

# 1. バージョン管理システム

## 1.1 バージョン管理システムとは

ファイルの変更履歴を効率的に管理することができるツールを「バージョン管理システム」と呼びます。バージョン管理システムを用いることで、誤って削除してしまったことを取り消して元のファイル群に戻すことなどが可能になります。また、複数人でファイルの編集作業などをするときにも、誤って他の人が行った変更を無かった状態に上書きしてしまうなどの事態を防ぐこともできます。

バージョン管理システムが最も利用されているのが、システム開発の現場です。システムは徐々に機能が実装されていくものであり、何人もの人による共同作業であることが多いため、上記バージョン管理システムの機能が最も有効に使われるわけです。現在のシステム開発は、バージョン管理システムなしで開発を行うことは考えられない状況となっています。

バージョン管理システムの主な機能

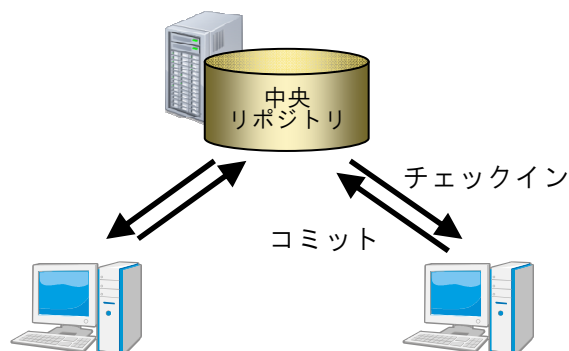
- ・ ファイル履歴（ファイルの作成日時、変更日時、変更箇所など）の管理
- ・ 複数人によるファイル編集における整合性の管理
- ・ 異なったバージョンの同時開発の管理 など

## 1.2 バージョン管理システムの種類

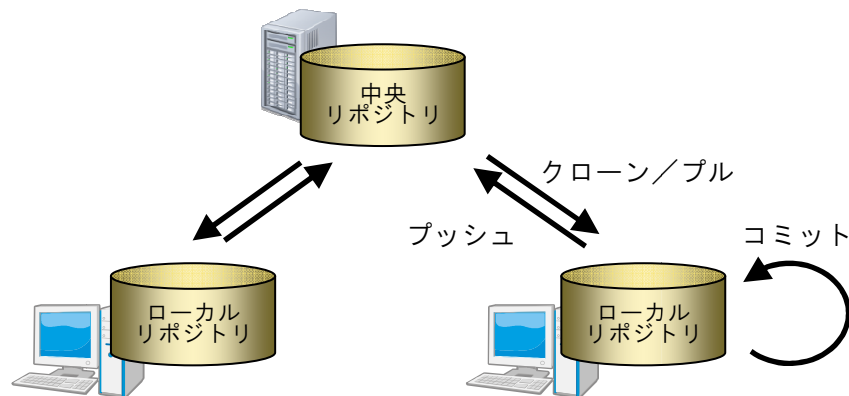
現在までに様々なバージョン管理システムが開発され、沢山の人々に利用されています。それぞれに特徴がありますが、大きく以下の部分で分類することができます。

バージョン管理システムの分類

- ・ 集中型 … リソース（ファイルや履歴情報）をシステムが集中管理（この部分を一般に中央リポジトリと呼ぶ）を行う方式。各利用者は中央リポジトリからリソースを取得（チェックアウト）、変更したリソースを中央リポジトリに登録（コミット）などを行う



- ・分散型 … リソースを、システム側（中央リポジトリ）だけではなく、各利用者側（ローカルリポジトリ）でも保持をする方式。中央リポジトリから複製を取得（クローン）してローカルリポジトリにリソースを置くことや、ローカルリポジトリ内だけで編集履歴を登録（コミット）ができる。



#### 代表的なバージョン管理システム

- ・CVS (Concurrent Versions System) … 集中型。テキストファイルの履歴管理
- ・Subversion … 集中型。CVS の改良版。Apache プロジェクトで管理されている
- ・Mercurial (マーキュリアル) … 分散型。Python による実装
- ・Git (ギット) … 分散型。Linux Torvalds によって開発が開始された

### 1.3 Git によるバージョン管理

Git は Linux OS の作者 Linux Torvalds 氏によって開発が行われ、Linux カーネル (OS の中枢部分) のソースコード管理に用いられている分散型バージョン管理システムです。

分散型であるため、各利用者（開発者）は自身の持つローカルリポジトリでの編集作業およびコミットを行うことができるため、複数人によるそれぞれ同時並行的なシステム開発を行うことが可能になります。またそれぞれの開発した編集内容を中央リポジトリに上げる（プッシュ）、中央リポジトリから編集内容を貰う（プル）などをすることで他の開発者への開発状況の提供が可能になります。さらには開発状況の共有をするマージ、異なったバージョンの並行開発を行うことのできるブランチなど、共同開発作業に適した様々な機能があります。

Git の各処理はコマンドラインから実行させるようになっています。また Linux 等の UNIX 環境での使用を前提としていますので文字コードは UTF-8、改行文字は LF が基本となっています。

## 1.4 Git における 3 つの領域

Git を使用するとき、以下の 3 つの領域をイメージしながら作業をする必要があります。

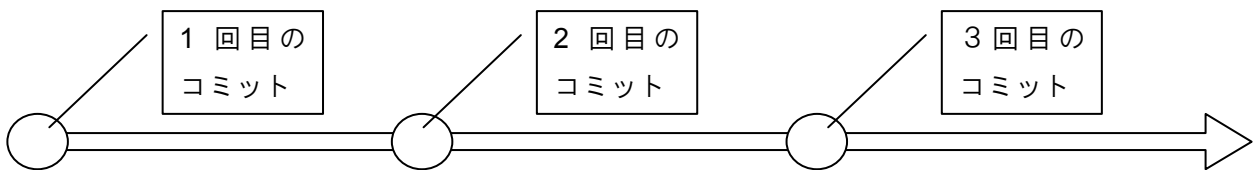
Git の 3 領域

- ・ワーキングツリー … 作業用フォルダ。編集するファイルが置かれる通常のフォルダ
- ・インデックス … 変更履歴としてリポジトリに登録するファイルを指定した情報
- ・リポジトリ … 変更履歴が登録される場所

インデックスに変更履歴を管理したいファイルの名前のみを指定することで、特定のファイルについて履歴管理から除外を行うなどといったことが可能になります。

## 1.5 コミット(Commit)

ファイルの編集状態を履歴として残す作業をコミットと呼びます。コミットを行う度に、前回コミットをしたときからの差分（ファイルの変更点）がリポジトリに記録されていきます。



コミットそれぞれには、自動的に英数字 40 桁の独自の名前（コミット ID）が付加されます。コミット ID でどのコミットであるか判別ができます。また、最新のコミットは、**HEAD** という特別な名称を利用して他のコミットと判別をすることもできます。

コミットするときには必ずコメント（コミットメッセージ）が必要になります。

## 1.6 リポジトリホスティングサービス GitHub

Git の共通リポジトリをホスティングしてくれる無料サービス（有料による機能もあり）として GitHub（ギットハブ）というものがあります。GitHub では Web ページでの操作や各 PC 用 GUI ツールが充実しているため、Git のコマンドを覚えなくても一定の作業ができるようになっています。

オープンソースソフトのソースコード公開や共同開発などで広く利用されています。

## 【演習：GitHub for Windows】

今回は、Git でのバージョン管理の基本を理解するために GitHub の Windows 用 GUI ツール「GitHub for Windows」を用いてのファイル公開と履歴管理の演習を行ってみましょう。

### 1. GitHub for Windows のインストール

- (1) GitHub のサイト (<https://github.com>) にアクセス
- (2) ページ下の Applications の所にある「GitHub for Windows」をクリック
- (3) 「download 1.0」のボタンをクリック
- (4) ダウンロードされた「GitHubSetup.exe」を実行
- (5) GitHub のアカウントを持っていない場合は、「SIGN UP」をクリックして登録する

### 2. ローカルの作業フォルダとファイルを作成する

- (1) ドキュメントのフォルダを開き、そこに「sample」というフォルダを新規作成
- (2) sample フォルダ内に「sampletext.txt」というファイルを新規作成
- (3) sampletext.txt の内容として以下の文章を記述して保存する

GitHub の練習用に作ったファイルです。  
うまく履歴管理できるでしょうか？

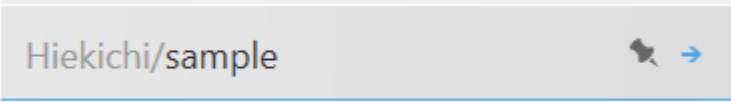
※ 必ず文字コードを「UTF-8」にして保存してください。TeraPad 等で「文字コードを指定して保存」ができます。

### 3. ローカルリポジトリに作業フォルダを登録する

- (1) 作成した sample フォルダを GitHub for Windows のウィンドウ上にドラッグ&ドロップ
- (2) 「new repository」作成画面になるので、「CREATE」ボタンをクリック

### 4. ローカルリポジトリにコミットを行う

- (1) Local の repositories として表示されている sample フォルダの水色矢印をクリック



- (2) uncommitted changes の COMMIT MESSAGE 欄に「初めてのコミット」と、EXTENDED DESCRIPTION 欄に「まだ書き始めたばかりの原稿です」と記入
- (3) 「COMMIT」ボタンをクリック
- (4) 右側の「初めてのコミット」と表示されている部分をクリック  
(3つのファイル「.gitattributes」「.gitignore」「sampletext.txt」が確認できます)

## 5. GitHub サーバのリポジトリに登録する

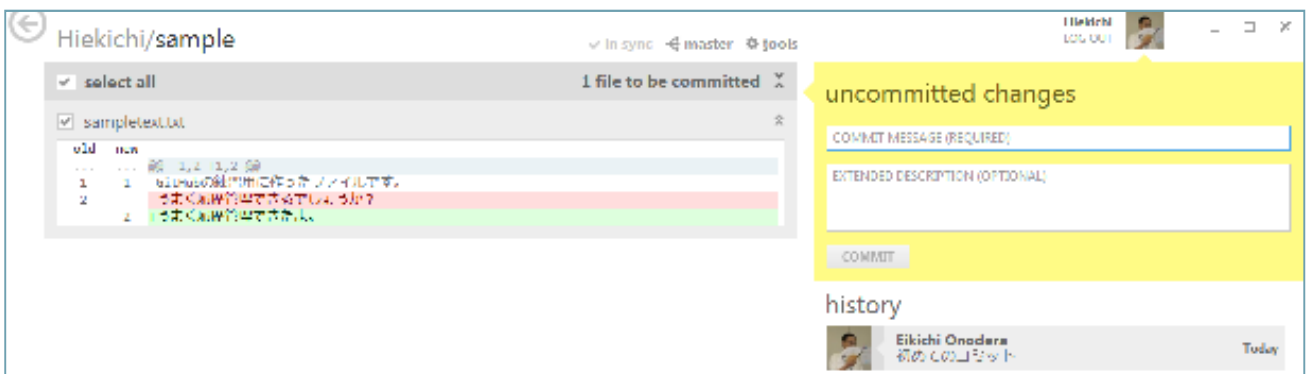
- (1) 上部中央にある「publish」ボタンをクリック  
(サーバへの登録が完了すると、この部分が「in sync」になります)
- (2) 左側上部にある「左向き矢印」をクリックして前画面に戻る

## 6. サーバ上のファイルを確認する

- (1) 左側にある「github」という表示の下にある自分のアカウント名をクリック
- (2) sample の「右向き水色矢印」をクリック
- (3) 「tools」をクリックし、「view on github」をクリック  
(Web ブラウザが起動し、自分の GitHub 上の sample リポジトリが確認できます)
- (4) 「sampletext.txt」をクリックし、内容を確認する
- (5) 「Files」をクリックし、ファイル名一覧に戻る

## 7. ローカルのファイルを編集し、ローカルコミットする

- (1) ドキュメントフォルダの sampletext.txt をエディタで開き、2 行目の『うまく履歴管理できるでしょうか?』を『うまく履歴管理できたよ。』に変更して保存、終了する
- (2) 「GitHub for Windows」アプリ上で「local」の「repositories」をクリック
- (3) sample の「右向き水色矢印」をクリック
- (4) 以下のように、左側に「どのように変更されたか」が表示され、右側に「前回のコミット後に変更があった」旨の表示がされているのを確認する



- (5) uncommitted changes の COMMIT MESSAGE 欄に「疑問形から断定に変更」と、EXTENDED DESCRIPTION 欄に「うまく履歴管理できないわけがない」と記入
- (6) 「COMMIT」ボタンをクリック

## 8. GitHub サーバ側に反映されているかの確認

- (1) Web ブラウザに表示されている GitHub のページで sampletext.txt をクリックして内容を確認する  
(変更が反映されていないはずです)

## 9. GitHub サーバのリポジトリにコミットを反映させる

- (1) 「GitHub for Windows」アプリ上で上部中央の「sync」ボタンをクリック
- (2) Web ブラウザに表示されている GitHub のページで sampletext.txt をクリックして内容を確認する  
(変更が反映されているはずです)

### 応用練習

- ・ sample フォルダに別のファイルを作成してコミットしてみましょう
- ・ コミットの履歴を確認してみましょう
- ・ コミット履歴を元に、以前コミットした状態に戻る方法を試してみましょう
- ・ 「GitHub for Windows」アプリの様々な機能を試してみましょう
- ・ GitHub の Web ページで行える様々な機能を試してみましょう
- ・ GitHub の Web ページで色々な人が公開しているソースファイルを確認してみましょう  
また「Clone in Windows」のボタンを使ってソースファイルをダウンロードしてぜひ読んで勉強してみましょう
- ・ GitHub の「Pull Request」とは何か、調べてみましょう

## 課題 1 (GitHub でリポジトリ作成 1)

C 言語や Web プログラミングの授業で作成した課題のフォルダを一つ選んで、ローカルの作業フォルダとして指定し、「ローカルリポジトリへのコミット」「GitHub への sync(push)」を行いなさい。

### [注意点]

- ・ 最初のコミットの COMMIT MESSAGE は「応用プログラミング A の課題 1」とする
- ・ ファイルに修正を加えながら 5 回以上コミットすること

**提出方法：**担当の先生にメールで、課題の URL (GitHub リポジトリのページ) を伝える