ソースコード管理入門

内容

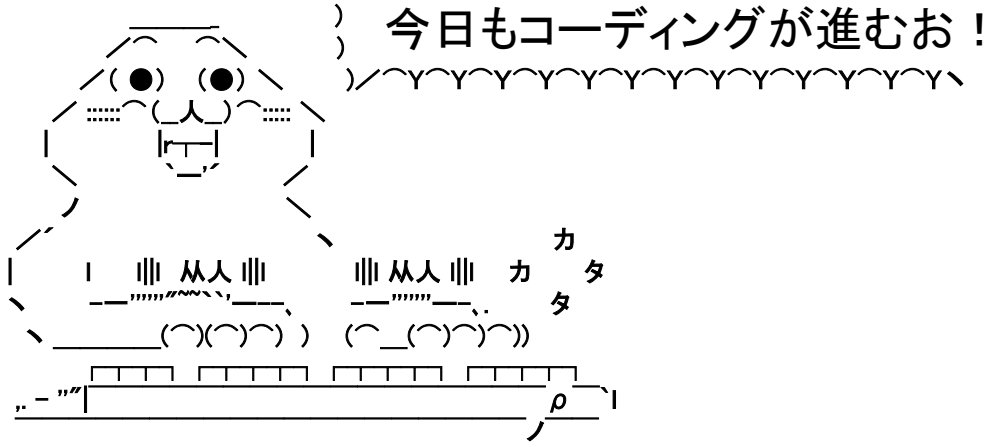
- ソースコード管理とは？
  - ソースコード・バージョン管理システム
  - 何故必要なのか？
  - バージョン管理システムのいろいろ
    - RCS、CVS、SVN、etc...
- 実際に使ってみる
  - RCSを使う
  - SVNを使う

ソースコード管理とは？

- ソースコードとその変更履歴を管理するシステム
- 多人数での開発を補助するためのシステム

必要性のあらまし

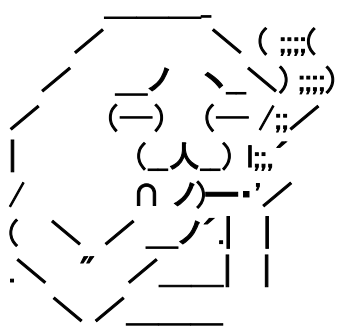
- ある日のやる夫・・・・



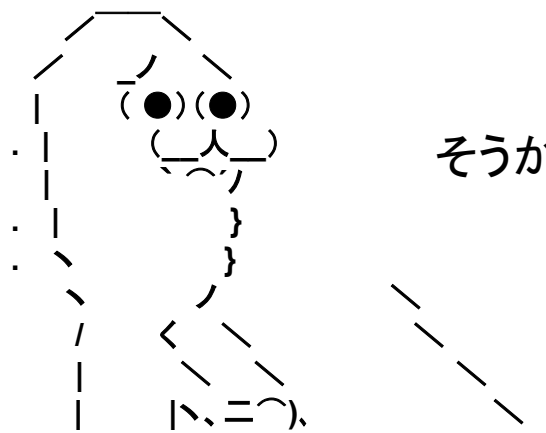
やる夫、悪いが使うライブラリの仕様が変わった
新しい奴に直してくれないか？

必要性のあらまし

- ある日のやる夫・・・・



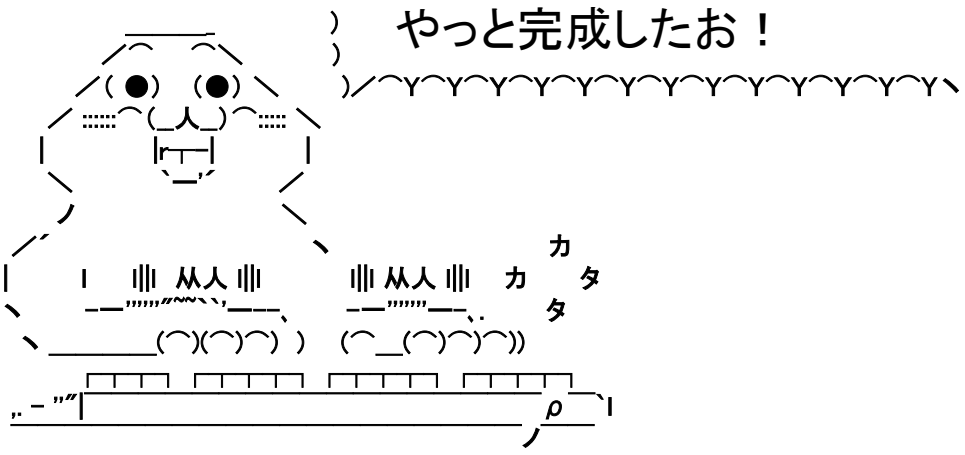
んー、ちっと面倒だお
３日ぐらいかかるかも・・・・



そうか、まぁそれで頼む

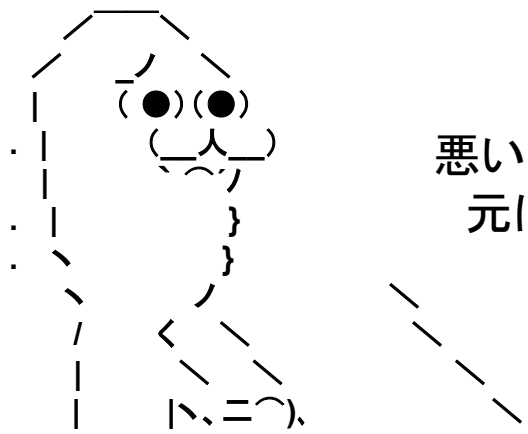
必要性のあらまし

- 三日後・・・・



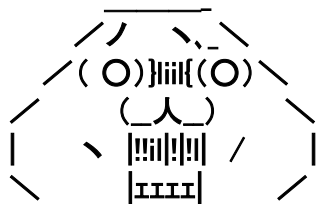
よし、これで今日は帰るお！

必要性のあらまし



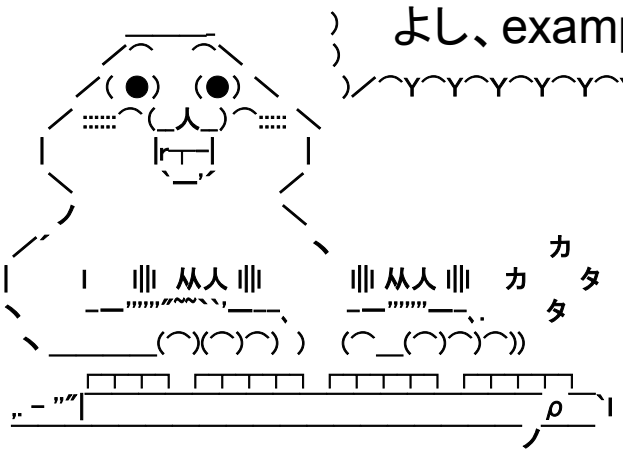
悪い、あのライブラリに欠陥が見つかった
元に直してくれないか？

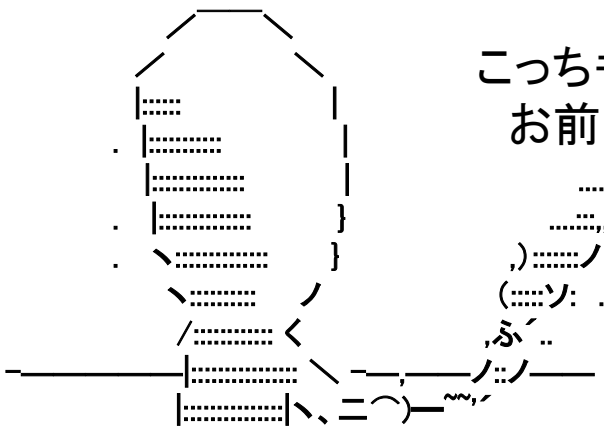
いまさら言われても
もう書き換えちゃったお！！



・・・などと、「少し前の状態」に戻さないといけないとき

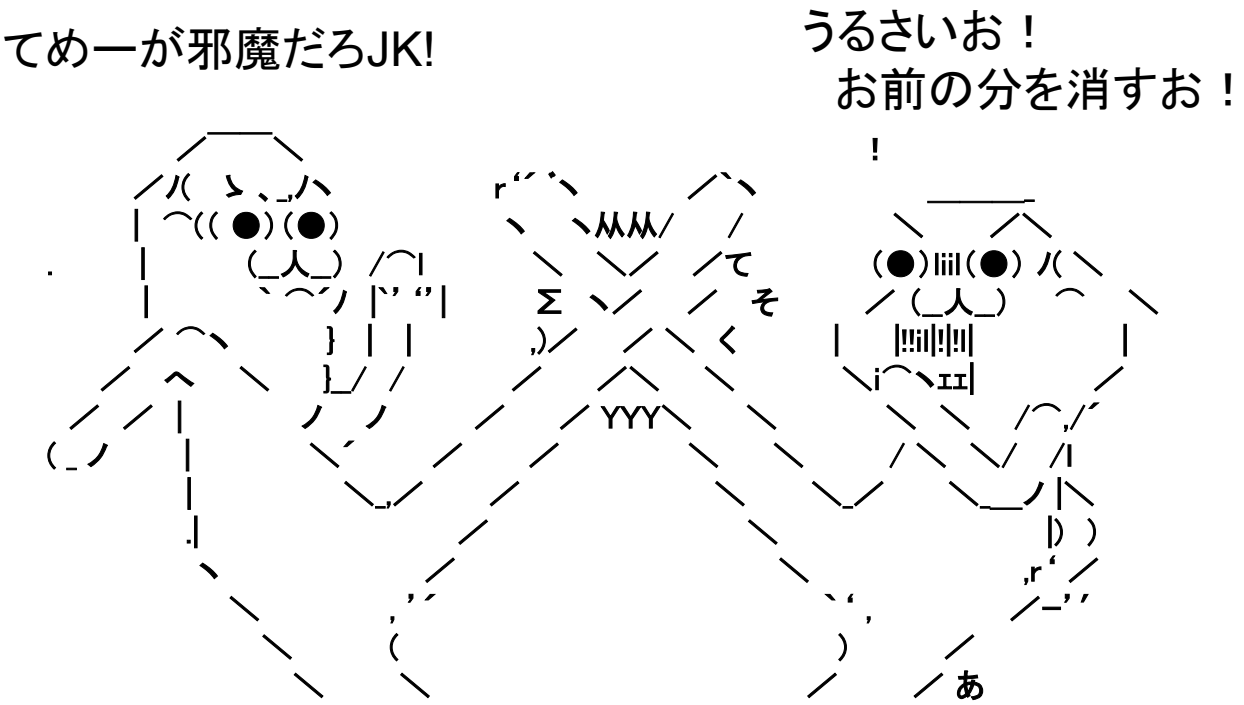
- 多人数で開発をしていると・・・・





こっちも example.c を書き換えた。
お前の分、邪魔

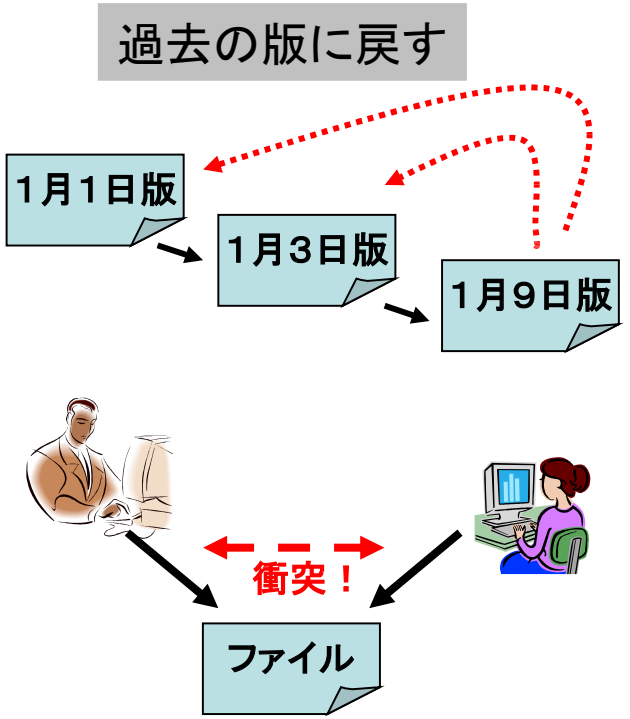
- 多人数で開発をしていると・・・・



・・・・・などと、「お互いの更新が被る」ときに

というわけで・・・

- いくらでも過去の状態に戻せるように
- 多人数で開発を行うときに修正がかち合わないように
  - かちあうことを衝突（コンフリクト）と言う
  - 衝突があったことを検知する、調整と修正を促す
- それらを実現するのが
  バージョン管理システム



バージョン管理システムの動作

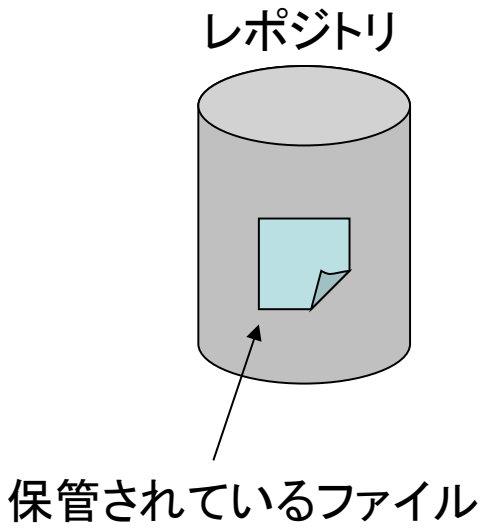
- ソースはレポジトリというデータベースで保管される

- 作業する場合は、
  1. レポジトリからファイルを取り出す
  2. 編集する
  3. レポジトリに反映させる

  という手順を踏む

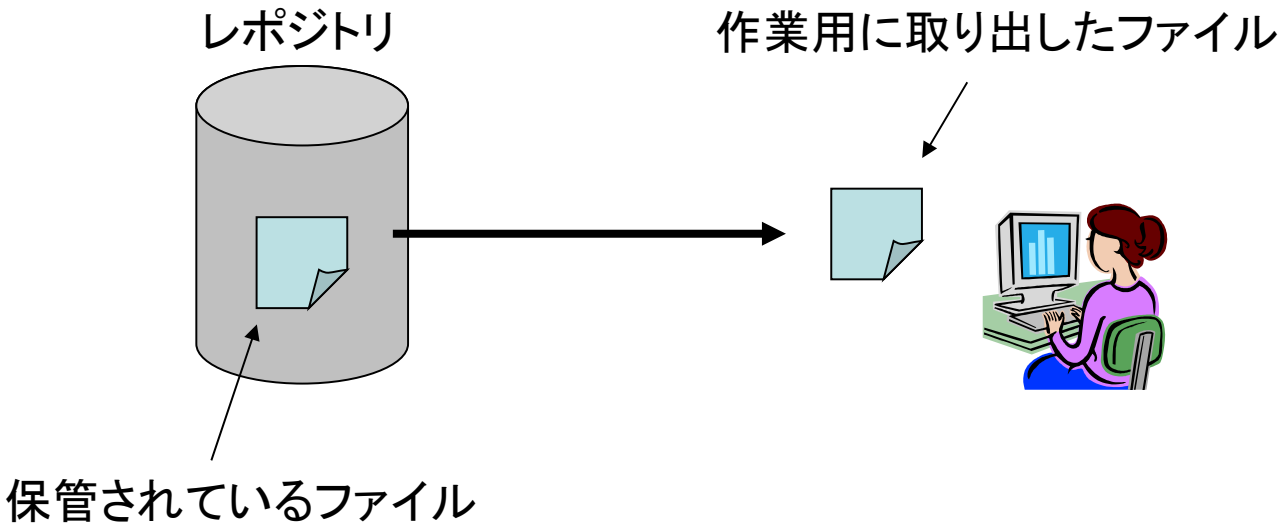
バージョン管理システムの動作

- ソースコードはレポジトリというデータベースで保管される



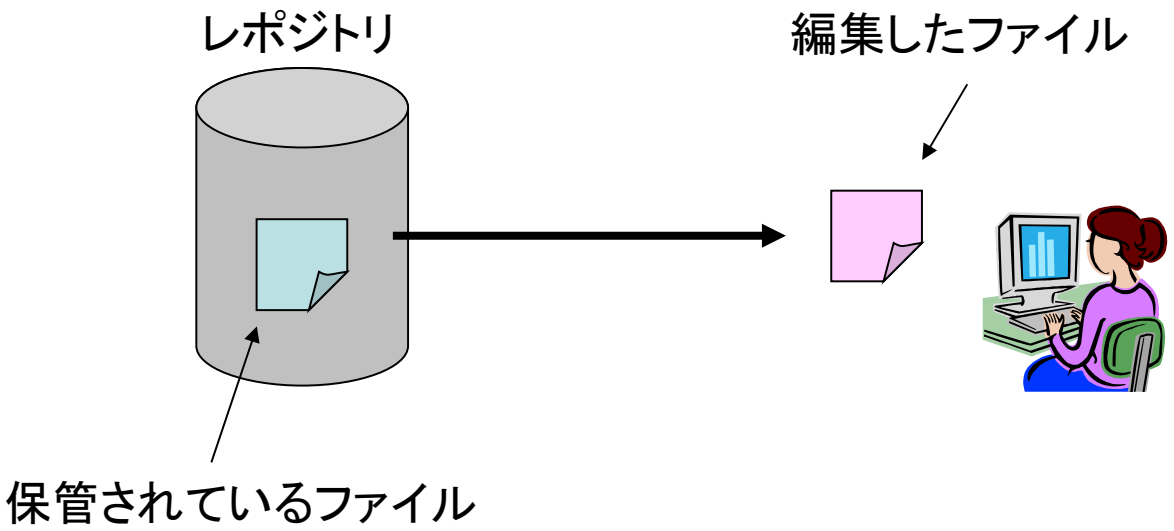
バージョン管理システムの動作

- 編集作業する場合は、レポジトリからファイルを取り出す（チェックアウト）



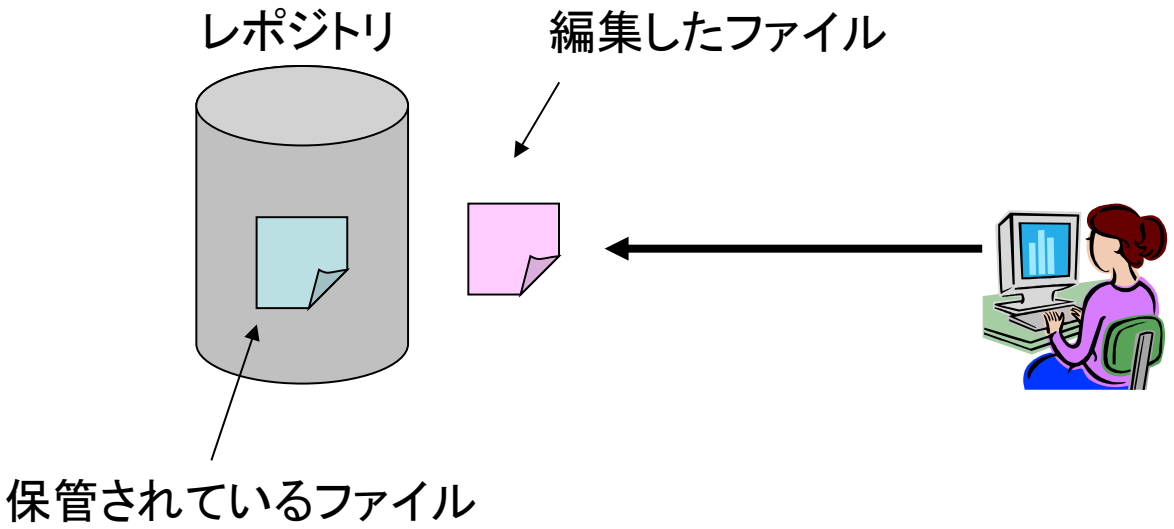
バージョン管理システムの動作

- 取り出したファイルを編集する



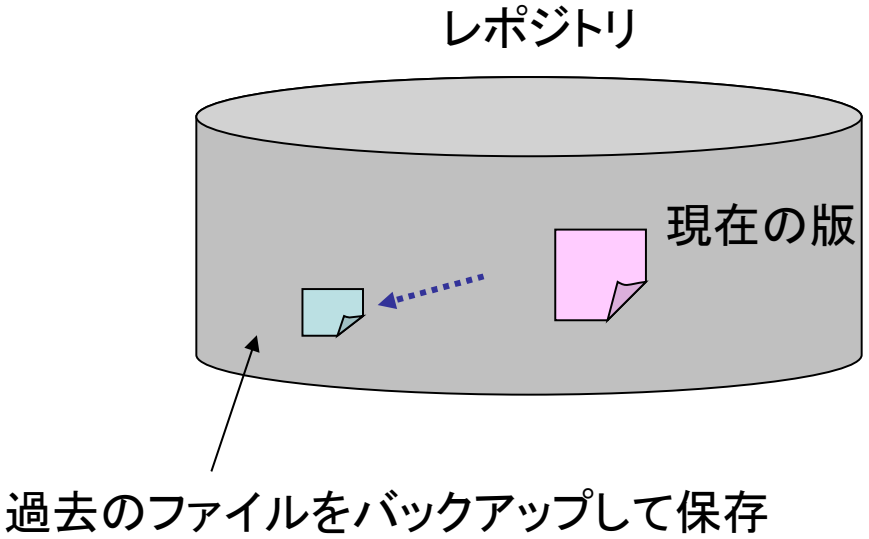
バージョン管理システムの動作

- 編集したファイルをレポジトリに送る（コミット）



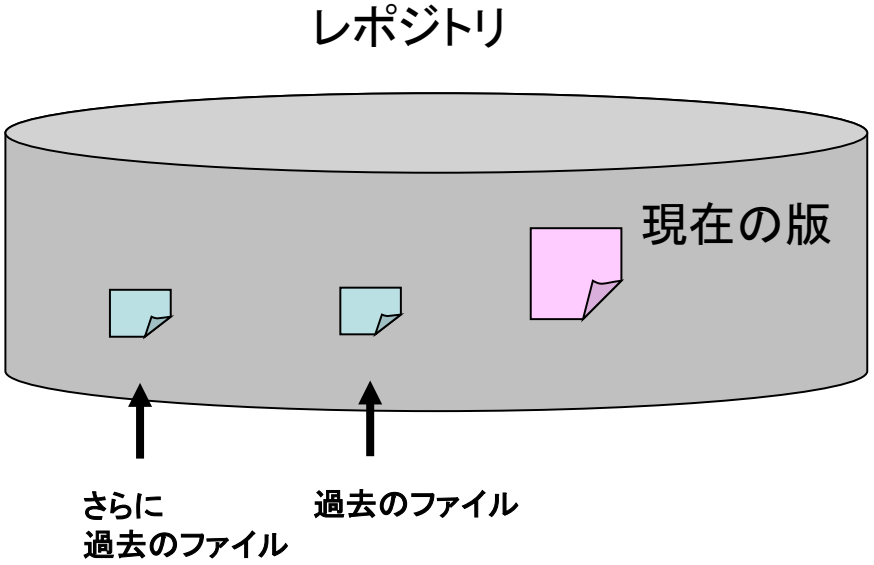
バージョン管理システムの動作

- レポジトリは新しい版を現在の版として保存
- 今までの版を過去の版としてバックアップ
  - 実際は差分の形で保存されるので、それほどサイズは大きくならない



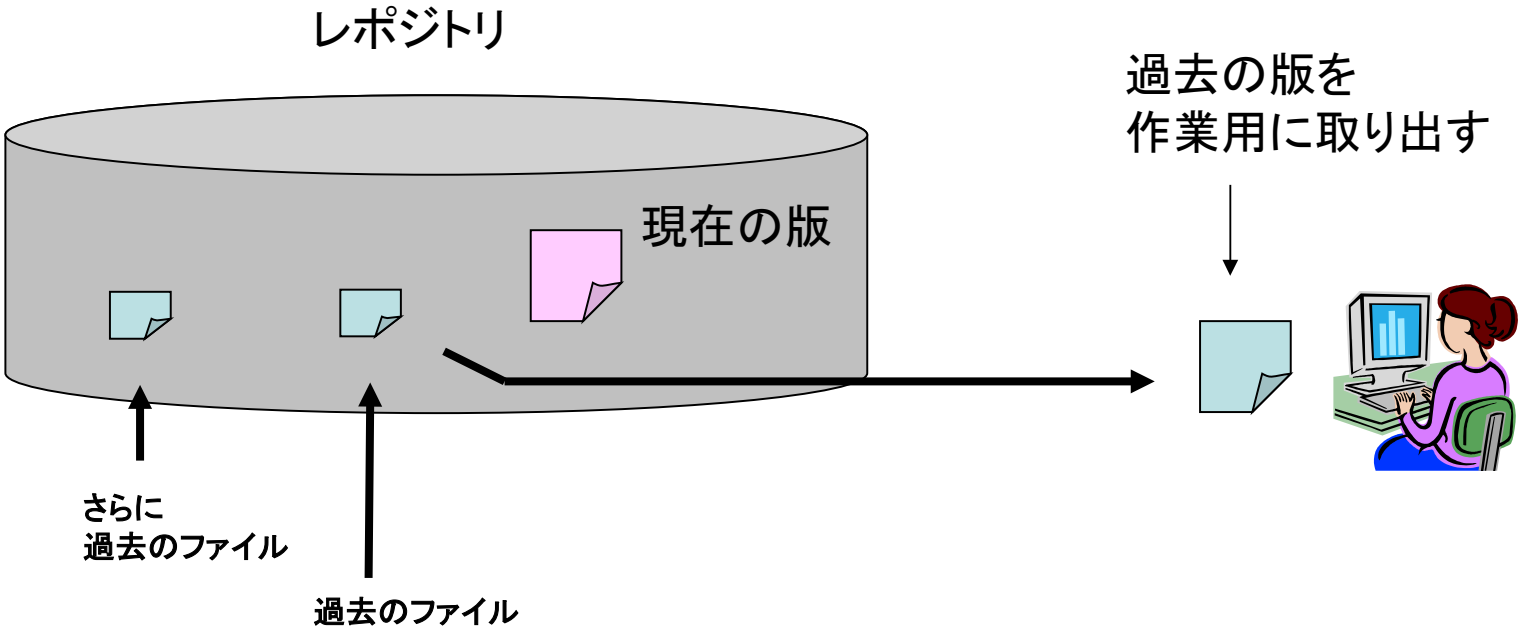
バージョン管理システムの動作

- このようにして、どんどん版を増やしながら作業を続けていく



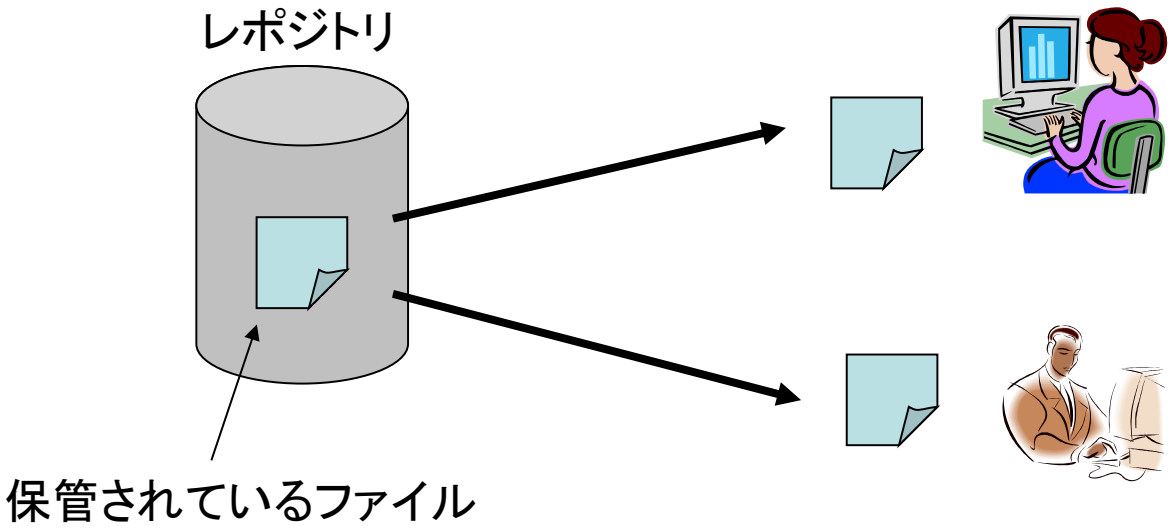
過去に戻したい場合は？

- 最初の場合のように、過去のファイルに戻したい場合は・・・
- 版を指定して取り出せる



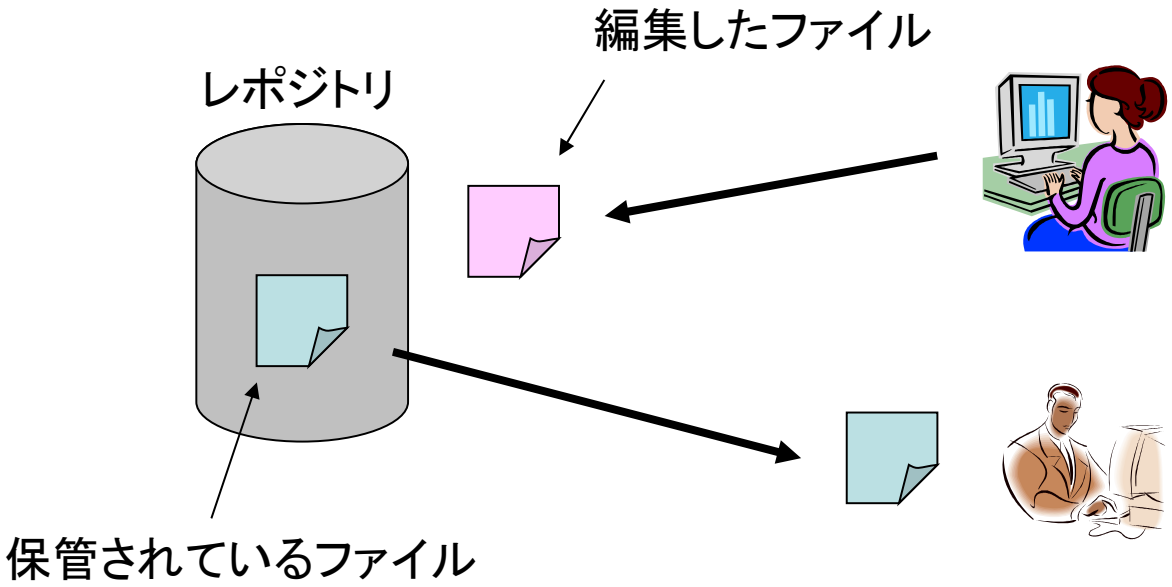
同時に編集した場合は？

- 二人が同時にファイルを取り出して編集した場合・・・



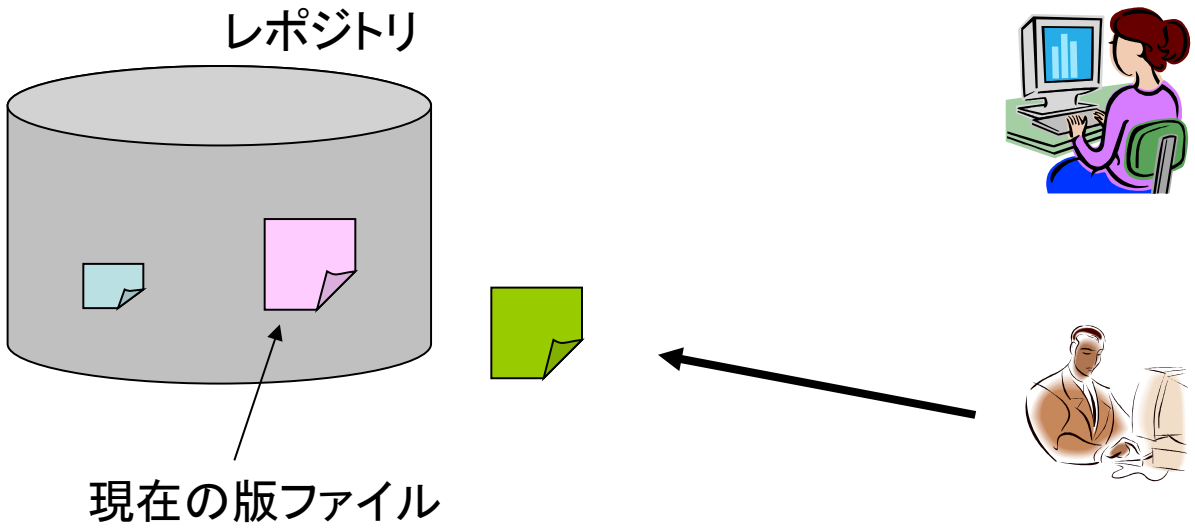
同時に編集した場合は？

- 二人が同時にファイルを取り出して編集した場合・・・
- 先にコミットした方は受理される



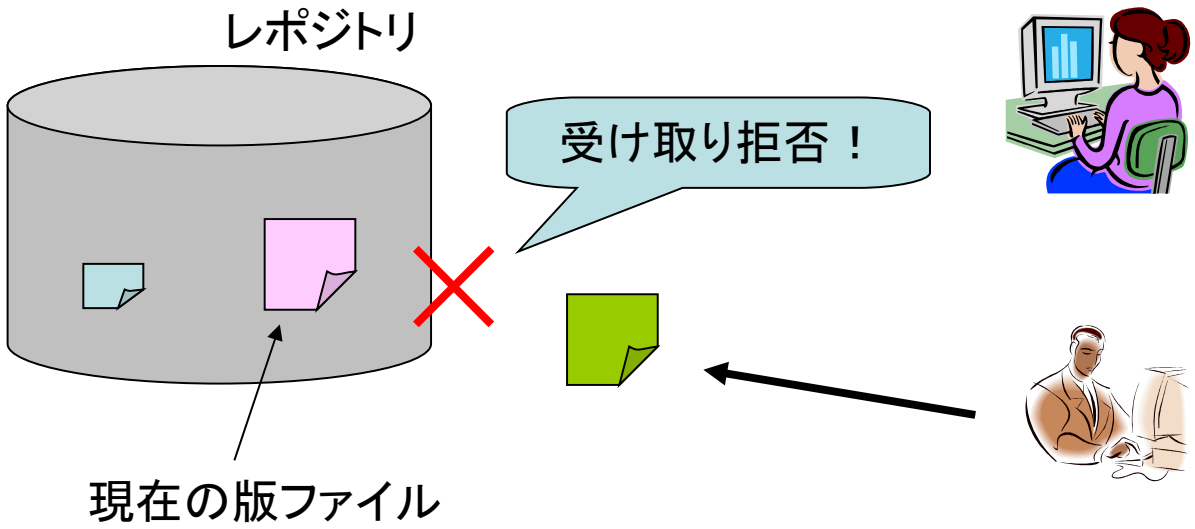
同時に編集した場合は？

- その後にコミットしようとすると・・・・



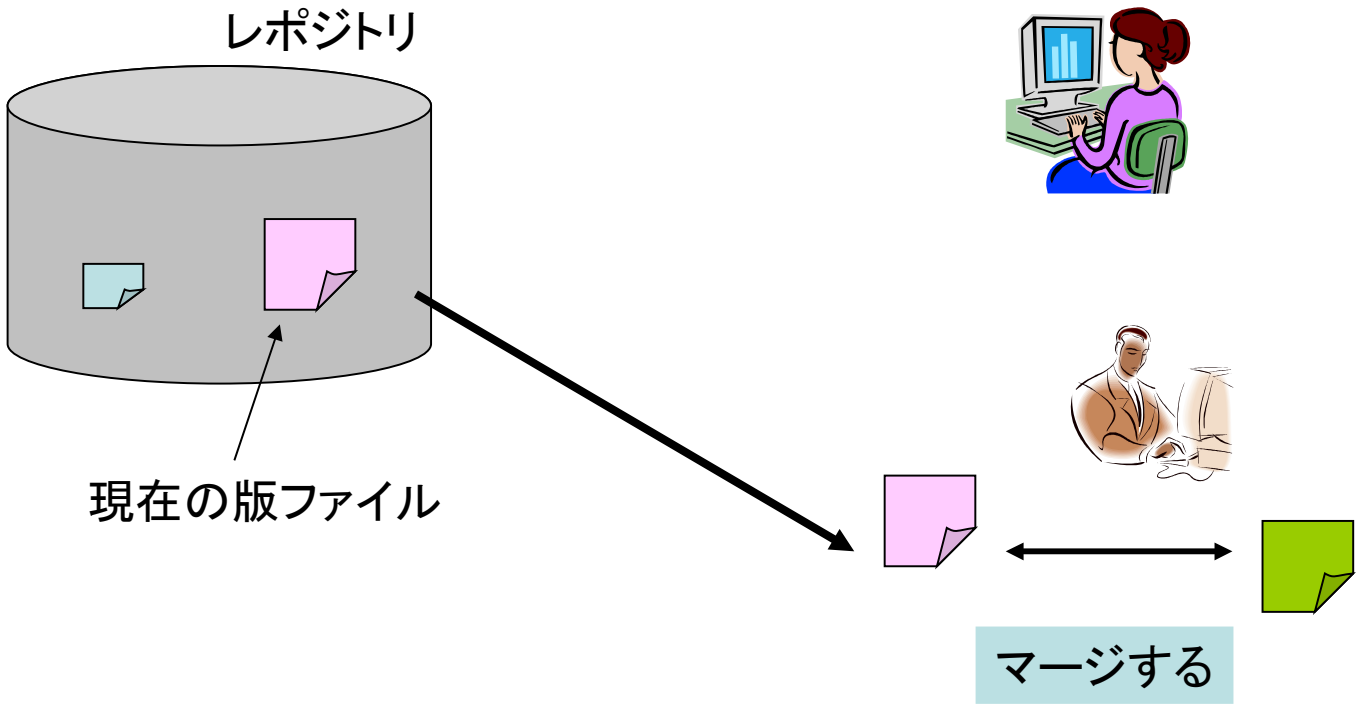
同時に編集した場合は？

- その後にコミットしようとすると・・・・
- 下の作業者が取り出した版が現在の版と違うので、拒否される



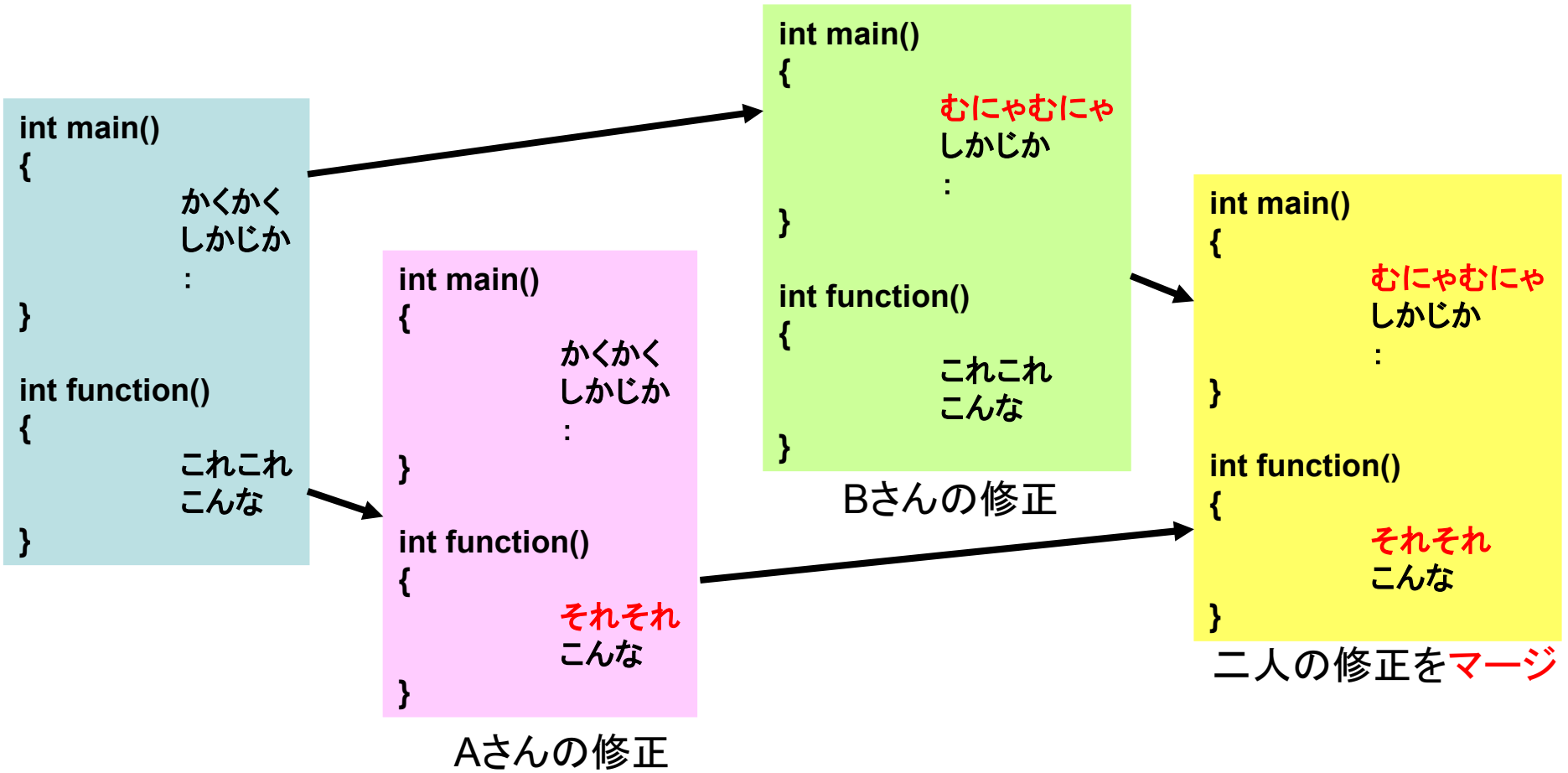
同時に編集した場合は？

- その場合、最新の版を取り出して、マージをおこなう



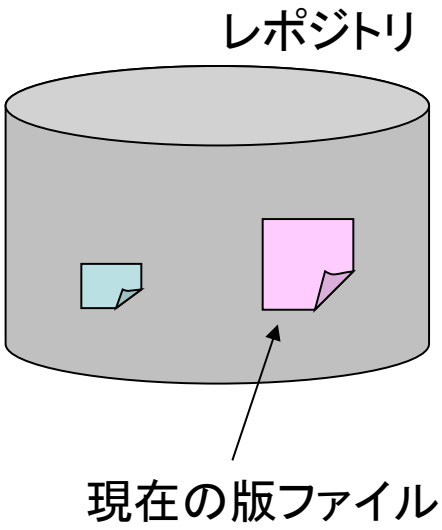
マージ

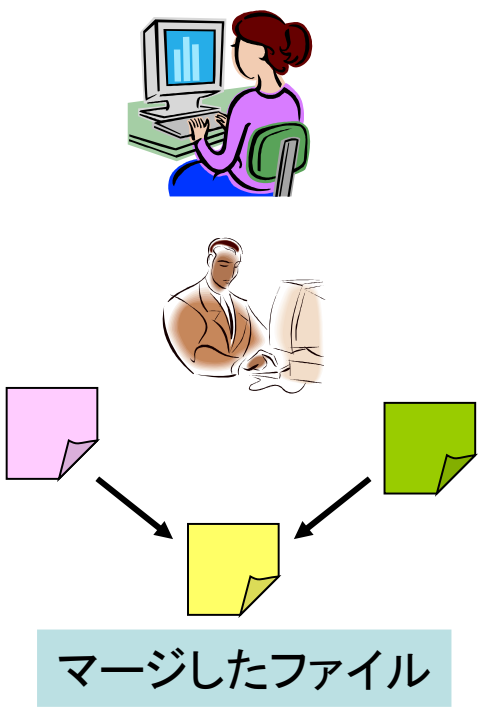
- それぞれの変更箇所をまとめることをマージという
  - それぞれが別の箇所を修正していれば自動的にマージしてくれる



同時に編集した場合は？

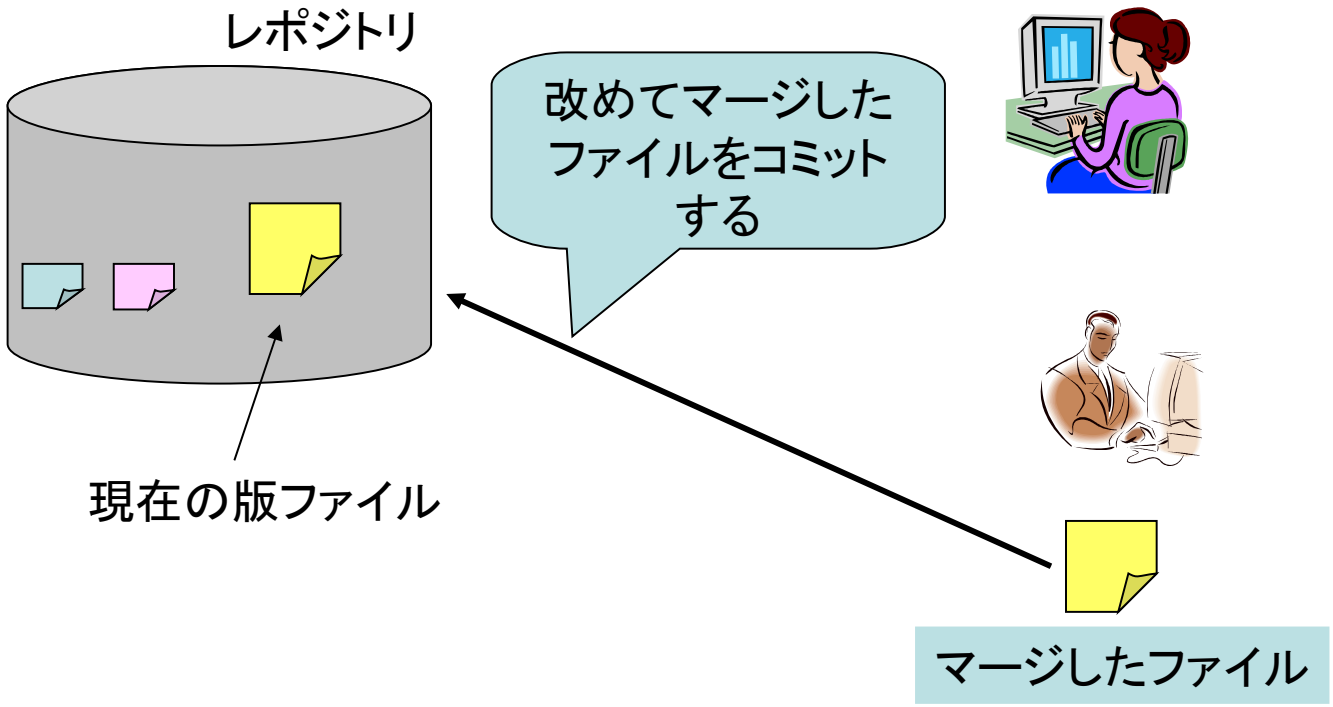
- マージをおこなったあとに、改めてコミットする





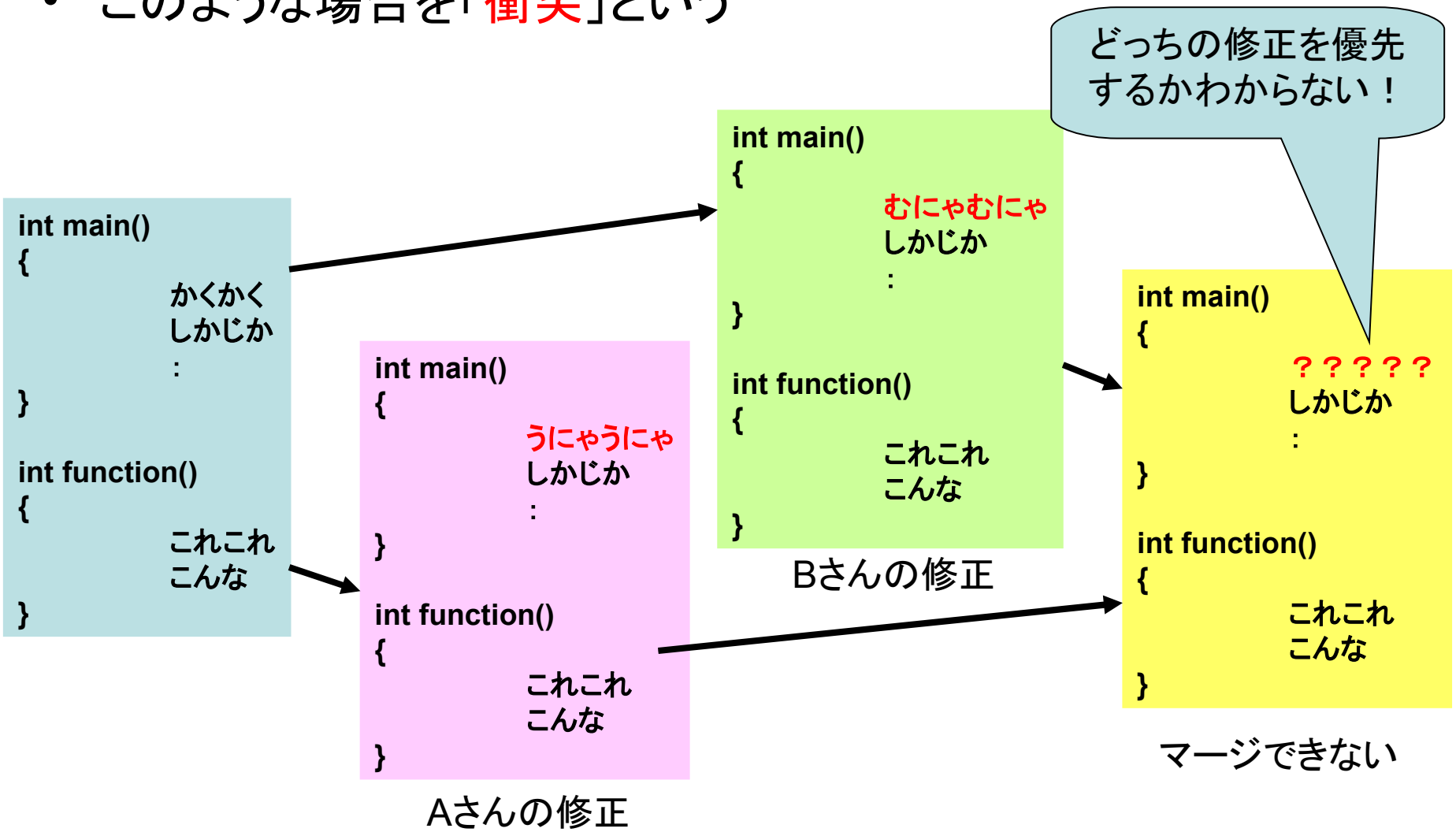
同時に編集した場合は？

- マージをおこなったあとに、改めてコミットする



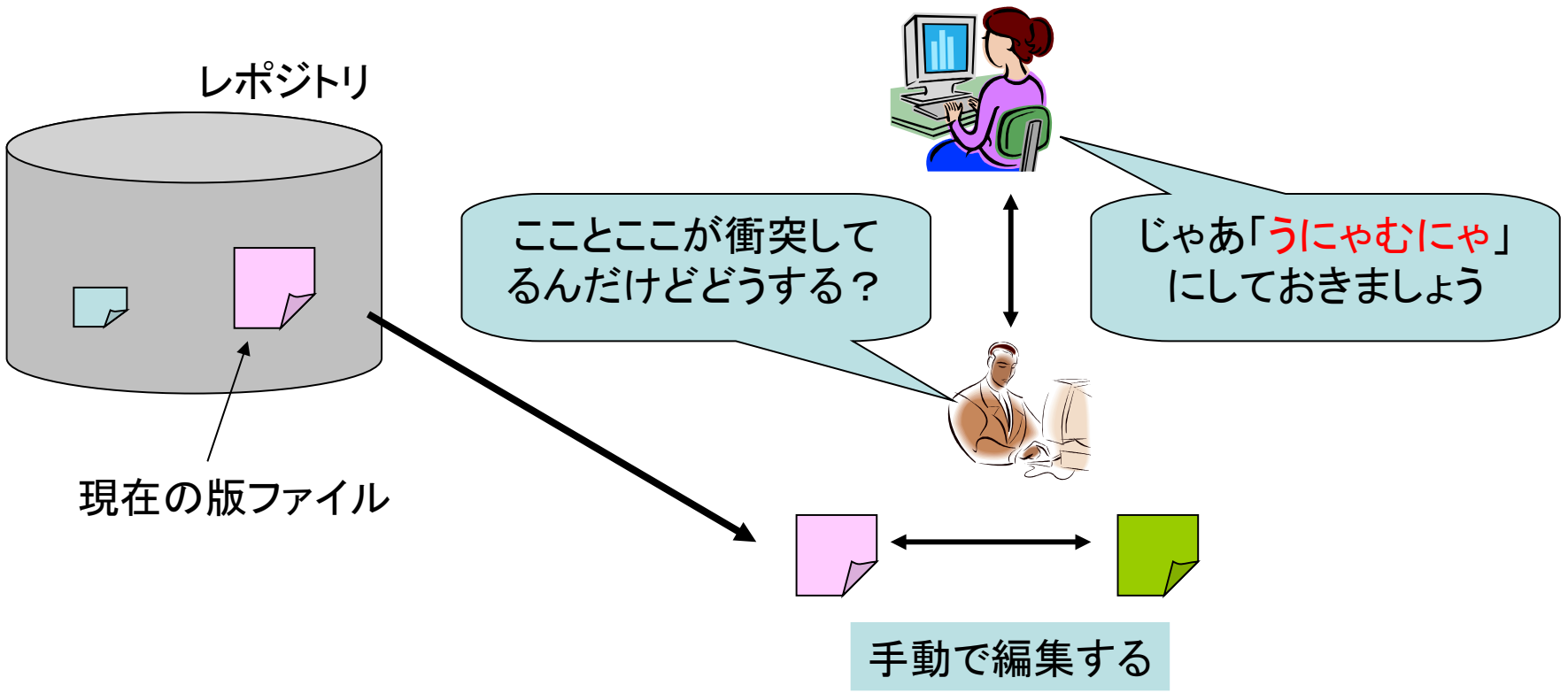
自動的にマージできない場合：衝突

- 同じ箇所を変更していた場合、自動的にマージできない
- このような場合を「衝突」という



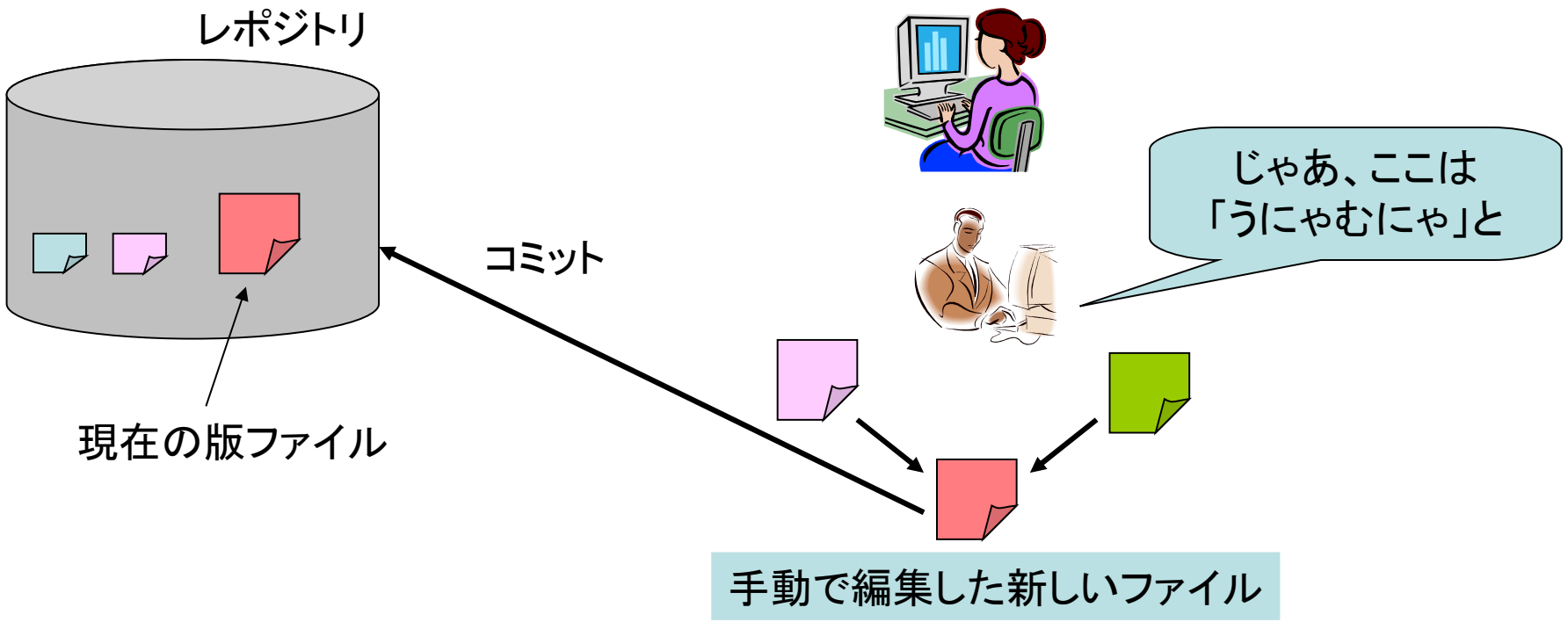
衝突した場合は？

- 手動で二つの変更箇所を編集
  - 連絡をとって調整したり・・・



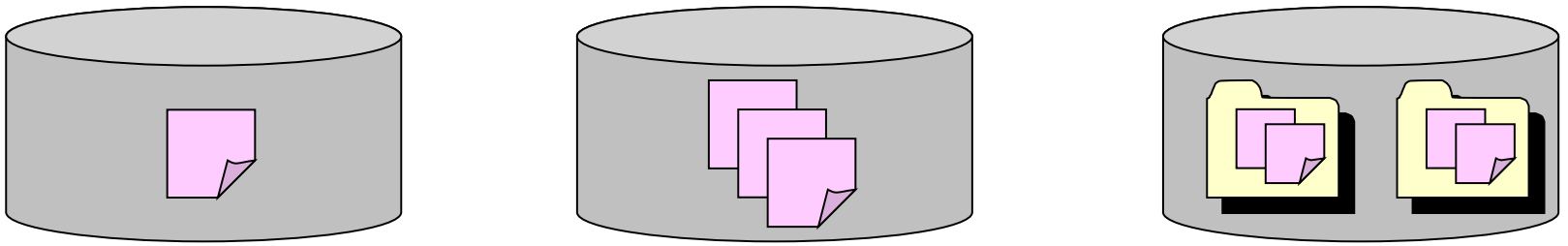
衝突した場合は？

- 衝突を手動で解消したあとに、改めてコミットする



管理の単位

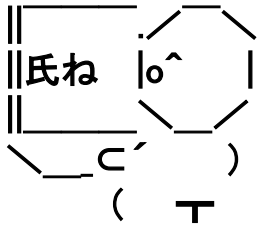
- バージョン管理システムによって、ファイルを管理する単位が違う
- RCS はファイル1つずつ
- CVS は複数のファイルをまとめて管理できる
- SVN はディレクトリごとで管理できる



1ファイル単位での管理

- 導入が楽
- たくさんのファイルにわたって編集を加える場合
  - 変数名や関数の宣言を変える場合など
  - 関係するファイルを全部ロックして編集しないと衝突する可能性がある
  - ロックしてる間は他の人が一切作業できない
    - うっかりチェックインし忘れるとさらに迷惑
    - 怒りのメールが来る

なんというバージョン管理システム・・・
チェックインをし忘れただけで同僚からメールが来た
このシステムは間違いなくRCS



複数ファイル単位での管理

- 多くのプログラムは複数のファイルからできているので、基本的にこっちを使う
- モジュールごと（CVS）、もしくはディレクトリごと（SVN）まとめてチェックアウトする
- コミットもまとめて行う

※ CVS は変更を加えてないファイルはバージョン上がらない

※ SVN は全部上がる

実際に使ってみる

Windows 用 RCS

- http://www.componentsoftware.com/Products/RCS/

- ダウンロードしてインストール
- 途中で「レポジトリをどうするか？」の二択のボタンが出るので、下を選ぶ（上でもいいけど）

- Linux/UNIX には普通に入ってる
- Cygwin にも入ってる

まずは編集するファイルを作る

- 「新しいファイルの作成」で適当なテキストファイルを作る
- 適当に編集して、保存

ファイルをレポジトリに追加

- ファイルを右クリックして、「Add to RCS」を選ぶ
- これでファイルがレポジトリで管理される！
  - インストール時の選択を下にすると「RCS」というディレクトリができて、その下で管理される
  - 上にした場合 C:\RCS 以下で管理される

- Linux/Unix の人は ci（ファイル名） でファイルがレポジトリに追加される

チェックアウトする（Windows版は不要）

- co コマンドでファイルをチェックアウトする
  - co –l（ファイル名）
- Windows 版は最初にレポジトリに追加したときに自動的にチェックアウトもされている

編集する

- 適当にファイルの中身を書き換えて保存

チェックインする

- 右クリックして「Check-in」を選択
- ポップアップウィンドウが出るので、コメントを入れる
  - 入れなくてもシステムは動くが、コメントは重要
  - どのような意図でそれを編集したか、などを書く
  - 多人数で開発してる場合は必須、自分一人でも覚え書きとして役に立つのでできれば書く

- Linux/Unix は ci（ファイル名）でチェックイン

過去の版を見る

- 右クリックして「Revision History」をえらぶ
- 今までの版が見れる
- Compare Revisions で特定の版同士の差分が見れる

- Unix/Linux の場合は rcslog（ファイル名）

過去の版を取り出す

- 「Revision History」をえらぶ
- 過去の版を選んで、「Retrive Revision」を選ぶ
- ファイル名を選択（同じ名前で上書きしてもいい）

- Linux/Unix では
  co –l –r（バージョン番号）（ファイル名）

簡単でしょ？

- というわけで、RCS は非常に簡単
- 文書やコンフィグファイルなど、一つのファイルで管理するものはこれで十分便利
  - 明日からエクセル・ワード・レポート・プレゼン資料なんかで使ってみよう！
- でも、複数ファイルにわたるプログラムなどを管理する場合は確かに面倒そう・・・・

SVNを使う

- というわけで、ファイル・フォルダをまとめて扱えるSVNを使ってみよう！
- 最初はとっつきにくいけど、慣れるととても便利

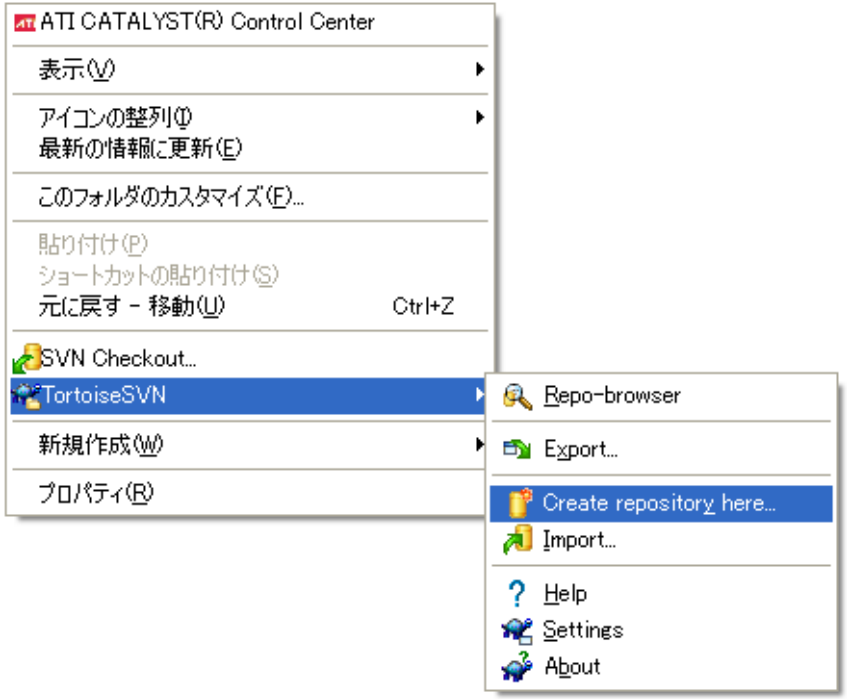
- Windows 用をダウンロード
  - http://tortoisesvn.net/downloads
  - インストールは適当に「ok」か「Next」押してればよし
  - 下の方にある、日本語ランゲージパックもあった方がいい人はダウンロードしてインストール

レポジトリを作る（１）

- RCS ではカレントディレクトリ以下にレポジトリが作られたが・・・・
- SVNではどこかにレポジトリを自分で作る必要がある
  - 自分のディスクの中、どこかのサーバ等

レポジトリを作る（2）

- レポジトリを作りたい場所に、空のフォルダを作る
- 空のフォルダの中で右クリックして「Trotoise SVN」→「Create repository here」

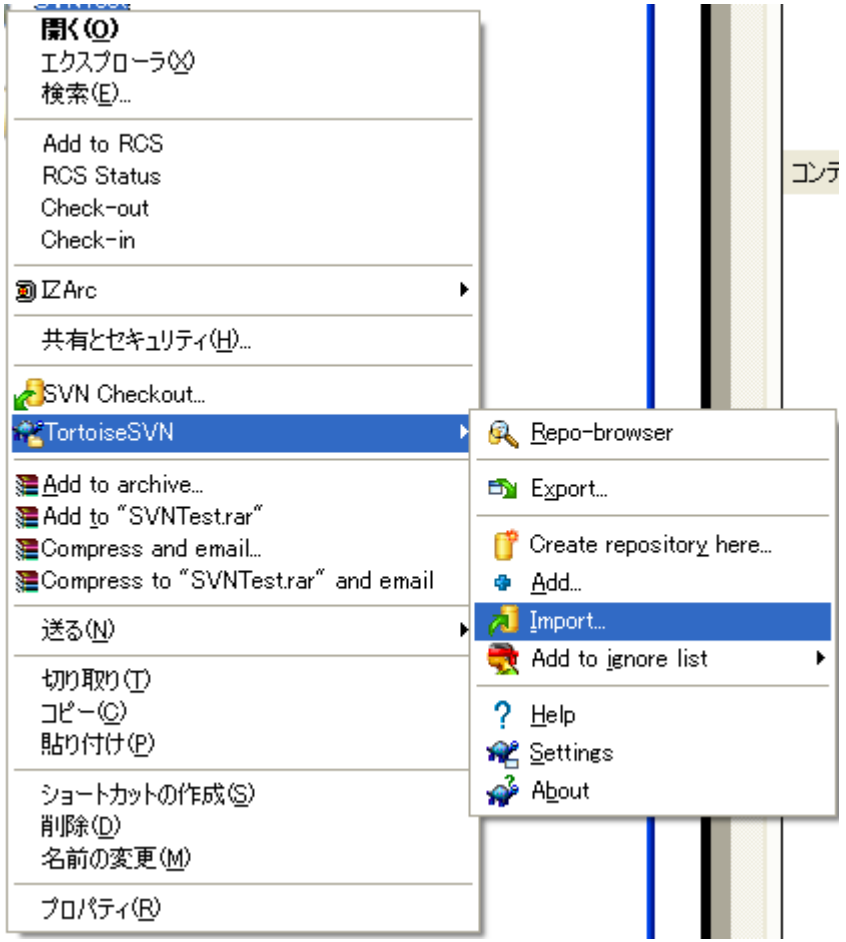


- データベース形式は FSFS でも BDB でもどちらでも
- 適当に必要なファイルが作られる

プロジェクトを追加する

- 適当にプロジェクト用のフォルダを作る
- 中にファイルを置く
- フォルダを右クリックして「TortoiseSVN」→「Import」

- URLを聞かれるので、さっき作ったレポジトリのディレクトリ名＋追加するディレクトリ名を指定



チェックアウトする（１）

- 追加するときに作ったフォルダはSVNで管理されていない
- 別途どこかにチェックアウトする必要がある
  - CVSも同じようにする必要がある
  - ここらへんがRCSと違う

チェックアウトする（2）

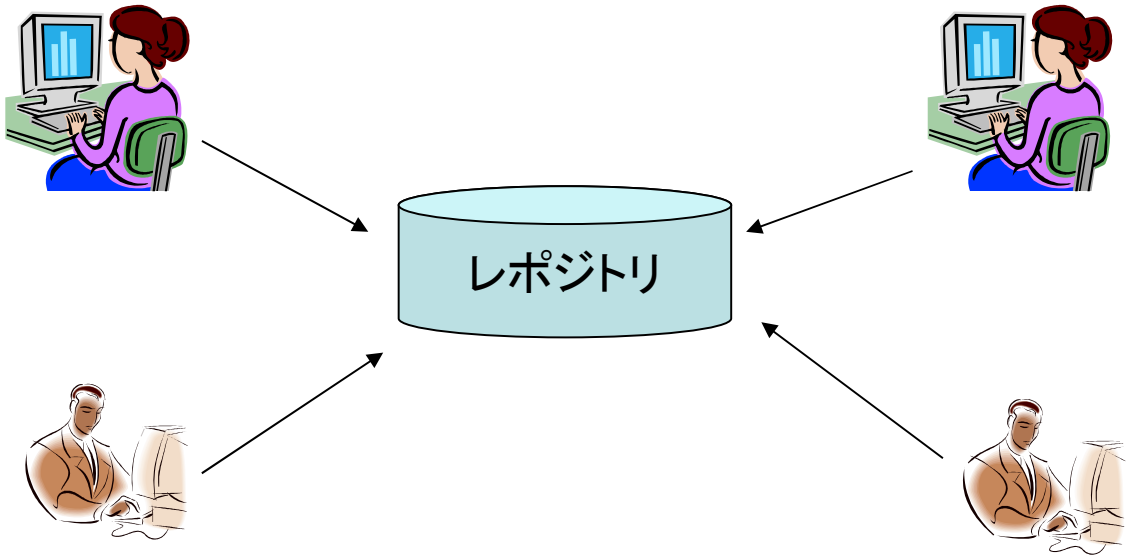
- 作業用に（また）新しいディレクトリを作る
- 空ディレクトリの中で右クリックして「SVN Checkout」
- URLにレポジトリ名を入れる
- チェックアウトされる！

編集してみる

- 適当にフォルダの中のファイルを変更する
- そのフォルダで右クリックして「SVN Commit」
  →変更がレポジトリに反映される

多人数で開発するには？

- どこかのサーバでレポジトリを立てる
- そのレポジトリに全員がアクセスをして開発



実験用レポジトリ

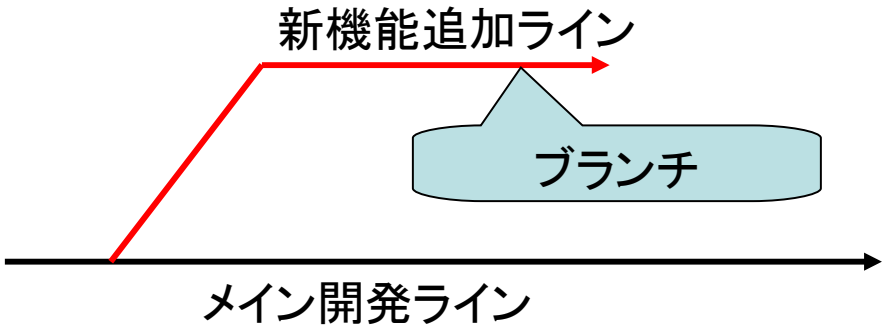
- 接続を試してみる
  - svn://nanv@yuuyake.ddo.jp/
  - パスワードは‘nandemov’
- 追加、編集、コミット、アップデートなどをしてみる

公開用レポジトリ

- フリーの subversion サーバ
  - http://www.xp-dev.com/
  - atwiki みたいな感じ
- ユーザーを作成すれば誰でもアクセス可能

ブランチ

- 今までの説明ではレポジトリは一本の線
- だが、その線を分岐したいときもある
  - 新規機能の追加など、かなり大きな改造
  - 開発に時間がかかるが、その間本家のレポジトリを不安定にしたくない
- ソースコード管理システムには、線を分岐する機能がある
  - 分岐した線のことをブランチという
  - 詳しくはは長くなるので別の機会に！



まとめ

- ソースコード・バージョン管理システム
  - 何故必要なのか？
- 実際に使ってみる
  - RCSを使う
  - SVNを使う