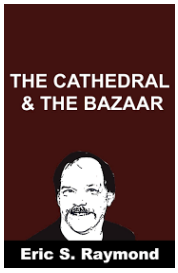


オープンソースソフトウェア

文教大学　情報学部

白石　亘

目次

はじめに

このテキストは「オープンソースソフトウェア」のテキストです。この授業ではオープンソースソフトウェア（以下、OSS と略す）を知るために、OSS に関連する幾つかのことについて学びます。

オープンソースソフトウェア（OSS）は無料で使用できて、しかもソースコードが公開されています。利用者はソースコードを修正して新たな機能を加えることもできます。利用者は無条件で使って良いのでしょうか。使うときに何か注意することはないのでしょうか。また開発者側のビジネスモデルはどうなっているのでしょうか。ソフトウェアという製品の特徴、オープンソースソフトウェアが生まれた背景、関連するビジネスについて理解を深めていきます。授業では、実際に OSS を調査してみたり、OSS を使ってみたり、OSS 開発で使われている開発ツールを使ってみることによって、理解を深めていきます。

授業の到達目標

この授業で目標とする到達点は次の 3 つに大別できます。

・OSS の開発スタイルについて説明できる

・バージョン管理システムを使うことができる

・OSS の利用方法と提供する側のビジネスモデルについて理解し説明できる

・OSS 製品を使ったサーバ構築ができる

授業の内容

・ソフトウェアの性質とソースコードについて

・ソースコードについて

・OSS にはどのような製品があるか調査

・著作権と利用許諾、ソフトウェアの権利保護について

・フリーソフトとオープンソース

・OSS を利用する上で注意すること、商用ソフトとの違い

・OSS 製品（Linux, Apache, PHP, MySQL, WordPress）を仮想マシンにインストール

・インストールしたソフト（Linux, Apache, PHP, MySQL, WordPress）の動作確認

・開発者のビジネスモデル

・開発ツール（git）について

・開発ツール（git）を使ったバージョン管理を体験する

参考文献

書籍

・「オープンソースソフトウェア」クリス・ディボナ他

・「フリーソフトと自由な社会」Richard M.Stallman エッセイ集

・「伽藍とバザール」

・「それがぼくには楽しかったから」リーナス・トーバルズ

・「リナックスの革命」ペッカ・ヒマネン

資料

・「ソフトウェアについてのコメント」（「フリーソフトと自由な社会」）

・「オープンソースソフトウェア」

・「GNU 宣言」（「GNU Emacs マニュアル」）

ソフトウェアの性質とソースコード

まずソフトウェアの特徴とソースコードの特徴について確認しましょう。

ソフトウェアとは？

そもそもオープンソースソフトウェア（OSS）とは何でしょうか？まずは言葉の定義から確認していきましょう。

コンピュータ上の情報のひとかたまりを「ファイル」と呼びます。コンピュータで実行できる「実行コード」が含まれているファイルのことを「実行ファイル」と呼びます。実行コードのもとになるのは「ソースコード」です。ソースコードはテキストファイルに記述します。このテキストファイルを「ソースファイル」と呼びます。ソースコードにはプログラミング言語の文法的な記述、動作環境の OS に対するシステムコール、ライブラリ

関数の呼び出しが含まれています。

ソースコードは、テキストエディタや IDE（Integrated Development Environment - 統合開発環境）と呼ばれるアプリのエディタで作成するのが一般的です。ソースコードはコンパイラによって実行コードにコンパイルされ、リンカーによってライブラリなどと結合されて実行ファイルが作成されます。これらの工程を「ビルド」と呼びます。

代表的な IDE には、Xcode、Visual Studio、eclipse のような開発ツールがあります。開発ツールには複数のソースファイルをひとまとめに管理する機能や、それぞれコンパイラ、リンカー、Make ツール、テキストエディタ（ソースエディタ）が用意されています。ターゲットとなる言語に特化したエディタには、プログラミング言語の構文が色付きで表示されたり、呼び出している関数の定義文に簡単に移動できたりする機能がついています。開発ツールの「ビルド」コマンドを実行することによって、ワンタッチで構文がチェックされ、コンパイルされて実行コードが生成され、複数の実行コードがあればそれらがリンクされて実行ファイルが生成されます。

このようにしてできた実行ファイルと、実行に必要な条件を保存しておくための設定ファイル、アイコンなどを表示するための画像ファイル、ヘルプファイルやマニュアル、などを含めた一式をソフトウェアと呼びます。広義のソフトウェアという単語はハードウェアとの対比で使われる場合もありますが、ここではコンピュータ上で実行可能なファイルと関連するファイル群のセット、とします。

実行コードとソースコード

コンピュータで実行できる形は実行コードですが、機械語とかバイナリコードとも呼ばれます。機械とはコンピュータのことで、バイナリとは英語で 2 進数のことです。 2 進数ですので、1 と 0 だけで記述されています。機械語という名の通り人間には理解しにくいです。バイナリコードはソースコードをコンパイルすると生成されます。初期の頃は人間が機械語を記述することもありましたが、プログラミング言語が発明されてからは、ソースコードは人間が理解しやすいプログラミング言語の文法で記述します。

フリーソフトの提唱者であるリチャード・ストールマン（Richard M. Stallman、RMS とも呼ばれる）の説明がわかりやすいので要約を紹介します。

／＊　参考文献：「ソフトウェアについてのコメント」（「フリーソフトウェアと自由な社会」から要約）

たとえば、C 言語で、画面に「Hello World!」と出力するプログラムは次のようになる。

```
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!" );
   }
}
```

しかし機械語では、上記のプログラムは次のようになる。言語で書けば 3 行のプログラムの、これはほんの一部。このような記述がもっとたくさん続く。

```
11000111 10111010 10010100 10010010 10101110
01101010 10011000 00111100 10110101 01111101
01001111 11111111 00101101 10000000 10100100
01001000 01100101 01101100 01101100 01101111
.....
```

これがバイナリという 2 進数の表現形式である。このように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。人間にとってはそのようなデータを理解するのは非常に困難である。機械語のプログラムに変更を加えるためには、動作環境であるコンピュータが機械語をどのように解釈するかについての詳細な知識も持っていなければならない。上記のような小さなプログラムであれば、可能かもしれないが、役に立つプログラムであれば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、上記の C 言語のプログラムを「Hello World!」と出力する代わりに、日本語を出力する場合を考えてみよう。新しいプログラムは次のようになる。

```
int main(){
   printf( " こんにちは " );
   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。しかし、バイナリ表現を書き換えようとすると、熟練したプログラマでさえ、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機械語のことではなく、C とか Java といったより高い水準のプログラミング言語で書かれたプログラムのことを指す。

プログラミング言語には、アセンブラ、C、Pascal、FORTRAN、COBOL、C++、Objective-C、C#、Java、PHP、Ruby、Python、JavaScript、Swift など様々な種類がある。 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バイナリコードと比べれば、はるかに扱いやすいといえる。

参考文献：「ソフトウェアについてのコメント」（「フリーソフトウェアと自由な社会」から要約） ＊／

上記の説明を補足します。たとえば Web サーバを介して動作する Web アプリケーションの場合、プログラムはサーバにソースファイルをアップロードするだけで動くものがあります。これをインタープリター（逐次翻訳実行型）といいます。ソースファイルが呼び出されてからソースコードが読み込まれて、逐次コンパイルされて実行されます。上記の中では PHP、Ruby、Python、JavaScript がインタープリターのプログラミング言語です。事前にビルドする必要はなく、実行環境にソースファイルを置いておくだけで実行してくれます。この場合はソースファイルだけが存在して、実行ファイルは存在しない、ことになります。ビルドの必要がないため、開発する人にとっては便利ですが、実行の直前にその都度コンパイルするために実行速度は遅くなります。

ソースコードを読みやすくする

ソースコードは人間が読むものです。しかしワープロのように見出しをつけたり、フォントを変えたりすることができません。プログラマは、そういった文字に対する装飾やレイアウトに頼ることなく、ソースコードの本文で読みやすく書くように工夫しています。わかりやすい変数名をつけたり、機能の単位で関数化したり、ソースファイルを分割したりして、人間ができるだけ扱いやすい形になるように記述することが良いこととされています。将来修正が必要になったときに費やす時間が少なくなるからです。コンピュータに同じことをさせるのに、ソースコードが異る、ことはよくあります。著作権法で保護される対象となっているのも頷けます。

特に、機能の単位で関数化したり分割することを「モジュール化」といいます。これによりプロジェクトに依存する部分、ハードウェアに依存する部分、データベースに書き込む部分、のように機能の単位を分けていきます。複数の開発者が同時にプログラミングする場合は、このモジュール化が重要になります。

ソフトウェアにはどんなものがあるの？

クローズドソースとオープンソース

一般的に商用のソフトウェアは実行コードだけを配布しています。ソースコードは付いていません。オープンソースに対してクローズドソースと呼ばれたりします。クローズドソースはいつごろからこのような形になったのでしょうか。

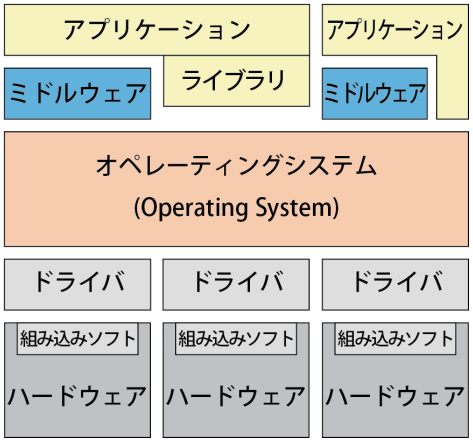
ほとんどのソフトウェアがソースコード付きで配布されていた時代がありました。まだパーソナルコンピュータがなくて大型汎用機の時代、ソフトがハードのおまけだった時代です。その後、1980 年代になってパーソナルコンピュータの登場でハードウェアが普及する時代になると、ソフトウェアだけを販売して利益が見込めるようになりました。ソフトウェア企業の登場です。儲かるソフトウェアの場合、他人に真似されたり、簡単にコピーされたりすることが問題になります。この頃にソースコードをクローズドにしたソフトウェアが市場に出回るようになります。

商用ソフトとフリーソフト

無料のソフトと有料のソフトがあります。無料の場合、有料版があるとか、広告が表示されるとか、ハードウェアのおまけとか、幾つかのタイプがあります。有料は文字通り、売っているソフトです。

用途別

ソフトウェアには役割によっていくつかのタイプに分けることができます。



・アプリケーションソフト

ダブルクリックして起動するタイプのソフトです。画面にウインドウやメッセージを表示してユーザの操作に応じます。ブラウザとかワープロ、メーラー、などなど、皆さんが普段使っているソフトです。

・ミドルウェア

アプリケーションソフトを「クライアント」とするならば「サーバ」的な役割の画面表示をしないソフトです。サービスと呼ばれたりします。データベース管理システム、ファイルサーバ、Web サーバなどが該当します。

・オペレーティングシステム

基本ソフトとも言われています。商用では Mac OS X, Windows, iOS, OS/360, HP-UX, Solaris、OSS では Linux, Android, BSD があります。

・ドライバと組み込みソフト

ハードウェアを OS に認識させるために必要になるのがドライバと呼ばれるソフトです。ハードウェア側には「組み込みソフト」が実装されていて、お互いに通信しながらハードウェアを制御します。アプリケーションからは OS を介して制御されます。

ソフトウェア開発工程におけるファイル管理

ソフトウェア開発の仕事には次のような工程があります。工程は何回も繰り返されます。

1．設計

　　要望を分析、実装方法を調査、ユーザと打ち合わせ

2．コーディング

プログラミング、テスト

3．ビルド

ソースファイルをコンパイルして実行コードを作成、ユーザに提供するファイルを作成

4．テスト

テストデータを作成、テストレポートを作成

これらの工程で重要になってくるのは、ソースファイルや設計書、マニュアル類を何処に保管するか、という問題です。ビルドを担当する人にとっては、どれが最新か、最新のリリースのもととなっているソースファイルがどれかがわからなくてはなりません。また、プロジェクトの規模が大きくなってくると、複数の開発者がいるために、さらに厄介な問題になります。

ソースコードを記述する、という行為は、多くの場合すでにあるソースファイルを変更する、作業になります。自分が今書いているソースファイルは、どこからか持ってきたファイルが元になっています。これを修正して、おそらくは元のファイルがあったところの近くにフォルダ名を変えて保存します。

●ソースファイルをバックアップする

ソースファイルは修正の前後にバックアップファイルを作成して、意図しない消失を防ぎます。ひとりで開発しているのであれば、外付けハードディスクや USB メモリにコピーを保管という方法もあるでしょうし、同じマシン上に圧縮ファイルを作っておくだけ、かもしれません。今では Dropbox とか iCloud を使う場合もあります。

チームで開発する場合、ファイル共有する必要が出てきます。ファイルサーバに保管というのが一般的でしょうか。この場合、ファイルにアクセスするためのルールを決めることになります。フォルダ名の付け方とか、フォルダの階層構造のについての決めごとです。誰がどこにファイルを保存したのかがわからないと非効率だからです。

●バージョン管理

コンピュータ上のファイルは簡単にコピーを作ることができます。そのため、似たようなファイルがたくさんできて、どれが最新かがわかりにくくなりやすい、という特徴をもっています。ソースファイルも同様です。フォルダ名、ファイル名を工夫しても、これがどんなソースだったがを記述するには限界があります。開いてみてみないとわからない、ビルドして実行してみないとわからない、ということになります。

このような事情があるために、版管理（バージョン管理）という仕組みがあると便利です。ツールとしては Git が広く使われています。修正のひとかたまりにコミットコメントをつけて保存しておき、あとで過去の状態に遡ることができる、ソースレベルで修正箇所を確認できる、ような機能が備わっています。

●課題管理

誰がどの仕事を受け持っていて、いつまでにやるか、今どういう状況か、ということを管理する必要があります。関係者が 1 つの場所に集まっている場合には、会議室やホワイトボードで課題を確認したり共有することができます。関係者の居所が離れている場合は、クラウドサービスを使うことができます。

バージョン管理と課題管理（ToDo チケット管理）の仕組みを組み合わせてうまく使うとさらに便利ということとなります。

インターネット上に置かれたサービスを利用することもできます。たとえば GitHub は、Git を使ったバージョン管理サービスを提供するサイトとして一般的です。

演習 1

グループを作る

演習 2 と演習 3 はグループに分かれて作業します。今のうちにグループを作ってメンバー同士で自己紹介してください。

slack

オープンソースではありませんが、無料で使えるチャットツールです。クラウドサービスです。slack 上にチームを作成して、それぞれを Invite してその上で自己紹介してみましょう。

ソフトウェアの権利保護

ソフトウェアの権利は著作権法で保護されることになっています。ソフトウェア特許という方法もあります。ソフトウェアの権利保護の観点から、著作物とアイデア、著作権法、ソフトウェア特許について知っておきましょう。 著作権法は「表現」特許法は「アイデア」を守るための法律です。

背景

もともとハッカーコミュニティでは人にソースを見てもらったり、他人のソースを再利用したりすることが当たり前でした。他人のコードをコピーして修正して使う、自分のコードも誰かが同じようにコピーして修正して使う。必要なときに見せてもらえる。このような関係者同士であれば、コミュニティの中では、権利を主張したり、権利保護する必要はありませんでした。

1980 年代になると、開発者の書いたプログラムを多くの利用者が利用する、という事になりました。ソフトウェアは利益を生む産業になったのです。ソフトウェアには、ハードウェアに比べてコピーが簡単で、大量に配布するときの手間が少ないという特徴があります。そして、何らかの権利保護が必要、ということになりました。根底には、ソフトウェア企業の権利を保護することで、より良いソフトウェアがリリースされ、社会がより良くなるだろうという期待があります。

ソフトウェア企業は、自社が多くの時間を使って書いたプログラムを、他社が無断でコピーしたり修正して使うことを嫌います。ソフトウェアにはアイデア、プログラミング、解決した課題、サポートノウハウ、などの様々な工夫が盛り込まれています。コピーを使う人はこうした多大な労力を払うことなく、出来上がった機能だけを利用して利益を得てしまいます。ソフトウェア開発者は何らかの形で利用者にソフトウェアを配布しますが、作者の意図に反した利用がされないように法律で保護しないと、ソフトウェアを開発する意欲が衰えて、馬鹿馬鹿しいから誰もソフトウェアを作らなくなる、市場にソフトウェアが流通しなくなる、ことになると多くの人が心配しました。

ソフトウェアを法律で保護する方法は、政府レベルで検討されていました。日本でもアメリカでもソフトウェアの権利をどうやって法律で守るか、が議論されていたのです。日本はプログラム権法を定めようとしていましたが、1980 年にアメリカが著作権で保護することを決めて、その圧力で日本も著作権法で保護することになったようです。結局ソフトウェアの権利は著作権法によることとなり、1986 年に日本の著作権法は改定されて、著作権法にソフトウェアについての条文が加えられました。ちなみにこの頃はソフトウェアだけで特許になる、ということは想定されていませんでした。特許は自然法則を利用するアイデアを保護する法だったからです。

著作物とアイデア

ソフトウェアの開発者は著作権者として著作権法の権利保護を受けることができるようになりました。プログラムは小説と同じように、著作物とみなされることになったのです。ただし小説の場合でもそうですが、ありふれた文章は著作物とはみなされません。先程の「Hello World」程度のプログラムは著作物とはいえません。

コンピュータソフトウェアの場合、表現が含まれる著作物とはソースコード、画面、メッセージ、マニュアルも含まれます。ありふれた表現は著作物と認められませんので、画面やメッセージがまねされた場合、誰が設計しても似たような画面になる、とされた場合は元の画面に創作性が認められない＝著作権が認められない、ことになります。これはグリーが DeNA を訴えた例で見ることができます。

アイデアはコンピュータに何をさせるか、という部分になります。プログラムの実行手順のような領域です。こちらは著作物ではありませんので必要であれば「特許」で保護することになります。プログラムを書くこと自体がアイデアを書き留めている行為とみなすこともできますので、どの程度のアイデアが「ありふれている」のかは、著作物と同様に判断が難しいのです。裁判で争ってみないと本当のところはわからないのが実情です。

著作権

著作権は登録したり申請することなく発生します。登録にも維持にも費用はかかりません。保護期間は、個人の場合死後 50 年、法人の場合は公開から 50 年です。皆さんが絵や文章を書いたらそれは著作物になり得るのです。そして著作物は著作権者の許可なしに利用することが禁じられています。著作権は「表現」を守るための権利です。

詳しくは文化庁のサイトを見ましょう。

著作権テキスト

http://www.bunka.go.jp/chosakuken/text/index.html

著作権契約

http://www.bunka.go.jp/chosakuken/keiyaku_intro/index.html

著作権法でソフトウェアに関係する権利は主に次の 3 つです。

著作者人格権

契約によっても譲渡できない、著作物の人格的部分です。公表権：著作物を公表する権利、氏名表示権：これは私が書いたものです、といえる権利、同一性保持権：私の書いたものと同じです、といえる権利があります。著作人格権を譲渡する契約は無効になる恐れがありますが、著作人格権を主張（行使）しない、という契約は可能です。

複製権

コピーを作る権利です。CD メディアを複製したり、ハードディスクにコピーを作ったり、ネットで転送したりする行為は、すべて「複製」に該当します。著作権法では複製後の「配布」については何も規定していません。配布は自由です。

翻案権

元の著作物を修正する権利です。これを「翻案」といいます。翻案した結果の著作物にも著作権が発生します。これを二次的著作物といいます。

特許

著作権は「表現」を守るための権利ですが、一方で、ソフトウェア特許を取得して権利を保護することもできます。著作権と違って出願申請・審査請求・登録の手続きが必要で費用がかかります。保護期間は出願から 20 年、毎年維持費を支払う必要があります。

特許は「アイデア」を守るための権利です。以前はアルゴリズムとハードウェアを組み合わせて特許にするのが一般的でしたが、近年はソフトウェアだけの特許が認められるようになってきました。この流れは、ビジネスモデル特許と同様にアメリカを発端として日本に波及しています。

アイデアを特許で守ろうとした場合、そのコンピュータソフトによって、これまでできなかったことができるようになった、ことなどを請求明細書に示して特許出願します。そのあと出願が審査されたときに、特許が成立するには、そのアイデアが新しいこと（新規性）、容易に考え出すことができないこと（進歩性）、先に出願がないこと（先願主義）が条件になります。

特許庁のホームページ「電子図書館」にアクセスしてみましょう。ソフトウェア関連特許も多く登録されていることがわかります。たとえばマイクロソフトは多数の特許を登録しています。権利数のわりには係争事例は少なく、クロスライセンスや和解に使われているようです。

http://www.ipdl.inpit.go.jp/homepg.ipdl

特許の場合、本当に権利が存在するかは、争いが起こらないとはっきりしません。これまでに表に出てきた数少ない事件を調査してみますと、著作権が認められなかったり、特許が無効だったという判決になるという事例もあります。著作権も特許も、その効力は争ってみないとわからない、というのが現実のようです。事件を調査すると少しわかってきます。ただし治郎吉商店の例もそうでしたが、弁護士による調停、裁判所調停、訴える前の和解、など公には目立たない方法で決着が付くことが多いようです。

判例のデータベースを紹介しておきます。

http://tyosaku.hanrei.jp/

権利保護

ソフトウェアが著作権法で保護されているということはどういうことでしょうか。 著作者に無断で利用することはできないということです。利用者は著作者から利用許諾を得られていない場合は、利用することが違法行為になります。著作権者はどのような条件が成立したらソフトウェアを利用していいと許諾するのかを自分で決めることができます。

たとえば著作権者は自分の書いたプログラムの利用方法に条件をつけることができます。また、自分の書いたプログラムが真似された場合は、模倣製品を差し止めることができます。著作権の場合は、このような強い権利が何の申請もなく付与されています。

特許で保護する場合、表現は問われません。同様の問題解決を特許取得者に断りなく実施した場合に差し止め請求、ロイヤリティ請求が可能です。

利用規約違反

配布したソフトウェアが作者の意図（ライセンス）に反した利用がされた場合に差し止めることができます。たとえば「ソースコードをつけて再頒布せよ」というライセンスを守らずにクローズドソースで販売していた場合、訴えられる可能性があります。実際に GPL ライセンスに違反したとして Free Software Foundation から訴えられて敗訴した事例があります。

模倣

著作権者に無断で表現を再利用することです。翻案権の侵害にあたります。見た目を真似したことを証明できれば、製品を差し止めたり、損害賠償を請求することができます。誰がやっても同じような表現になる場合、独自性が低い場合など、著作権侵害を証明することが難しい場合もあります。

特許違反

特許を持っている側は、自分の権利を侵害したことを知った場合、製品を差し止めたり、ロイヤリティを請求することができます。訴えられた側は、特許の無効審判請求をしたり、侵害していないことを証明したりして対抗します。

フリーソフト

「free（フリー）」という英語には「自由」という意味と「無料」という意味があります。フリーソフトは「無料」の場合もありますが、「自由」の方を重要視していて、「unfetterd」が近い意味を持ちます。反対語は「proprietary」誰かに所有されているという意味です。

リチャード・ストールマン

提唱者であるリチャード・ストールマンは 1971 年から MIT の AI ラボに働いていました。そこにはハッカーコミュニティが存在したと言っています。ソフトウェアは、誰でも自由に使え、ソースコードを見せあって、改良を加えながら利用していくものでした。1981 年に Symbolics 社が AI ラボ出身者によって設立され、MIT のメンバーの多くが移籍。ハッカーコミュニティが崩壊して、もともとソフトウェアを自由に使えた環境が変化してしまいます。ストールマンは将来の選択を迫られていたその頃、あるプリンタドライバに機能が不足していたために、提供者にソースコードの提供を求めたが拒否されたときに腹を立てた経験などから、自分がプロプライエタリ（Proprietary）なソフトのユーザや開発者にはなりたくない、と考えたようです。

リチャード・ストールマンは、ソフトウェア企業がソフトウェアを所有しているという概念を持ち、海賊版が違法だと主張するとき、次の 3 つの仮定があると述べています。

1）ソフトウェア企業はソフトウェアを所有する自然権を持っている

2）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

3）ソフトウェア企業の支配権がなければ、ソフトウェアは作られない

これに対して、ストールマンは次のように反論しています。

1）著作権は自然権ではなく、コピーをするというユーザの自然権を制約するもの

2）ソフトウェアで、コンピュータユーザはどのような社会を持つか自由であるべきだ

3）フリーソフトウェア運動でも大量の役に立つソフトは作られる

このような考えを実践するために、自分でフリーソフトを書くことにしたようです。努めていた MIT をやめることも必要と判断しました。クローズソフトに対抗して、ソースも含めてソフトウェアは自由に使えるように、すべてのソフトウェアをフリーソフトで提供しようと、1984 年にリチャード・ストールマンは GNU プロジェクトを立ち上げます。そのときにフリーソフトウェアの定義は以下の条件をみたすものであるとしています。

- ユーザ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おり、コミュニティがそのユーザによる改良から利益を受け取ることができる。

それからストールマンはフリーソフトを書き続けました。やめたあとも MIT の所長に仕事場と設備は提供されたようです。プロジェクトの最初の作品が GNU Emacs です。Unix でフルスクリーンエディタが使えるようになりました。エディタの次はコンパイラ、リンカー、デバッガ、OS とフリーソフト開発を計画していました。まだ途中段階のフリーソフト計画でしたが、すでに GNU Emacs だけでも使いたいという人が現れ始めました。

Free Software Foundation

まだ当時はインターネットが普及していなかったため、多くの人に配布するには手間がかかりました。そこでストールマンは、Emacs を使いたい人には $150 でテープにコピーをとって郵送する、という方法にしました。これがフリーソフトウェア流通ビジネスの始まりです。このあと 1985 年に Free Software Foundation（フリーソフトウェア財団）、ソフトウェア開発の活動資金を稼ぐための団体を設立しました。流通の手数料を受け取ったり、寄付金を受け付けたりしています。

http://www.fsf.org

GNU Public License

ストールマンは、その頃にコピーレフトを宣言します。そして、GNU ソフトウェアが私有ソフトウェアに化けるのを防ぐための頒布条件が必要、ということで GNU Public License（GPL）を作成します。

このライセンスはライセンス自体もフリーです。これは誰でも利用できるものとして認知されるようになりました。ソフトウェア開発者の中にはストールマンと同様にソフトウェアはフリーであるべき、またそうであっても良い、と考える人がいます。そのような開発者が自分のプログラムを世の中に配布するときに、利用許諾（ライセンス）を添付します。そこで GPL が採用されていきます。開発者がライセンス条文を作るのは手間ですし、自分の書いたプログラムを公開したことが元になって、悪用されたり訴訟事になったら大変です。GPL が適用されていれば、自分で条文を作るよりも安心、という一面もあります。Ruby の作者であるまつもとゆきひろさんは、「GNU フリーソフトには今まで世話になってきたので、自分のプログラムを公開する場合も同じライセンスと決めていた」と語っています。

Linux を開発したリーナス・トーバルスもそう考えた一人かもしれません。Linux には GPL が適用されていましので、GNU ソフトと親和性がありました。1992 年に Linux と GNU ソフトが結合して、完全なフリーオペレーティングシステムが完成しました。これを GNU/Linux と呼んでいます。

参考文献：「GNU 宣言」、「GNU Emacs マニュアル」

Open Source Initiative（オープンソース・イニシアティブ）

1998 年に Bruce Perens と Eric Laymond が立ち上げた団体です。オープンソースソフトウェアを促進することを目的としています。ライセンスを承認したりして、オープンソースソフトウェアの普及に努めています。

http://opensource.org/

オープンソースの定義（The Open Source Definition）

ここでは自由な配布、ソースコードがついていること、など次のオープンソースを１０の要件で定義していま

す。日本語のサイトは次です。

http://opensource.jp/osd/osd-japanese.html

要約は次のとおりです。

Introduction（はじめに）

オープンソースは、ソースコードが入手できるというだけでなく、頒布についての以下の基準を満たしている必要がある。

１．Free Redistribution（再頒布の自由）

頒布物の一部として配っても良い。販売、無料配布のどちらでも良い。販売した人からロイヤリティや手数料を請求してはいけない。

２．Source Code（ソースコード）

ソースコードを付けること。つけるのが難しい場合は、無料ダウンロードなどで簡単に入手できること。ソースコードが何処にあるかわかりにくくしたり、中間コードしかない、のはだめ。

３．Derived Works（派生ソフトウェア）

もとのソースを変更したり派生の製品を認めること。同じライセンスで頒布することも認めること。

４．Integrity of The Author's Source Code（作者のソースコードの完全性）

ソースの変更は認めるものの、元の作者の作った部分を完全な形で残す努力はしていい。ライセンスが変更箇所をパッチファイルで頒布する場合に限り、制限をつけていい。修正コードの配布を認めなくてはならないが、派生物に異なる製品名、バージョン名をつけさせることは要求しても良い。

５．No Discrimination Against Persons or Groups（個人やグループに対する差別の禁止）

特定の個人やグループを差別してはいけない。

６．No Discrimination Against Fields of Endeavor（利用する分野に対する差別の禁止）

ビジネス用途はだめとか、遺伝子研究に使ってはだめとか、利用分野を制限してはならない。

７．Distribution of License（ライセンスの頒布）

プログラムについているライセンスは再頒布するプログラムにも適用できること。追加の作業が必要であってはならない。

８．License Must Not Be Specific to a Product（特定製品でのみ有効なライセンスの禁止）

プログラムが特定の再頒布物に含まれていることを条件にライセンスが付与する、というのはだめ。特定の再頒布物に含まれていたプログラムを一部取り出しても、全体と同じライセンスが適用されなくてはならない。

９．License Must Not Restrict Other Software（他のソフトウェアを制限するライセンスの禁止）

ライセンスは一緒に頒布するソフトウェアのライセンスを制限してはいけない。オープンなことも含めて。

１０．License Must Be Technology-Neutral（ライセンスは技術中立的でなくてはならない）

ライセンスの許可が特定の技術に強く依存してはならない。たとえば同意するためのダイアログを出せない環境では同意できないなどということがあってはならない。

伽藍とバザール

Eric Laymond は Linux の開発が成功した事例を見て、自身の論文「The Cathedral and the Bazaar（伽藍とバザール）」で、オープンソースプロジェクトについてその良さを語っています。

```
http://cruel.org/freeware/cathedral.html
```

伽藍方式とは、従来からあるソフトウェア開発の方法で、当時のソフトウェア開発というと、ウォーターフォール・モデルというのが一般的でした。これは、大規模簿なソフトであっても要求定義～設計～実装～テスト～レビューという流れで作業することを差しています。ひとりでやった場合でも、たとえばリチャード・ストールマンのソフトウェア開発はこれに当たるとされています。

一方でバザール方式というのは、機能ごとに誰かが実装して、コミッターと呼ばれる管理者がコミュニティの利益になると判断した場合にリリースする、ような開発者たちがよってたかって１つのプロジェクトを作っていく様子を差しています。

レイモンドの文章を読んで次の問いに答えなさい。

１）バザール方式は誰のどのプロジェクトで採用されたか

２）伽藍方式と比べて、バザール方式は何が優れているのか

２）バザール方式が成立するための条件とは何か

これらについて「伽藍とバザール」を読んで 1000 文字程度にまとめなさい。

利用許諾書

著作権者は利用者に条件をつけることができます。この条件 – 利用許諾条件、ライセンスとも言います – を書いた書類を「利用許諾書」（License）と呼びます。著作権者は利用許諾書に、使用する条件、複製する条件、翻案する条件、再配布の条件などを記述しておきます。

オープンソースソフトウェアによく使われるライセンス（利

用許諾書）

オープンソースソフトウェアにはライセンスが付いていますが、代表的ないくつかのライセンスがあります。開発者は独自にライセンスを記述してもかまわないのですが、すでに実績のあるライセンスがコピーフリーで存在しますので、自分のソフトもこれと同じでいい、という具合に採用されてきました。よく知られた「実績のある」利用許諾書たちです。

http://opensource.org/licenses/index.html

GPL

General Public License（GPL、GNU 一般公的使用許諾）、Copyleft（コピーレフト：コピーライトの反対という意味）と呼ばれているライセンスです。利用者に実行・複製・改良することの自由を保証させたい、というリチャード・ストールマンが提唱しました。ソフトウェアの利用者は自由に使いたいはず、という立場に立っています。著作権者がコピーレフトを使って、二次著作物にも同じライセンスを適用する、という仕組みで、自由なソフトウェアを広げていこうとしています。1989 年、リチャード・ストールマンがいた MIT(マサチューセッツ工科大学）で発表されました。元々は gcc、emacs、make などのライセンスでしたが、Linux が採用したことによってメジャーになりました。

ソフトウェアを再配布する際にソースを公開することを条件にしています。 このライセンスが適用されたソースを修正して派生製品を作った場合も、再配布の際にはその派生製品のソースも含めてすべてのソースを公開する必要があります。

再配布しない場合、つまり自分で使う場合はソースを公開する必要はありません。

GPL について詳しくは、八田真行さんのサイトを見てみましょう。

```
http://www.opensource.jp/osd/osd-japanese.html
```

BSDL

Berkley Software Distribution License の略です。著作権者、ライセンス条文、無保証の 3 点を明記しておけば、ソースコードを開示しなくても製品を再配布できる、というライセンスです。

ソースコード公開は自由で、必ずしも公開する必要はありません。BSDL 製品のソースを入手して、少しだけ改良して、コンパイルしてソースなしで販売する、ということも可能です。ただし「BSDL の○○製品を利用している」ことを明示する必要があります。

再配布時に著作権表示を義務づけているため、リチャード・ストールマンはこれがフリー（自由）ではないと指摘しています。が、共存は可能であるとも述べています。しかし BSDL の目的で配布するパッケージに GPL でライセンスされた製品が含まれていると問題になります。このため GPL ライセンスの製品と混在させて再配布することは難しいと考えられています。

```
http://www.gnu.org/philosophy/bsd.ja.html
```

商用ソフトの利用許諾書

治郎吉商店の「文庫番」の例を見てみましょう。この製品は商用ソフトウェアですが、かなりシンプルな例です。

文庫番 v3 使用許諾書
本使用許諾書の内容に同意いただける場合に限り、「文庫番」(以下「本製品」) を下記の条件で使用することができます。本製品の使用を継続される場合は同意いただいたものとみなします。
下記の条件に違反した場合は、使用許諾は自動的に取り消されます。
1. 本製品に含まれているソフトウェアや文書などのすべての著作物の著作権は、株式会社治郎吉商店 (以下「当社」) に帰属します。
2. 本製品は 1 ライセンスにつきコンピュータ 1 台にインストールして使用できます。使用者が特定の 1 人に限られていて、インストールされている本製品を同時に使用しない場合は、本製品を複数のコンピュータにインストールして使用できます。第三者に本製品を使用させることはできません。
3. お客様は、本製品に含まれているソフトウェアについて、著作権法上認められたバックアップのための複製をすることができます。本製品の内容を改変することはできません。
4. 本製品の使用に伴ってお客様または第三者が直接または間接に損害を被った場合でも、当社は責任を負いません。
5. お客様は、ユーザ登録から 90 日間、本製品の無料サポートが受けられます。ユーザー登録の方法は製品に同梱されたユーザ登録カードを参照してください。

演習 2

OSS 製品を調査する

OSS には様々なタイプの様々なソフトウェアがあります。中にはよく知られた定番のような製品もあります。それらの中から、今年は次のソフトを使ってみたいと思います。

Linux

OS（オペレーティングシステム)。リーナス・トーバルズが始めたプロジェクト。様々なパッケージ（ディストリビューション）がある。ライセンスは GPL。

Apache、NGINX

Web サーバ。ホストの 80 番ポートで HTTP プロトコルを待っている（リッスンしている）サービス。httpd ともいう。主に HTML ファイルを返信する。ライセンスは Apache License。

PHP

インタープリター言語。C 言語の文法に似た書式で記述する。Web アプリケーション開発に向いている。ライセンスは PHP License。

MySQL（MarierDB）、PostgreSQL、SQLite

リレーショナルデータベースマネージメントソフトウェア（RDBMS）。SQL という言語でデータを操作できる。MySQL のライセンスは GPL、または商用（Commercial）ライセンス。PostgreSQL のライセンスは BSD License。

vi、Emacs

テキストエディタ。vi はビル・ジョイ、Emacs はリチャード・ストールマンの作品です。Unix 系 OS に附属していますので、どちらかを使えるようになりましょう。設定ファイルなどのテキストファイルを編集するときに使います。

WordPress

ブログサイト構築ツールです。

git、GitLab

バージョン管理ツールです。

上記の OSS について、次の事柄などを調べましょう。調べたことをグループで一つのメモにまとめてください。

1）サイトの場所（英語、日本語）

2）製品の機能概要、特徴

3）沿革、初期の開発者

4）利用許諾書、利用条件、配布条件（ライセンス）

5）ソースコードの場所

slack 上で一つのドキュメントに統合してみてください。

コミュニティ

役割

OSS を開発しているメンバーの仲間になることを「コミュニティに参加する」などといいます。役割はいくつあります。

・ユーザとして利用する（これはソフトウェアをテストすることにもなります）

・不具合を報告する

・ドキュメントを作成する

・ホームページを制作する

・パッケージを制作する

・ソフトウェアを修正する

参加

コミュニティに参加するには、まずその製品のユーザであることが大前提です。まずは使ってみてから、メーリングリストや RSS に登録して情報を収集しましょう。

バグらしき現象を発見したらバグレポートをメールで送りましょう。使い方や仕様に疑問を感じたりしたら質問メールを送りましょう。ユーザ会やオフ会に参加しましょう。そうしてだんだんと輪の中に入っていき、自分でもできる仕事を見つけていく、というのが一般的な流れのようです。

共同開発

一般的にオープンソースプロジェクトの開発はオープンなネット上で行われます。小規模のソフトウェアの場合は 1 人で開発することもありますが、オープンにする目的のひとつに開発を手伝ってくれる人を募集している、という意味が含まれていますので、オープンソースソフトウェアは複数のプログラマが共同でひとつの製品を開発するスタイルになります。

共同開発するためには、コンフリクトを起こさないようにバージョン管理ツールを使ったり、チケット発行などのツールを使って、プロジェクトを効率よく管理したり、生産性を高める工夫をしているのが一般的です。

関連→　Subversion, CVS, trac、git,

バージョン管理ツールを使った共同作業

OSS の開発に広く使われているバージョン管理ツール（VCS: Version Control System）を体験しましょう。代表的な「git」を紹介します。OSS の共同作業のために開発されましたが、MacOS に標準装備されていたり、広く使われています。関連のツール「GitHub」はクラウドサービス、または自社サーバで運用するソーシャルサービスです。

git

分散型バージョン管理システムです。2005 年にリーナス・トーバルスが最初のバージョンを書きました。浜野純さんがメンテナーです。

GitHub

git をエンジンとして使った、ソーシャルサービスです。オープンソース製品開発用には無料で使えます。「GitHub Enterprise」という自社サーバにインストールして使う製品も用意されていて、社外へのアクセスが制限されているセキュアな環境で GitHub を使いたいという要望に答えています。

GitLab

GitLab という OSS があります。これをインストールすると、自分で GitHub のようなサーバを構築することができます。GitLab サーバをレンタルするサービスもあります。

演習 3

グループメンバーは、GitHub に無料アカウントを作成して、メンバーを招待して、そこでファイルを共有しましょう。git を色々なケースを想定して使ってみましょう。

解説

演習 2 で作成して Slack にアップしたメモを GitHub のリポジトリにアップして共有してみましょう。

開発者の目的

無料で配布する意図

なぜオープンソースにしたのか、なぜ無料なのかについて考察してみます。著作権者として、作品をまねされるのはいやだけれど、利用されると嬉しいことがあります。

テストケースを増やして、ソフトウェアの価値を上げたい

最も大きな動機はテストケースが増えることだと思います。ソフトウェアは使われて初めて価値が出ます。実績が少なければ価値が上がりません。

無料で配布することによって多くのユーザが使う可能性が高まります。様々なユーザが様々な環境で使えば、ソフトウェアの不具合があぶり出されて、バグフィックスによって信頼性が上がっていきます。性能的、機能的にも向上していくことでしょう。こうして良くなったソフトウェアを開発者自身も利用できます。

営業的効果（宣伝）

無料で配る目的は主に宣伝です。ソフトそのもののであったり、開発者自身、開発会社、そのソフトと連動するハードや周辺のソフトなどを目立たせたいという意図があります。開発者が趣味で作った場合でも、成果を人に見せたい、という動機は宣伝目的といえるでしょう。

宣伝目的にもいくつかのタイプがあります。

1）製品に有料版がある

2）会社の技術力をアピールしたい

3）人材募集の際に有利になりたい

自分たちに必要なツールが無料で手に入る

宣伝にならないのに開発プロジェクトを手伝っている人や会社もあります。そのソフトが自社に有用な場合です。自社のハードウェアに搭載するために必要な機能を実装していたりします。

同等のソフトが市場にないか、あったとしても買えば支払わなくてはならない対価を節約できる、という場合もあります。特に従業員の多い会社はライセンス料金が高額になります。

開発者を訓練・募集できる

プログラマを OSS プロジェクトに参加させて人材を育成する、という目的もあるようです。一方で人材を引きつける道具、ととらえることもできます。OSS を出している会社は技術力があると認められやすいからです。

売るための機能を開発するコスト、トランザクションコストが発生しない

製品はできた、使えるようにはなっている、としても売るための機能はまだ十分ではありません。次のようなことを決める必要があります。

- 価格　　直販か流通経由か、有償サポートをどうするか
- パッケージ　　実行ファイルやマニュアルを利用者に手渡す方法
- マニュアル　　インストール方法から、機能の詳細まで
- ライセンスキー　　ライセンスキーが入力されている場合はプログラムの動きを変える。

ハッカー的価値観

仕事の目的には、サバイバルー社会ー趣味、という段階があります。サバイバルというのは生きていくために必要な金銭を得るという目的、社会というのは社会的に認められたい、という動機です。趣味というのはやっていることが楽しいという動機です。

Linux の開発者リーナス・トーバルズは、これら仕事の動機は「生き残り」「社会生活」「娯楽」という 3 つの基本的カテゴリーに分類できると主張しています。

／／　**参考文献：「リナックスの革命」ペッカ・ヒマネン**

インターネットの発達した今日では、趣味的な仕事の結果が社会に大きく貢献する事例は増えていく、と思います。

一方で、もともとフリーソフトにお世話になりながら自分がプログラミングしているのであるから、成果品はフリーソフトにお返しする、という考え方もあります。

プロジェクトによってビジネスモデルが異なる

開発者はどうやって生計を立てているのでしょうか。利用者からはお金をいただかない、だけで、開発者には別の方面からお金が入ってきます。大きく分けて 2 つのパターンがあります。

1）OSS を開発することで報酬を得ている場合

A) 報酬を支払っている会社が宣伝効果を狙っている。B) 技術者のトレーニングと認識している。C) 自社がそのソフトで利益を得ている。

直接報酬を得てはいないが、案件を受注するときに有利に働く情報となる（Ruby）

ソフトを使って社内の生産性を上げている（eclipse）

社内に詳しい人がいるだけで生産性が上がる（PostgreSQL）

自社に必要な機能を実装させている（Linux）

製品にバンドルする場合など、利用によっては商用ソフトが必要になる製品（MySQL）、開発ツールで言語は無料だが生産性を高めようとするユーザ向けに別にクラスライブラリを売っている製品（PHP）もあります。

2）OSS を開発することで報酬を得ていない

完全にプライベートな時間で作ってしまうケースもあります。 リチャードストールマンのケースは、MIT に半年間在席している間にフリーソフトのプログラミングをして、残りの半年はカリフォルニアの大学で講師をして 1 年分を稼いでいたようです。

まつもとゆきひろさんの場合も、Ruby の開発は趣味だったようです。

／／ **参考資料：「オープンソースのビジネスモデル」PDF（「ソフトウェアライセンスの基礎知識」から抜粋）**

売るとしたらさらに開発コストがかかる

ソフトウェアを売る、というと何かおいしい商売のように見えますが、実はそうともいえない事情があります。一般的にハードウェアを売る場合は、同じものを作るための製造装置や原材料などの製造コスト、お客様にものを届けたり設置したりする配達コストが発生します。これに対してソフトウェアは複製コストや配達コストはゼロに近い、という性質があります。一方でトランザクションコスト（商品を説明するためのコスト、売って代金を回収するためのコスト）、知ってもらうための宣伝コストは発生します。

ソフトウェアは、複製コストと配達コストがゼロに近いために、販売元でない人でも複製と配達が簡単にできてしまいます。これは「買っていない人が使えてしまう」という状況が生まれやすいということになります。著作権法上、作者に無断で複製することは違法行為ではありますが、摘発されなければお咎めはありません。ハードウェアを盗めば証拠品が残りますが、ソフトウェアは消すのも簡単です。

このように違法コピーから製品を守るためには、コピーしづらい、またはコピーしてもそのままでは使えないようにする仕組みが必要になります。 たとえば次のような機能を実装します。

・コピープロテクト

・シリアル番号による製品管理

・ネットワークプロテクト

・ライセンスサーバ

一方で、ソフトウェアは使ってみなくては良さがわからない、という側面があります。コピーを制限するための機能と同時に、次のような簡単に使えるようにする仕組みも必要です。

・お試し版

・機能縮小版をあえてリリースして、ライセンスキーを入力すると製品版の機能が使えるようになる仕組みとか...

・アプリ内課金

・サブスクリプション

このように、「ソフトウェアを売る」ことにすると、開発者は売るための機能を実装する必要に迫られて、製品の機能や性能とは関係ないところで結構な手間が発生します。

ひとつ売るのも手間がかかります。問い合わせに答えたり、代金の支払いを確認したり、サポートしたり、といった慣れない作業に開発者は追われてしまいます。かといって、このような仕事を誰か代わりの人に依頼すると、人件費（固定費）が発生して利益が出にくくなります。

治郎吉商店の場合

治郎吉商店にはオープンソースで公開している「YAO」という C/C++ 用データベースライブラリがあります。有料の製品である「文庫番」で使うために何年もかけて開発しました。

もともと文庫番 v1 は Mac OS（Classic）で動くアプリケーションでした。このときデータベースエンジンは「NeoAccess」という有料のオブジェクトデータベースライブラリを使っていました。後に Apple 社が OS を Mac OS X に移行したとき、NeoAccess が Mac OS X 対応にならなかったため、新しい文庫番を出せなくなってしまいました。そこで自社でデータベースライブラリを作ろう、ということになったのです。

YAO の販売は検討しました。しかし Neo 社が商売を辞めたようにライブラリの販売という商売は難しいと感じていました。一般にソフトウェアの価値はテストケースが多い方が高くなります。ライブラリも多くの開発者に使ってもらうと価値が上がります。何もしないで隠しているよりは販売するほうが目立ちますのでテストケースは増えそうですが、ではいくらで販売するか？安くすればたくさん売れて高くすれば少ししか売れないのでしょうか？究極は無料です。しかし無料ですと怪しいソフトになりそうです。また動作保証がされない印象がありますし利用者が増えそうにありません。

一方でこのライブラリ開発を手伝ってくれたプログラマがいます。社外の人や学生アルバイトにそれなりの報酬で手伝ってもらいましたが、この人達が使えるようにしておいた方が良さそうだと思いました。さらにソー

スコードをつけて配布してしまえば、不具合があっても各人が修正することもできますし、フィードバックしてもらえるかもしれません。バイトで開発に参加したライブラリを自由に使えたら仕事の励みにもなりそうだと思いました。こうしてオープンソースで公開してしまおう、ということになりました。

デメリット

OSS にすると開発者には次のようなデメリットがあります。

・製品からは収入を得ることができません。

・ソフトウェアの設計図といわれるソースコードが付いていますから、ソフトウェアに使われているアイデアを秘密にしておくことはできません。

・第三者に使ってもらうだけの完成度は求められます。公表するために必要になる作業もあります。特にドキュメント作成の負担は大きいと言われています。いくら売るための実装コストやトランザクションコストが発生しないといっても、公表しない方が手間はかからないのです。自分で、または社内で使うだけのほうが手間はかからない。

利用者

無条件で使っていいの？

無料で使えますが、無条件ではありません。利用許諾書に書かれていることは守らなくてはなりません。

ソースコードが公開されていますので、ユーザ自身がソフトウェアを修正することができます。元の開発者がいなくなってもソフトウェアを保守し続けることができます。

商用ソフトと比較

商用ソフトは多機能高性能、これに対してオープンソースソフトは単機能で普通の性能、という傾向があります。商用ソフトは売るためにセールスポイントを増やそうとします。そのため機能を拡大していきますので、自然と多機能になっていく傾向があります。

また古いバージョンは新しいＯＳで動かなくなる方が都合がいい、そうすれば買い換え需要が発生してもうかる、という事情があります。

一方で買ってもらったからにはそれなりの納得感を出さなくてはなりません。どこが違うのでしょうか。

たとえば Oracle と PostgreSQL

どちらも SQL が使える RDBMS（リレーショナルデータベースマネージメントシステム）です。テーブルやフィールドの数がそこそこで、レコード件数もそれほど多くない場合には両者の差を感じることはありません。しかし負荷が高くなってきたり、運用中に危機的な状況になってきたり、分散処理をしたくなったり、ハイエンドのマシンで動かしたい場合など、差が出てきます。このようなときに Oracle にしておいて良かったね、となるかもしれません。組織によっては、管理者エンジニアの負担を軽くするため、または品質管理上の理由で、それほど機能が必要のないシステムにも Oracle を採用する例もあります。

このように、

・サポートが必要
・インストールマニュアルが日本語
・トレーニングコースが充実している
・手間を減らすためにお金を払った方がいい

などなど、商用ソフトを採用する事情はあります。

リスク

OSS は無保証です。利用者の自己責任で使うことになります。

お金を払っていないためサポートが保証されません。

マニュアルやメッセージが英語だったりします。

製品の機能・仕様がわかりにくいことがあります。売っている人がいない、ということは親切に教えてくれる人が少ない、ということでもあります。

このように商用ソフトと OSS の間を埋めるようなサービスが立ち上がる余地があります。 これが OSS を取り巻くビジネスです。

OSS 派生のビジネスモデル

OSS は自己責任で利用するソフトウェアです。ユーザが製品を発見して、ソースを入手して、コンパイル、実行、学習、運用、トラブル対策して、という作業をすることになります。うまくいったりうまくいかなかったり、するうちにうまくいくケースをまとめておきたくなるでしょう。そしてまとめたものはパッケージや手順書に発展していきます。

ビルド済みの実行ファイルの形式で配布しているサイトもありますし、サイトやマニュアルを日本語化している人たちもいます。このようなサイトを利用すれば、すべてのユーザが面倒な作業を経験する必要はありませ

ん。

ここではOSSから派生したビジネスについて考察します。どのようなビジネスモデルがあるか見ていきましょう。

配布する

FSF（フリー・ソフトウェア・ファウンデーション）やRedHat

商用ライセンスを用意する

MySQL、PHP

サポートする

技術サポート、コールセンター、Cygnux Solutions

サービスを販売する、サービスに付加価値をつける

レンタルサーバ（redmine）、GitHub

使い方を教える

コンサルティング

顧客のシステムに採用する

システムインテグレーション、OSSをベースにした業務用システム開発

このように商用ソフトとOSSの間を埋めるようなサービスが立ち上がる余地があります。OSSに詳しくなると、ただ使うだけでなくビジネスにすることができるのです。

フリーであるため使えばいいので、自分の努力と熱意で詳しくなっていくことができます。開発会社ならば生産性が高まるとか、ユーザをサポートする（コンサルティング、セットアップ、技術サポートなど）ことで対価を得られるとか、個人であれば就職・転職に有利になるとかが考えられます。

演習 4

OSS 製品をセットアップする

４つの製品を使えるようにします。仮想マシンに OS からインストールします。

検証方法（動作確認）

ソフトウェアごとに動作確認の方法が異なります。

Apache

ブラウザでサーバアドレスを開く。同じマシンでは 127.0.0.1(localhost) でブラウザに何か表示されることを確認する。ドキュメントルートを探して、index.html を修正したり、index2.html（ファイル名は何でもいい）を作成して表示してみる。

ipconfig でサーバとクライアントの IP アドレスを調べる。別のマシンで index2.html（ファイル名は何でもいい）を表示してみる。

Apache と PHP

phpinfo を表示する

MySQL

コンソールでコマンドを実行。（別紙参照）

WordPress

ブログを書き込みしてみる

作業の進め方

インストールの手順は簡単な場合もあります。しかしメッセージをよく読まないで「次へ」ボタンや「OK」ボタンを押していくのは感心しません。次の３つを心がけてください。

1）注意深くやること

2）英語のメッセージを無視しないで、その意味を理解しようと努力すること

3）手順をメモしながら、操作手順を再現できるようにしておくこと

グループ演習の進め方

グループでやることで、うまくいくケース、失敗するケースを多く体験できます。自分だったらそうやらないようなことを他人はやってくれますので経験値がぐっと上がります。最終的には個人個人でセットアップできるようになりましょう。

グループメンバー自己紹介と役割分担

はじめに自己紹介をしましょう。次に役割を決めます。

みんなが同じ立場ですと、議論がしにくいかもしれませんので、役割を決めましょう。リーダー、サブリーダー、発表者、メモ係、調査係、兼務も OK です。時間が許せば、グループ活動の成果はみんなの前で発表してもらいます。

方針を立てる

一口に Linux といっても様々なディストリビューション（パッケージ）があり、また同じディストリビューションであってもバージョンがいくつかあります。利用許諾書（ライセンス）を確認することも忘れないでください。

数多くあるソフトの中から、どの OS のどのバージョンをインストールするか、ターゲットを決めてください。ターゲットは作業の途中で変えても構いません。決めたとき、変えたときの理由をメモしておきましょう。**「なぜ？」をメモする習慣をつけましょう。**

1）まずグループで方針を立てましょう。どのソフトをセットアップするかの理由をみんなで決めましょう。たとえばこんな具合です。

□　今回は企業向けに安定性の高い Linux と技術的に枯れたミドルウェアでいく。

□　旧バージョンには不満だったので新しいバージョンを入れてみよう。

□　とにかく軽いバージョンにしよう

□　目的のアプリケーションを動かすのには、この Linux が最適だ。

□　たまたまサイトにインストール方法が載っていた。

□　日本語のサイトがわかりやすかった。

□　ネットで関連情報が多いからこれでいいのでは？

□　その他（

2）次に Apache、PHP、MySQL のバージョンを決めます。ここでも 決めた理由を明らかにして、利用許諾書（ライセンス）を確認してください。

3）上記アプリをサポートする OS を選びます。いくつか候補がある場合はどれに決めたか過程をメモします。

4）グループ全員でひとつのバージョンをインストールしてもかまいませんが、できれば異なるパッケージやバージョンをインストールしてみて、違いを見るのもいいでしょう。

このほか、VMWare Player との相性を考慮に入れておくと OS などを決める作業が速いかもしれません。

演習

演習 1　slack アカウントとチャネル

グループ発表

演習 2　OSS 製品を調査する

グループ発表

演習 3　git

git ローカルで使う

GitHub にアカウントを作成、チームを組んで、リポジトリを共有

利用方法をマニュアルにまとめる

発表

演習 4　oss 製品をセットアップ

プランを作成　方針を立てる

プランファイルを GitHub で共有（これで講師がチェックできる）

役割分担

作業

結果を随時プランに書き込んでいく

レポートを作成、GitHub 上で 1 つのファイルに纏める

附録

仮想マシン

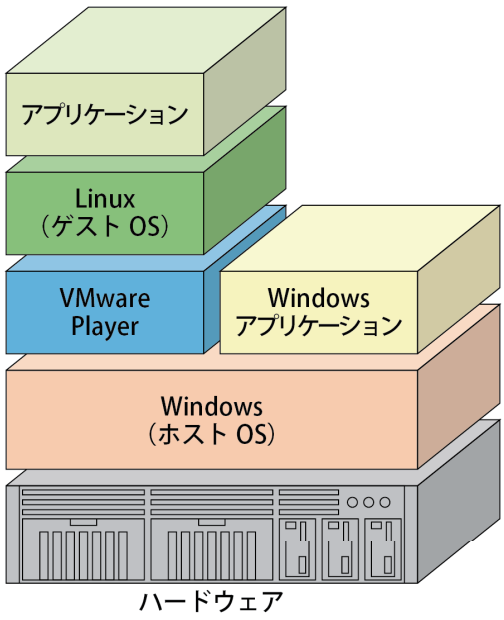
授業では、仮想マシン上に、Web アプリケーションの開発環境を実装します。仮想マシンを作るためのソフトはいくつかあります。大学のマシンにインストールされているのは VMWare Player です。では VMWare とは何でしょうか。

VMWare とは？

Windows、Mac OS X、Linux で動作する仮想マシン作成ソフト。製造元の、VMware 社 (VMware Inc.、本社カリフォルニア州) は 1998 年に設立され、2004 年 1 月に EMC コーポレーションによって買収された。2007 年 8 月にニューヨーク証券取引所で株式公開した。2003 年には日本法人であるヴイエムウェア株式会社 (VMware K.K.) が設立された。

VMWare Player は同社の無償製品で、ホスト OS 上で、仮想マシン環境を構築して、ゲスト OS をインストールできるようにします。ゲスト OS 上ではサービスやアプリケーションを動かすことができます。

仮想マシンに与えるネットワークアドレスの割り当て方がいくつかあります。これをブリッジに変更します。

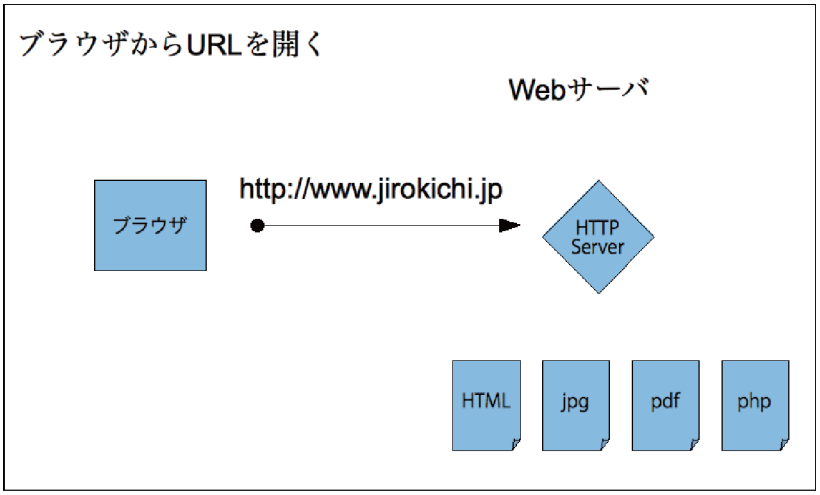


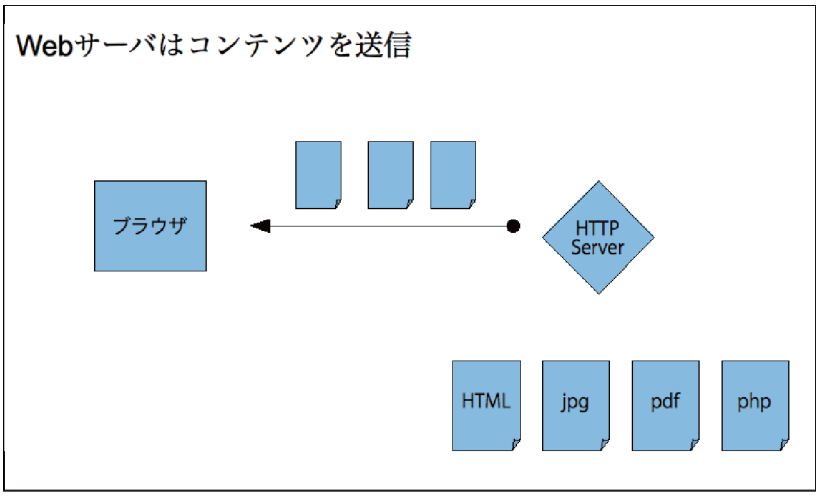
Web アプリケーションとは？

Web アプリケーションとは、ブラウザのウインドウ内で動くアプリケーションです。HTTP プロトコルを使って Web サーバ（HTTP Server）にアクセスし、データベースやファイルから必要な情報を取得し、HTML 形式にしてブラウザに表示します。

Apache

このような Web サーバを使ったアプリケーションを機能させるための Web サーバソフトが Apache です。





Apache HTTP Server とは、サーバ IP アドレスのポート番号８０番にやってくる TCP/IP リクエストを受信して、主に HTML ファイルを返信するサーバソフトウェア（サービス、ミドルウェアとも呼ばれる）です。

通常はドキュメントルートの HTML ファイルを返すが、URL でファイル名が省略されたときのデフォルトのファイル名を決めておくことができる。設定は conf ファイルに記述する。

起動と終了、確認方法

PHP

Web サーバに PHP ファイルを要求すると、PHP ファイルに記述されたプログラムが実行される。これが CGI（Common Gateway Interface）と呼ばれる仕組み。Perl, Ruby, Python, Java, もちろんＣで記述しても良い。

Apache の conf ファイルに CGI の呼び出し方法を指定しておくことで、拡張子が .php のファイルを要求されたときにプログラムが実行されるようになる。実行された PHP プログラムは、HTML ファイルの中身をロードしたり、HTML の特定のタグをプログラムで置き換えてクライアントに返すことができる。

データベースアクセスは、PHP の場合、pgconnect コマンドを使う。

MarierDB, MySQL

確認方法

・コマンドライン

・phpMyAdmin

WordPress

ファイル群を所定のディレクトリーに置くだけで動作します。ブログ作成ツールですので、ブログを追加したりサイトを編集してみましょう。

IP アドレスとドメイン名

IP アドレスとはホストコンピュータを識別するための番号です。広くインターネット上でユニークなグローバルアドレスと、ひとつの LAN 内で使うためのプライベートアドレス、自分自身のアドレスがあります。有効な番号を示すためサブネットマスクを指定します。

アドレスは４つの８ビット値がドットで繋がった形式です。0.0.0.0 - 255.255.255.255 の値があります。１つの８ビット値で 255 個のホストアドレスを割り当てることができます。

プライベートアドレス

プライベートな所内 LAN 上で、パソコンやプリンタに割り振られているアドレスです。多くの場合、パソコ

ンは起動時にこのアドレスを、ルータなどの DHCP（Dynamic Host Configuration Protocol）サーバから取得します。手入力で固定のプライベートアドレスを設定しておくこともできます。

所内 LAN 上のマシンの数によって、使えるプライベートアドレス空間が 3 種類あります。クラス A（10.0.0.0 - 10.255.255.255）、クラス B（172.16.0.0 ～ 172.31.255.255）、クラス C（192.168.0.0 ～ 192.168.255.255）です。台数が多い LAN ではクラス A、少ない LAN ではクラス C が使われます。

グローバルアドレス

世界中で唯一のアドレスです。202.11.16.169 とか、157.112.145.14、27.96.xx.xx のような形です。インターネットアドレスとなります。すべてのユーザはグローバルアドレスを持っていないとインターネットにアクセスできません。通常はドメイン名でアクセスされますので、ユーザが意識する場面は少ないですが、プロバイダの契約書に書かれていると思います。

自分自身のアドレス

127.0.0.1 です。localhost とはネットを見に行くかどうかが異なりますが同じ意味になります。

IPv4 では、自分のアドレスは「localhost（127.0.0.1）」になります。

ドメイン名

サーバコンピュータには IP アドレスは設定されているがドメイン名はついていない。ドメイン名は IP アドレスに変換されてはじめてそのサーバにアクセス可能になる。このためドメイン名を IP アドレスに変換する仕組みが必要になる。これを名前解決と呼ぶ。conf ファイルの設定にもよるが、クライアントコンピュータは、ローカルの「hosts」ファイルを使った名前解決を試み、そこで解決できない場合は DNS による名前解決を試みる。

Linux コマンド

Linux サーバの基本的な操作方法について説明します。ソフトウェアのインストールや設定に必要になります。

基本

pwd　カレントディレクトリ。print working directory の略。ワーキングディレクトリの表示。

cd　change directory。ディレクトリを移動する。カレントディレクトリを変更する。

　　cd /　ルートディレクトに移動。/（スラッシュ）はハードドライブのルートのこと。

　　cd /home　これでログインユーザのホームディレクトリに移動する。

ls　list segments　カレントディレクトリにあるディレクトリやファイルを一覧表示する。

　　ls -al　-a は全てのファイル、隠しファイルも表示する。-l オプションはファイルとディレクトリを

一つ 1 行で詳しく表示。パーミションもわかる。

cp　　コピー

rm　　リムーブ。削除

mv　　ムーブ、移動。

cat　　ファイルの中身を見る。

more　ファイルの中身を見る。

unzip　zip ファイルを解凍する。

su　　スイッチユーザ、ルートユーザに変更する。パスワードが求められる。

sudo　ルートユーザにスイッチ。このあとに続けて vi などと実行コマンドをタイプする。「sudo vi test.html」とすると、ルート権限で vi を実行して test.heml ファイルを作成、というコマンドになる。パスワードが求められる。

sudo　ルート権限で実行。パスワードが求められる。

chmod　ファイルやディレクトリのアクセス権を変更する。アクセス権は ls -l で確認できる。r: read(読み込み)、w:write（書き込み)、x:（実行）の権限を示している。

chown　ファイルやディレクトリの所有者を変更する。

UNIX コマンドは 1980 年代に広く知られるようになりました。このころからコマンドは変わっていません。これを機会に覚えましょう。

Linux の階層構造とパーミション

よくはまるところにファイルやディレクトリへのアクセス権（パーミション、permission）があります。最近では MacOS も Windows もセキュリティが厳しくなりましたので、知っている人も多いと思いますが、Linux にもアクセス権があります。これをターミナルのコマンドラインで確認したり変更することがよくあります。次のコマンドを実行するとファイルやディレクトリの所有者やアクセス権が表示されます。

```
ls -al
```

rwxrwxrwxs などと表示されますが、r は読む権限、w は書き込み権限、x は実行権限（r：read、w：write、x：execute）です。これらが 3 セット繰り返されていますが、最初のが所有者、次がグループメンバー、3 つめがゲストユーザに対するアクセス権を意味します。

root　Linux には root と呼ばれるユーザがいます。システムをインストールしたときの最初のユーザです。

たとえば MacOS では背景の OS である UNIX システムでターミナルを使おうとすると root ユーザが必要になりますが、こちらは「root はあとから有効にするユーザ」であり、システムに最初に登録するユーザは

MacOS 上の管理者です。su　　スイッチユーザの略。ログインユーザをルートユーザにスイッチするという意味になります。続くプロンプトでルートのパスワードが求められます。exit するまでルートユーザになっています。

sudo　続くコマンドをルートユーザで実行する、という意味になります。続くプロンプトでルートのパスワードが求められます。ログインユーザは元のままです。

vi を覚えよう

vi は古くからあるスクリーンエディタです。作者はビル・ジョイです。コマンドラインから実行できるためテキストファイルを編集したいときに便利です。が、GUI 環境に慣れたユーザにとっては驚きの操作環境ではあります。vi a.txt で「a.txt」を作成して編集、という意味になります。すでに「a.txt」が存在すればそれを編集モードで開きます。

終わりに

オープンソースソフトウェアを題材にソフトウェアについて学んできました。最後にオープンソースソフトウェアのプロジェクトを成功させた要因として、インターネットの果たした役割と UNIX の精神、そしてモジュール化についてを考察して締めくくりたいと思います。

インターネットとオープンソースソフトウェア

配布のコストと手間が劇的に下がりました。インターネット以前は、ソースが無料であっても手紙を出して送料を添えて、フロッピーディスクを送ってもらうなどの手間と時間がかかりました。今はネットでコミュニティのサイトを探してソースをダウンロードすることができます。思い立ったらすぐにダウンロードしてコンパイルして実行することができます。

開発者側もインターネットのおかげで仕事がやりやすくなっています。ソースの管理、利用者への配布、開発コミュニティの運営など、インターネットなしの状況は考えられなくなっています。結果としてここ１０年くらいでオープンソースプロジェクトは劇的に発展しました。インターネット環境が良くなったことが理由のひとつと考えられます。

インターネットがなくてはオープンソースプロジェクトは成立しなかったとってもいいでしょう。皆さんには当たり前のように存在するインターネット社会は、ほんの３０年前、私が新入社員の頃にはなかったのです。そこから将来を考えると、この先インターネット社会にはどんな発展を見ることができるか、今後も楽しみで

す。

オープンソースソフトウェアとスマートフォン

Android は Linux ベースです。オープンソースソフトウェア製品です。このほかにも Linux を組み込んだ電機製品が数多く存在します。世の中はオープンソースソフトウェアだらけになっています。

Apple は開発言語 Swift をオープンソースにしました。これでサーバ用途などへの移植が進んでいます。プロプライエタリなメーカーもオープンソースを採用しています。

ソースコードをモジュール化する

UNIX の世界では小さくすることが良いこと、とされています。機能やソースを小さな部品の集まりで実装していくことが奨励されています。ソフトウェアの内部構造も同じです。ソースファイルは機能ごとに分割され、さらに小さな機能ごとに関数に記述されます。

階層構造（レイヤ）も重要です。ハードウェアレイヤ、データベースレイヤ、ネットワークレイヤなどとアクセス対象によって統一された名前の関数群を記述していきます。これにより移植性に優れたソフトウェアになります。

成功したオープンソースソフトウェアの内部にはこうしたモジュール化の努力が積み重なっているようです。仕事は小さな単位に分割してから片付けていきましょう。


PDP-11/45
digital PDP-10
DIGITAL EQUIPMENT CORPORATION/MAYNARD, MASSACHUSETTS
PDP-11/40