バージョン管理システム

バージョン管理システムはファイルの作成日時/変更日時、変更内容を管理するシステムである。

バージョン管理システムを使用すれば最新版以外にも以前の任意のバージョンを取り出したり、バージョン間の差異を表示したりすることができる。バージョン管理システムは主にソフトウェア開発におけるソースコードの管理のために使用されるが、システムの設定ファイルの管理や個人的な文書の管理に利用することもできる。

ソフトウェア開発プロジェクトやシステム管理においては複数の人間 (開発者やシステム管理者) が同一のファイルを編集することがあるので、最新バージョンの管理や整合性を保つための仕組みがバージョン管理システムには必要である。

個々のファイルのバージョンを個別に管理する初期の頃のローカルなシステムから、ソースコードツリー全体をネットワーク上で管理する近年のシステムまで、様々なバージョン管理システムがある。

バージョン管理システムは 3 つのタイプに大別できる。

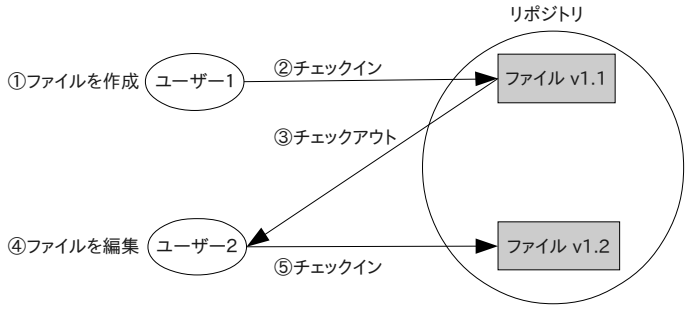
- 初期の頃のローカルなバージョン管理システム
- 集中型バージョン管理システム
- 分散型バージョン管理システム

バージョン管理システムは、ほぼ同義で SCM (Software Configuration Management) と呼ばれることもある。

バージョン管理システムの基本的な仕組み

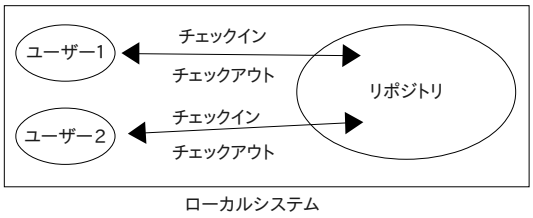
バージョン管理システムではファイルの各バージョンをリポジトリと呼ばれるデータベースで管理する。

編集したファイルをリポジトリに登録することをチェックイン (あるいはコミット)、編集するためにリポジトリからファイルを取り出すことをチェックアウトという。



初期の頃のローカルなバージョン管理システム

ローカルなバージョン管理システムでは各ユーザーは、同一のホストにログインし、同一のディレクトリ下で作業を行う。



ローカルなバージョン管理システムとして、SCCS (Source Code Control System) や RCS (Revision Control System) がある。

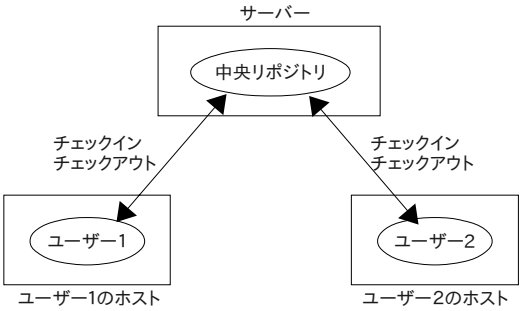
SCCS は、1972 年に IBM のベル研究所の Marc J. Rochkind によって開発され、その後 UNIX のソースコードの管理に使われた。現在ではほとんど使われていない。

RCS は、1980 年代初頭にパデュー大学の Walter F. Tichy によって開発された。複数のユーザーが作業するプロジェクトの管理には適していないが、コマンドで直接リポジトリを操作するため軽量であり、個人的なファイルの管理やシステムの設定ファイルの管理に利用できる。

集中型バージョン管理システム

集中型バージョン管理システムでは参加するすべてのユーザーは中央にあるただ1つの共用のリポジトリにネットワークを介してアクセスする。

各バージョンの参照はこの中央リポジトリからのチェックアウトにより行い、更新もこの中央リポジトリに対するチェックインにより即座に反映する。



集中型バージョン管理システムとしては、CVS (Concurrent Versions System) や SVN (Subversion) がある。

CVS は 1985 年にニュージーランドのアムステルダムにある VU 大学の Dick Grune によって開発され、1990 年代に広く使われた。

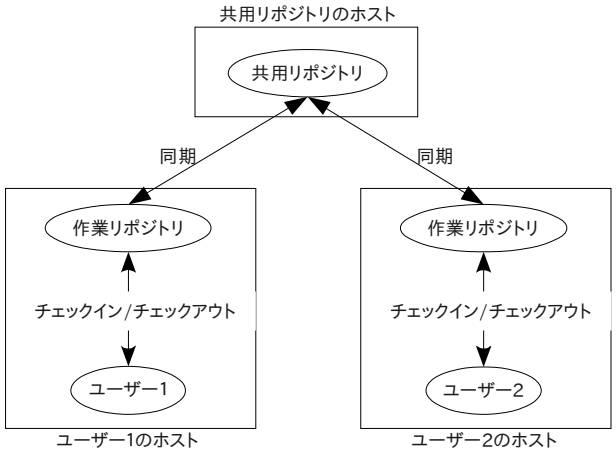
Subversion は CVS の持つ問題点を改善するバージョン管理システムとして 2000 年に CollabNet 社によって開発され、オープンソースのプロジェクトとなった。その後、Apache プロジェクトの 1 つとなった。

分散型バージョン管理システム

分散型バージョン管理システムでは、各ユーザーは自分専用のコピーとなる作業リポジトリを持つ。

チェックイン/チェックアウトは自分の作業リポジトリ内で完結し、他のリポジトリにに影響を与えることなく、ネットワーク接続も必要ない。

他のリポジトリとの同期はチェックイン/チェックアウトのタイミングとは関係なく、明示的に行うことができる。



分散型バージョン管理システムとしては、Linux カーネルの管理に使われている Git (ギット)、Mozilla Firefox の管理に使われている Mercurial 等がある。

2005 年、Linux カーネルの開発に利用されていたバージョン管理システム BitKeeper のフリー版が事情により利用できなくなり、Linus Torvalds 氏と、やはり Linux カーネルの開発者の一人であった Matt Mackall 氏がそれぞれ代わりとなるバージョン管理システムの開発に取りかかった。

Linus Torvalds 氏が開発したのが Git であり、Matt Mackall 氏が開発したのが Mercurial である。

Git は Linux カーネルの開発に使われ、Mercurial は Mozilla Firefox の開発に使われることとなった。Git の開発に初期の頃から関わった日本人の濱野純 (Junio C Hamano) 氏が Git のメンテナーとなっている。

Git は現在、Linux カーネル以外にも Xorg、Ruby on Rails、Perl、Qt、MeeGo といったオープンソースの主要なソフトウェアの開発に使われている。

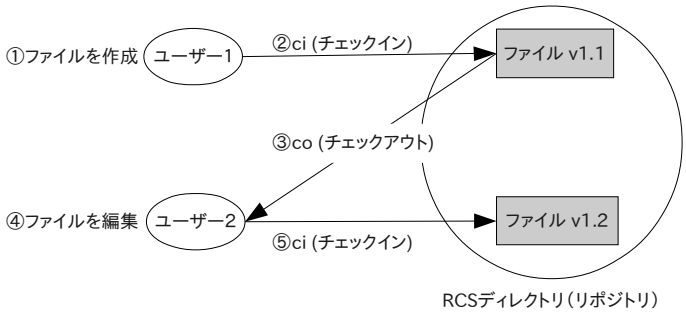
RCS

RCS (Revision Control System) は 1980 年代初頭にパデュー大学の Walter F. Tichy 氏によって開発されたローカルホスト上で使用するバージョン管理システムである。

複数のユーザーが作業するプロジェクトの管理には適していないが、コマンドで直接リポジトリを操作するため軽量であり、個人的なファイルの管理やシステムの設定ファイルの管理に利用できる。

チェックインは ci コマンド、チェックアウトは co コマンドで行なう。レポジトリはコマンド実行ディレクトリ下の RCS ディレクトリ[24]である。

CentOS5.5 ではパッケージ rcs-5.7-30.1.i386.rpm で提供されている。



RCS のコマンド

コマンド	機能
ci	RCS にチェックインする
co	RCS からチェックアウトする
ident	ファイルのリビジョン情報を表示する
rcs	rcs ファイルの属性を変更する
rcsclean	ワークファイルを削除する
rcsdiff	リビジョン間の差異を表示する
rcsmerge	RCS ファイルのリビジョンを併合する
rlog	RCS ファイルのログを表示する

24 RCS ディレクトリを作成しない場合は、ファイル毎にバージョン管理用のファイルができる。

プログラムソースコードの管理例

RCS ディレクトリを作成し、このディレクトリでバージョン管理を行なう。メインプログラムとサブルーチンファイルを作成する。

```
# mkdir RCS

# vi prog.c
  void calc();
  main()
  {
   int x = 1, y = 2;
   calc(x, y);
  }

# vi calc.c
  #include <stdio.h>
  void
  calc(int a, int b){
   printf("a + b = %d\n", a + b);
  }
```

サブルーチンとメインプログラムを RCS ディレクトリにチェックイン (ci) する。

```
# ci calc.c
  RCS/calc.c,v  <--  calc.c
  enter description, terminated with single '.' or end of file:
  NOTE: This is NOT the log message!
  >> c prog calc.c                              ← 適当なコメントを付ける
  >> .                                          ← .でコメント終了
  initial revision: 1.1
  done

# ci prog.c
RCS/prog.c,v  <--  prog.c
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> c prog prog.c                                ← 適当なコメントを付ける
>> .                                            ← .でコメント終了
initial revision: 1.1
done

# ls RCS/
calc.c,v  prog.c,v
```

calc.c を編集するためにチェックアウト (co) し、編集を加えて、またチェックイン (ci) する。
-l は lock オプションである。これを付けると co してから ci するまでの間は他の人は変更ができない。

```
# co -l calc.c
  RCS/calc.c,v  -->  calc.c
  revision 1.1 (locked)
  done

# ls
  RCS  calc.c

# vi calc.c
  #include <stdio.h>
  void
  calc(int a, int b){
   printf("a + b = %d\n", a + b);
   printf("a - b = %d\n", a - b);      ←この行を追加
  }

# ci calc.c
  RCS/calc.c,v  <--  calc.c
  new revision: 1.2; previous revision: 1.1
  enter log message, terminated with single '.' or end of file:
  >> .
  done
```

calc.c のログを表示。revision 1.2 や revision 1.1 の履歴等が表示される。

```
# rlog calc.c
RCS file: RCS/calc.c,v
Working file: calc.c
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2;     selected revisions: 2
description:
c prog calc.c
----------------------------
revision 1.2
date: 2010/03/11 09:13:43;  author: root;  state: Exp;  lines: +1 -0
*** empty log message ***
----------------------------
revision 1.1
date: 2010/03/11 09:11:52;  author: root;  state: Exp;
Initial revision
=============================================================
```

calc.c のバージョン 1.1 と 1.2 の差分の表示。中で diff コマンドが実行されているのがわかる。

```
# rcsdiff -r1.1 -r1.2 calc.c
=============================================================
RCS file: RCS/calc.c,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
5a6
>  printf("a - b = %d\n", a - b);
```

prog.c と cal.c のバージョン 1.1 を RCS から取り出し、コンパイル、実行する。ファイルの編集はしないので、チェックアウト時にロックする必要はない。

```
# ls
RCS                             ← 現在のディレクトリにファイルはない。

# co prog.c                     ← prog.c の最新版を取り出す。
RCS/prog.c,v  -->  prog.c
revision 1.1
done

# co -r1.1 calc.c               ← calc.c のバージョン 1.1 を指定して取り出す。
RCS/calc.c,v  -->  calc.c
revision 1.1
done

# gcc calc.c prog.c             ← コンパイル

# ./a.out                       ← 実行
a + b = 3
```

cal.c の最新版を RCS から取り出し、コンパイル、実行する。現在のディレクトリにある calc.c v1.1 は、最新版 1.2 で上書きされる。

```
# ls
RCS  a.out  calc.c  prog.c

# co calc.c
RCS/calc.c,v  -->  calc.c
revision 1.2
done

# gcc calc.c prog.c             ← コンパイル

# ./a.out                       ← 実行
a + b = 3
a - b = -1
```

prog.c を RCS から取り出してリビジョン情報を埋め込む。その後、チェックイン、チェックアウトして先頭行のリビジョン情報を表示、確認する。

```
# co -l prog.c
  RCS/prog.c,v  -->  prog.c
  revision 1.2 (locked)
  done

# vi prog.c
  /* $Id$ */           ← 追加 ( リビジョン情報を埋め込む。)
  void calc();
  main()
  {
   int x = 1, y = 2;
   calc(x, y);
  }

# ci -l prog.c          ← チェックインし、すぐにロックして取り出す。(ci,co -l)
   RCS/prog.c,v  <--  prog.c
   new revision: 1.3; previous revision: 1.2
   enter log message, terminated with single '.' or end of file:
   >> .
   done

# cat prog.c
```

```
      /* $Id: prog.c,v 1.3 2010/03/11 09:27:38 root Exp root $ */
      void calc();
      main()
      {
       int x = 1, y = 2;
       calc(x, y);
      }
```

システム設定ファイルの管理例

/etc/hosts を RCS で管理することにする。今回は RCS ディレクトリは作成しない。まずは/etc/hosts を RCS にチェックインする。

この際、単に"ci /etc/hosts"とすると/etc/hosts はなくなってしまうことに注意する。

```
# cat /etc/hosts
   127.0.0.1       localhost.localdomain localhost
   172.17.220.130  lx30.example.com lx30
   172.17.220.131  lx31.example.com lx31

# ci /etc/hosts
   /etc/hosts,v  <--  /etc/hosts
   enter description, terminated with single '.' or end of file:
   NOTE: This is NOT the log message!
   >> hosts file
   >> .
   initial revision: 1.1
   done

# ls -l /etc/hosts*                          ← /etc/hosts はなくなる
  -r--r--r-- 1 root root 1347  3月 11 19:23 /etc/hosts,v
```

RCS (/etc/hosts,v) から/etc/hosts をチェックアウトし、編集後チェックインする。この際、ci コマンドに-u オプション (ci, co -u) を付け、チェックイン後、すぐにアンロック状態 (編集する訳ではない) で取り出す。

```
   # co -l /etc/hosts
   /etc/hosts,v  -->  /etc/hosts
   revision 1.2 (locked)
   done

   # vi /etc/hosts
   127.0.0.1       localhost.localdomain localhost
   172.17.220.130  lx30.example.com lx30
   172.17.220.131  lx31.example.com lx31
   172.17.220.132  lx32.example.com lx32   ← この行を追加。

   # ci -u /etc/hosts                       ← /etc/hosts を残す。
   /etc/hosts,v  <--  /etc/hosts
   new revision: 1.3; previous revision: 1.2
   enter log message, terminated with single '.' or end of file:
   >> .
   done

   # ls -l /etc/hosts*
   -r--r--r-- 1 root root 1159  3月 11 19:44 /etc/hosts
   -r--r--r-- 1 root root 1640  3月 11 19:44 /etc/hosts,v
```

/etc/hosts のログを表示

```
# rlog /etc/hosts
RCS file: /etc/hosts,v
Working file: /etc/hosts
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2;     selected revisions: 2
description:
hosts file
----------------------------
revision 1.2
date: 2010/03/11 10:28:42;  author: root;  state: Exp;  lines: +1 -0
*** empty log message ***
----------------------------
revision 1.1
date: 2010/03/11 10:23:48;  author: root;  state: Exp;
Initial revision
=============================================================
```

/etc/hosts のバージョン 1.1 と 1.2 の差分を表示

```
# rcsdiff -r1.1 -r1.2 /etc/hosts
===================================================================
RCS file: /etc/hosts,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
32a33
> 172.17.220.132        lx32.example.com lx32
```