になるかどうかを気にせずに小説を書ける日が来るかどうか、私には何とも言えません。経済的希少性がなくなった世界では、可能だと思います。経済的希少性のない社会を築くためには、経済と法律に対する企業の支配を取り除かなければならないと思います。実際には、これは鶏が先か卵が先かという問題です。どちらを先にしたらよいのでしょうか。企業支配を取り除かずに、人々が無性にお金をほしがる必要のない世界にたどり着くにはどうしたらよいのでしょうか。企業支配を取り除くにはどうしたらよいのでしょうか。私にはわかりません。しかし、まず著作権制度との妥協を提案し、次にそれらの作品を書く人たちに収入の流れを確保するために、妥協的著作権制度の下支えを受ける形で自発的な支払い方式を提案しようとしているのは、それらわからないことがあるからです。

Q: 政治献金制度によってアメリカの政治家たちが企業の利益にがんじがらめになっている状態で、この妥協的な著作権制度を実現できると本当に思っているのですか。

RMS: それは困った問題です。実現方法がわかればよいのですが、恐ろしく難しい問題になるでしょう。問題の解決方法がわかっていたら、解決して

いるはずですし、私はもっと胸を張っていられたところですが。

Q:　あなたは企業支配に対してどのように闘うのですか。と言うのも、企業が裁判につぎ込む費用は、合計で見るととてつもない額になっています。あなたが触れたDeCSS（CSS復号化）裁判では、被告側にかかった費用は150万ドルほどだそうです。企業サイドがどれだけの費用をかけたかは神のみぞ知るです。この巨額の費用の捻出方法については何かお考えはありますか。

RMS:　私には提案があります。映画を丸々ボイコットしようと提案しても、そんな提案は人々に無視されてしまうでしょう。それではあまりにも過激に感じられてしまうからです。ですから、結果的にはほとんど同じことになる少し異なる提案をしたいと思います。それは、非常に良いというはっきりとした理由がない限り、映画を見に行かないようにしようというものです。こうすれば、ハリウッド映画を完全にボイコットするのと実質的に同じ結果になるはずです。広い意味では同じことになりますが、意図は非常に異なります。多くの人々は、映画が良いと思っているかどうかと無関係に映画に行っています。皆さんがそのような習慣をやめて、とても良いと思うしっかりとした理由のある映画だけを見るようにすれば、映画会社からかなりの資金力を奪うことができます。

司会者:　今日の講演全体を理解するための１つの方法は、社会に変革を迫るような根源的なテクノロジーが現れると、それを支配する者との間で争いが起きることを認識することなのだろうと思います。今日の私たちは、過去に起きたことを繰り返しているのです。この角度から考えると、これから長期的に起きることについて、絶望したり、まして悲観的になったりするいわれはないのかもしれません。しかし、短期的には、テキスト、イメージ、その他あらゆる形態の情報の支配に対する闘いは、苦しく、広範なものになるでしょう。たとえば、私はメディア学の教師ですが、イメージの利用に関しては、近年いまだかつてない形で制限を受けるようになってきています。評論を書くために静止画を使いたいと思うとき、映画から取り出したものでさえ、使用許可を得るのが難しくなってきて

いますし、使用料は以前よりもはるかに高くなっています。学術調査や「公正利用」の法的分類についての論文を書くときでさえそうです。それでも、この長い移行期の長期的な展望は、短期的に起きていることほどひどいものではないと思います。ただ、私たちが今経験していること全体は、西側社会で繰り返されている技術資源の支配をめぐる闘いの新しいバージョンだと理解する必要があります。

また、古いテクノロジーの歴史自体、複雑な事象だったことも理解する必要があります。たとえば、印刷機がスペインに与えた影響は、イギリスやフランスに与えた影響とは根本的に異なります。

Q:　著作権の議論を聞くときにいつも引っかかるのは、「180 度方向転換したい、あらゆる支配を脱したいのだ」という話になることが多いことです。しかし、3 分類説には、著作権制度に対する一定の理解が含まれているように感じられました。実際、現在の著作権のあり方についての評論の中には、存続期間を特許権や商標権並に見直せばはるかにうまく機能するはずだというものがあります。講演者は、この戦略をどう思われるでしょうか。

RMS:　著作権の存続期間の短縮は、私も良いアイディアだと思います。著作権を 150 年も存続させるような方法を取ってまで（現行法では、そうなる場合があるのですが）、作品の公刊を奨励する必要は間違ってもありません。企業は、雇用を伴う作品の 75 年という著作権は、そのような作品を制作できるようにするために決して長いものではないと言っていました。そのような企業には、主張を裏付けるために、今後 75 年間の予想貸借対照表を提出してみろと言いたいものです。彼らが本当に望んでいたことは、古い作品の著作権を延長し、その利用に制限を加え続けられるようにすることです。しかし、どっかにタイムマシンをしまってあるというのでもない限り、今著作権を延長することによって 1920 年代の作品制作を奨励することがどうやってできるでしょうか。ある映画には、確かにタイムマシンが登場していました。ですから、映画会社の思考回路はそれに影響を受けているのかもしれません。

Q:　あなたは「公正利用」の概念を拡張することを考えたことがありますか。また、私たちに対して特に説明しておきたいニュアンスはありますか。

RMS:　2種類の作品について、すべての人に対し非営利の本文に一切の変更を加えない複製を認めるというアイディアは、公正利用概念の拡張と思われるかもしれません。これには、現在言われている公正利用よりも大きな意味があります。しかし、あなたが進歩のために国民が一定の自由を交換するという立場に立つのであれば、さまざまな場所に線を引くことができるでしょう。国民が交換したい自由がどれで、確保しておきたい自由がどれかということです。

Q:　話をちょっと先に進めますが、一部の娯楽分野には、公開演奏という概念があります。たとえば、著作権法は、季節がきたときにクリスマスキャロルを歌うことを禁じてはいませんが、公開演奏することを禁じています。公開利用を無制限の非営利逐語コピーに拡張するのではなく、それよりも狭く現在の公正利用の概念よりも大きいところに拡張するのはどうでしょうか。

RMS:　私もそれで充分だと思っていたことがありましたが、Napsterによって考えが変わりました。というのも、ユーザーは、非営利の本文に一切の変更を加えない再頒布のためにNapsterを使っているからです。Napsterサーバ自体は営利活動ですが、実際にサーバに情報を登録している人々は非営利で行っていますし、それは自分のWebサイトに情報を登録するのと同じように簡単です。Napsterが巻き起こした興奮と、関心、利用度の高さを見ると、Napsterは非常に役に立っているという結論になります。ですから、現在の私は、すべての情報の本文に一切の変更を加えない複製を非営利で公開再頒布できる権利がすべての人に必要だと考えています。

Q:　最近教わった考えですが、Napster問題は、公共図書館のアナロジーだというのです。Napster論争を聞いたことのある方は、このアナロジーをお聞きになったことがあると思います。これについてコメントしてい

ただけませんか。Napster は営業を継続できるようにすべきで、制限を課すべきではないとする Napster 擁護論者は、「公共図書館に言って本を借りた人は、使用料を払うわけではないし、数十回でも数百回でも、追加使用料を払わずに借りることができる。Napster に違いはあるのか」という言い方をします。

RMS:　Napster と公共図書館は、正確には同じだとは言えません。しかし、出版社が公共図書館をペイパーユース（利用ごとに支払う方式）の小売店のように変えたいと考えていることには触れておく必要があります。ですから、出版社は、公共図書館に反対しているのです。

Q:　著作権についてのあなたのお考えは、アフリカ向けの安い一般向けの薬の製造のような特許法関連の問題にも何らかの関連がありますか。

RMS:　いいえ、両者に類似点はまったくありません。特許権の問題と著作権の問題はまったく別個のものです。これらが互いに関係を持つという考え方は、「知的財産権」のような用語が使われ、これらの問題をひとまとめに考えることを促す動きが進められていることの不幸な結果の 1 つです。私が今日話してきたことは、コピーの価格は重要な問題ではないということです。しかし、アフリカ向け AIDS 薬にとって重要な問題は何ですか。価格です。価格以外の何ものでもありません。

私が話してきた問題が起きたのは、デジタル情報テクノロジーがすべてのユーザーにコピー作成能力を与えたからです。しかし、薬のコピーを作成する能力を与えてくれるものはありません。私は、自分が手に入れた薬のコピーを作る能力を持っていませんが、実際、そんな能力を持っている人はいないのです。薬は、そのようにして作られるものではありません。薬は、一般向けのものでも、アメリカから輸入されるものでも、高い建設費用のかかる中央集権化された工場以外では作れません。いずれにしても、薬は少数の工場で作られます。そして、問題は単純に薬を作るのにいくらかかるのか、アフリカの人々が買えるような値段で売られるかどうかです。

ですから、AIDS 薬の問題は非常に重要な問題ですが、これは今日の話とはまったく異なる問題です。実際に特許がコピーの自由の問題に接近する分野は１つだけあります。それは、農業です。多少なりともコピー可能で特許の対象にもなるのは、主として生き物です。生物は、生殖時に自分のコピーを作ります。まったく正確なコピーである必要はありません。遺伝子をシャッフルし直します。いずれにしても、農民たちは数千年もの間、自分自身のコピーを育てるという生き物の能力を利用してきました。農業とは、基本的に育てているものをコピーすることであり、毎年コピーし続けることです。植物や動物の種が特許の対象となり、遺伝子に特許が認められて動植物の中で使われると、農民は今までしてきたことを禁止されることになります。

特許の対象となっているある変種を畑で育てていたカナダのある農民は、こう言いました。「わざとやったわけではないんだ。花粉が飛んできて、遺伝子の渦巻きがうちの倉庫に紛れ込んだだけだ」　しかし、この主張は無視され、彼は作物を破棄しなければならなくなりました。これは、政府がどこまで独占主義者側に立てるかということの極端な例を示しています。

この問題については、コンピュータ内のもののコピーに適用されるのと同じ原則に従い、農民には種を保存し、家畜を育てる権利を無条件に与えるべきです。種苗会社に制限を加える特許はあり得ますが、農民に制限を加える特許はあってはなりません。

Q:　モデルを成功に導くためには、ライセンス以上のものが必要です。それについてお話しいただけますか。

RMS:　ええ、もちろん。もっとも、私は答えを知っているわけではありません。しかし、フリーな機能情報を発展させる上で非常に重要なものの１つは、理想主義だと思っています。この種の情報の自由が非常に重要だということ、自由な情報はフルに活用できるということを人々は認識しなければなりません。制限のある情報は、フルに活用できません。フリーでは

ない情報は、人々の間に分裂を持ち込もうとし、人々を孤立無援の絶望的な気分に押し込めます。こういったことを認識すれば、「協力して自分たちが使いたい情報を作ろう。そうすれば、その情報は、何ができるかを命令できる権力者の支配を脱することができる」と考えることができます。

これは、[フリーソフトウェアコミュニティの発展を] 大幅に加速させます。異なるさまざまな分野でこの考え方がどの程度機能するものか、私にはわかりませんが、教育の分野で教科書が必要になったときには、これが機能するのではないかと思います。世界中には教師がたくさんいます。有名な大学で教えているわけではない教師たち、たとえば高校で教えているかもしれないし、単科大学で教えているかもしれない。あまりいろいろなものを書いたり出版したりしていないかもしれませんし、求められることも少ないかもしれません。しかし、多くの教師は、明晰な人々ですし、自分のテーマのことを良く知っています。さまざまなテーマの教科書を書き、それを世界中で共有することができます。そして、その教科書を使って学んだ人々から大変感謝されることでしょう。

Q:　それはまさに私が提案したことです。しかも面白いことに、私が知っているのは教育の歴史です。私が進めているのは、教育電子メディアプロジェクトですが、実例がまだ見つかりません。あなたはご存知ですか。

RMS:　いいえ。私がこのフリー百科事典とフリー学習教材を提案したのは 2 年前で、そのときには物事がうまく回るようになるまで、おそらく 10 年はかかるだろうと思っていました。百科事典は、すでに回転し始めました。ですから、事態は私の予想よりも速く進んでいます。次は、フリーの教科書を書き始める人が何人か出てこなければなりません。自分の得意なテーマについて 1 冊、あるいはその一部を書くのです。1 冊のうちの数章を書いて、残りを書いてくれる人を募るのも良いでしょう。

Q:　私が探していたものは、実際にはそれ以上のものです。あなたが述べられたような構造を築く上で重要なのは、他のすべての人々が貢献できる

ようなインフラストラクチャを構築する人です。教材を寄付しようにも、幼稚園から高校卒業までを貫くインフラストラクチャがありません。

情報はあちこちから入りますが、それらはフリーライセンスのもとでリリースされているわけではありません。ですから、それをフリー教科書で利用できないのです。

RMS:　実際には、著作権は事実を対象とすることはできません。対象となるのは、書き方だけです。ですから、任意の場所である分野について学習し、教科書を書けば、その教科書をフリーにすることはできます。

Q:　しかし、1人の生徒が学校の全過程を履修するために必要なすべての教科書を私一人で書くことはできません。

RMS:　それはそうでしょう。私も、フリーオペレーティングシステム全体を書いたわけではありません。私が書いたのは一部だけで、あとは私と一緒に他の部分を書く人を誘ったのです。私は実例を示し、「私はこちらのほうに行きます。私と一緒に来てくれれば、そこにたどり着くことができます」と言ったのです。そして、充分な人々が参加したので、ここまでたどり着くことができました。ですから、この巨大な仕事を全部成し遂げるにはどうしたらよいか、という考え方をすると圧倒されてしまうかもしれません。大切なのは、そのような考え方をしないことです。一歩前進することを考えましょう。あなたが一歩前進したら、他の人が数歩前進します。それらの力を合わせていくうちに、最終的に仕事が完成するのです。

人類が自滅しないとすると、私たちが世界のためにフリーな教育インフラストラクチャ、フリーな学習教材を作るために今日していることは、人類が続く限り、役に立ちます。完成するまでに20年かかったとしても、それが何だというのでしょう。ですから、仕事全体の大きさを考えるのではなく、あなたがしようとしている部分のことを考えるのです。そうすれば、人々にそれが可能だということを示すことになります。そして、他の人々もそれぞれができることをするようになるでしょう。

初出: 2001 年 4 月 19 日に MIT で開催された Communications Forum における講演記録に編集を加えたもの。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第20章
フリーソフトウェア：自由と協力

イントロダクション

マイク・ユリツキー: 私はマイク・ユリツキーです。スターン・スクール・オブ・ビジネス教授で、先端工学センターの共同所長の1人でもあります。私たちコンピュータ科学科のメンバー全員を代表して、皆さんを歓迎いたします。講演者はエドが紹介してくれることになっていますが、その前に私からも一言ご挨拶させていただきたいと思います。

大学の役割は、論争を促し、面白い議論をする場を提供することです。そして、大きな大学の役割は、特に面白い議論を提供することです。この講演会はまさにそのようなものになるでしょう。私は、オープンソースの議論は特に面白いと思っています。言わば…… [聴衆の笑い]。

リチャード・M・ストールマン (RMS): 私がしているのは、フリーソフトウェアです。オープンソースは別の運動です [聴衆の笑い] [拍手]。

ユリツキー: 私がこの分野に入った1960年代には、基本的にソフトウェアはフリーでした。私たちはそのサイクルに入っていきました。その後、ソフトウェアメーカーが市場拡大のために別の方向に引っ張っていきました。PCの登場以来着手されたさまざまな開発プロジェクトは、それとまったく同じサイクルに入っていきました。

フランスにピエール・レヴィという面白い哲学者がいます。彼は、この方向への移行について論評している人です。技術だけではなく、社会、政治面での改革

にからめてサイバースペースへの移行を、人類の福祉を向上させる関係の変化として論じています。今回の討論がその方向になることを期待しています。つまり、いつもは大学内の安らぎの場として機能しているさまざまな学科を横断するものになることです。非常にスリリングな議論を期待しています。では、エド……。

エド・ショーンバーグ: 私は、クーラントインスティテュート・コンピュータ科学科のエド・ショーンバーグです。このイベントにようこそ。通常、紹介者は公開講演会では無用の長物ですが、今回は非常に有意義な役割を果たされたと思います。マイクがいとも簡単に実証されたように、紹介者は、たとえば不適切なコメントを加えることによって、[講演者が] 整理して誤りを正せるようにするとともに[聴衆の笑い]、論点をはっきりさせてくれます。

では、ご紹介するまでもない人物について、できる限り簡潔に紹介させていただきましょう。リチャードは、ローカルな活動からグローバルな思考を始めた典型的な例を提供する人物です。彼の出発点は、大昔のMIT AIラボでプリンタドライバのソースコードを参照できないことから起きた問題にあります。彼は、ソフトウェアの製作形態について、知的財産権の意味について、ソフトウェアコミュニティが実際に何を代表しているかについて、私たち全員が再考を余儀なくされるような首尾一貫した哲学を打ち立てました。では、リチャード・ストールマンさん、どうぞ [拍手]。

フリーソフトウェア：自由と協力

RMS: 誰か時計を貸してくれますか [聴衆の笑い]。ありがとう。この演台に立つ機会を与えてくれたMicrosoftに感謝の意を表したいと思います [聴衆の笑い]。数週間前、私はどこかで思いがけず著書が発禁になった著者のような気分になりました [聴衆の笑い] [1]。もっとも、その事件についてのすべての記事は、著者名を書き間違えていました。MicrosoftはGNU GPLをオープンソースライセンスと呼んでおり、ほとんどの記者がそれに追随したからです。もちろん、ほとんどの方々は、私たちの仕事がオープンソースとはまったく無関係であることも、

[1] この講演の数週間前、Microsoftのクレイグ・マンディー副社長がフリーソフトウェアを攻撃する講演を行った（フリーソフトウェアを「オープンソース」と呼びながら）。

「オープンソース」なる用語が作られる前に私たちがほとんどの仕事を終えていたことも、無邪気にご存知ないのです。

私たちは、フリーソフトウェア運動に属しており、私はフリーソフトウェア運動とは何か、何を意味し、私たちが何をしてきたのかということを話していくつもりです。そして、今回のスポンサーにビジネススクールが含まれているということから、フリーソフトウェアがビジネスや社会生活のその他の分野とどのような関係を持つかということについて、いつもよりも突っ込んだお話をしたいと思います。

さて、皆さんの中にはコンピュータプログラムを書いたことがないけれども、料理なら作ったことがある人がたくさんいると思います。料理を作るとき、特に料理の得意なほうでなければ、たぶんレシピを使うはずです。そのレシピは、友達からもらったコピーの場合もあるでしょう。そして、まったくの初心者でなければ、レシピを書き換えた経験があると思います。レシピには一定のことが書かれていますが、まったくその通りに従う必要はありません。一部の材料はなしで済ませて、キノコ好きならキノコを加えるようなこともあるでしょう。また、医者に塩分を控えるように言われていれば、塩加減を調節するはずです。腕次第で、さらに大きな変更を加えることができます。そして、レシピを書き換え、友達のために料理して気に入ってもらえたら、友達の中には、「ねえ、レシピをちょうだいよ」という人が出てくるかもしれません。そのようなとき、あなたはどうしますか。あなたが変更したレシピを書いて、友達のためにコピーを作ってあげるところでしょう。どのような分野のものであれ、機能的に役に立つレシピの扱い方としては、これが自然なものだと思います。

さて、レシピはコンピュータプログラムに非常に似ています。そして、コンピュータプログラムはレシピに非常に似ています。自分が望む結果を得るために、遂行すべき一連の手順を書いたものです。ですから、コンピュータプログラムでも同じことをする、つまり友達にコピーを手渡すのが自然なのです。プログラムがもともと書かれた目的と自分のしたい仕事にずれがあるなら、プログラムに変更を加えるでしょう。そのプログラムは、あなた以外の誰かのためにはすばらしい仕事をしたかもしれませんが、あなたのしたい仕事はそれではないのです。そして、あなたが書き換えたプログラムは、他の誰かにとっても役に立つかもしれ

ません。彼らは、あなたがしているのと同じような仕事を持っているのです。そのような人は、「コピーをもらえるかな」と言うでしょう。もちろん、そう言われたら、意地悪な人でなければコピーを上げるでしょう。まともな人ならそうするものです。

ここで、レシピがブラックボックスの中に閉じ込められていたらどうなるか考えてみましょう。どんな材料が使われているかはわかりませんし、まして変更することはできません。さらに、友達のためにコピーを作ったら、あなたは海賊と呼ばれ、何年も牢屋に閉じ込められそうになります。そのような世界は、レシピの共有に慣れていたすべての人々の怒りを集めるでしょう。しかし、私有ソフトウェアの世界とは、まさにそういうものなのです。他者に対する当たり前の親切が禁じられ、妨害される世界なのです。

私がそれに気づいたのはなぜでしょうか。それは、幸運にも1970年代にソフトウェアを共有していたプログラマのコミュニティの一員だったからです。このコミュニティの起源は、基本的にコンピューティングの草創期までさかのぼることができます。しかし、1970年代には、ソフトウェアが共有されているコミュニティは、少しまれな存在になっていました。私が働いていた研究所はむしろ極端な例で、オペレーティングシステム全体がコミュニティのメンバーによって開発されたものでしたし、それをすべての人と共有していました。やってきた人はすべて歓迎され、ソースコードを見たり、コピーを取ったりして、自分がしたいことを何でもすることができました。これらのプログラムには、著作権表示は含まれていませんでした。私たちのライフスタイルは協力であり、そのスタイルで安全に暮らしていました。そのために闘うことはなかったし、闘う必要もありませんでした。ただ単に、そのように暮らしていただけです。そして、私たちが知る限り、私たちはそのように暮らし続けることができたはずです。ですから、フリーソフトウェアはありましたが、フリーソフトウェア運動はありませんでした。

しかし、私たちのコミュニティは、いくつかの災難のために破壊されてしまいました。そして、最後に止めをさされたのです。私たちが仕事のために使っていたPDP-10コンピュータ[*2]は、製造中止になりました。私たちのシステム

[*2] Programmed Data Processor model 10。1970年代、多くの研究所、政府組織で使われていたメインフレームコンピュータ。

(Incompatible Timesharing System: ITS) は、60年代の初めから開発が始まったもので、アセンブリ言語で書かれていました。60年代にオペレーティングシステムを書くということは、そういうことだったのです。もちろん、アセンブリ言語は、ある特定のコンピュータアーキテクチャに縛られています。そのアーキテクチャが廃止されれば、すべての仕事は灰になります。使えなくなってしまうのです。私たちの災難とはそういうものでした。私たちのコミュニティの20年ほどの仕事が灰になったのです。

しかし、その前に準備をするというか、どうすべきかがわかるというか、こうなったときにどうすべきかという心構えができたというか、そういう経験をしました。どういうことかというと、Xeroxが私の勤務先だったAIラボにレーザープリンタをくれたのです。これは本当にすばらしい贈り物でした。何と言っても、Xeroxの社外に出た初めてのレーザープリンタだったのです。非常に高速で、1秒に1ページ印刷できましたし、そのほかさまざまな面で優れていました。ただ、ちょっと信頼性に欠けていたのです。それは、オフィス用高速コピー機をプリンタに改造したものだったからなのでしょう。そして、コピー機には紙詰まりがありますが、そうなっても誰かがそばにいて直すことができます。しかし、プリンタが紙詰まりを起こしたとき、プリンタを見ている人はいません。そのため、紙詰まりを起こしたまま、長時間放置されることがよくありました。

この問題の対処方法について、アイディアはありました。プリンタが紙詰まりを起こすたびに、プリンタ駆動マシンが私たちのタイムシェアリングマシンに通知を送り、そのマシンがプリントアウトを待っているユーザーに通知を送れば、ユーザーはプリンタのところに行って紙詰まりを直すでしょう。紙詰まりが起きているのを知っているのはそのユーザーだけですからね。プリントアウトを待っていて、プリンタが紙詰まりを起こしていることがわかっていれば、永遠に座って待っているのはいやだから、直しに行くというものです。

しかし、ここのところで完全に暗礁に乗り上げてしまったのです。と言うのも、プリンタを駆動するソフトウェアがフリーソフトウェアではなかったからです。ソフトウェアはプリンタに付属していましたが、バイナリ形式だけでした。私たちは、ソースコードを手にすることはできませんでした。Xeroxは、ソースコードをくれようとはしなかったのです。ですから、私たちはプログラマとしての能

力を持つにもかかわらず（私たちは自前のタイムシェアリングシステムを書いたほどなんですから）、プリンタソフトウェアにこの機能を追加することは不可能でした。

そして、私たちは黙って待っていなければなりませんでした。プリンタは、ほとんどの時間、紙詰まりを起こしていたので、プリントアウトが終わるまで1時間から2時間もかかりました。「プリンタが紙詰まりを起こすことはわかっているから、1時間待ってから、プリントアウトを取りに行こう」と思って1時間待ったあと、プリンタを見に行くと、その間ずっと紙詰まりを起こしていたというようなこともありました。その間、ほかの誰も直していなかったのです。そこで、紙詰まりを直して、もう30分待って戻ってみると、プリントアウトが終わる前にまた紙詰まりを起こしている。3分印刷して、30分紙詰まりで止まっている。イライラは極限まで達します。さらに頭に来るのは、自分たちならこの問題を直せるのに、誰か他人が自己中心主義のために私たちをブロックし、ソフトウェアの改良を妨害していることです。当然、私たちは憤りを感じていました。

その後、カーネギーメロン大学（CMU）の誰かがソフトウェアのコピーを持っているという話を聞きつけました。私は、しばらくしてからCMUに出かけ、彼のオフィスに行って言いました。「こんにちは、私はMITから来た者です。プリンタのソースコードのコピーをいただけないでしょうか」　彼は、答えました。「いいえ、私はあなたにコピーを上げないように約束しました」[聴衆の笑い]。私は唖然としました。非常に腹が立ち、どうしたらよいのかわかりませんでした。私にできたことは、きびすを返して、オフィスを立ち去ることだけでした。ひょっとすると、ドアをばたんと閉めたかもしれません[聴衆の笑い]。そしてその後もそのことについて考えました。私は単なる孤立した馬鹿に出くわしたわけではなかったのです。これは重要で、多くの人々に影響を与える社会現象だったのでした。

私は幸いにも、ちょっと苦い味を味わっただけで済みました。他の人々は、始終この世界と折り合いをつけなければなりませんでした。ですから、私は長い間考えました。彼は、私たちMITの仲間への協力を拒むと約束しました。彼は私たちを裏切ったのです。しかし、彼は私たちにだけそうしたわけではありません。あなたにもしたかもしれません[聴衆の1人を指して]。また、彼はきっとあなた

にもそうしたと思います［聴衆の別の1人を指して］［聴衆の笑い］。そしておそらく、あなたにもそうしていたかもしれない［第3の聴衆を指して］。彼はおそらく、この部屋にいるほとんどの人々に対してそうしたはずなのです。1980年にまだ生まれていなかったごく一部の人々を除いて。彼は、地球上のほぼすべての人々との協力を拒むように約束したのです。彼は、NDA（情報非開示契約）に署名したのです。

これは、私にとってNDAとの初めての遭遇でしたが、重要な教訓を残しました。なぜ重要かというと、ほとんどのプログラマが未だにこの教訓を学んでいないからです。これが私にとってNDAとの最初の遭遇で、犠牲者は私でした。私と研究所全体が犠牲者になりました。そして、この事件から学んだ教訓とは、NDAには犠牲者がいるということです。NDAは無罪でも無害でもありません。ほとんどのプログラマは、署名を促されたときに初めてNDAと遭遇します。そして、そのようなときには必ず何らかの誘惑があります。署名すれば、何か見返りが得られるのです。そこで、言い訳をでっち上げます。彼らは言います。「どうせ彼はコピーを手に入れられないのだから、彼を欺く陰謀に加わってはいけないという理由はない」　あるいはこう言います。「みんなそうしているのに、自分だけ逆らえないよ」　あるいはこういいます。「私がサインしなくても、誰かがサインするさ」　さまざまな言い訳が、彼らの良心を抑えつけてしまうのです。

しかし、私自身がNDAにサインしないかと誘われたときには、私の良心はすでにこの問題に敏感になっていました。私を助けないように、ラボ全体の問題解決を妨害するような約束をした人間に対したときに、いかに腹が立ったかが蘇ってきました。ですから、私は180度方向転換して、私に何の害も与えていない他者にまったく同じことをするわけにはいきませんでした。憎んでいる敵と役に立つ情報を共有しないように約束してくれと言われたなら、イエスと言ったでしょう。悪いことをしている相手なら、共有を得られないだけの理由があります。しかし、見知らぬ他人は、私に対して何もしていません。どうしてそのような虐待を受けなければならないのでしょうか。自分がすべての人々に対して、ひどい仕打ちをすることを許してはなりません。そんなことをすれば、社会に対する略奪者になってしまいます。そこで私はこう言いました。「このすばらしいソフトウェアパッケージを私に提供しようとしてくれてどうもありがとう。しかし、あなた

が要求している条件では、私は自分の良心にかけてそのパッケージを受け入れるわけにはいかないので、そのパッケージはいりません。どうもありがとう」　というわけで、私はソフトウェアなどの有用な技術情報全般について、自ら承知した上でNDAにサインしたことはありません。

情報には、その他の種類のものもあります。それらはまた別の倫理的な問題を抱えています。たとえば、個人情報です。皆さんがボーイフレンドと自分の間に起きたことについて、私に話をしたいのだけれども、誰にも言わないでね、と頼むようなことがあります。それは一般的に役に立つ技術情報ではないので、私は皆さんのために秘密を守ることを約束するでしょう。

その情報は、少なくとも、おそらく一般的に役に立つものではありません［聴衆の笑い］。ごく小さな可能性、本当に可能性の問題ですが、皆さんが私にあるすばらしい新しいセックステクニックを教えてくれるかもしれません［聴衆の笑い］。そして、私は人類のその他の人々に対してその情報を伝えれば、皆が利益を得られるという倫理的義務感［聴衆の笑い］を感じるかもしれません。そのような場合には、内緒にしてという約束に条件を付けなければならないでしょう。

誰がそれをほしがっていて、誰が誰に腹を立てていて、といった連続テレビドラマ的な些細な話なら、私は秘密を守ることができます。しかし、知ることによって人類がとてつもなく大きな利益を得られる情報なら、私は出し惜しみをするわけにはいきません。科学技術の目的は、人々がよりよく生きられるようにするために役に立つ情報を開発することです。そのような情報を隠す約束をするなら、あるいは情報を秘密にするなら、私たちは自らの使命に反することになります。そして、私はそのようなことはすまいと決心したのです。

しかし、その一方で私のコミュニティは崩壊し、私はまずい状況の中に取り残されました。PDP-10が陳腐化してしまったので、ITS全体が陳腐化してしまいました。ですから、以前のようにオペレーティングシステム開発者として仕事を続けることは不可能になっていました。私の仕事はコミュニティの一員となって、コミュニティのソフトウェアを使い、改良することに依存していました。そのような可能性はもう閉ざされてしまったので、私は倫理的にジレンマに陥ってしまいました。自分はこれからどうしたらよいのか。もっとも簡単な可能性は、私の決心を覆すことでした。それは、世界の変化に自分を適応させるということです。

世界が変わったことを受け入れ、自分の原則を放棄し、私有オペレーティングシステムのNDAに署名し、おそらくは私自らも私有ソフトウェアを開発するということです。そうすれば、楽しくコーディングでき、豊かになれることはわかっていました。特に、MIT以外のところで働けば。しかし、あとで人生を振り返ったときに、「私は、人々を分断する壁を立てるために人生を費やしてしまった」とつぶやきつつ、自分の人生を恥じなければならなくなってしまうでしょう。

そこで、私は別の選択肢を探し、簡単な方法が見つかりました。ソフトウェアの世界を離れて他の仕事をするのです。私には他に特別な技能はありませんでしたが、ウェイターにはなれただろうと思います［聴衆の笑い］。高級レストランには雇ってもらえないでしょうが［聴衆の笑い］、どこかでウェイターになることはできたでしょう。多くのプログラマは、私にこういうことを言います。「プログラマを雇う人間は、これとこれとこれを要求する。それができなければ、自分は飢え死にだ」　これは、彼らが使う用語そのままです。ウェイターなら、飢え死にすることはないでしょう［聴衆の笑い］。ですから、実際には彼らには何の危険もないのです。しかし、ここが重要なポイントですが、何かもっと悪いことが起きると言えば、他人を傷つけるような仕事をする言い訳になるのです。本当に飢え死にしそうなら、私有ソフトウェアを書く言い訳も立つでしょう［聴衆の笑い］。誰かが銃で狙っていると言うのなら、それも許されるでしょう［聴衆の笑い］。しかし、私は非倫理的なことをしなくても生き残れる方法を見つけましたから、それは言い訳にはなりません。もっとも、ウェイターになるのは私にとって面白いことではありませんし、オペレーティングシステム開発者としての技能を無駄にすることになります。しかし、自分の技能を悪用することは避けられたでしょう。私有ソフトウェアの開発は、自分の技能の悪用です。私有ソフトウェアの世界で生きることを他者に勧めるのも、自分の技能の悪用です。悪用するくらいなら無駄にしたほうがましですが、とても良いわけではありません。

そこで、私はまた別の選択肢を探すことにしました。状況を打開して世界をより良い場所にするためにオペレーティングシステム開発者ができることは何か。そして、必要なものは、まさにオペレーティングシステム開発者だということに気づきました。私にとっても誰にとっても問題でありジレンマであったことは、新しいコンピュータのオペレーティングシステムがすべて私有ソフトウェアだっ

たことです。フリーオペレーティングシステムは、古い陳腐化したコンピュータのためのものでした。ですから、新しいコンピュータでは、新しいコンピュータを手に入れて使いたければ、私有オペレーティングシステムを使うことを余儀なくされたのです。では、一人のオペレーティングシステム開発者が別のオペレーティングシステムを書き、「すべての人よ、ここに来てこれを共有せよ。すべての人を歓迎する」と言ったらどうなるでしょうか。すべての人々がジレンマを克服し、新たな選択肢を手にすることができます。私は、問題を解決するために自分ができることがあることに気づいたのです。私は、それを実現できる技能をちょうど持っていました。そして、私の人生をかけてできることの中で、それは想像できる限りもっとも役に立つ仕事でした。そして、それは他の誰もがまだ解決しようとしたことのない問題でした。ただ存在し、悪化し、私以外に解決しようとする者のいない問題でした。ですから、私はこう感じました。「私は選ばれた者だ。私はこの仕事をしなければならない。私がしなければ誰がするというのだ」

そこで、私はフリーオペレーティングシステムを開発することにしました。でなければ、開発しようとして死ぬ……もちろん、歳を取ってからですが［聴衆の笑い］。

当然ながら、私はどのようなオペレーティングシステムを開発するかを決めなければなりませんでした。設計上の技術的な判断をいくつかしなければなりませんでした。私は、いくつかの理由から、Unix互換システムを作ることにしました。まず第1に、私は自分が本当に気に入っていたあるオペレーティングシステムが陳腐化するのを目の当たりにしましたが、それは特定のコンピュータ専用に書かれていたからです。また同じ目に遭うのはいやでした。ポータブルなシステムが必要だったのです。Unixはポータブルなシステムでした。Unixの設計に従ったことで、同じようにポータブルで仕事のできるシステムを作るチャンスを手にしました。さらに、細部でUnixに対する互換性を保つ必要がありました。それは、ユーザーが互換性のない変更を嫌うからです。私が自分の好きなようにシステムを設計していたら、きっとそういう仕事は楽しかったとは思いますが、互換性のないシステムを作っていたでしょう。細部が異なるものになっていたはずです。そのようなシステムを作ったら、ユーザーはきっとこう言っていたはずです。「ええ、確かにこれはすばらしいシステムですが、互換性がありませんね。必要な作

業を考えると、切り替えは不可能です。Unix の代わりにあなたのシステムを使うためにそこまでする余裕はありません。ですから、私たちは Unix を使い続けるつもりです」

人々がいて、フリーシステムを使い、自由と協力のすばらしさを満喫するコミュニティがほしければ、人々が使えるシステム、人々が簡単に移行できるシステム、最初の時点で引っかかるような障害を持たないシステムを作らなければなりませんでした。そして、Unix に対して上位互換性を持つシステムを作ることにすると、さし当たっての設計上の判断はみな終わっていました。Unix はさまざまな部品から構成されており、それらは多少なりともドキュメント化されているインターフェイスを介して通信しているからです。ですから、Unix に対して互換性を保ちたければ、部品を 1 つずつ互換部品に取り替えていかなければなりません。他の設計上の判断は、1 つの部品内の問題であり、あとでその部品を書く誰かが決めればよいのです。最初から決めておく必要はありませんでした。

仕事を始めるためにその他にしなければならないことは、システムの名前を決めることだけでした。私たちハッカーは、プログラムのためにウケを取れる名前を探すものです。名前が受けている様子を想像することは、プログラムを書く楽しみの半分を占めます［聴衆の笑い］。そして、私たちの間には、既存のプログラムに似ているものを書いていることを示すために、入れ子細工のような再帰的頭字語を使う伝統があります。これこれはあれではない（this one's not the other）という文句を考えれば、再帰的頭字語になります。たとえば、60 年代から 70 年代にかけて、テキストエディタの Tico がたくさんありましたが、一般に何とか TECO と呼ばれていました。そこで、ある賢いハッカーが自分のエディタに Tint という名前を付けました。Tint Is Not TECO の頭文字を並べたわけです。これが最初の再帰的頭字語でした。私は 1975 年に最初の Emacs テキストエディタを開発しましたが、その後 Emacs を真似たエディタが無数に作られ、それらの多くは何とか Emacs と呼ばれていました。1 つは Fine Is Not Emacs で、また Sine Is Not Emacs、Eine Is Not Emacs と呼ばれるものがありました。そして、MINCE は、Mince Is Not Complete Emacs です［聴衆の笑い］。これは、機能縮小版でした。Eine はその後ほとんど完全に書き換えられ、新バージョンは Zwei Was Eine

Initially[*3]になりました［聴衆の笑い］。

そこで、私は何とかIs Not Unixという再帰的頭字語を探しました。26字全部を試してみましたが、単語になるものはありませんでした［聴衆の笑い］。うーむ。そこで他の方法を試しました。ちょっと縮めてみたのです。こうすれば、何とか's Not Unixという3文字名前を試すことができます。そして、「GNU」という単語に行き当たりました。GNUは、英語でももっとも面白い単語です［聴衆の笑い］。これだ。もちろん、面白いという理由は、辞書によれば「new」と発音されることにあります。この単語がさまざまな言葉遊びに使われるのはそのためです。ご存知かもしれませんが、これはアフリカに生息する動物の名前です。そして、アフリカでの発音には、クリック音が含まれていました［聴衆の笑い］。おそらく、今でも残っているでしょう。ヨーロッパの植民地主義者どもは、アフリカに着いたとき、このクリック音の発音を学習しませんでした。そこで、発音自体は無視し、「自分たちは発音するつもりはないが、ここには別の音があったはずである」ということを意味する「g」を付けました［聴衆の笑い］。そこで、今晩私は南アフリカに発ち、クリック音の発音方法を教えてくれる人を見つけたいと思っています［聴衆の笑い］。そうすれば、GNUが動物を指すときの正しい発音方法がわかるはずです。

しかし、システム名としてのGNUの正しい発音は、「guh-NEW」です。「g」をハードに発音します。「new」オペレーティングシステムを話題にすると、人に混乱を与えることになるでしょう。何しろ私たちはもう17年もこのシステムを開発しており、もうとても「new」システムだとは言えないからです［聴衆の笑い］。しかし、このシステムは今も、そしてこれからも永遠に、GNUです。たとえ、多くの人々が間違えてLinuxと呼んだとしても［聴衆の笑い］。

そこで、1984年の1月に私はMITを辞職してGNUの部品を書き始めました[*4]。しかし、ありがたいことに、MITは従来通り設備を使えるようにしてくれました。当時の私は、すべての部品を書き、GNUシステムを完成させたら、「こっちにおいで」と言って人々に使ってもらおうと思っていました。しかし、実際にはそ

*3　EineとZweiは、それぞれドイツ語で1、2を表す。
*4　GNUプロジェクトの最初の声明は「GNU宣言」に書かれている。

うはなりませんでした。私が最初に書いたのは、Unix の一部の部品の代わりに使える良い部品で、バグも少なくなっていましたが、とてもすばらしいというほどのものではありませんでした。わざわざそれらを手に入れてインストールしようという人はいませんでした。しかし、1984 年の 9 月になって、私は GNU Emacs を書き始めました。これは、私にとって 2 度目の Emacs の実装で、1985 年の初めまでには動くようになっていました。これで、すべての編集作業で Emacs を使えるようになりました。私は Unix の vi エディタの使い方を学ぶ気にはなれませんでしたので、これでずいぶん気が楽になりました [聴衆の笑い]。それまでは他のマシンでファイルを編集し、ネットワーク経由で保存して、テストしていましたが、GNU Emacs が使える程度に動くようになってからは、他の人々も使いたいと思うようになりました。

そこで、私は頒布方法の細部を詰めなければならなくなりました。もちろん、anonymous FTP ディレクトリにはコピーを置いておきましたし、ネットにアクセスできる人々にとってはそれで充分でした。tar ファイル[*5]をほどくだけですから。しかし、1985 年当時は、多くのプログラマでさえ、ネットへのアクセスを持っていませんでした。彼らは、「どうやったらコピーを手に入れられるの？」とメールしてきましたので、私は答え方を決めなければなりませんでした。「テープを書くのではなく、もっと多くの GNU ソフトウェアを書くために時間を使いたいので、Internet にアクセスできる友達を見つけて、ダウンロードしてもらい、それをテープに書き込んでもらってください」と答えても良かったでしょう。遅かれ早かれ、彼らはそのような友達を見つけられたはずです。彼らはコピーを手に入れられたでしょう。

しかし、私は無職でした。実際、1984 年 1 月に MIT を辞職してからずっと、私は無職でしたから、フリーソフトウェアの仕事を通じてお金を稼ぐ方法を探していました。そこで、私はフリーソフトウェアビジネスを始めたのです。私は、次のように発表しました。「150 ドル送っていただければ、Emacs のテープを郵送します」　すると、ぽつぽつと注文が入るようになってきました。1985 年の中頃には、ぽたぽたとしずくが垂れるような感じで注文が届くようになりました。

---

*5　Unix のアーカイブプログラム。gzip との組み合わせで、フリーではない ZIP 圧縮フォーマットに対する GNU 対抗版を形成する。

月に8本から10本の注文がありました。そして、必要なら、私はたったそれだけで生活していけます。私の生活費はいつも安く、基本的に学生のように暮らしています。このような生活は、何をすべきかがお金によって左右されないという点で気に入っています。やらなければならない重要に思うことをすることができるのです。私は、やる価値があると思うことをできる自由を手に入れました。普通のアメリカ人が慣れている消費過剰のライフスタイルに流されないようにすることは、非常に重要です。このライフスタイルに足をすくわれると、お金を持っている人々に生計を立てるための手段を支配されます。自分にとって本当に必要なことをできなくなってしまうのです。

まあ、それはいいでしょう。しかし、人々は私にこう尋ねました。「150ドルもするのにフリーソフトウェアというのはどういうことなんだね？」[聴衆の笑い]。彼らがこのようなことを尋ねてきたのは、英語の「フリー」という単語が持つ複数の意味に混乱したからです。フリーの1つの意味は価格、もう1つの意味は自由です。私がフリーソフトウェアと言うときには、価格のことではなく、自由のことを言っています。ですから、フリービール（無料ビール）ではなく、フリースピーチ（言論の自由）のほうをイメージしてください[聴衆の笑い]。プログラマの収入を確実に減らすために人生のうちのこれだけの時間を費やしたりはしなかったでしょう。それは、私の目標ではありません。私はプログラマで、自分自身の収入は気にしていません。収入を得るために全人生を捧げたりはしませんし、収入を得ることについてはどうこう思いません。誰にとっても倫理が同じであれば、他のプログラマが収入を得ることに反対したりはしません。プログラマの値段が下がることを望んでいるわけではないのです。問題にしているのは、そんなことではありません。問題は自由です。プログラマかどうかにかかわらず、ソフトウェアを使っているすべての人々にとっての自由が問題なのです。

ここで、フリーソフトウェアの定義を明らかにしておく必要があるでしょう。ただ、「自分は自由を信じている」と言ったところで、中身は空っぽです。現実のディテールに触れておくべきでしょう。信じることのできる自由にはさまざまなものがありますし、それらは互いに矛盾を生じます。ですから、リアルな政治課題は、重要な自由とはどれなのか、すべての人々に確実に与えなければならない自由は何かということです。

では、ソフトウェアを使うという特定の分野でのこの問題に対する私の答えを明らかにしておきましょう。特定のユーザーとしての皆さんが次の自由を持つとき、そのプログラムは皆さんにとって「フリーソフトウェア」です。

- **自由0：任意の目的のために好きなようにプログラムを実行できる自由。**
- **自由1：自分のニーズに合うようにプログラムを書き換えて自分のために役立てる自由。**
- **自由2：プログラムのコピーを頒布して隣人を助ける自由。**
- **自由3：改良版を公開し、自分の仕事を他者のために役立てて、コミュニティの構築を助ける自由。**

皆さんがこれらの自由をすべて持っていれば、そのプログラムは皆さんにとってフリーソフトウェアです。これは非常に重要なことです。私がこういう言い方をするのはそのためですが、詳しくはあとでGNU一般公開使用許諾書（GPL）について説明するときに触れます。今は、より基本的な問いであるフリーソフトウェアとは何かということについて説明しておきます。

自由0は、ごく当然のことでしょう。好きなようにプログラムを実行する自由さえ認められていないとすれば、それは恐ろしく制限のきついプログラムです。しかし、ほとんどのプログラムは、少なくとも自由0だけは与えています。そして、法的には自由0は自由1、2、3の当然の結果として付随するものです。著作権法は、そういう仕組みになっています。フリーソフトウェアが典型的なソフトウェアと異なるのは、自由1、2、3の部分ですから、それらを詳しく説明し、なぜ重要なのかを説明していきましょう。

自由1は、自分のニーズに合わせてソフトウェアを書き換え、自分のために役立てる自由です。これは、バグの修正を意味する場合もありますし、新機能の追加を意味する場合もあります。異なるコンピュータシステムへの移植を意味することもありますし、すべてのエラーメッセージをナバホ語に翻訳することを意味することもあります。加えたいと思うあらゆる変更は、自由に加えられなければなりません。

プロのプログラマがこの自由を非常に効果的に活用できるのは当然ですが、この自由を謳歌できるのは彼らだけではありません。当たり前の知性を持つ人なら、誰でもちょっとしたプログラミングを学ぶことができます。プログラミングには難しい仕事とやさしい仕事があり、ほとんどの人々は難しい仕事をこなせるところまでは学習しないでしょう。しかし、簡単な仕事ができる程度までなら多くの人が学習できます。50年ほど前は、非常に多くのアメリカ人が車の修理を学びましたし、アメリカが第二次世界大戦で自動車化された陸軍を持ち、勝つことができたのはそのためです。簡単な修理ができる人がたくさんいるということは、非常に重要なことです。

皆さんが人間らしい方で、テクノロジーなどまったく学びたくないとすれば、きっとたくさんの友達をお持ちでしょうし、自分に好意を持たせるのは得意技の1つでしょう［聴衆の笑い］。おそらく、その中の何人かはプログラマだと思います。だとすれば、そのプログラマの友達に、「こいつを書き換えてもらえないかな？　この機能を追加してくれるとうれしいんだけど」と頼むことができるでしょう。そうすれば、多くの人々の役に立ちます。

この自由がなければ、社会に現実的で物理的な害を与えます。ユーザーはソフトウェアに囚われてしまいます。それがどのようなものかは、レーザープリンタの例を使ってご説明した通りです。私たちにとっては満足のいかないものでありながら、ソフトウェアが私たちを囚われの身にしていたために、私たちはそれを直すことができませんでした。

しかし、これは人々の気持ちにも影響を与えます。コンピュータが使うたびにコンスタントにイライラするようなものなら、それを使っている人々もイライラを募らせていくことでしょう。そして、仕事にそのようなコンピュータを使っているのなら、仕事もイライラするものになります。人々は、仕事を嫌うようになるでしょう。そして人は、気にしないようにすることによってイライラから自分を守るものです。ですから、「今日は仕事に顔を出しました。私がしなければならないのは、それだけです。仕事が進まなくても、それは私の問題ではありません。上司の問題です」というような態度の人々が現れます。こうなったら、それらの人々にとって良くないことですし、社会全体にとっても良くないことです。これが自由1、自分自身を助けるという自由です。

自由2とは、プログラムのコピーを頒布して隣人を助ける自由です。思考、学習能力を持つ者にとって、役に立つ知識を共有することは友情の基本です。このような人たちがコンピュータを使うとき、友情はソフトウェアの共有という形を取ります。友達同士は、相手と共有し、相手と助け合うもので、これが友情というものの本質ではないでしょうか。実際、好意的精神、つまり自発的に隣人を助けるという精神は、社会のもっとも重要な資源です。生きられる社会と弱肉強食のジャングルを隔てるのは、まさにこれです。その重要性は数千年前から世界の主な宗教が認めてきたもので、これらは正面からこの考え方を育てようとしてきました。

私の幼稚園時代の先生たちも、この態度、共有の精神を実地訓練で私たちに教えようとしました。先生たちは、私たちが実際に共有すれば、私たちが学んだのだと判断しました。先生たちはこう言いました。「幼稚園にキャンディを持ってきたら、一人占めにすることはできませんよ。他の子と分け合わなければなりません」　社会は、この協力の精神を教えるように作られていました。なぜ、そうしなければならないのでしょうか。人間は、完全に協力的なものではないからです。人間にはこういう部分もあり、ああいう部分もあります。人間にはさまざまな性質があります。ですから、社会をより良いものにしたければ、共有の精神を育む仕事をしなければなりません。100%になることは決してないでしょうが、それは理解できることです。人は自分のことも心配しなければなりません。しかし、協力の精神を少し大きくできれば、私たち全員がよくなります。

今日の合衆国政府によれば、教師はまったく逆のことを教えるべきものとされています。「ジョニー、学校にソフトウェアを持ってきたのかい？　他の人にそれを分けてあげたらいけないよ。共有はいけない。共有するってことは、海賊だってことだよ」　彼らが「海賊」という言葉を使うとき、それはどういう意味なのでしょうか。彼らは、隣人を助けることは、船を攻撃するのと倫理的に同じだと言っているのです［聴衆の笑い］。

仏陀やキリストなら、どういったと思いますか。あなたのお気に入りの宗教指導者を考えてみてください。よくわかりませんが、マンソンなら違うことを言ったでしょう［聴衆の笑い］。L・ロン・ハバードならどういったでしょうか。しかし……。

質問者（Q）：［聴取不能］
RMS: もちろん、彼は死んでいます。しかし、彼らはそれを認めないでしょう。それで？
Q: 他の人たちも死んでいます［聴衆の笑い］［聴取不能］。チャールズ・マンソンも死んでいます［聴衆の笑い］。彼らは皆死んでいます。キリストも、仏陀も……。
RMS: はい、そうですね［聴衆の笑い］。ですから、それについては推測の域を出ないわけですが、L・ロン・ハバードは、その点について、他の人たちよりも悪いということはないでしょう［聴衆の笑い］。いずれにしても……［聴取不能］。
Q: L・ロンは、いつもフリーソフトウェアを使っていました。フリーソフトウェアが彼をザヌーから解放したのです［聴衆の笑い］。
RMS: いずれにしても、私がソフトウェアはフリーでなければならないと思うもっとも重要な理由はこれです。私たちには、社会のもっとも重要な資源を汚染させてやっていけるほどの余裕はありません。これは、きれいな空気やきれいな水のような物理的な資源ではありませんが、間違いありません。心理的社会的資源ではありますが、汚染してはならないことは同じであり、私たちの生命に非常に大きな違いを残します。私たちが取る行動は、他の人々の考えに影響を与えます。私たちが、「分かち合うな」と言って歩き、人々がそれに耳を傾けたら、社会に影響を与えたということになります。もちろん、悪いほうのです。以上が自由2、隣人を助ける自由です。

ああ、そういえば、言い忘れていました。この自由がなければ、社会の心理的社会的資源に害を与えるだけではなく、浪費という物理的現実的害悪も引き起こすのでした。プログラムが所有権者を持ち、その所有権者が、ソフトウェアを使うためにはすべてのユーザーが料金を支払わなければならないような条件を設定したら、「気にすることはない。そんな条件がなくても、自分はそうしていたよ」という人がいるかもしれません。しかし、それが浪費なのです。意図的に仕組まれた浪費です。そして、ソフトウェアの面白いところは、ユーザーが少ないからといって材料を減らさなければならないというわけではないことです。車を使う人が少なければ、車の製造台数を減らしますが、これは節約につながります。車の製造には、資源を割り当てたり、割り当てなかったりすることができます。ですから、車が価格を持つのは良いことだと言えるでしょう。価格があるので、本

当は不要な車を作るために無駄な資源が大量につぎ込まれることを防げます。しかし、新しく車を作るために資源がいらないのだとしたら、車を作らないことにしても何も良いことは起きません。もちろん、車のような物理的なものの場合には、新しいもの、新しいコピーを作るためには必ず資源が必要です。

ところが、ソフトウェアでは話が違ってきます。誰でもコピーを作れるのです。そして、コピーを作るのはごく簡単なことです。ごくわずかな電流を除き、資源はいりません。ですから、値段を吊り上げてソフトウェアを使いにくくしても、節約できる資源、より適切な分野に割り当てられる資源はありません。ソフトウェアには当てはまらない前提に基づいて構築された経済理論の結論の部分を説く人がよくいます。前提が当てはまる分野で有効な経済理論をソフトウェアの分野に持ち込み、結論の正しさを主張しようとするのです。彼らは結果だけを取り出し、ソフトウェアにも同じことが当てはまると言おうとするのですが、ソフトウェアに関する限り、この議論には根拠がありません。前提に誤りがあるのです。結論の正しさを判断する上で、そこに達する過程や依存している前提条件を検証することは非常に重要です。というわけで、これが自由2、隣人を助ける自由の説明です。

自由3は、ソフトウェアの改良版を公開してコミュニティの構築を助ける自由です。以前、私はよくこう言われました。「ソフトウェアがフリーなら、ソフトウェアのために必要な仕事をした人たちには、報酬が与えられないことになる。そうしたら、誰が働くと言うのだろうか」　もちろん、彼らはフリーの2つの意味に混乱しているわけで、彼らの理屈は誤解に基づいたものになっています。しかし、いずれにせよ、彼らはこの理屈を持ち出します。今は、彼らの理屈と経験から得られた事実を比較できます。そして、フリーソフトウェアの開発のために報酬を得ている数百人の人々を見つけることができますし、10万を越える人々が自発的に開発を行っています。さまざまな動機に基づいて、無数の人々がフリーソフトウェアの開発に携わっています。

私が初めてGNU Emacs、すなわち人々が本当に使いたがった初めてのGNUシステムの構成要素をリリースし、GNUがユーザーを持ち始めたという手応えを掴んだ頃、「ソースコードにバグを見つけました。バグフィックスをお送りします」というメッセージが届きました。「新機能を追加するコードをお送りします」と

いう別のメッセージも届きました。その後も、別のバグフィックス、別の新機能、別の何とか、さらに別の何とかが続き、ついにはこういったメッセージがあまりにも速いペースで届くために、すべてのメッセージを活用するのが私にとっては大仕事になってしまいました。Microsoftには、このような問題はありません［聴衆の笑い］。

そのうち、この現象は多くの人々に注目されることとなりました。1980年代には、フリーソフトウェアはあまりお金にならないから、非フリーソフトウェアほど優れたものにはならないだろうという考え方が主流でした。もちろん、自由とコミュニティを価値とする私のような人間は、「いずれにしても、私たちはフリーソフトウェアを使う」と言っていましたが。自由には、単純な技術的利便性を若干犠牲にするだけの価値があります。しかし、1990年頃には、私たちのソフトウェアのほうが実際に良いということが指摘され出しました。フリーソフトウェアは、私有ソフトウェアの類似品よりも強力で、信頼性が高かったのです。

1990年の初めに、ソフトウェアの信頼性を科学的に計測する方法を発見した人がいました。彼の方法はこういうものです。異なるシステムで同じ仕事、本当にまったく同じ仕事をするいくつかの類似するプログラムを取り出します。Unix風の基本的なユーティリティは確かにいくつもあります。そして、これらのプログラムがする仕事は、多少なりとも同じ、あるいはPOSIX準拠のものです。ですから、これらすべてのプログラムは、する仕事という点では同じです。しかし、これらのプログラムは、別個に書かれ、別々の人々によって管理されています。コードは違うのです。ですから、これらのプログラムを取り出し、ランダムなデータを与えて実行し、クラッシュやハングの頻度を調べれば、信頼性が測れるはずです。彼らは実際に計測を行い、もっとも信頼できるのはGNUプログラムだという結論を得ました。市販されている私有ソフトウェアの類似品は、どれもGNUより信頼性が低かったのです。彼はこの結果を公開し、すべての開発者に知らせました。数年後、彼はそのときの最新バージョンを使って同じ実験をしましたが、結果は同じでした。GNUバージョンの信頼性がもっとも高かったのです。癌専門の診療所や911[*6]の中にはGNUシステムを使っているところがありますが、そ

*6　アメリカの多くの地域では、911は緊急電話番号となっている。

れはGNUの信頼性が高いからです。言うまでもなく、これらの仕事では信頼性が非常に重視されます。

いずれにしろ、ユーザーにさまざまなことを認め、自由を与える最大の理由として、この利益に注目する人たちさえいます。もっとも、私のフリーソフトウェア運動についての話を聞いてこられた方はお気づきかと思いますが、私が言っているのは、物理的現実的利益だけではなく、倫理の問題や、どのような社会で暮らしたいか、よき社会を築くには何が必要かという問題もあります。両方が重要なのです。フリーソフトウェア運動とは、そういうものです。

別のグループの人々、**オープンソース運動**と呼ばれている人々は、実利だけを口にします。彼らは、これが正義の問題だとは思っていません。隣人と共有する自由やプログラムの動きを見て、気に入らないところに変更を加える自由が人々にとって大切だとは思いません。しかし、彼らは人々にそれを認めることは有益だと言います。ですから、企業に出かけて行ってはこう言うのです。「人々にこれをさせると、もっと金儲けができますよ」　ですから、2つの運動が人々をある程度似た方向に導くのは確かですが、その理由はまったく異なります。哲学が根本的に異なるのです。

あらゆる問題の中でもっとも深い部分にある倫理的な問題のところで、2つの運動は異なります。フリーソフトウェア運動の私たちは、こう言います。「あなたには、これらの自由を持つ資格があります。あなたがこれらのことをするのを他人が禁止するようなことは、あってはなりません」　オープンソース運動の彼らは、こう言います。「はい、そのつもりなら彼らはあなたを止めることができます。しかし、私たちはあなたがそれらのことをできるようにしておくようにしようと彼らが思うよう努力します」　オープンソース運動は確かに貢献してきました。かなりの数の企業に、かなりの量のソフトウェアをフリーソフトウェアとしてコミュニティにリリースするよう説得してきました。オープンソース運動は、私たちのコミュニティに大きな貢献を果たしており、実際のプロジェクトでは［彼らと］協力しています。しかし、哲学的には、大きなずれがあります。

残念ながら、オープンソース運動は企業の支援を受けていますが、私たちの仕事を取り上げたほとんどの記事は、オープンソースという表現を使っているため、多くの人々は無邪気に私たちもオープンソース運動の一部だと考えています。私

が両者を区別しているのはそのためです。私は、このコミュニティを作り上げ、フリーオペレーティングシステムを開発したフリーソフトウェア運動がまだここに存在することを皆さんに意識していただきたいと思っています。皆さんが他の人を無意識に誤解に導かないよう、皆さんには、ぜひこのことを知っていただきたいのです。

それは、皆さんがどの立場を取るかを考えるためでもあります。

皆さんがどちらの運動を支援するかは、皆さんが決めることです。フリーソフトウェア運動と私の考え方を支持してくださるかもしれませんし、オープンソース運動を支持するかもしれません。あるいは、どちらも支持しないかもしれません。皆さんの政治的な立場を決めるのは、皆さん自身です。

しかし、フリーソフトウェア運動に賛同していただけるなら――どの立場を取るかを決めることによって生き方が変わり、その中に発言に値する問題が含まれていることを理解していただけるなら――自分はフリーソフトウェア運動に賛同していると言っていただきたいと思います。そして、「フリーソフトウェア」という用語を使い、私たちが存在していることを人々に知らせることも、賛同の形態の1つに他なりません。

ですから、自由3は、現実的物理的にも心理的社会的にも非常に重要です。この自由がなければ、コミュニティによる開発が行われなくなり、強力で信頼性の高いソフトウェアが作られなくなるという現実的物理的な害悪が発生します。しかし、それだけではなく、科学的協力の精神、人類の知恵を前進させるために協力して働くという思想に悪影響を与えるという、心理的社会的害悪も発生します。科学の発展は、人々が協力して働けるかどうかに非常に大きく依存しています。しかし、最近は、科学者の小さなグループが、互いに相手の科学技術ギャングと戦争を起こしているかのような行動を取ることが目立ってきています。でも、彼らが知識を分かち合わなければ、彼ら全体が後退してしまうのです。

以上がフリーソフトウェアと典型的なソフトウェアを区別する3つの自由です。自由1は自分のニーズに合った変更を加えられる自助の自由、自由2はコピーの頒布によって隣人を助ける自由、自由3は変更を公開して他者も使えるようにすることによりコミュニティの構築を助ける自由です。皆さんがこれらの自由をすべて持つなら、そのプログラムは皆さんにとってフリーソフトウェアです。さて、

私がそのように特定のユーザーを使ってフリーソフトウェアを定義するのはなぜでしょうか。それはあなたにとってフリーソフトウェアか［聴衆の1人を指して］。それはあなたにとってフリーソフトウェアか［聴衆の別の1人を指して］。それはあなたにとってフリーソフトウェアか［第3の聴衆を指して］。わかりますか？

Q: 自由2と自由3の違いをもう少し説明していただけますか［以下聴取不能］。

RMS: はい、2つは確かに関連を持っています。再頒布する自由がなければ、変更版を配布する自由もありません。しかし、これらはまったく別の行為です。

自由2は、正確なコピーを作り、それを友達に渡して、友達がそれを使えるようにすることです。あるいは、正確なコピーを作り、それを無数の人々に販売し、それらの人々が使えるようにすることです。

自由3は、改良を加え、あるいは自分では改良だと思う変更を加え、誰か他の人があなたに同意することです。ですから、これらは違います。そうそう、それから1つ重要なポイントがありました。自由1と自由3は、ソースコードを操作できることと関連しています。バイナリのみのプログラムに変更を加えるのは、とてつもなく難しい［聴衆の笑い］。ソースがなければ、日付を表現するために4桁の数字を使うというような簡単な変更[*7]でも［聴衆の笑い］大変なのです。ですから、現実的でやむにやまれぬ理由から、ソースコードアクセスはフリーソフトウェアにとっての必要不可欠な前提条件です。

それで、フリーソフトウェアを「皆さんにとって」というような言葉を使って定義する理由ですが、どうでしょう。それは、同じプログラムがある人々にとってはフリーソフトウェアでありながら、別の人々にとっては非フリーソフトウェアになってしまうからです。それはずいぶん逆説的な話に聞こえるかもしれませんが、どのようなときにそうなるか、例を挙げておきましょう。非常に大きな例、おそらく今までででもっとも大きな例は、MITで開発され、フリーソフトウェアにするというライセンスのもとでリリースされたX Window Systemです。MITのライセンスでMITバージョンを入手した場合、自由1、2、3はすべてあります。

---

*7　古いプログラムの多くが年を2桁で格納していたために起きたY2K問題を指す。2桁しか使っていなかったので、00が2000を指すのか1900を指すのかあるいは下2桁が00の別の年を指すのかが曖昧になっていた。2000年が来る前、この問題を解決するために数千のコンピュータシステムで合わせて数百万ドルの費用がかかった。

それは、あなたにとってフリーソフトウェアです。しかし、コピーの入手者の中には、Unix システムを販売しているさまざまなコンピュータメーカーが含まれており、それらのメーカーは自分のシステムで動作するように X に必要な変更を加えました。おそらく、X の数十万行のうちの数千行ほどだろうと思います。メーカーは、変更後のシステムをコンパイルし、Unix システムにバイナリを追加して、Unix システムの残りの部分と同様の NDA のもとで販売しました。数百万の人々は、こうなったあとのコピーを手にしました。彼らは X Window System を持っていましたが、これらの自由をどれ1つとして持っていませんでした。このような X は、彼らにとってはフリーソフトウェアではなかったのです。

どこから見るかによって X がフリーソフトウェアかどうかが変わるのは確かに変なことです。開発者グループの中から見れば、「私にはこれらの自由はすべてある。X はフリーソフトウェアだ」ということになり、ユーザーから見れば、「ほとんどのユーザーはこれらの自由を持たない。これはフリーソフトウェアではない」ということになります。X を開発した人々は、これを問題だとは思いませんでした。彼らの目的は、人気を集めること、つまりはエゴでした。彼らは、プロとして大成功を収めたかったのです。彼らは、「やった。たくさんの人々が自分たちのソフトウェアを使っている」と感じたかった。そして確かにそうなりました。多くの人々が彼らのソフトウェアを使っていましたが、自由はありませんでした。

GNU プロジェクトの場合、GNU ソフトウェアに同じようなことが起きたらそれは失敗です。私たちの目的は、人気を集めることではなく、人々に自由を与え、協力を勧め、認めることだからです。人に協力を強制してはいけませんが、協力を求める人に対してそれを認め、その自由を与えることは絶対に必要です。数百万の人々が非フリーバージョンの GNU を実行しているなら、それは成功でもなんでもありません。目標からはかけ離れたところに道が逸れてしまったということになります。

そこで、私はこのようなことが起きないようにする方法を探しました。私が考え出した方法は、「コピーレフト」と呼ばれているものです。コピーライト（著作権）の一種のようでありながら反転しているからコピーレフトにしました［聴衆の笑い］。法的には、コピーレフトは著作権を基礎としています。私たちは既存の著作権法を使いながら、まったく異なる目標を実現しました。それはこういう仕

組みです。まず私たちは、「このプログラムは、著作権法の保護を受けています」と言います。もちろん、デフォルトでは、これはコピー、頒布、変更を禁止すると言う意味です。しかし、そのあとでこう言います。「あなたには、これのコピーを頒布する権利があります。あなたには、これを書き換える権利があります。あなたには、変更版や拡張版を頒布する権利があります。あなたは、それを自由に変更できます」

しかし、これには条件があります。そして、その条件こそ、私たちがこのような面倒なことをしている理由です。この条件がなければ意味がないというほどのものです。その条件とは、このプログラムの一部を含む何かを頒布するときには、プログラム全体をまったく同じ条件で、条件を足したり引いたりしないで頒布しなければならないというものです。ですから、皆さんはプログラムを書き換え、書き換えたあとのバージョンを頒布できますが、そのときには、皆さんからプログラムを手渡された人々も、皆さんが私たちからもらったのと同じ自由を手にすることができなければならないということになります。しかも、それは皆さんが私たちのプログラムからコピーした部分だけではなく、皆さんがプログラムに付け加えたすべての部分に適用されます。彼らにとって、プログラム全体がフリーソフトウェアでなければなりません。

これで、プログラムを変更、再頒布する自由は、不可侵の権利になります。これは、独立宣言から借りてきた概念です。私たちが確保した権利は、絶対取り上げることのできないものです。コピーレフトのこのアイディアを具体化したライセンスがGNU一般公開使用許諾書（GPL）です。GPLは、私たちのコミュニティを寄生虫のように侵食しようとする人々にノーを突きつける強さを持った厳しいライセンスです。

自由の理想を尊重しない人々は無数にいます。彼らは、非フリープログラムをばら撒くための出発点として私たちの仕事を大喜びで取り込み、人々に自由を放棄させようとします。このようなことを許していたら、フリープログラムを開発しつつ、自分たちのプログラムの改良版と絶えず競争しなければならなくなってしまいます。これは面白くないことです。

「コミュニティに貢献するためなら自分の時間を差し出す気にもなるが、あの会社の私有プログラムを改良するために自分の時間を差し出す理由がどこにある

だろうか」と思う人は、たくさんいると思います。これを悪いとは思わないものの、そうするなら報酬がほしいと思う人もいるかもしれません。私自身は、そんなことなら何もしないでしょう。

しかし、どちらのタイプの人々も、つまり、私のように、「非フリープログラムが私たちのコミュニティに足がかりを持つのを手助けするのはいやだ」と思う人にも、「企業のために働くのはいいけど、もっといい報酬をくれなくちゃね」と思う人にも、どちらの人にでも、GNU GPL は役に立ちます。GPL は企業に対して、「私の仕事を取り込み、自由を抜き取って頒布することはできない」と規制しています。それに対し、コピーレフトではない X Window ライセンスのようなライセンスは、それを認めています。

ですから、ライセンスという観点から見ると、フリーソフトウェアは大きく２種類に分類できます。コピーレフトのようにすべてのユーザーに対してソフトウェアの自由を守るライセンスを持っているプログラムと、非フリーバージョンを認めるコピーレフトの保護のないプログラムです。後者のようなプログラムは、誰かが自由を取り除いてしまうことができます。そのようなプログラムは、非フリーバージョンという形で皆さんの手元にやってくるかもしれません。

そして、現在もこの問題は存在します。今でも、私たちのフリーオペレーティングシステムで使われている非フリーの X Window があります。非フリーバージョンの X Window 以外サポートしていないハードウェアさえあります。これは、私たちのコミュニティの大きな問題の１つです。にもかかわらず、私は X Window を悪いものだと言うつもりはありません。X の開発者は、できるはずの最良のことをしなかっただけです。しかし、私たちみんなが使える多くのソフトウェアを彼らがリリースしたことは間違いありません。

完璧ではないことと悪いことには大きな違いがあります。善と悪の間には、さまざまな色合いがあります。私たちは、最良のことをしなかったからお前は良くないと言いたくなる気持ちを断ち切らなければなりません。X Window の開発者は、私たちのコミュニティに大きな貢献をしたのです。しかし、もう少し良いことができただろうとは言えます。プログラムの部品をコピーレフトで守っていれば、彼ら以外の開発者が非フリーバージョンを頒布することを防げたでしょう。

さて、Microsoft が現在 GPL を攻撃しているのは、もちろん、GNU GPL が著

作権法を使ってユーザーの自由を守っているからです。Microsoft は、私たちが書いたすべてのコードを取り込み、それを私有プログラムにして、誰かに何らかの改良を加えさせたいと思っています。あるいは、Microsoft が必要としているのは、互換性のない変更だけなのかもしれません［聴衆の笑い］。

Microsoft の営業力があれば、私たちのバージョンを駆逐するものは、私たちのバージョンよりも優れている必要はありません。単に私たちのものとは異なる互換性のないものにすればよいのです。あとは、すべての人のデスクトップにそれを置くだけです。ですから、Microsoft は、本当は GNU GPL を嫌っているわけではありません。GNU GPL があると、それができないから邪魔なのです。GNU GPL は、「取り込んで拡張」方式を認めません。GNU GPL は、「あなたのプログラムに私たちのコードを使いたければ、使ってかまわないが、あなたも同じように共有を認めなければならない」と要求しています。皆さんが加えた変更は、誰もが共有できなければなりません。つまり、双方向の協力です。本当の協力は、そういうものです。

多くの企業は、IBM や HP のような大企業も含めて、この基礎を認めた上で私たちのソフトウェアを喜んで使おうとしています。IBM と HP は、GNU ソフトウェアの改良に大きく貢献しています。そして、両社は新しいフリーソフトウェアの開発も行っています。しかし、Microsoft はそうしたくないので、企業は GPL を受け入れられないと放言しています。もし、その企業に IBM や HP や Sun が含まれないのなら、Microsoft の言い分も正しいのかもしれません［聴衆の笑い声］。これについてはまたあとで触れることにしましょう。

そろそろ歴史の話を締めくくらなければなりません。私たちは、ただフリーソフトウェアを書くためにではなく、もっと包括的な仕事、つまり全体がフリーソフトウェアになっているオペレーティングシステムの開発をするために、1984 年に立ち上がりました。そのため、私たちは次々に部品を書かなければなりませんでした。もちろん、私たちはいつも近道を探すようにしていました。仕事は非常に大きいものだったので、私たちの仕事は永遠に終わらないと言われていました。私は、少なくとも完成する見込みはあると思っていましたが、近道を探す価値は間違いなくありました。ですから、私たちは周囲を観察することを怠りませんでした。私たちが採用し、取り込めるようなものを誰かが書いていないだろうか。

そうすれば、その部品についてはゼロから書かなくても済みます。たとえば、X Window System です。X は確かにコピーレフトの対象にはなっていませんでしたが、フリーソフトウェアだったので、私たちは X を使うことができました。

私は、当初から GNU にウィンドウシステムを組み込みたいと思っていました。私は GNU を発足させる前に、MIT で2つのウィンドウシステムを書いていました。そして、1984 年の段階では、Unix はウィンドウシステムを持っていませんでしたが、私は GNU にはウィンドウシステムを搭載しようと思っていました。しかし、私たちが GNU ウィンドウシステムを書くことはありませんでした。それは、X が登場したからです。私は言いました。「やった！　大きな仕事を1つやらずに済ませることができる。X を使おう」　X を採用して、GNU システムに組み込もうと言ったのです。そして、X と共存できるように GNU のほかの部分を適宜改造しました。また、私たちはテキストフォーマッタの TEX やバークレイのライブラリコードのように、他の人が書いたフリーソフトウェアを見つけました。当時、バークレイ Unix はありましたが、それはフリーソフトウェアではありませんでした。このライブラリコードは、もともと浮動小数点数の研究をしていたバークレイのほかのグループが作ったものでした。私たちはこれらの部品に合わせてシステムを改造しました。

1985 年の 10 月に、私たちはフリーソフトウェア財団（FSF）を設立しました。ですから、GNU プロジェクトのほうが先だったのです。このことに注意してください。FSF は、GNU プロジェクトが発表されてから約2年後に設立されたのです。FSF は、ソフトウェアの共有と改造の自由を促進する資金を調達するための非課税組織です。そして 1980 年代に私たちが集めた資金で行った重要な仕事の1つは、GNU の部品を書くプログラマを雇うことでした。そして、シェルや C ライブラリなどの重要なプログラムはこのような形で作られました。他の部分も同様です。絶対的に重要な tar プログラムは、まあ全然面白いものではありませんが［聴衆の笑い］、こうやって書かれました。GNU grep もそうだったと思います。私たちは、このようにして目標に近づいていきました。

1991 年までの段階で、主要な部品で足りないものは1つだけで、それはカーネルでした。ではなぜ、カーネルを後回しにしたのでしょうか。どのような順番で仕事を進めていくかという問題とはあまり関係がなかったと思います。少なくと

も技術的な理由ではありません。いずれにせよ、システム全部を作らなければならないのです。理由の1つは、どこかにカーネルの出発点になるものがないかと思っていたことにあります。そして、そういうものはありました。カーネギーメロン大学で開発されたMachです。Machは、カーネル全体ではなく、下半分だけでした。そのため、ファイルシステムやネットワークコードといった上半分を書かなければなりませんでした。しかし、Machの上半分に当たるこれらのシステムは、基本的にユーザープログラムとして動作します。その分、これらのシステムはデバッグしやすくなるはずです。本物のソースレベルデバッガを同時実行してデバッグできるからです。こうやって仕事を進められれば、カーネルの上位レベルは短期間のうちに完成させられるだろうと思いました。しかし、実際にはそうはなりませんでした。これら、互いにメッセージを送り合う非同期マルチスレッドプロセスは、非常にデバッグが難しいことがわかりました。そして、私たちがブートストラップのために使っていたMachベースのシステムは、デバッグ環境が最低で、信頼性がありませんでした。GNUカーネルが動作するようになるまでには、何年もの時間がかかりました。

しかし、幸いなことに、私たちのコミュニティはGNUカーネルを待たなくても済みました。1991年にリーナス・トーバルズがLinuxという別のフリーカーネルを開発したからです。彼は、古いモノリシック（一体型の）設計を使っていましたが、彼の仕事は私たちの仕事よりもはるかに速く進みました。ですから、この設計上の判断は、私のミスの1つです。いずれにしてもGNUプロジェクトのことを知りながら、私たちに連絡を取ってきませんでしたので、私たちはLinuxのことを知りませんでした。しかし、彼はネット上の他の人々と他の場にはLinuxを発表しました。そこで、他の人々がLinuxとGNUシステムの他の部分を結合して完全なフリーオペレーティングシステムを作る仕事をしてくれました。基本的に、GNUとLinuxの組み合わせを作ったわけです。

しかし、彼らは自分の仕事の意味を理解していませんでした。彼らは言いました。「私たちはカーネルを持っている。まわりを見てカーネルに組み合わせられるほかの部品を探そう」　そして、彼らはまわりを見回したところ、あらびっくり、必要なものはすべてすでにあることがわかりました。彼らは言いました。「なんて幸運なんだろう［聴衆の笑い］。それらは全部揃っている。私たちは必要なも

のをすべて見つけることができた。これらすべてのものを取り込み、1つにまとめれば、システムになる」

彼らは、見つけた部品の大半がGNUシステムだったことを知りませんでした。ですから、自分たちがしていることは、GNUシステムとLinuxの隙間を埋めていくことだということを理解していませんでした。彼らはLinuxを取り出し、Linuxからシステムを作れると思っていました。そこで、彼らはシステムをLinuxシステムと呼びました。[ある聴衆の発言]「しかし、X Window SystemとMachを見つけたのよりも、それは幸運だったのではないですか」　[RMS、答えつつ話を続ける]その通りです。違いは、XやMachを開発した人々には、完全なフリーオペレーティングシステムを作るという目標がなかったことです。そういう目標を持っていたのは、私たちだけでした。システムを実現したのは、私たちの膨大な作業でした。私たちは、他のプロジェクトよりもシステムの多くの部品を作ったのです。これは偶然ではありません。これらの人々、彼らが書いたのは、システムの役に立つ部品でしたが、彼らがそれらを書いたのは、システムを完成させるためではありませんでした。彼らには彼らの理由があったのです。

Xを開発した人々の場合、ネットワークをまたがるウィンドウシステムを設計したら良いプロジェクトになるだろうと考え、実践したわけです。彼らの仕事は、私たちが優れたフリーオペレーティングシステムを作る上で大いに役に立ちましたが、それは彼らが望んだことではありません。彼らは、そういうことについて考えもしなかったのです。偶然の事故のような利益です。しかし、彼らがしたことが悪いと言っているわけではありません。彼らは、大規模なフリーソフトウェアプロジェクトをやりました。これは、すばらしいことです。しかし、彼らには究極的なビジョンはありませんでした。ビジョンがあったのは、GNUプロジェクトです。

ですから、他の人が作らなかった小さな部品を隅々まで作ったのは、私たちです。私たちがそういう仕事をしたのは、それがなければ、完全なシステムにならないことがわかっていたからです。tarやmv[*8]のように本当に退屈でロマンのない仕事でも[聴衆の笑い]、私たちはしました。あるいはld。ldに特に面白いわく

---

*8　ファイルを移動したりファイル名を変えたりする簡単なプログラム。

わくするようなところはありませんが［聴衆の笑い］、私はldを書きました。そして、ディスクI/Oを最小限に押さえ、より高速で、より大きなプログラムを処理できるように考えて作りました。私はいい仕事をしたいと思っています。自分が書いているときには、プログラムのさまざまな部分を改良したいと思っています。しかし、私がそうしたのは、より良いldについて優れたアイディアがあったからではありません。そういう仕事をしたのは、フリーのldが必要だったからです。そして、私たちは、ほかの誰かがその仕事をしてくれるとは思っていませんでした。そこで、私たちは自分でそれをするか、それをしてくれる人を見つけなければなりませんでした。

現時点で、このシステムには数千の人々とさまざまなプロジェクトが貢献していますが、このシステムが存在するのはこのおかげだといえるプロジェクトは1つだけで、それはGNUプロジェクトです。それ［システム］は、基本的にGNUシステムであり、他のものをそれに追加したのです。

システムをLinuxと呼ぶ習慣は、GNUプロジェクトがしてきたことが通常は言及されないということですから、私たちにとって大きな打撃です。カーネルであるLinuxは、フリーソフトウェアの非常に役に立つ部品だと思いますし、Linuxについて言うべきことはいいことしかありません。いや、実際には、Linuxにも悪いことはいくつかあります［聴衆の笑い］。しかし、基本的に、Linuxについてはいいというつもりです。しかし、GNUシステムを「Linux」と呼ぶ習慣は、誤りです。ここにお集まりの皆さんには、システムをGNU/Linuxと呼ぶという小さな努力をしていただければと思います。そうすれば、私たちの活動が認められるきっかけになります。

［聴衆の中の1人が叫ぶ］「マスコットが必要なんじゃないですか！　ぬいぐるみの動物が」［RMSの応答］はい、ありますよ。［同じ聴衆の応答］本当に？［RMSの応答。大笑いを引き起こす］マスコットの動物、いますよ。ヌーです。そうそう、ペンギンを描いたら必ず隣にヌーも描いておいてください。しかし、質問は最後まで我慢してください。まだ、お話しなければならないことがあります。

なぜ、私はこのことにこだわっているのでしょうか。皆さんを煩わしい気分にさせ、皆さんの私に対する評価を下げるような［聴衆の笑い］クレジット問題のことなどを考えるのでしょうか。私がこのようなことをすれば、エゴを満足させ

たがっているからだと考えるほうがきっとおられることでしょう。もちろん、私は「Stallmanix」というように呼んでくれといっているわけではありません［聴衆の笑い］［拍手］。

私がこのシステムをGNUと呼んでほしいとお願いしているのは、GNUプロジェクトにクレジットを与えたいからです。そして、それには非常に具体的な理由があります。それは、誰かがクレジットをもらうこと自体よりもはるかに重要な意味があります。最近、私たちのコミュニティでまわりを見回せば、システムについて話し、システムについて書いているほとんどの人々は、GNUに言及せず、自由の目標、その政治的社会的理想についても言及しません。システムがGNUから生まれたものである以上、これらに触れないわけにはいかないのです。

Linuxの思想、哲学はまったく異なります。それは、基本的にリーナス・トーバルズのノンポリ思想です。ですから、システム全体をLinuxだと考えると、「すべてはリーナス・トーバルズから始まっているに違いない。彼の哲学は、私たちがしっかりと検討しなければならないものであるに違いない」というように考えがちです。そして、GNUの哲学を聞くと、「おいおい、これは理想主義が過ぎて、あまりにも非現実的だよ、俺はLinuxユーザーでGNUユーザーじゃないんだ」と言います［聴衆の笑い］。

何たる皮肉でしょう。彼らが知ってさえいれば。彼らが気に入っていて、あるいはそれを越えて熱狂しているシステムは、私たちの理想主義的な政治哲学が生み出したものだということを知ってさえいれば。

知っていたとしても、彼らが私たちと同じ考えになるとは限りません。しかし、少なくともGNUを真剣に捉え、充分に検討し、チャンスを与える気にはなるでしょう。彼らは、GNUと自分の生活の関係を理解するでしょう。「自分が使っているのはGNUシステムで、GNUには哲学がある。私がとても気に入っているこのシステムが存在するのは、この哲学のためだ」ということがわかれば、少なくともよりオープンな気持ちでGNUについて考えるようになるでしょう。だからといって、誰もが私たちと同じ考え方になるというわけではありません。人々にはそれぞれの考え方があります。それはかまわない。人々は、それぞれの意思を持つべきです。しかし、私はこの哲学が実現したことについて、この哲学にクレジットを与えておきたいのです。

私たちのコミュニティを見回せば、このシステムを Linux と呼ぶ習慣はほとんどすべての場所に浸透しています。大半の記者は、システムを Linux と読んでいます。正しくないのですが、彼らはそうするのです。システムをパッケージングする企業も、ほとんどがそう［Linux と］言っています。ほとんどの記者が、記事を書くときに、それが政治問題であるとか社会問題であるとは考えていないのです。彼らは一般に、純粋なビジネスの問題として、あるいはどの企業が成功するかという問題としてシステムを見ますが、そんなことは社会にとってごくごく小さな問題です。そして、人々が使えるように GNU/Linux システムをパッケージ化している企業も、大半がそれを Linux と呼んでいます。そして、彼らはパッケージにフリーではないソフトウェアを追加しているのです。

GNU GPL は、GPL の対象となっているプログラムからコードの全部または一部を取り出し、コードを追加してより大きいプログラムを作った場合には、そのプログラム全体を GPL のもとでリリースしなければならないと規定しています。しかし、同じディスク（ハードディスクであれ CD であれ）に別個のプログラムを書き込み、それらのプログラムがまったく別個のライセンスを持つことは可能です。それは単なる集成版であり、誰かに同時に 2 つのプログラムを頒布するだけのことであり、私たちがとやかく言うことではありません。しかし、本当はそうではないのですが、製品内で GPL の対象となっているプログラムを使っている場合には、製品全体がフリーソフトウェアでなければならないという規則にできればと思うことがあります。GPL は、そこまでの範囲をカバーするものではありません。対象は、プログラム全体です。少し離れた距離で通信し合う、たとえば互いにメッセージを送り合う 2 つの別個のプログラムがある場合、一般に両者は法的には別々の存在です。ですから、システムにフリーではないソフトウェアを追加している企業は、哲学的政治的に悪い考え方をユーザーに広めていることになります。これらの企業はユーザーに対して、「フリーではないソフトウェアを使うことに問題はないんだ。うちではボーナスとしてシステムに非フリーソフトウェアを追加しているよ」と言っているようなものです。

GNU/Linux システムの使い方を取り上げている雑誌を見ると、ほとんどのものは「Linux 何とか」という名前になっています。ですから、ほとんどの場合、彼らはシステムを Linux と呼んでいるのです。そして、これらの雑誌には、GNU/Linux

システム上で実行できる非フリーソフトウェアの広告が満載されています。これらの広告には、同じメッセージが含まれています。「非フリーソフトウェアは役に立ちます。とても優れているので、お金を払ってもほしくなるくらいです」[聴衆の笑い]。

そして、彼らはこういったものを「付加価値パッケージ」と呼びます。この言い方には、パッケージの価値についての考え方が表現されています。「自由ではなく、実利的な便益を評価せよ」と言っているのです。私はそのような価値を認めませんから、こういったパッケージを「自由除去パッケージ」と呼んでいます[聴衆の笑い]。フリーオペレーティングシステムをインストールした人は、自由な世界に生きているのです。私たちが皆さんのために何年もかけて築き上げてきた自由を満喫できるのです。しかし、これらのパッケージは、皆さんを鎖につなごうとしているのです。

そして、GNU/Linux システムのトレードショウを見るなら、どれも「Linux」ショウという名前になっています。そして、ショウは非フリーソフトウェアを展示するブースでいっぱいになっており、非フリーソフトウェアに承認の証印を付けようとしているようなものです。私たちのコミュニティは、見渡す限り、非フリーソフトウェアを奨励する習慣に満ち溢れており、GNU の開発目的である自由の思想を否定するものになっています。そして、人々が自由の思想に出会う場所は、GNU との接点、フリーソフトウェアとの接点、フリーソフトウェアという用語、それだけです。ですから、お願いするわけです。システムを GNU/Linux と呼んでください。システムがどこからなぜ生み出されたのかを人々に知らせるようにしてください。

もちろん、名前を使っただけで歴史を説明できるわけではありません。でも、4文字余分に入力すれば GNU/Linux と書けるわけですし、余分に発音しなければならないのは2音節だけです。でも、GNU/Linux の音節数は、Windows 2000 よりは少ないです [聴衆の笑い]。こう呼んだからといって、GNU について大したことを言っているわけではありませんが、彼らが GNU を知る準備をしているのです。彼らは GNU について、それが何なのかについて聞いたら、GNU と自分や自分の生活の関係を理解することでしょう。そしてそれは間接的ですが、非常に大きな差を生みます。ですから、私たちを助けてほしいのです。

Microsoft が GPL を「オープンソースライセンス」と呼んだことに注意してください。Microsoft は、人々が自由の問題を考えることを望まないのです。Microsoft は、人々が非常に狭く消費者として考えるように、あるいは、消費者としてもあまり合理的に考えず、Microsoft 製品を選ぶように、人々を誘導します。Microsoft は、人々が市民として、政治的意識を持った人として考えることを望みません。市民は彼らの敵です。少なくとも、Microsoft の現在のビジネスモデルの敵です。

では、フリーソフトウェアと私たちの社会の関係について話したいと思います。皆さんの中の一部の方々にとって関心のある副次的なトピックは、フリーソフトウェアがビジネスとどのような関係を持つかということでしょう。

フリーソフトウェアは、ビジネスにとって非常に役に立ちます。結局のところ、先進国のほとんどの企業は、ソフトウェアを使います。ソフトウェアを開発するのは、そのうちのごくわずかに過ぎません。

フリーソフトウェアがソフトウェアを使うすべての企業にとって非常に役に立つというのは、ソフトウェアをコントロールできるからです。基本的に、フリーソフトウェアとは、プログラムがすることをユーザーがコントロールできるソフトウェアです。個人としても、集合的にも、充分な注意を払っていれば、何らかの影響力を行使できます。注意が足りなければ、発言しても賛同が得られず、ほかの人の好みを受け入れることになります。しかし、注意していれば、何らかの影響力があります。それに対し、私有ソフトウェアでは、ユーザーには発言権はありません。

フリーソフトウェアなら、書き換えたいところを書き換えることができます。社内にプログラマが 1 人もいなくてもかまいません。そんなことは問題ではないのです。ビルの壁を動かしたい場合でも、自分が建設会社である必要はありません。大工さんを探してきて、「この仕事はいくらでやってもらえるだろうか」と尋ねるだけのことです。同様に、使っているソフトウェアに変更を加えるために、自分がプログラミング会社である必要はありません。プログラミング会社に出向いて、「この機能をいくらで実装してもらえるだろうか。仕事はいつまでに終わるだろうか」と尋ねるだけです。そして、その会社が仕事をしないというなら、別の会社を探すまでです

フリーソフトウェアのサポートには自由市場があります。ですから、サポート

が気になる企業は、フリーソフトウェアを選べば、大きな利益を手にすることになります。私有ソフトウェアの場合、サポートは独占体制です。ソースコードを持っているのは1社だけか、Microsoftの共有ソースプログラムであれば、巨額のお金を払った数社だけです。いずれにしても、ごくわずかです。サポートを提供できる企業はあまりありません。ですから、有力な顧客でなければ、あまり相手にしてもらえないということです。あなたの会社を相手にしないのだとすれば、皆さんの会社はMicrosoftにとってあまり重要ではないのでしょう。プログラムを使ってしまえば、異なるプログラムに切り替えるのは大仕事ですから、皆さんがサポートを受けるために身動きが取れなくなってしまうことをMicrosoftは知っています。そのため、バグレポートの特権のために料金を支払わなければならない羽目になるのです［聴衆の笑い］。そして、料金を支払うとこう言うわけです。「わかりました。あなたのバグレポートを記録しました。数か月後には、アップグレード版が発売されます。そのときにバグがフィックスされたかどうかはわかります」［聴衆の笑い］

フリーソフトウェアのサポートプロバイダは、そんなわけにはいきません。彼らは、お客さんを喜ばせなければならないのです。もちろん、優れたサポートの多くは無料で受けられます。Internetに問題をポストすれば、次の日には答えが返ってくるでしょう。しかし、これはもちろん保証できることではありません。確実を期するなら、サポート企業と契約して料金を支払ったほうがよいでしょう。もちろん、これはフリーソフトウェアビジネスの進め方の一例でもあります。

企業にとってのフリーソフトウェアのもう1つの長所は、セキュリティとプライバシーです。これは個人の場合でも変わりませんが、ビジネスの文脈で取り上げただけのことです。私有プログラムの場合、それらのプログラムが本当は何をしているのかさえわかりません。

プログラマは、ユーザーが知っていたら嫌がるような機能をわざと潜り込ませることができます。たとえば、開発者がマシンに潜り込めるような裏口を作ることができます。その裏口は、ユーザーがしていることを監視して情報を送り返すことができます。これは、特異なことではありません。Microsoftの一部のソフトウェアはこれをやっていましたが、それはMicrosoftに限ったことではありません。ユーザーを監視している私有プログラムはほかにもあります。そして、ユー

ザーはプログラムがそんなことをしていることを知ることさえできません。そして、プログラマが心底正直だったとしても、すべてのプログラマは誤りを犯します。誰の罪とも言えないようなバグがセキュリティに影響を与えることがあります。しかし、ポイントは、それがフリーソフトウェアではないので、バグを見つけられないということにあります。もちろん、修正することもできません。

自分が実行するすべてのプログラムのソースをチェックする時間があるような人はいません。しかし、フリーソフトウェアには大きなコミュニティがあり、そこにはチェックをする人が含まれています。プログラムに意図せぬバグがあったら(そういうものは確実にありますが)、彼らが早晩それを見つけてフィックスしてくれますので、皆がその恩恵を蒙ります。また、トロイの木馬とか監視機能を意図的に追加したらどうなるかを考えるので、それらが入り込む可能性はずっと下がります。私有ソフトウェアの開発者たちは、それらが見つからないことを知っています。彼らがそういうことをしても見つからないで逃げ通せるでしょう。しかし、フリーソフトウェアの開発者は、そのような機能を加えたらユーザーに見つかることを意識しなければなりません。私たちのコミュニティでは、ユーザーが気に入らない機能をユーザーに押し付けられるとは誰も思っていません。ユーザーが嫌う機能があれば、それを取り除いた改訂バージョンが作られることは見えています。改訂版ができたら、すべてのユーザーがそちらのバージョンを使うようになるでしょう。

実際、そんな機能が追加されないことは、ずっと前からわかり切ったことです。要するに、プログラマが書いているのはフリープログラムです。人々が自分のバージョンを気に入ってくれたらと思いながら書いているわけです。多くの人々が嫌うような機能を潜り込ませたら、自分のバージョンではない、別のバージョンが人気を集めてしまいます。フリーソフトウェアの世界の王様は、ユーザーだということです。私有ソフトウェアの世界では、お客さんは王様ではありません。ただのお客さんなので、自分が使っているソフトウェアに意見することはできないのです。

このような点で、フリーソフトウェアは民主主義を機能させるための新しいメカニズムだと言うことができます。今、スタンフォード大学に所属しているレッ

シグ教授*9は、コードが一種の法のように機能すると述べたことがあります。ほとんどすべての人がさまざまな意図と目的のために使うコードを書く人は、人々の生活を支配する法律を書いているのです。フリーソフトウェアの世界では、この法律が民主的に作られます。それは、民主主義の古典的な形態を取るわけではありません。つまり、大規模な選挙を行い、「この機能をどうすべきか皆で投票しました」と言うわけではありません［聴衆の笑い］。その代わりに、ある機能を自分の思うように実装したい人がその仕事をするのです。同じ機能を別の形で実装したい人はそうします。同じ機能がこのようにもあのようにも実装されるのです。そして、多くの人々がこの方向を望んだら、この方向が採用されます。このように、自分が望む方向に向かって一歩進むだけで、すべての人々が社会的な決定に貢献するのです。

皆さんは、個人として自分が好きなだけ歩みを進めることができます。企業は、自分にとって役立つ方向に自由に投資できます。そして、すべてのユーザーがそれぞれにとって必要な行動を取ったら、ソフトウェアが進むべき方法がそれによって決まります。

また、既存のプログラムから部品を取り出し（もちろん、通常はかなり大きな部品ですが）、自分である程度のコードを書き足して自分のニーズにぴったり合ったプログラムを作れるということは、非常に便利なことが多いです。プログラムをゼロから作らなければならないとしたら、既存のフリーソフトウェアパッケージから大きな部品を取り込むことができなければ、法外な費用がかかることでしょう。

ユーザーが王様であるということから生み出されるもう1つの効果は、互換性が高まり、標準化が進むということです。なぜかって？　ユーザーは、この2つが好きだからです。ユーザーは、意味もなく互換性のない機能を持つプログラムを拒否するものです。もっとも、ある種の非互換性を実際に必要とするユーザーのグループが存在するなら、その機能は実装されるでしょう。それはかまわないことです。しかし、ユーザーが標準に準拠した機能を求める場合には、私たちプログラマは標準に従いますし、そういうものだということを知っています。それに対し、私有ソフトウェアの開発者たちは、わざと標準に従わないことに利益を

*9　ローレンス・レッシグは本書の「序文」を書いている。

感じています。それは、彼らがユーザーに利益を与えているからではなく、ユーザーに非互換性を押し付け、ユーザーをその枠内に縛り付けられるからです。しかも、彼らはファイルフォーマットをひんぱんに取り替え、最新バージョンの購入をユーザーに強制するようなことさえします。

公文書管理官たち*10は最近困っています。コンピュータで 10 年前に書かれたファイルはアクセスできないことが多いのです。それらのファイルは、今はもうなくなってしまった私有ソフトウェアで書かれています。それらのファイルがフリーソフトウェアで書かれていれば、最新の形式に変換して実行できたかもしれません。そして、それらの記録が失われたり、アクセスできなくなったりすることはないでしょう。彼らは、解決策としてフリーソフトウェアに言及しつつ、National Public Radio*11でこのことについて不満を述べていました。実際、自分のデータの保存のために非フリープログラムを使うということは、自分で自分の首をしめるようなものです。

以上、フリーソフトウェアがほとんどの企業に与える影響について話してきましたが、ソフトウェアビジネスという特定の狭い分野にはどのような影響を与えるでしょうか。答えは、ほとんどない、というものです。私が聞いた限り、ソフトウェア産業の 90%はカスタムソフトウェアの開発に従事しています。つまり、リリースされないソフトウェアです。カスタムソフトウェアでは、このような問題、つまりフリーか私有かという倫理的な問題は発生しません。ユーザーが 1 人だけで、権利を持っているのもそのユーザーなら、問題はありません。そのユーザーは、自由に何でもできます。ですから、ある企業によって社内用に開発されたカスタムプログラムは、顧客企業がソースコードの保管とすべての権利の保持にこだわるセンスを持ってさえいれば、本質的にフリーソフトウェアです。

時計、電子レンジ、自動車のイグニッションスイッチなどに埋め込まれるソフトウェアにも、これらの問題はありません。これらの場所には、インストールすべきソフトウェアをダウンロードしたりはしないからです。これらは、ユーザーに関する限り、本物のコンピュータではありませんから、倫理的な重要性を帯びる

*10 公文書管理官は、Internet 経由で数千ものファイルを格納、共有している。
*11 National Public Radio は、毎日ニュース、音楽を放送するラジオ局をこの講演が行われた当時で 620 局も運営している私的な非営利組織。

ところまでこれらの問題が発生することはないのです。ですから、ソフトウェア産業の大半の部分は、今までと同じように続いていきます。しかも面白いことに、この産業のそれだけ多くの部分がこのような仕事に従事しているため、フリーソフトウェアビジネスに可能性がまったくないとしても、フリーソフトウェア開発者はカスタムソフトウェアを書けば日々の糧を得られるわけです［聴衆の笑い］。この仕事は非常にたくさんあります。割合はそれだけ大きいのです。

しかし、フリーソフトウェアビジネスは実際に存在します。あとで行うプレスコンファレンスでは、フリーソフトウェア企業2社の人々が加わります。そして、もちろん、フリーソフトウェアビジネスをしているわけではないものの、役に立つフリーソフトウェアを開発してリリースしている企業もありますし、それらの企業が作ったフリーソフトウェアは、非常に大量です。

では、フリーソフトウェア企業はどのように仕事をしているのでしょうか。一部の企業は、コピーを販売しています。コピーは自由にできるのですが、これらの企業は月に数千本のコピーを販売しています。他の企業は、サポートやさまざまなサービスを販売しています。私自身、80年代後半は、ソフトウェアサービスを売っていました。私は、「基本的に時給200ドルいただければ、私が書いたGNUソフトウェアにご希望の変更を加えましょう」と言っていました。確かにかなり高い値段ですが、私が作ったプログラムのことなのだから、他人よりもずっと短い時間で仕事を終わらせるだろうと考えたようです［聴衆の笑い］。私は、そのようにして生計を立てていました。実際、私はそれまで以上の収入を得ていました。私は、授業もしました。大きな賞[12]をもらった1990年までは、そういうことをしていましたが、それからはその必要はなくなりました。

しかし、1990年は最初のフリーソフトウェア企業であるCygnus Supportが設立された年でもあります。この会社の仕事は、基本的に、私がしていたのと同じようなことでした。私にその必要があれば、私がこの会社で働くことは可能だったでしょう。しかし、私にはその必要はありませんでしたし、特定の企業に縛られない立場でいたほうが運動にとって有益だろうと思いました。フリーな立場で

---

[12] 彼が言っている「大きな賞」は、「天才寄付金」とも呼ばれているマッカーサー奨学金で、例外的な業績を収め、創造的な仕事を継続、拡張していくことが見込まれる個人に与えられる5年間の寄付金である。

あれば、利害関係なしに、さまざまなフリーソフトウェア、非フリーソフトウェア企業について、褒めることも非難することもできます。そのほうが運動のためになるだろうと感じたのです。しかし、生計を立てるために必要だと思ったら、私は Cygnus で働いていたはずです。この会社は、所属しても良い倫理的な企業でした。この会社で仕事をしたからといって、恥じる理由はありませんでした。そして、Cygnus は初年度から利益を出しました。3 人の設立者が持っていただけの非常に小さな資本で設立されましたが、毎年成長し、利益を出しました。そこで、彼らは欲を出して外部の投資家を募り、会社をめちゃめちゃにしてしまいました。しかし、彼らが欲を出すまで、成功した期間は数年ありました。

この経緯は、フリーソフトウェアのある長所を良く示しています。フリーソフトウェアを開発するために資本を用意する必要はないということなのです。フリーソフトウェアは便利ですし、役に立ちます。ちょっとした資本があれば、人を雇っていくつかのソフトウェアを書いてもらうことができます。ごく少人数で大きな仕事ができるのです。実際、フリーソフトウェア開発の効率のよさは、フリーソフトウェアへの移行が重要だという理由の 1 つです。Microsoft が GNU GPL は悪いと言っていますが、ここからはその嘘も暴かれます。彼らが GPL に反対するのは、Microsoft が非フリーソフトウェアを開発するための資金を調達しにくくするからであり、私たちのフリーソフトウェアをつまみ食いして、私たちと共有するつもりのない自分たちのプログラムに私たちのコードを取り込むことを不可能にするからです。基本的に、私たちは彼らにそうして資本を調達させる必要はありません。いずれにしても、仕事を終わらせるだけで、現に終わらせつつあります。

私たちは、フリーオペレーティングシステムを完成させられないだろうと言われたものです。しかし、私たちはシステムを完成させましたし、それ以上のことをしています。それでも、世界が必要としている汎用公開ソフトウェア全体と比べれば、私たちが作ったものはまだ 1 桁足りません。世界と言っても、ユーザーの 90%以上がまだ私たちのフリーソフトウェアを使っておらず、OS が GNU/Linux で Web サーバが Apache という組み合わせが Web サーバの半分以上になっているだけの世界のことです。

Q: ［聴取不能］……さっき、なんとおっしゃいました？　Linux ですか。

RMS: GNU/Linux と言いました。

Q: 本当に？

RMS: はい、カーネルについて言うときには、Linuxと呼びます。それがカーネルの名前だからです。カーネルは、リーナス・トーバルズが書いたものであり、作者に敬意を表するために、作者が選んだ名前で呼ぶべきです。

一般に、企業のほとんどのユーザーはGNU/Linuxを使っていません。ほとんどのホームユーザーもまだ私たちのシステムを使っていません。彼らが使うようになれば、将来のフリーソフトウェアビジネスは自動的に10倍の数のボランティアと顧客を手にするはずです。そのときには、仕事の数もその桁になります。現時点では、私たちがその仕事をこなせる自信は充分にあります。

Microsoftが私たちを絶望させたがっているだけに、これは重要です。Microsoftは、このように言っているのです。「お前らが実行できるソフトウェアを手に入れる唯一の手段、技術革新の恩恵を蒙るための唯一の手段は、私たちに権利を譲ることだ。私たちに支配権を与えよ。お前たちが実行しているソフトウェアでできることを私たちに決めさせよ。そうすれば、私たちはお前たちから大量のお金を搾り取り、そのうちの一定の割合をソフトウェアの開発に当て、残りを利益として手に入れる」

皆さんはそこまで絶望的な考え方をしてはいけません。自由を放棄するところまで絶望してはならないのです。それは非常に危険です。

Microsoftが、いやMicrosoftに限らず、フリーソフトウェアを支援しない人々のもう1つの特徴は、短期的現実的な利益しか考えない価値観です。今年どれだけの売上を上げられるか。今日どれだけの仕事を終わらせられるか。短期的で狭い思考方法です。彼らは、自由のために犠牲を払う人間がいるなどと想像するのもばかばかしいという先入観を持っています。

昨日[*13]は多くの人々が、同胞の自由のために犠牲を払ったアメリカ人たちについて演説を行いました。彼らの一部は、非常に偉大な犠牲を払っています。彼らは、この国の誰もが知っているある種の自由のために自らの命さえ捧げたのです(少なくとも、一部の場合は。ベトナムでの戦争は無視しなければならないかもしれません)。

---

*13 講演前日は、全米の公休日である戦没将兵記念日だった。

しかし、幸いにして、ソフトウェアを使う上で私たちの自由を確保するために大きな犠牲を払う必要はありません。まだ、グラフィカルユーザーインターフェイスプログラムがなければ、コマンドラインインターフェイスを学ぶといったごく小さな犠牲で充分です。まだ、そのようなフリーソフトウェアパッケージはないので、そのような形で仕事をすることに満足してください。また、特定のフリーソフトウェアパッケージを開発しようとしている企業にお金を払ってあげるようにしてください。そうすれば、そのパッケージは数年で完成します。私たちは皆、さまざまな小さな犠牲を払うことができます。しかも、長期的に見れば、その犠牲から利益を引き出すことさえできるのです。ですから、実際には、犠牲というよりも投資です。投資から誰がいくらの利益を得るかということを細かく計算せずに、社会の改良に投資することが自分たちのためになるということを理解できる程度の長期的な視点を持てばよいのです。

これで、私の話はほとんど終わりです。

最後に、トニー・スタンコが提案しているフリーソフトウェアビジネスへの新しいアプローチ、「FreeDevelopers」について触れておきたいと思います。これは、最終的に参加したすべてのフリーソフトウェア作家に利益の中の一定の割合を支払うことを目指す経営構造体です。彼らは、フリーソフトウェアを基礎とすることによって膨大なコストを削減できるために、インド政府の大規模ソフトウェア開発契約を勝ち取る見通しとなっています。

では、質問にお答えすることにしましょう。

質疑応答

Q:　Microsoft のような会社がフリーソフトウェア契約を組み込むにはどうしたらよいでしょうか。

RMS:　確かに、Microsoft は事業の軸足をサービスのほうに傾けようと計画しています。そして、Microsoft がしようとしていることは汚く危険なことで、プログラムにサービス、サービスにプログラムというように両者を

ジグザグに結ぶことです。このサービスを使うためには、この Microsoft のプログラムを使わなければならず、そのためにはこのサービスを使わなければならなくて、さらにこの Microsoft プログラムが必要になる。このように結び付けられるわけです。Microsoft の計画はこのようなものです。

さて、面白いことに、このようなサービスを売ることは、フリーソフトウェアか非フリーソフトウェアかの倫理的な問題を引き起こしません。Microsoft がネット越しにサービス販売事業のための会社を持つことは、まったく問題のないことです。しかし、Microsoft がしようとしていることは、それらの会社を使ってソフトウェアとサービスを今まで以上に独占し、拘束しようとすることで、それは最近もある記事に書かれていました。ネットをマイクロソフトカンパニータウンに変えると表現している人たちもいます。

これは重要なことです。Microsoft に対する反トラスト法裁判では、Microsoft をオペレーティングシステム部とアプリケーション部に分割することが争われていますが、これは無意味で、効果のない分け方です。

しかし、その記事を読んで以来、Microsoft をサービス部とソフトウェア部に分割し、両者に一定の距離を強制する分割方法なら、効果的で意味があると思っています。サービス部はインターフェイスを公開し、すべての人がそのサービスとやり取りするクライアントを書けるようにしなければならなくするのです。おそらく、サービスを受けるために料金を支払わなければならなくなるでしょうが、それはかまいません。それはまったく異なる問題です。

Microsoft がこのようにサービスとソフトウェアに分割されたら、Microsoft サービスのライバルを蹴散らすために Microsoft ソフトウェアを使うことはできず、Microsoft ソフトウェアのライバルを蹴散らすために Microsoft サービスを使うことはできなくなります。すると、私たちはフリーソフトウェアを書けるようになりますし、皆さんは Microsoft

サービスにフリーソフトウェアの仕事を持ち込むかもしれませんが、それはかまわないことです。

結局のところ、Microsoft はもっとも多くの人々を服従させた私有ソフトウェアメーカーに過ぎず、他社はそれよりも少ない人々しか服従させられなかったのです。やる気が足りなかったわけではありません［聴衆の笑い］。単に多くの人々を服従させることに失敗しただけです。ですから、問題は Microsoft ではなく、Microsoft だけではありません。Microsoft は、私たちが解決しようとしている問題の最大例に過ぎません。その問題とは、協力して倫理的な社会を築こうとするユーザーの自由が私有ソフトウェアによって損なわれるということです。ですから、Microsoft は確かに私がこの演台に立つ機会を提供してくれましたが、私たちは Microsoft にそれほど気を取られているわけにはいきません。演台を用意したくらいでは、重視できないのです。Microsoft は究極の目的ではありません。

Q:　先ほど、オープンソースソフトウェアとフリーソフトウェアの哲学的な違いを話していただきましたが、Intel プラットフォームのサポートだけに関心が集中している最近の GNU/Linux ディストリビューションの傾向についてはどのように感じておられますか。そして、プログラムの正しい書き方に従い、どの環境でもコンパイルできるソフトウェアを書くプログラマがどんどん減っているように見えることについてはいかがでしょうか。単純に Intel システムだけで動くソフトウェアを作ることについてはどうですか。

RMS:　倫理的な問題はないと思います。実際、コンピュータメーカーは、自社製品に GNU/Linux システムを移植することがあります。最近では、HP がこれをやりました。そして、HP はコストのかかりすぎる Windows の移植にはわざわざお金を使いませんでした。しかし、GNU/Linux は、5 人のエンジニアが数か月仕事をしただけでサポートできました。GNU/Linux のサポートは、簡単にできる仕事だったのです。

もちろん私は、プログラムを移植しやすくする GNU パッケージ、autoconf

を使うことを皆さんにお勧めしています。autoconf はぜひ使ってください。あるいは、システムのそのバージョンではコンパイルできないバグフィックスを行った誰かが、その内容を送ってきたら、autoconf にかけてみてください。しかし、私はこれを倫理的な問題だとは見ていません。

Q:　コメントが２つあります。１つは、あなたが最近 MIT でなさった講演です。私は講演録を読んだだけですが、誰かが特許について質問したとき、あなたは「特許はまったく異なる問題です。それについてはコメントはありません」とおっしゃいました。

RMS:　はい。実際には、特許について言いたいことはたくさんありますが、1時間かかってしまいますから［聴衆の笑い］。

Q:　私が言いたいのはこうです。何か問題があるらしい。企業が特許と著作権を有形の財産のように呼ぶことには理由があるだろうということです。国家権力を使って独占への道を切り開きたいなら、そういうことをするでしょう。そして、これら２つの共通点は、同じ問題を扱っているということではなく、動機が同じだということ、つまり、一般に奉仕するためではなく、私的利益のために独占を得たいという動機があることなのではないでしょうか。

RMS:　企業が望んでいることについてのご指摘は正しいものです。しかし、企業が知的財産権という用語を使いたがる理由は、もう１つあります。それは、人々が著作権の問題と特許の問題をしっかり考えないように仕向けることです。著作権法と特許法はまったく異なりますので、ソフトウェア著作権とソフトウェア特許の効果はまったく異なります。

ソフトウェア特許はプログラマに対する規制であり、プログラマが特定の種類のプログラムを書くことを禁止しますが、著作権はそのようなことはしません。著作権法上は、少なくとも自分で書いたものは、頒布することが認められています。ですから、これらの問題を分けて考えることは非常に重要なのです。

これら２つは、非常に低い水準では、ほんの少しだけ共通点を持ってい

ますが、他のすべてについては違います。ですから、明晰な思考を心がけてください。著作権について論じるか、特許権について論じることはあっても、知的財産権について論じるようなことはしないでください。私には、知的財産権についての意見はありません。著作権についての見解、特許権についての意見、ソフトウェアについての意見はあります。

Q:　冒頭で、コンピュータプログラムはレシピと同様の機能的な言語だと言われました。しかし、料理のレシピからコンピュータプログラムまでの間には大きな距離がありますし、英語からコンピュータまでの間にも大きな距離があります。「機能言語」の定義は、非常に広いということです。DeCSS、DVD で問題になっているのは、このことではないでしょうか。

RMS:　それらの問題は部分的には似ていますし、部分的には異なります。物は、本質的に機能的ではありませんから。問題の一部は引き写して考えることができますが、全部そうすることはできません。残念ながら、これについても話し始めたら 1 時間かかってしまいます。しかし、すべての機能的な作品は、ソフトウェアと同じ意味でフリーでなければならないということは言っておきます。教科書、マニュアル、辞書、レシピなどです。

Q:　私は今オンラインミュージックについて考えていました。オンラインミュージックには、似ているところと違うところがあります。

RMS:　その通りです。あらゆる種類の公刊されている情報について保証すべき最小限の自由は、非営利の本文に一切の変更を加えない再頒布です。機能的な作品の場合には、変更済みバージョンの営利目的での公刊の自由が必要だと思います。それは、社会にとってとても役に立つことですから。それ以外の作品、娯楽的なもの、美的なもの、個人の思想を述べたものについては、おそらく変更を認めるべきではないでしょう。それは、営利的な頒布を対象とする著作権を認めてもかまわないということでもあります。

合衆国憲法によれば、著作権の目的は公共の福祉にあることを忘れないでください。目的は、特定の私的な主体の行動を変え、彼らがより多く

の本を公刊するように仕向けることにあります。それによる利益は、社会が問題を論じ学習することです。そして、社会には文学があり、科学論文があります。著作権の目的は、これらを奨励することです。著作権は著作者のためにあるわけではなく、まして出版社のためにあるわけではありません。著作権は読者のために、人が書き他者が読むときに発生するコミュニケーションから利益を得るすべての人たちのために存在するのです。そして、この目的については私も同意見です。

しかし、コンピュータネットワークの時代に入り、従来の方法は維持できなくなっています。従来の方法に従うなら、現在はすべての人のプライバシーに侵入し、すべての人を攻撃するような厳格な法律を必要とするようになってしまいます。隣人と共有したために何年も監獄暮らしをしなければなりません。印刷機の時代とは状況が変わってしまったのです。当時の著作権は、業界に対する規制でした。出版社の行動を制限していたのです。現在は、出版社が一般人に対して押し付けた制約になっています。ですから、たとえ同じ法律であっても、権力関係が 180 度ひっくり返ってしまったのです。

Q:　同じ物を作るのは認められるわけですね。では、他の音楽から新しい音楽を作るのは？

RMS:　それは面白い……。

Q:　そしてユニークで新しい作品です。そして、協力の精神にも満ち溢れています。

RMS:　そうですね。そして、私は何らかの形の公正利用の概念が必要になるだろうと思います。数秒のサンプルを作り、音楽作品の中でそれを利用するのは、明らかに公正利用とみなされるべきです。皆さんがこの問題について考えるなら、公正利用の標準的な考え方さえ、これを含むことになるでしょう。法廷が認めるかどうかはわかりませんが、認めるべきです。それは、今までの制度を変更することにはなりません。

Q:　私有フォーマットで公的な情報を公刊することについてはどう思われま

すか？

RMS:　それはいけません。政府は、どちらの方向であれ、市民が政府とやり取りするために、市民に対して非フリープログラムへのアクセスを要求すべきではありません。

Q:　私は GNU/Linux ユーザーです……。

RMS:　ありがとう［聴衆の笑い］。

Q:　4 年ほど前からですが。私にとってずっと大きな問題で、おそらく私たち全員にとって本質的なことでもあるのですが、それは、Web のブラウズです。

RMS:　はい。

Q:　GNU/Linux システムを使うときの決定的な弱点の 1 つは、Web ブラウズです。というのも、そのための主流のツールである Netscape は……。

RMS:　フリーソフトウェアではありませんね。

そのことについてお答えしましょう。もっと深いところに触れるために、その問題には触れておきたいと思います。おっしゃる通り、GNU/Linux システムで Netscape Navigator を使うといういやな傾向は以前からのものです。実際、市販パッケージになっているすべてのシステムに Netscape Navigator が同梱されています。これは皮肉なことで、私たちはフリーオペレーティングシステムを作るために一所懸命働いてきましたが、店に行くと、GNU/Linux のいくつかのバージョンが置いてあり、それらは Linux と呼ばれていて、フリーではありません。もちろん、一部がフリーではないということですが。いずれにしても、Netscape Navigator があり、他の非フリープログラムが含まれています。ですから、自分が何をしているのか良く把握していない限り、実際にフリーシステムを見つけるのは非常に難しいということになっています。何をしているのかがわかっていれば、Netscape Navigator をインストールすることはできないはずです。

さて、実際には、ずいぶん前からフリーのWebブラウザはありました。私がいつも使っていたLynxというフリーのWebブラウザがあります。これはグラフィカルではない、テキストのみのフリーWebブラウザです。テキストのみということには、広告を見なくて済むという大きな利点があります［聴衆の笑い］［拍手］。

しかし、いずれにしても、Mozillaというフリーのグラフィカルプロジェクトがあり、利用できるところまで成長してきています。私は、これをときどき使っています。

Q:　Konqueror 2.01は非常に優れていますが。

RMS:　そうでしたね。それもフリーのグラフィカルブラウザです。ですから、私たちはついにこの問題を解決できるところまできたということです。

Q:　フリーソフトウェアとオープンソースの間の哲学的倫理的な違いについて話していただけますか。両者は両立できないようなものだと感じていらっしゃいますか。

［録音テープ交換。質問の末尾と回答の先頭は録音できず］

RMS:　……自由、および倫理……あるいは、どちらの言葉を使うにしても、私たちにこれらのことを認めたほうが利益になると企業の方々に判断されるとよいと思っています。

しかし、お話ししたように、実際の仕事では、個人の政治的な立場が影響を与えることはまずありません。GNUプロジェクトを支援しようという人がいたとして、私たちが「あなたは私たちの政治的主張に同意しなければなりません」などと言うことは決してありません。私たちは、GNUパッケージの中ではシステムをGNU/Linuxと呼ばなければならないこと、それをフリーソフトウェアと呼ばなければならないことは言わせていただきます。私たちがGNUプロジェクトについて話をしているわけではないときにあなたがどういう表現を使われるかは、あなたの問題です。

Q:　IBM は政府部局向けに新しい大型マシンを売り込むために、セールスポイントとして Linux を使っていると言っています。Linux です。

RMS:　はい、もちろん、本当は GNU/Linux システムです［聴衆の笑い］。

Q:　その通りです。営業のトップに教えてやってください。彼は GNU のことを何も知りません。

RMS:　そうですね。問題は、IBM が自分の利益のためにどう言ったらよいかをすでによく考えた上で言っていることです。そして、より正確で、公平で、正しい言い方は何かという問題は、そのような企業にとって第 1 の問題にはならないのです。小さな企業でも、ボスはいます。そのボスが同じように歪んだ考え方をしていたら、彼も同じような決定を下すでしょう。巨大企業ではありませんが。これは恥ずべきことです。

IBM がしていることには、より重要で現実的な問題がもう 1 つあります。彼らは、「Linux」に 10 億ドルのお金をつぎ込んでいると言っています。しかし、「に」の部分にも引用符を付けるべきでしょう。と言うのも、フリーソフトウェアを開発している人々に払っているのは、その一部だからです。その部分は、間違いなく私たちのコミュニティに対する貢献です。しかし、他の部分は、私有ソフトウェアを書いたり、GNU/Linux に私有ソフトウェアを移植したりする人々に支払われています。その部分は、私たちのコミュニティに対する貢献ではありません。しかし、IBM は両方をひとまとめにして言っています。一部は宣伝であり、部分的には誤っていますが、部分的には貢献です。ですからこれはなかなか複雑です。IBM がしていることの一部は貢献であり、一部はそうではなく、一部はあいまいなところです。ですから、皆さんもそれを一緒くたにして「やった、IBM が 10 億ドル出したぞ」と考えるわけにはいきません［聴衆の笑い］。それは単純化のし過ぎというものです。

Q:　GNU 一般公開使用許諾書に盛り込まれている思想についてもう少し説明していただけませんか。

RMS:　GNU GPL の思想ですか。その一部は、X Window について今説明した

ような現象からコミュニティの自由を守りたかったということです。Xと同じことは他のフリープログラムにも起きました。実際、私がこの問題について考えたとき、X Windowはまだリリースされていませんでしたが、私は他のフリープログラムでこの問題が起きているのを見ていました。たとえば、TEXです。私は、ユーザーが確実にすべての自由を持てるように保証したいと思いました。そうしなければ、自分がプログラムを書き、多くの人々がそのプログラムを使っても、彼らは自由を持たないだろうということがわかっていたからです。そんなことになったら、何の意味があるでしょうか。

しかし、私が考えていたもう１つのことは、コミュニティに対して、自分は他人に踏みつけられるだけのドアマットではない、身体の中をうろうろしている寄生虫の餌食ではないという感覚を与えたいということでした。コピーレフトを使わなければ、こう言っていることになります。[屈従的な言い方で]「私のコードを持っていってください。好きなようにしてください。私はノーとは申しません」　すると、いろいろな人たちがやってきてこう言うでしょう。[非常に強い言い方で]「ああ、俺はこいつの非フリーバージョンを作りたかったんだ。もらっていくよ」
もちろん、彼らは何らかの改良を加えるでしょう。そして、それらの非フリーバージョンがユーザーにアピールし、フリーバージョンを駆逐していくことになります。そうだとすれば、皆さんはいったい何を達成したのでしょう。何らかの私有ソフトウェアバージョンに寄付しただけではありませんか。

そして、人々がこれを見ると、つまり、私が作ったものを他人が取り出して何も返さないところを見ると、意識の低下を引き起こすことがあります。これは、ただの理屈ではありません。私は実際にそうなったところを見ています。私が1970年代に所属していた古いコミュニティが消えたときに起きたこともそうでした。一部の人々が、非協力的になり始めたのです。そして、私たちは、彼らがそれによって利益を上げているのだろうと思いました。彼らは確かに利益を上げることを考えているか

のように行動していました。私たちは、そういう連中が協力の成果を持ち出して何も返さないことが可能だということに気づきました。私たちにできることは何もありませんでした。それは非常に力の抜けることでした。そのような風潮を好まない私たちは議論さえしましたが、流れを止めるアイディアは浮かんできませんでした。

GPL は、それを止めさせるために作られています。確かに、私たちのコミュニティに参加してコードをするのは歓迎します。コードを使ってあらゆる種類の仕事をすることができます。しかし、変更を加えたバージョンをリリースするときには、私たちのコミュニティの一員として、自由世界の一員として、私たちのコミュニティにそれを開放しなければなりません。

実際のところ、私たちの仕事の成果を受け取った上で何も返さない方法はいくつもあります。何もソフトウェアを書く必要はないわけですから。多くの人々が GNU/Linux を使いながら、ソフトウェアを一切書いていません。そうすることは義務付けられていないのです。しかし、ある種のことをした場合、その人は仕事の成果を寄付しなければなりません。私たちのコミュニティが言いなりにはならない、ただのドアマットではないというのは、そういうことです。そして、GPL は人々に強さを与える上で役立ったと思います。私たちは、皆の足に踏みつけられるだけの存在になるつもりはありません。

Q:　フリーでありながら、コピーレフトの対象ではないソフトウェアのことですが、誰もがそれを取り出して私有ソフトウェアに変えられるわけですから、誰かがそれを取り出して変更を加えて全体を GPL のもとでリリースするということはできないでしょうか。

RMS:　はい、可能です。

Q:　だとすると、その方法を使えば将来のすべてのコピーが GPL 化されないでしょうか。

RMS:　そのブランチからはそうなります。私たちは、一般にそういうことをし

ませんが、その理由を説明しておきます。私たちは、そうしたければすることができます。X Window を取り出し、GPL の対象となるコピーを作って、それに変更を加えることは不可能ではありません。しかし、X Window を改良しつつ、GPL 化しない私たちよりもずっと大きなグループの人々がいます。私たちが GPL バージョンを作ると、彼らから分かれていくことになりますが、それは彼らの扱い方としてあまり良いものではありません。彼らは私たちのコミュニティの一員ですし、私たちのコミュニティに貢献しているのですから。

第２に、彼らは私たちよりもずっと多くの仕事を X に対して行っているので、私たちのバージョンは彼らのバージョンよりも劣ったものになるでしょうし、人々は私たちのバージョンを使わないでしょう。GPL 化は私たちにとって裏目に出ます。そんな面倒を引き起こす必要があるでしょうか。

ですから、X Window に対する改良コードを書いたら、その人は X 開発チームと協力すべきだと私は言っています。彼らにそれを送り、彼らに扱いを任せるのです。彼らはフリーソフトウェアの非常に重要な部品を開発しているのですから、私たちは彼らと協力すべきです。

Q:　X ということで言えば、約２年前の X Consortium は、フリー、オープンソースから非常に遠いものでしたが……。

RMS:　はい、実際には、X はオープンソースではありませんでした。X Consortium はそうだったと言うかもしれません。また、私は X Consortium がそう言ったかどうか覚えていません。しかし、X はオープンソースではなく、制限を受けていました。確か、営利目的で頒布することができませんでした。あるいは、変更版を営利目的で頒布することができないとか、何かそういう制限がありました。フリーソフトウェア運動でもオープンソース運動でも認められないような制限がありました。

コピーレフト以外のライセンスを使ったときに残されるのはこういう可能性です。実際、X Consortium は非常に厳格な方針を取っていました。

彼らは、少しでもコピーレフト化されているプログラムには、X を頒布しないと言っていました。私たちは自分たちのディストリビューションには X を組み込もうとしませんでした。

このようにして、多くの人々がコピーレフトを避けるように圧力を受けてきたのです。そして、その後は X のすべてのソフトウェアが広範にオープンになりました。プログラマに過度に寛容になるよう圧力をかけてきたのと同じ人々が、あとになって「では、制限を設けることにしよう」と言う。それはあまり倫理的なことではありません。

しかし、だからと言って、私たちが資源をかき集めて GPL バージョンの対抗 X を本当に作るべきでしょうか。そんなことをする意味はありません。私たちがしなければならないことは、他にたくさんあります。対抗 X を作るくらいなら、そちらをしましょう。私たちは X の開発者たちと協力できます。

Q:　GNU は商標ですか。GNU GPL に商標を認める条項を組み込むのは現実的ですか。

RMS:　私たちは実際に GNU に商標登録を申請しています。しかし、だからといって GNU 商標で何かをするつもりはありません。その理由を説明するためには、非常に長い時間が必要です。

Q:　GPL の対象となるプログラムに商標を表示することを要求することもできますが。

RMS:　いいえ、私はそう思っていません。ライセンスは、個々のプログラムを対象とします。そして、あるプログラムが GNU プロジェクトの一部であれば、そのことについて嘘をつく人はいません。全体としてのシステムの名前は、また別の問題です。そして、それは余談というもので、これ以上論じる価値はありません。

Q:　すべての企業に対して強制的にソフトウェアをフリー化させられるボタンがあったとして、あなたはそれを押しますか。

RMS:　公刊されているソフトウェアだけを対象としてなら押すでしょうね。人には、プライベートにプログラムを書いて使う権利があると思っています。人と言った中には企業も含まれます。これはプライバシーの問題です。そのプログラムが人類にとって非常に役に立つのに、隠しておくなら、そういうことは良くないかもしれません。それは事実です。良くないことではありますが、それは良くないことの種類が異なります。同じ分野の問題ですが、異なる問題です。

しかし、私はすべての公刊されたソフトウェアがフリーソフトウェアであるべきだと思っています。そして、公刊されたソフトウェアがフリーソフトウェアでないとしたら、それは政府の介入の結果だということを覚えておいてください。政府がソフトウェアをフリーではないものにするために介入してくるのです。政府はプログラムの所有権者に特別な権力を法的に与えています。ですから、私たちが特定の形態でプログラムを使おうとすると、所有権者たちは警察力を使ってそれを制止することができます。私はこのような事態を終わらせたいと思っています。

エド・ションバーグ:リチャードの講演は、すばらしい量の知的エネルギーを生み出してくれました。そのうちの一部はフリーソフトウェアを使うこと、そして、おそらくフリーソフトウェアを書くことに導くことだろうと思います。

まもなく、講演を終了しなければなりません。リチャードは、一般人からは政治的に救いようもなくナード的だと見られている職業に、政治的で倫理的な議論を巻き起こしたということをぜひ言っておきたいと思います。私たちの職業ではこのようなことは今までありませんでした。このことについての彼の非常に大きな貢献に私たちは感謝したいと思います［聴衆の拍手］。

初出: 2001 年 9 月 29 日に NYU で開催された講演会の記録。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

第21章
避けたほうがよい用語

曖昧であるとか、鵜呑みにしてほしくない考え方を包んでいるため、使わないようお勧めしている語句がたくさんある。

BSD スタイル（BSD-style）

「BSD スタイルのライセンス（使用許諾書）」という表現は、重要な違いのある複数のライセンスをひとまとめにしているので、混乱を招く。たとえば、宣伝条項を含むオリジナルの BSD ライセンスは GPL との互換性を持たないが、改訂された BSD ライセンスは GPL と互換性を持つ。

混乱を避けるために、個別のライセンスを指定し、「BSD スタイル」という曖昧な用語を避けたほうがよい。

RAND（reasonable and non-discriminatory：合理的かつ非差別的な）

特許権の制約を受けた標準を推進し、フリーソフトウェアの可能性を封じる標準策定団体は、準拠プログラム 1 本ごとに固定料金を要求する特許実施権取得方針を取っている。彼らは、このようなライセンス（特許実施権）を「RAND」（合理的かつ非差別的な）と称することが多い。

この用語は、通常合理的でも非差別的でもない特許実施権の欺瞞をごまかしてしまう。このライセンスが特定の人物を差別しないのは事実だが、フリーソフトウェアコミュニティを差別しているのも事実であり、その分非合理的である。つまり、RAND の前半分は欺瞞的であり、後半分は偏見に満ちている。

標準策定団体は、これらのライセンスが差別的であることを認め、これらに言

及するときに「合理的かつ非差別的」とか「RAND」という用語を使わないようにすべきである。それまでは、このごまかしに加担したくない方は、この用語を拒否したほうがよい。特許権を持つ企業が撒き散らしたというだけの理由でこの用語を受け入れ、使ってしまえば、それらの企業に表現主体を委ねることになる。

代替語として、私は「統一料金のみ」あるいは「UFO（uniform fee only）」というものを提案する。この用語は、これらのライセンスに含まれる条件が統一的な特許使用料だけだということを正確に表現している。

オープン（open）*1

「フリーソフトウェア」の代わりに「オープン」という単語を使うのは避けていただきたい。私たちよりも理想主義的でない価値観を持つ別のグループは、スローガンとして「オープンソース」を掲げている。彼らに触れる場合には、彼らの名前を使うのは適当なことだが、私たちと彼らをひとまとめにしたり、私たちの仕事に彼らのラベルを貼ったりしないでいただきたい。そのようなことをすれば、人々は私たちが彼らの支援者であるかのように誤解する。

海賊行為（piracy）

出版社は、禁止されているコピーを「海賊行為」と表現することが多い。言外のうちに、違法コピーは、海上で船を攻撃し、乗員乗客を誘拐、殺害するのと倫理的に同じだと言いたいのである。違法コピーが誘拐や殺人と同じだと思わないなら、「海賊行為」という単語は使わないほうがよい。「禁止されているコピー」とか「無許可コピー」といった中立的な言葉を使うべきである。私たちの一部では、「隣人との情報の共有」のような肯定的な意味の言葉を選ぶ場合もある。

クローズド（closed）*1

非フリーソフトウェアを「クローズド」と呼ぶのは、明らかに「オープンソース」という用語を意識している。フリーソフトウェア運動は、最近のオープンソース運動と混同されるのを避けたいと考えているので、フリーソフトウェア運動と

---

*1　【訳注】http://www.gnu.org/philosophy/words-to-avoid.html より追加。

オープンソース運動をひとまとめにするのを奨励するような用語を避けている。そこで、非フリーソフトウェアを「クローズド」と呼ぶことも避けている。「非フリー」あるいは「私有」と呼ぶ。

コンテンツ、内容（content）

満足感、安心感を表現したい場合には、content と言っていただいてまったく問題はないが、書かれた仕事、その他作者を持つ仕事の意味でこの言葉を使うと、それらの仕事に対して特定の態度を示すことになる。つまり、箱に詰めて売ることを目的とする交換可能な商品とみなすことになる。実質的に、それは仕事に対して敬意を払わないことになる。

この用語を使う人々は、作品の著作者（彼らは「創造者（creator）」と呼ぶが）の名のもとに著作権の強化を図る出版社であることが多い。「コンテンツ」という用語は、彼らの実際の感覚を暴露する。

彼らが「コンテント（満足）プロバイダ（content provider）」という用語を使うなら、反対派は「不満プロバイダ（malcontent provider）」という言葉を使ってもよいはずだ。

商用、市販（commercial）

「商用」という用語を「非フリー」の同義語として使わないようにしていただきたい。これでは、2 つのまったく異なる問題を混同してしまうことになる。

ビジネスとして開発されたプログラムは商用である。商用プログラムは、ライセンス次第でフリープログラムにも非フリープログラムにもなる。同様に、学校や個人が開発したプログラムも、ライセンス次第でフリープログラムにも非フリープログラムにもなる。どのような主体がプログラムを開発したかという問題とユーザーがどのような自由を持つかという問題は、まったく無関係である。

フリーソフトウェア運動の最初の 10 年間は、フリーソフトウェアパッケージの大半が非商用だった。GNU/Linux オペレーティングシステムのコンポーネントは、フリーソフトウェア財団や大学のような非営利の組織か個人によって開発された。しかし、1990 年代に入って、フリーの商用ソフトウェアが現れるようになった。

フリーの商用ソフトウェアは私たちのコミュニティに対する貢献であり、私たちはこれを奨励しなければならない。しかし、「商用」が「非フリー」を意味すると考える人々は、この組み合わせを自己矛盾だと考え、可能性を潰してしまう。このような意味で「商用」という言葉を使わないように注意が必要である。

創造者、クリエータ（creator）

著作者を創造者と呼ぶと、暗黙のうちに著作者を神（「創造主」）に喩えることになる。この用語は、出版社が、著作者の精神を一般人よりも上に置き、著作者の名のもとに著作権の強化を正当化するために使うものである。

ソフトウェアの販売（sell software）

「ソフトウェアの販売」という用語は曖昧である。厳密に言って、フリープログラムのコピーと一定の金額を交換することは「販売」である。しかし、通常「販売」という用語には、その後のソフトウェアの利用形態に対する所有権者からの制約が付随する。言おうとしていることの意味に応じて、「有料でプログラムのコピーを頒布する」とか「プログラムの使用に所有権者からの制約を課す」と言い分ければ、正確になり、混乱を避けられる。

この問題については、「フリーソフトウェアの販売」を参照していただきたい。

ソフトウェアをあげる（give away software）

「フリーソフトウェアとして頒布する」という意味で「あげる」という用語を使うと誤解を招く。この言葉は、「ただで」と同じ問題を持っている。問題が自由ではなく、価格であるかのような意味を含んでしまう。たとえば、「フリーソフトウェアとしてリリースする」と言えば、混乱を避けられる。

知的財産権（intellectual property）

出版社や法律家は、好んで著作権を「知的財産権」と呼ぶ。この用語には、隠された前提条件がある。それは、物理的なものとそれを財産と考える私たちの考え方を基礎におけば、コピーの問題をもっとも自然な形で考えられるというものである。

しかし、このアナロジーは、物理的なものと情報の間の決定的な違いを見過ごしている。ものの複製を作るのは大変なことだが、情報はほとんど労力をかけずにコピー、共有できるのだ。このアナロジーを基礎として考えることは、この違いを無視することである。

アメリカの法制度でさえ、このアナロジーを全面的に受け入れてはおらず、著作権と物理的なものとしての財産権を同じようには扱っていない。

このような考え方に縛られたくなければ、自分の議論と思想から「知的財産権」という用語を取り除くことだ。

「知的財産権」という用語にはもう1つの問題がある。それは、著作権、特許権、商標権など、共通点をほとんど持たない独立した法制をひとまとめにしてしまうことである。これらの法律は別々の起源を持ち、別々の行為を対象とし、別々の運用形態を持ち、別々の政策問題を提起する。たとえば、著作権についてある事実を学んでも、特許法ではその通りにはならないと考えたほうがよい。たいていの場合はそのような食い違いが起きる。これらの法律はそれくらい異なるので、「知的財産権」という用語は過度の一般化を引き起こす。「知的財産権」についての見解は、ほとんど必ず馬鹿げたものになる。そのような広い水準では、著作権法に起因する政策問題も、特許権に起因する政策問題も、その他のものに起因する政策問題も論じることはできない。

「知的財産権」という用語は、これら別々の法律が持つごくわずかな共通点に人々の視野を狭める。つまり、売買できるさまざまな抽象を確立することにばかり力を入れ、それらが国民に課せられた制約であり、それらの制約がどのような善悪を引き起こすかを無視することになる。

特許権、著作権、商標権が引き起こす問題を明確に考えたいなら、さらにはこれらの法律が求めていることを学びたいなら、まず最初に以前耳に入ってきた「知的財産権」という用語を忘れ、それらを無関係なテーマとして扱うことである。明確な情報を提供し、明確な思考を奨励するつもりなら、「知的財産権」について言ったり書いたりしてはならない。代わりに、著作権、特許権など個々の法律の問題として話題を提出するのである。

テキサス大学ロースクールのマーク・レムリー教授によれば[*2]、「知的財産権」という用語が普及したのは、1967年に世界知的財産権機関（WIPO）が設立されてからであり、最近の流行に過ぎない。WIPOは、著作権、特許権、商標権保持者の利益を代表しており、彼らの権力を増強させるよう、政府に圧力をかけている。WIPOのある条約は、米国内で役に立つフリーソフトウェアパッケージの検閲に使われているデジタルミレニアム著作権法（DMCA）が引いた線をなぞるようにして作られている[*3]。

デジタル権管理（Digital Rights Management）

「デジタル権管理」ソフトウェアは、実際には、コンピュータユーザーに制限を課すために作られている。この用語における「権利」という単語の使用は、無意識のうちに制限を押し付けられる多数の視点を無視し、制限を押し付ける少数の視点から問題を見るように人を誘導する。

「デジタル制限管理」とか「拘束ウェア」と言ったほうがよい。

盗用（theft）

著作権の擁護者は、著作権侵害を表現するときに「盗まれた」とか「盗用」という単語を多用する。同時に、彼らは法制度を倫理的権威として扱うことを要求してくる。コピーが禁止されているなら、コピーは悪いことに違いないということになる。

そこで、法制度（少なくともアメリカ合衆国の）は、著作権侵害を「窃盗」とみなす考えをとっていないことを指摘すべきである。著作権擁護者は、権威に訴えているが、権威が言っていることを歪曲しているのである。

一般に、法律が善悪を決めるという考え方は誤っている。法律は、高々正義を実現するための試みである。法律が正義や倫理的行為を決めるということは、本末転倒である。

---

*2　*Texas Law Review* 誌の1997年3月号に掲載された James Boyle, "*Romantic Authorship and the Rhetoric of Property*"に対する彼の書評の脚注123を参照。

*3　反WIPO運動については http://www.wipout.net/を参照。

フリーウェア（freeware）

「フリーソフトウェア」の同義語として「フリーウェア」という用語を使わないようにしていただきたい。「フリーウェア」という用語は、1980 年代に実行可能形式だけでソースコード抜きでリリースされたプログラムのために多用された。今日、フリーウェアという用語の明確な定義はない。

また、英語以外の言語を使うときには、「free software」とか「freeware」といった英単語を借用するのを避けるようにしていただきたい。その言語が持っているより曖昧さのない単語があれば、それを使っていただきたい。次に示すのは、「フリーソフトウェア」を各国語の曖昧さのない用語に翻訳したものである。

- アイスランド語：frjls hugbnaur
- イタリア語：software libero
- インドネシア語：perangkat Lunak bebas
- エスペラント語：libera softvaro
- オランダ語：vrije software
- 韓国語：ja-yu software
- スウェーデン語：fri programvara
- スペイン語：software libre
- スロバキア語：slobodny softver
- スロベニア語：prosto programje
- チェコ語：svobodny software
- 中国語：zi4you2ruan3jian4[*4]
- デンマーク語：fri software OR frit programmel
- ドイツ語：freie Software

*4　【訳注】http://www.gnu.org/philosophy/words-to-avoid.html より追加。

- トルコ語：ozgur yazilim
- 日本語：自由（な）ソフトウェア
- ノルウェー語：fri programvare
- ハンガリー語：szabad szoftver
- フィンランド語：vapaa ohjelmisto
- フランス語：logiciel libre
- ヘブライ語：tochna hofshit*5
- ポーランド語：wolne oprogramowanie
- ポルトガル語：software livre
- ロシア語：svobodny programy*5

それぞれの言語で用語を作れば、外国の不思議な販売概念をオウム返しにしているのではなく、自由を意味しているのだということをはっきりと示すことができる。自由と称することは、最初は奇妙でうるさく感じられるかもしれないが、意味が正確に理解されるようになれば、問題が何かということも理解してもらえる。

フリーで（for free）

プログラムがフリーソフトウェアであると言いたいのなら、「フリーで」入手できると言ってはならない。この用語は、「価格ゼロで」という意味になる。フリーソフトウェアは自由の問題であって、価格の問題ではない。

フリーソフトウェアのコピーは、無料で入手できることが多い。たとえば、FTPを介したダウンロードなどである。しかし、フリーソフトウェアのコピーは、CD-ROMの形で有料で頒布されてもいる。一方、私有ソフトウェアのコピーは、宣伝用に無料で入手できる場合がある。また、一部の私有パッケージは、特定のユーザーに対しては無料で配られる。

---

*5　【訳注】http://www.gnu.org/philosophy/words-to-avoid.html より追加。

プログラムが「フリーソフトウェアとして」流通していると表現すれば、混乱を避けられる。

ベンダー（vendor）[*6]

ソフトウェアパッケージの開発者の一般的な呼称として「ベンダー」という用語を使わないようにしていただきたい。多くのプログラムはコピーを販売するために開発されるが、その開発者は間違いなくベンダーである。この中には一部のフリーソフトウェアパッケージも含まれる。しかし、個人のボランティアや組織によって開発される多くのプログラムは、コピーの販売を意図していない。それらのプログラムの開発者はベンダーではない。

保護（protection）

出版社の代理人は、著作権を話題にするときに「保護」という用語を使いたがる。この単語は、破壊や被害を避けるという意味を持っているため、人々が著作権によって制約を受けるユーザーの立場ではなく、著作権から利益を受ける権利保有者や出版社の立場にあるかのような錯覚を起こさせる。

「保護」という言葉を避け、中立的な用語を使うのは簡単なことである。たとえば、「著作権保護は非常に長い期間存続する」ではなく「著作権は非常に長い期間存続する」と言えばよい。

著作権を支持するのではなく、批判したいのなら、「著作権による制約」という用語を使うこともできる。

初出: 第 1 稿は 1996 年に執筆。このバージョンは、"*Free Software, Free Society: Selected Essays of Richard M. Stallman*", 2002, GNU Press (http://www.gnupress.org/); ISBN 1-882114-98-1 の一部である。

*6　【訳注】http://www.gnu.org/philosophy/words-to-avoid.html より追加。