
PERFORCE 2010.2
PERFORCE 概要

2011 年 3 月

This manual copyright 2005–2011 Perforce Software.

All rights reserved.

Perforce software and documentation is available from <http://www.perforce.com>. You may download and use Perforce programs, but you may not sell or redistribute them. You may download, print, copy, edit, and redistribute the documentation, but you may not sell it, or sell any documentation derived from it. You may not modify or attempt to reverse engineer the programs.

Perforce programs and documents are available from our Web site as is. No warranty or support is provided.

Warranties and support, along with higher capacity servers, are sold by Perforce Software.

Perforce Software assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

By downloading and using our programs and documents you agree to these terms.

Perforce and Inter-File Branching are trademarks of Perforce Software. Perforce software includes software developed by the University of California, Berkeley and its contributors.

All other brands or product names are trademarks or registered trademarks of their respective companies or organizations.

© copyright 2005-2011 PERFORCE Software.

All rights reserved.

PERFORCE のソフトウェアおよび関連文書は <http://www.perforce.com> より入手できます。プログラムは、ダウンロードしてご利用になれますが、販売または再配布することは禁じます。関連文書は、ダウンロード、印刷、コピー、編集、再配布することを認めますが、販売することは禁じます。また、いかなるものであれ、本書を元にして作成した文書を販売することも禁じます。プログラムについては、変更を加えること、またリバース・エンジニアリングを試みることも禁じます。

当社 Web サイトより入手した PERFORCE プログラムおよび関連文書は無条件受け取りとなります。保証もサポートもいたしません。保証、サポートは、より高機能のサーバとともに、Perforce Software より有償で提供いたします。

Perforce Software は、本書中の誤りまたは不正確な記述について、いっさい責任も負担も負いません。当社のプログラムおよび関連文書をダウンロードして使用すると、以上の条件に同意なされたことになります。

PERFORCE、インター・ファイル・ブランチ TM は、Perforce Software の商標です。PERFORCE のソフトウェアには、カリフォルニア大学バークレイ校およびその協力者によって開発されたソフトウェアが含まれています。

その他のブランドまたは製品名は、それぞれ当該各社または団体の商標または登録商標です。

目次

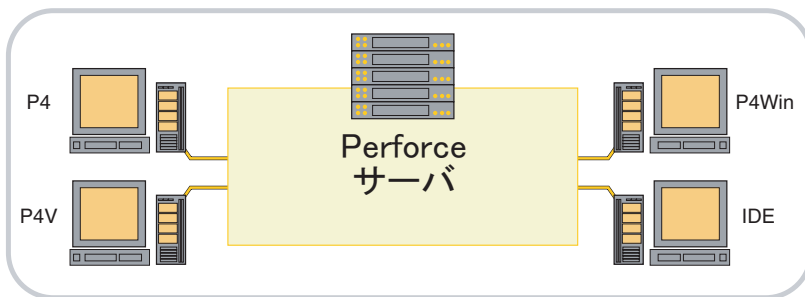
Perforce 概要	5
Perforce の動作	5
Perforce サーバ	5
Perforce クライアント・プログラム	6
サーバへの接続	7
ディポ内のファイルとクライアント・ワークスペースとのマッピング	7
その他の構成オプション	8
Perforce の操作	9
サーバからのファイルの取得	9
ワークスペースの同期	9
Perforce におけるファイル参照	10
Perforce の構文	10
ビューでワイルドカードを使用する	11
ファイルの特定リビジョンを参照する	11
Perforce 構文とステータスバー	12
サポート対象のファイル・タイプ	12
ファイルの操作	13
チェンジリストを使用する	13
チェンジリスト番号の動作	13
ファイルの編集	14
新しいファイルの追加	14
ファイルの削除	14
不要な変更の破棄	14
ファイルのチェックイン	15
ファイルの衝突解決	15
同時並行開発	16
ファイルの比較	17
個別ファイルの変更履歴を調べる	17
ファイル・グループの変更履歴を調べる	18
Perforce 構文とステータスバー	18
コードラインの管理	19
ブランチ操作の基本	19
コードラインの作成	19
コードライン間で変更を伝達する	21
コードライン間の相違を解決する	22
複雑なブランチ構造をコピーする	22
コードライン間の変更履歴を追跡する	23

ブランチに関する詳細説明	24
ラベルに関する注意	24
次の段階	25
作業と障害追跡	25
ファイルにラベルをタグ付けする	25
エディタとマージ・ツール	25
プロテクションと権限付与	26
ユーザとライセンス	26
Perforce の詳細情報	26

PERFORCE 概要

PERFORCE の動作

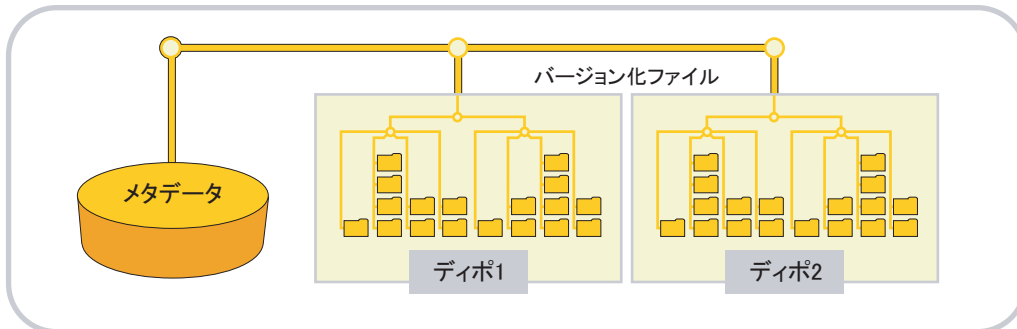
PERFORCE はクライアント / サーバ・アーキテクチャに基づくソフトウェア構成管理 (SCM) システムです。PERFORCE クライアント・プログラムのユーザは PERFORCE サーバに接続し、PERFORCE クライアント・プログラムを使用してサーバのファイル格納場所と個々のクライアント・ワークステーションとの間でファイルのやり取りを行います。



この文書では、PERFORCE サーバのインストールおよび構成が既に行われており、起動していることを前提としています。PERFORCE サーバの設定方法については、『システム管理者ガイド』を参照してください。

PERFORCE サーバ

PERFORCE サーバではファイルの主要格納場所、すなわちディポを管理します。1つのサーバに対し複数のディポが存在する場合があります。ディポには、PERFORCE の管理下にある全ファイルのあらゆるリビジョンが含まれています。PERFORCE はディポにあるファイルを、大型のハードドライブのようにディレクトリ・ツリーとして編成します。ディポ内のファイルは、ディポ・ファイル、またはバージョン化ファイルと呼びます。サーバではデータベースを保守し、チェンジログ、ユーザの権限、どの時点でどのユーザがどのファイルをチェックアウトしているかを追跡します。データベースに保存される情報を、メタデータと呼びます。



PERFORCE サーバではオペレーティング・システム自体の機能を使用してデータベースおよびバージョン化ファイルの管理が行われるため、専用のファイル・システムやボリュームは必要ありません。

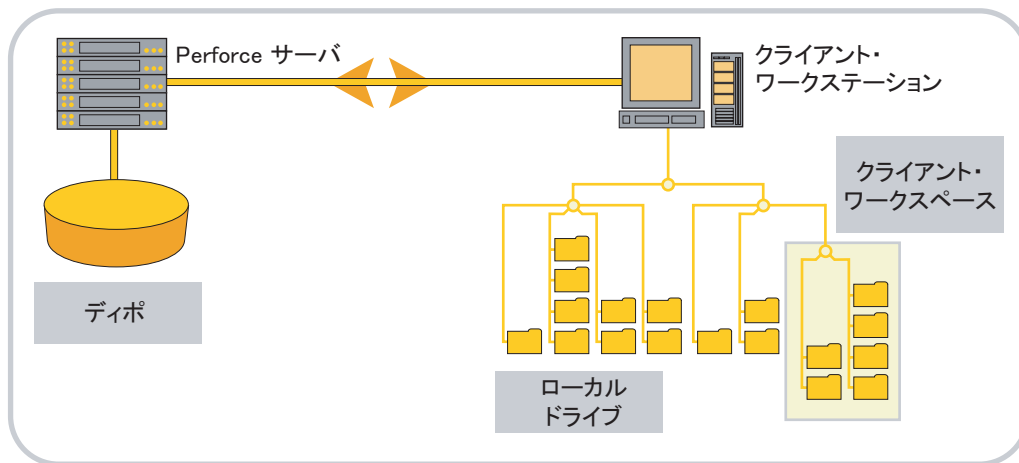
PERFORCE クライアント・プログラム

PERFORCE サーバとのやり取りを行うには、PERFORCE クライアント・プログラムを使用します。PERFORCE クライアント・プログラムにより、ファイルのチェックインおよびチェックアウト、衝突の管理、開発用ブランチの作成、バグの追跡、要求の変更などを行うことができます。PERFORCE クライアント・プログラムには、以下のツールが含まれます。

- ・ P4 - すべてのプラットフォームを対象とする PERFORCE コマンドライン・クライアント
- ・ P4V - Mac OS X、UNIX、Linux、Windows を対象とする PERFORCE ビジュアル・クライアント
- ・ P4Web - ブラウザ・ベースのクライアントである PERFORCE Web クライアント
- ・ 市販 IDE および 生産性ソフトウェアと共に機能する、統合ツール (プラグイン)

日本語版 PERFORCE は Linux 版 P4V および P4Web をサポートしておりませんのでご注意ください。

PERFORCE では、ディポにあるファイルを直接操作することはありません。その代わりに、PERFORCE クライアント・プログラムを使用して、クライアント・ワークスペースと呼ばれるローカルのワークステーションの特定領域を管理します。クライアント・ワークスペースには、ディポの一部のローカル・コピーが含まれます。



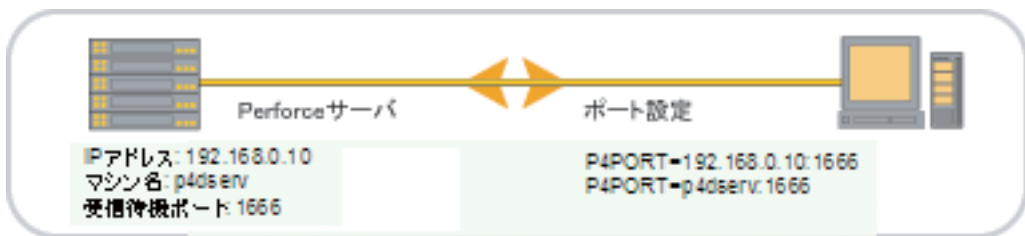
クライアント・ワークスペースにファイルを取得するとき、PERFORCE クライアント・プログラムがそのファイルを PERFORCE サーバに要求します。ネットワークのトラフィックを最小限に抑えるため、PERFORCE サーバはユーザがどのファイルを取得したかを記録します。PERFORCE クライアント・プログラムでは、PERFORCE サーバに持続的に接続している必要はありません。

PERFORCE を使用するには、PERFORCE クライアント・プログラムを設定し、自分の組織の PERFORCE サーバに接続するようする必要があります。また、クライアント・ワークスペースがローカル・ハードドライブのどこにあるか、ディポからどのファイルを取得して作業するつもりかをクライアント・プログラムに伝えなければなりません。

サーバへの接続

PERFORCE で操作を行うには、PERFORCE サーバに接続しなければなりません。PERFORCE クライアント・プログラムは TCP/IP 接続により PERFORCE サーバとの通信を行います。各 PERFORCE クライアント・プログラムでは、通信する PERFORCE サーバのアドレスおよびポートが分かっている必要があります。

アドレスおよびポートは P4PORT 環境変数に格納されます。使用する PERFORCE クライアントにより、この変数の構成処理は「ポートの設定」、または「サーバへの接続」と呼ばれます。



クライアント・プログラム関連文書およびオンラインヘルプに、ポートの設定方法に関する情報が記述されています。自分の組織の PERFORCE サーバへ接続するポートの設定が分からない場合は、PERFORCE 管理者に問い合わせてください。

ディポ内のファイルとクライアント・ワークスペースとのマッピング

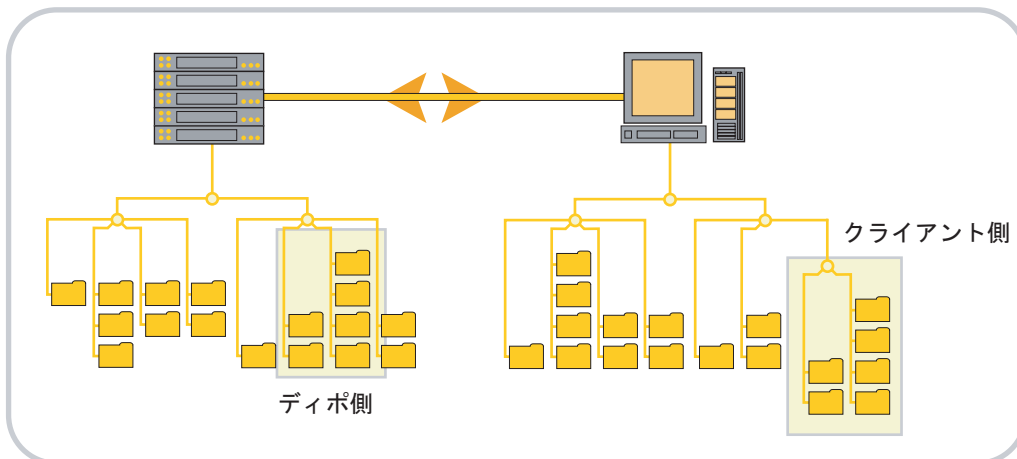
PERFORCE クライアント・プログラムは、ローカルディスクの特定領域、つまりクライアント・ワークスペースにあるファイルを管理します。クライアント・ワークスペースで大部分の操作が行われます。同一のコンピュータ上に、複数のクライアント・ワークスペースを持つことができます。クライアント・ワークスペースの最上位ディレクトリを、クライアント・ワークスペース・ルートと呼びます。

クライアント・ワークスペース・ルート配下のディポ・ファイル格納場所を制御するには、PERFORCE サーバ上のファイルとディレクトリを、ローカルのハードドライブの対応領域にマッピングしなければなりません。これらのマッピングにより、クライアント・ワークスペース・ビューが構成されます。

クライアント・ワークスペース・ビューにより、

- ・ ディポ内のどのファイルをクライアント・ワークスペースに表示するかが決定されます
- ・ ディポ内のファイルをクライアント・ワークスペース内のファイルにマッピングします

クライアント・ワークスペース・ビューは 1 つ以上の行、すなわちマッピングから構成されます。ワークスペース・ビューの各行は、ディポ内のファイルの部分集合を指定する「ディポ側」と、ディポ側で指定されたファイルのクライアント・ワークスペース・ルート配下での場所を制御する「クライアント側」の 2 つから成ります。



クライアント・ワークスペース・ビューを作成しただけでは、ファイルはサーバからコンピュータに転送されません。ビューにより設定されるのは、ファイル転送時のディポとクライアント・ワークスペースとの関係を制御するマッピングだけです。

クライアント・ワークスペースを定義するマッピングの設定方法について詳しくは、PERFORCE クライアント・プログラムの関連文書およびオンラインヘルプを参照してください。

その他の構成オプション

その他の PERFORCE クライアントのオプションにより、PERFORCE におけるさまざまな操作についてデフォルトの動作を制御することができます。例えば、複数プラットフォーム間での開発のために復帰 / 改行の解釈を制御したり、PERFORCE 内で使用する好みのテキスト・エディタやマージ・ユーティリティを指定したりすることができます。これらのオプションとその他のオプションについて詳しくは、各 PERFORCE クライアント・プログラムの関連文書を参照してください。

PERFORCE の操作

サーバからのファイルの取得

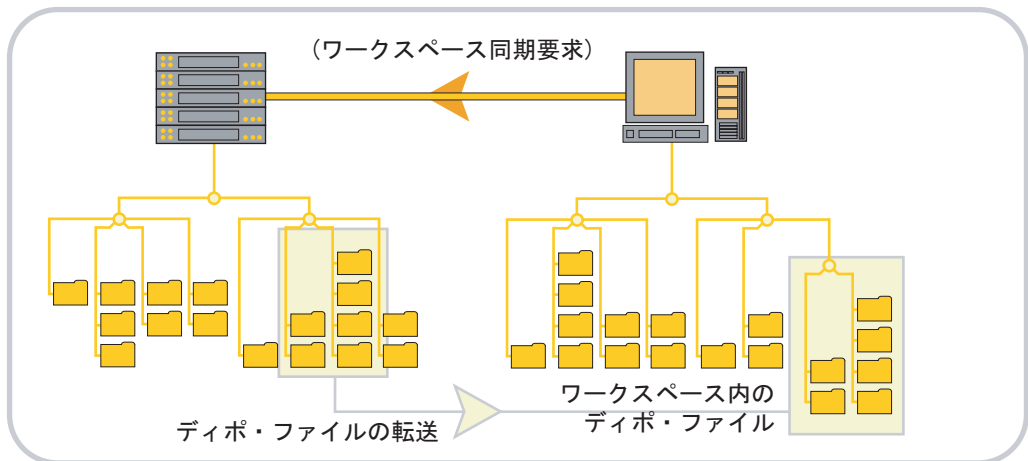
PERFORCE サーバはディポ、つまりシステム内にある全ファイルのあらゆるバージョンのアーカイブを管理します。クライアント・ワークスペースには、ディポのファイルの部分集合をコンピュータの領域へマップする、クライアント・ワークスペース・ビューがあります。

クライアント・ワークスペースにディポのファイルを配置するには、それらをサーバから取得しなければなりません。PERFORCE では、サーバから取得したファイルでワークスペースを更新することを、ワークスペースを同期と呼ぶことがよくあります。他のシステムでは「更新」、「チップ・リビジョンを取得」、または単に「ファイルの取得」と言います。

ワークスペースの同期

ワークスペースを同期すると、PERFORCE クライアント・プログラムがクライアント・ワークスペース・ビューを使用してディポ内のファイルをクライアント・ワークスペース内のファイルとマップし、その結果をクライアント・ワークスペースの内容と比較します。そして、必要に応じてワークスペース内のファイルの追加 / 更新 / 削除を行い、ワークスペースの内容がディポと同期するようにします。

ワークスペースを同期すると、それぞれのファイルの最新（「ヘッド」）リビジョンのコピーが取得されます。（他の SCM システムではこれを「チップ」リビジョンと呼ぶことがあります。）



PERFORCE クライアント・プログラムはワークスペースのファイルの権限を管理します。デフォルトでは、ワークスペースに同期されたファイルは読み取り専用であり、それらを編集目的でチェックアウトした場合に書き込み可能になります。

また、PERFORCE クライアント・プログラムでサポートされているオプションにより、ファイルの以前のリビジョンや指定した時期に格納されたファイルのリビジョン、または他のユーザがユーザ定義の識別ラベルでタグ付け（ラベル付け）したリビジョン群を取得することが可能です。

PERFORCE におけるファイル参照

PERFORCE はディポ内のファイルを、大型のハードドライブのようなディレクトリ階層に編成します。PERFORCE クライアント・プログラムでは一連の規則を使用して、ディポおよびワークスペースにあるファイルの仕様が定義されます。ディポとワークスペースとのマッピング（クライアント・ワークスペース・ビュー）を設定しているか、ロードしているワークスペースにディポから取得したファイルがあるか、ファイルをチェックインまたはチェックアウトしているか、などに関わらずファイルの参照規則はすべてのオペレーティング・システムにおいて共通です。

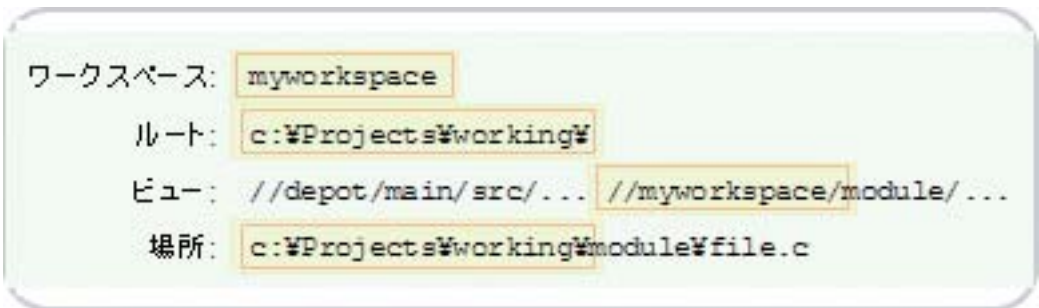
PERFORCE の構文

PERFORCE においては、クライアント・ワークスペース・ルートへの相対パス（「クライアント構文」）、またはディポ・ツリーの最上位ディレクトリへの相対パス（「ディポ構文」）、またはローカルのファイル・システムでの相対パスまたは絶対パス（「ローカル構文」）によりファイルを参照することができます。

クライアント構文またはディポ構文によるファイル指定は、常に 2 つのスラッシュ (/) で始まり、その後にクライアント・ワークスペースまたはディポの名前、ファイルの完全パス名（クライアント・ワークスペース・ルートまたはディポ・ツリーの最上位ディレクトリへの相対パス）が続きます。クライアント構文とディポ構文のパスの構成要素は、ローカルのオペレーティング・システムで使用されるコンポーネント・セパレータに関係なく、常にスラッシュ (/) で区切られます。

構文	例
ディポ構文	//depot/main/src/file.c
クライアント構文	//myworkspace/module/file.c
ローカル構文	C:\Projects\working\module\file.c

ディポ・ファイルをローカルのハードドライブにマップする際、クライアント・ワークスペース名はクライアント・ワークスペース・ルートの別名になります。



例えば、クライアント・ワークスペース名が myworkspace、ワークスペース・ルートが C:\Projects\working である場合、
 //depot/main/src/... //myworkspace/module/...
 というビューのマッピングは、
 //depot/main/src/file.c というディポ・ファイルを
 C:\Projects\working\module\file.c
 としてクライアント・ワークスペースにマップします。

ビューでワイルドカードを使用する

PERFORCE においてワークスペース・ビューを構成する際、以下のワイルドカードを使用することができます。

ワイルドカード	意味	例
*	1つのディレクトリ内のスラッシュを除くすべての文字に一致します	/src/*.c は、/src/file.c および /src/file2.c に一致しますが、/src/lib/file.c には一致しません。
...	現在の作業ディレクトリおよびすべてのサブディレクトリの配下にあるすべてのファイルに一致します。	/src/... は /src とその配下にある全ファイルと全サブディレクトリに一致します。
%%1 - %%9	ビュー内のファイル名の一部を置き替える位置指定子	%%1/%%2/... を %%2/%%1/... にマッピングすると、/web/images/file.gif は /images/web/file.gif にマップされます。

また、これらのワイルドカードはコマンドライン・クライアントでのファイル指定にも使用されます。PERFORCE 構文およびワイルドカードについてより詳しくは、『コマンド・リファレンス』を参照してください。

ファイルの特定リビジョンを参照する

PERFORCE ではファイル・リビジョンの指定に # の文字を使用します。PERFORCE のファイル・リビジョンは、最初のリビジョンを #1 として、連続して増加する整数により表されます。ファイルの最新のリビジョンはサーバ上の当該ファイルの最も大きい番号が付いたリビジョンで、ヘッド・リビジョンと呼ばれます。一番最近ワークスペースに同期したリビジョンは、所有リビジョンと呼ばれます。0 番のリビジョンはヌル・リビジョンと呼ばれ、データを含みません。

PERFORCE で作業する際、開発ブランチ（コードライン）はディレクトリ・パスで表されます。ファイルはコードラインごとに、最初のリビジョン #1 から順次増加するリビジョン番号の組を持っています。個々のコードラインにおけるファイルの祖先は**反跳レコード**内に保持されます。PERFORCE のブランチ機能について詳しくは、19 ページの「コードラインの管理」を参照してください。

file.c#3/4 という表記は、現在ワークスペースに file.c のリビジョン #3 を同期させており、file.c の最新リビジョンが #4 であることを示します。他の SCM システムと異なり、PERFORCE では個々の開発ブランチのファイル・リビジョンを指定するために紛らわしいバージョン番号（例えば「file.c のリビジョン 1.2.3」）を使用しません。

構文	参照先	備考
file.c#3	file.c の 3 番目のリビジョン	「file.c の 3 番目のリビジョンと同期」
file.c#head	サーバに保存されている file.c の最新リビジョン、すなわちヘッド・リビジョン	ディポからファイルの最新バージョンを取得するには、ワークスペースをヘッド・リビジョンに同期します

構文	参照先	備考
file.c#have	一番最近ワークスペースに同期された file.c のリビジョン、すなわち <i>所有リビジョン</i>	ファイルへの変更を破棄すると、PERFORCE クライアント・プログラムはワークスペースにあるファイルのコピーを <i>所有リビジョン</i> に復元します。
file.c#none file.c#0	file.c の実在しない、またはヌルのリビジョン	PERFORCE クライアント・オプションを使用して [ファイルをワークスペースから削除] すると、実際にはワークスペース内のファイルを <i>ヌル・リビジョン</i> に同期することになります

PERFORCE 構文とステータスバー

ヘッド・リビジョン (#head)、所有リビジョン (#have)、ヌル・リビジョン (#none) の構文がコマンドライン・クライアントおよびグラフィカル・クライアント・プログラムのステータス・ウィンドウで使用されます。詳しくは『P4 ユーザ・ガイド』を参照してください。

サポート対象のファイル・タイプ

PERFORCE のファイル・タイプには 6 種類の基本ファイル・タイプが含まれます。

- text ファイル
- binary ファイル
- Macintosh のネイティブ apple ファイル
- Mac リソース・フォーク
- シンボリックリンク (symlink)
- unicode および utf16 ファイル

デフォルトでは、ディポにファイルを追加するとき、PERFORCE がファイル・タイプを自動的に決定します。新しいファイル・タイプで編集目的の作業状態にすることにより、ファイル・タイプを変更することができます。既にファイルが作業状態にある場合、異なるファイル・タイプを指定して再度作業状態にすることができます。

6 種類の基本ファイル・タイプに修飾子 (+w、+x、+k など) を付加することにより、ロック動作、クライアント・ワークスペース内のファイルアクセス権、サーバ上でのリビジョン格納方法などを制御することができます。ファイル・タイプおよび付加可能な修飾子の全リストが『コマンド・リファレンス』に記述されています。

ファイルの操作

チェンジリストは PERFORCE での基本操作単位です。すべての SCM システムに共通する基本的なファイル編集操作（ファイルの編集・追加・削除、変更の取り消し、ファイルのチェックイン）はチェンジリストで実行されます。

チェンジリストを使用する

ワークスペース・ビューを設定し、ワークスペースをディポと同期させたら、PERFORCE で操作を開始することができます。ワークスペース内でファイルを操作する前に、PERFORCE クライアント・プログラムを使用してチェンジリスト内のファイルを開く必要があります。チェンジリストは、ファイルのリスト、それらのリビジョン番号、ファイルに対する変更、行った操作を説明するコメントから構成されます。

チェンジリストには以下の 2 つの役割があります。

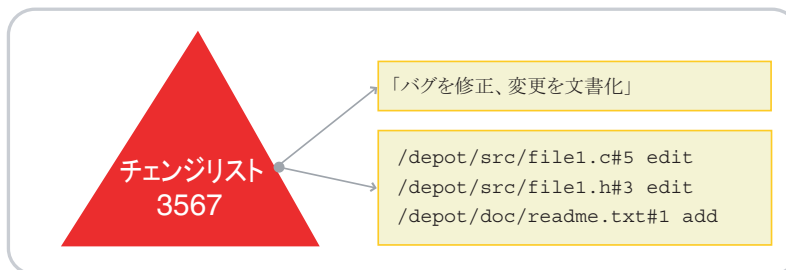
- ・ 関連するファイルへの変更操作をグループ化し、ユーザの作業を論理的単位に編成する
- ・ 関連するファイルへの変更操作が共にチェックインされるようにし、ユーザの作業の整合性を保証する

3 つのファイルに対する変更が必要なソフトウェア変更作業を行っている場合、それら 3 つのファイルを 1 つのチェンジリストで作業状態にします。チェンジリストをディポに戻すと、他のユーザはそのチェンジリストが 3 つのファイルに対して行われた変更にリンクされていることを確認できます。

PERFORCE のチェンジリストはアトミックな変更トランザクションです。チェンジリストが 3 つのファイルに影響する場合、それらに対する変更がすべてディポにコミットされるか、またはどの変更もコミットされないかのいずれかとなります。チェンジリストのサブミット中に PERFORCE クライアント・プログラムと PERFORCE サーバとのネットワーク接続が切断された場合にも、サブミット全体が失敗します。

チェンジリスト番号の動作

まだディポにサブミットされていない変更を含むチェンジリストを、*作業中チェンジリスト*といいます。ディポにコミットされた変更を含むチェンジリストを、*サブミット済みチェンジリスト*といいます。各チェンジリストには、*チェンジリスト番号*（PERFORCE により生成）、およびチェンジリストのコメント（変更を行ったユーザが入力）が含まれます。



PERFORCE でファイルを作業状態にすると、ファイルは *default* チェンジリストで作業状態になります。チェンジリストのファイルをディポにチェックインすると、*default* チェンジリストにチェンジリスト番号が割り当てられます。

処理中の作業を複数の作業中チェンジリストに分割することができます。default チェンジリスト以外の作業中チェンジリストには、チェンジリスト作成時に番号が割り当てられます。(チェンジリストをディポにサブミットする際に、新しい番号が作業中チェンジリストに割り当てられることがあります。)

ファイルの編集

ファイルを編集するには、ファイルをチェンジリストにチェックアウトします。PERFORCE クライアント・プログラムは、クライアント・ワークスペース内にコピーされたファイルを書き込み可能にして、そのファイルが編集目的で作業状態にされたことを PERFORCE サーバに伝えます。

自分の変更を他のユーザが利用できるようにするには、チェンジリストをディポにサブミットしなければなりません。チェンジリストをサブミットすると、他のユーザがワークスペースを同期させ、その変更のコピーを取得することができます。

新しいファイルの追加

ファイルを追加するには、クライアント・ワークスペースでファイルを作成し、チェンジリストでそのファイルを追加目的の作業状態にします。PERFORCE クライアント・プログラムによりファイル・タイプが決定され(書き換え可能)、そのファイルを追加予定であることが PERFORCE サーバに伝えられます。

新しいファイルを他のユーザが利用できるようにするには、追加されたファイルを含むチェンジリストをディポにサブミットしなければなりません。チェンジリストをディポにサブミットすると、他のユーザがワークスペースを同期させ、その新しいファイルのコピーを取得することができます。

ファイルの削除

ファイルを削除するには、チェンジリストでファイルを削除目的の作業状態にします。ファイルは直ちにワークスペースから削除されます。PERFORCE クライアントにより、ファイルが削除予定であることがサーバに伝えられます。ただしそのファイルは、チェンジリストをサブミットするまでディポでは削除済みとしてマークされません。

チェンジリストをサーバにサブミットすると、そのファイルが削除済みとしてマークされていることを他のユーザが確認できます。削除されたファイルのローカル・コピーは、そのユーザがワークスペースをディポと同期させるまで、他のユーザのワークスペースに残ります。

削除済みファイル・リビジョンは実際にはディポから削除されません。削除済みファイルより前のリビジョンは、ファイル削除前のリビジョンをクライアント・ワークスペースに同期させることによりいつでも復元することができます。

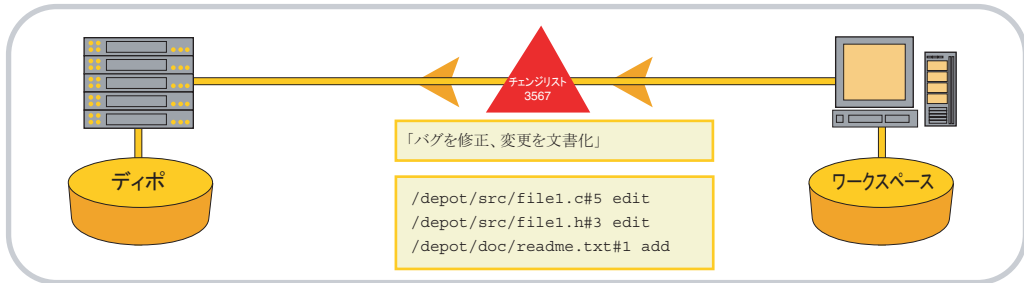
不要な変更の破棄

ファイルを元に戻すことにより、チェンジリスト内のファイルに対して行った変更を破棄することができます。ファイルを元に戻すと、そのファイルはチェンジリストから削除され、クライアント・ワークスペース内のファイルのコピーがワークスペースに最後に同期されたリビジョンに復元されます。

編集目的または削除目的で作業状態にされているファイルを元に戻すと、最後に同期したファイルのバージョンがワークスペースに復元されます。追加目的の作業状態にあるファイルを元に戻すと、そのファイルはチェンジリストから削除されますが、ファイルのローカル・コピーはクライアント・ワークスペース内に残ります。

ファイルのチェックイン

作業状態にしたファイルへの変更が完了した後、作業結果を他のユーザが利用できるようにするには、チェンジリストをサブミットして変更内容をディポにチェックインします。

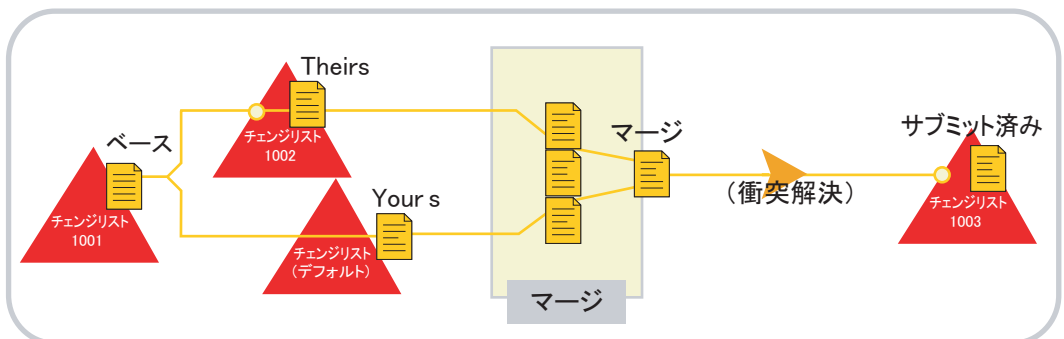


一部サブミット済みのチェンジリストというものはありません。チェンジリストのサブミットはアトミック・トランザクションであり、チェンジリスト内のファイルがすべてサブミットに成功するか、またはどのファイルもサブミットに失敗するかのどちらかになります。

ファイルの衝突解決

同時に 2 人のユーザが同じファイルを編集すると、それらの変更が衝突する場合があります。ある変更が別のユーザにサブミットされた変更と衝突する場合、PERFORCE は衝突しているファイルの解決およびチェンジリストの再サブミットを要求します。チェンジリストはアトミック・トランザクションであるため、衝突が解決されるまで、チェンジリスト内のファイルに対するいかなる変更もディポには表れません。

衝突解決処理により、どのような作業を行う必要があるかを、自分のファイルで他のユーザのファイルを上書きするか、自分のファイルを破棄し他のユーザの変更を生かすか、または衝突している 2 つのファイルをマージして 1 つのファイルにするかという選択肢の中から決定することができます。要求にしたがって、PERFORCE は 2 つの衝突するテキスト・ファイルおよびそれらのファイルの元となったファイルとの間で 3 ウェイ・マージを行うことができます。



同時並行開発

PERFORCE はチームの同時作業を支援します。衝突解決および3ウェイ・マージ処理により、複数のユーザが互いの作業に影響を及ぼすことなく同一ファイルを同時に操作することが可能です。

ファイルの衝突解決のための3ウェイ・マージ処理はテキスト・ファイルに対する変更の衝突解決には役立ちますが、グラフィックやコンパイル済みコードなどのバイナリ・ファイルでは効果がない場合もあります。マージする効果のないファイルを取り扱う場合は、それらのファイルをロックして他のユーザが自分の作業と衝突する変更を加えないようにすることができます。

PERFORCE では2種類のファイルのロック方法がサポートされています。ファイルのロックによりファイルがチェックインされないようにする方法と、*排他的オープン*によりファイルがチェックアウトされないようにする方法です。

- 自分の作業中のファイルへ他のユーザが変更をチェックインできないようにするには、ファイルをロックします。ロックされたファイルを他のユーザがチェックアウトすることは可能ですが、ロックしたユーザが変更をサブミットしない限り、ロックされたファイルを持つチェンジリストをサブミットすることはできません。(変更をサブミットする前に、ロックされたファイルを持つチェンジリストのサブミットを可能にするには、ファイルのロックを解除します。)
- ファイルが複数ユーザにより同時にチェックアウトされないようにするには、+1 *排他オープン* ファイルタイプ修飾子を使用します。+1 ファイルタイプ修飾子を持つファイルは一度に一人のユーザのみが作業状態にできます。PERFORCE 管理者はタイプマップ・テーブルという特別なテーブルを使用して、特定のファイルタイプを自動的に排他的オープンの対象として指定することができます。

例えば、変更の衝突解決を許容しない IDE で作業を行っているユーザの場合、作業中のファイルをロックして作業内容をサブミットするために、PERFORCE クライアント・プログラムに切り替える必要がないようにしたいと考えるかもしれません。また、電子資産を取り扱うユーザはすべての .gif ファイルまたは .mpg ファイルを、自動的に排他的オープンの対象にしたいと考えるかもしれません。

編集対象	ロックされているか	意味
file (type)	ロックされていない	全ユーザが file のチェックアウトおよび変更のサブミットが可能。 file を作業状態にしている間に別のユーザが file への変更をサブミットした場合、チェンジリストのサブミットの際に自分の変更と他ユーザの変更との衝突を解決しなければなりません。
file (type)	ロックされている	全ユーザが file のチェックアウト可能であるが、file への変更のサブミットは、ロックしたユーザが file への変更をサブミットするか、file のロックを解除しない限り不可。
file (type+1)	ロックされていない またはロックされている	一度に1ユーザのみがチェンジリストで file を作業状態にできます。ロックの状態には関係しません。他のユーザはそのファイルをチェックアウトできないため、そのファイルに関わるチェンジリストをサブミットすることはできません。

ファイルのロック、ファイルタイプ修飾子の排他的オープン、タイプマップ・テーブルに関して詳しくは、『コマンド・リファレンス』および『システム管理者ガイド』を参照してください。

ファイルの比較

PERFORCE を使用して、同じファイルの 2 つのリビジョン、またはディポ内にある任意の 2 つのファイル、またはディポ内のファイルとそのワークスペース内のコピーを比較することができます。

p4 diff および p4 diff2 コマンドは、UNIX および Linux のシステムに含まれている標準の diff プログラムの出力と同様の出力を生成します。その他の PERFORCE クライアント・プログラム (P4V を含む) には P4Merge が含まれています。P4Merge はファイルの差分がグラフィカルに表現されます。例を示します。

```

1017.15.8.22 - performe@proliant2: ~/p4...
Range: 未入力
118:114e144.152
< dd = pos * r;
< fv = ( pos / dd ) * ( -1 * CONST_G * w->mass
* m / ( dd * dd ) );
---
> dd = "pos * r + 10.0:
>
> //fv = ( pos / dd ) * ( -1 * CONST_G * w->mass
* m / ( dd * dd ) );
> //instead of using real gravity, we'll use
a version that's
> //actually useful, which is more like a spring
connecting everything
> //in the world to the center.
>
> fv = ( pos / dd ) * ( -1 * CONST_G * w->mass
* m * dd );
> if ( w->heavyz ) fv = fv * 1000.0;
125a164
> if ( w->nofric ) fric = 0.0;
141a181.182
> if ( !isworld ) returns;
---
144#187
> if ( !init && !init )
> if ( !init && !springs )
167a209
> if ( !p->isworld ) continue;
[performe@proliant2 bruno:~/win5082]$

```

```

//depot/Jamgraph/MAN/src/aparticle.cpp#1
//depot/Jamgraph/MAN/src/aparticle.cpp#2
16の差分 (行末の相違を無視) タブスペース4 エンコードシステム
p = new UParticle( pos.X + dx, pos.Y);
else
p->name = "none";
if ( n > 1 )
p->name = "some";
else
p->name = "none";
p->next = w->PARTS;
w->PARTS = p;
s = new GSpring(1);
s->K = SPRING_K;
s->PART = p;
s->next = springs;
springs = s;
p->initn = n - 1;
p->Q = Q * 0.9;
GVector GParticle::NearBy()
{
double dx, dy;
int i = w->n+1;
}

```

個別ファイルの変更履歴を調べる

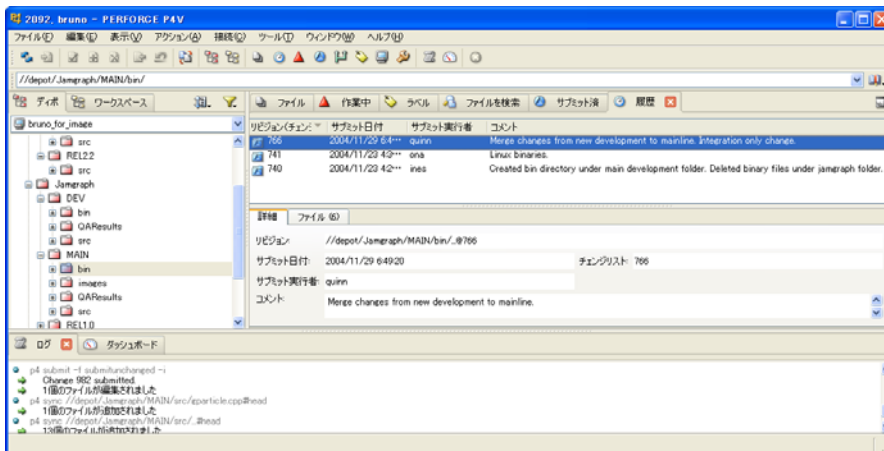
ファイルの履歴は、ファイルごとに一連のファイル・リビジョンにより表されます。ファイルの各リビジョンは、チェンジリストに関連付けられています。ファイルを、ワークスペース内のリビジョンまたはディポに保存された任意のリビジョンと比較することができます。

リビジョン	チェンジリスト	サブmitt日付	サブmitt実行者	ファイルタイプ	コメント
1	quinn	2003/04/02 01:...	quinn	text	A toy. Soon to be hopefully a toy power...
2	quinn	2003/04/09 12:...	quinn	text	Jamgraph functional. Usame...
3	quinn	2003/04/04 11:27:07	quinn	text	A few bug fixes and a few new options. Run "jamgraph -h" for info.

この P4V 画面コピーでは、ディポに `//depot/Jamgraph/MAIN/src/gparticle.cpp` というファイルの 3 つのリビジョンがあることを示しています。最新のリビジョンである #3 はチェンジリスト 720 でサブミットされています。

ファイル・グループの変更履歴を調べる

ディレクトリの履歴は一連のチェンジリストにより表されます。ディレクトリには個別のリビジョン番号が付けられていません。少なくとも 1 つのファイルを含むチェンジリストはすべて、ディレクトリの履歴の一部であると見なされます。



この P4V 画面コピーでは、`//depot/Jamgraph/MAIN/bin` にある少なくとも 1 つのファイルに影響を及ぼした最新のチェンジリストは、チェンジリスト #766 であることを示しています。

PERFORCE 構文とステータスバー

PERFORCE では、ある形式の構文により、番号付きチェンジリストのサブミットの際にディポに存在しているファイル、または簡略記憶ラベルによりタグ付けされたファイル、または特定の日付および時刻のファイルが表現されます。これらの構文形式 (`@changelist`、`@labelname`、`@date`、または `#start,end`) は通常コマンドライン・クライアントでのみ使用されますが、グラフィカル・クライアント・プログラム・ステータス・ウィンドウにも現れます。詳しくは、『P4 ユーザ・ガイド』を参照してください。

コードラインの管理

コードラインとは、同時に進化する、関連付けられたファイル集合のことです。新製品やリリースなどの目的に応じて関連するファイルのグループを構成するには、ブランチを作成します。ブランチ間の変更を伝播するには、チェンジリストを反映します。特定の状態にあるファイルのスナップショットを作成するには、ラベルを作成するか、日付またはチェンジリスト番号を指定してファイルをまとめて参照することができます。

注 | PERFORCE ブランチとブランチ・マッピングを混同しないでください。ブランチ・マッピングは、2つのブランチ間の関係を定義する仕様です。

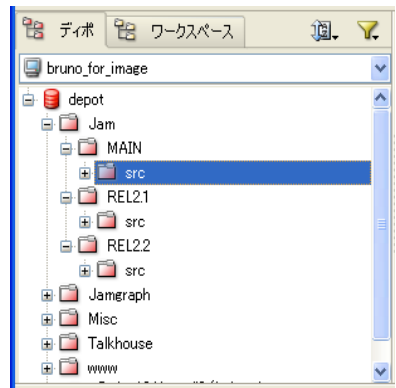
ブランチ操作の基本

ブランチ機能は2つ以上の関連するファイル・セット間の変更を管理する方法です。PERFORCEのインター・ファイル・ブランチ機能により、一方のファイル・セットに加えられた変更をもう一方にコピー、すなわち反映することにより任意のファイル・セットをディポ内の新しい場所にコピーすることができます。新しいファイル・セット（コードライン）は元のファイルから独立して進展しますが、一方のコードラインでの変更を反映機能によって他方に伝達することができます。

ほとんどのソフトウェア構成管理システムではブランチ機能を何らかの形式でサポートしていますが、PERFORCEのメカニズムのユニークな点は、ユーザはブランチ機能がなかったときと同様に、ファイルコピーを作成するような操作でPERFORCEのブランチ機能を使用できることです。

SCMシステムを使用せずにプログラムを作成しているとします。作成したプログラムをリリースする準備はできています。ここでコードをどう処理するでしょうか。おそらく、すべてのファイルを新しい場所にコピーするでしょう。ファイル・セットの1つがリリース・コードラインとなり、そのファイル・セットに対してリリースに向けてのバグ修正が行われます。他のファイルは、開発ファイル・セットになり、これらのファイルにコードの新機能が追加されます。

PERFORCEではディポ内のファイルをディレクトリ階層により、大型のハードドライブのように編成します。新しいコードラインを作成すると、ディポの中にサブディレクトリとして現れます。例えば、進行中の開発作業は `//depot/Jam/MAIN`、リリース 2.1 は `//depot/Jam/REL2.1`、リリース 2.2 は `//depot/Jam/REL2.2` のように示されます。



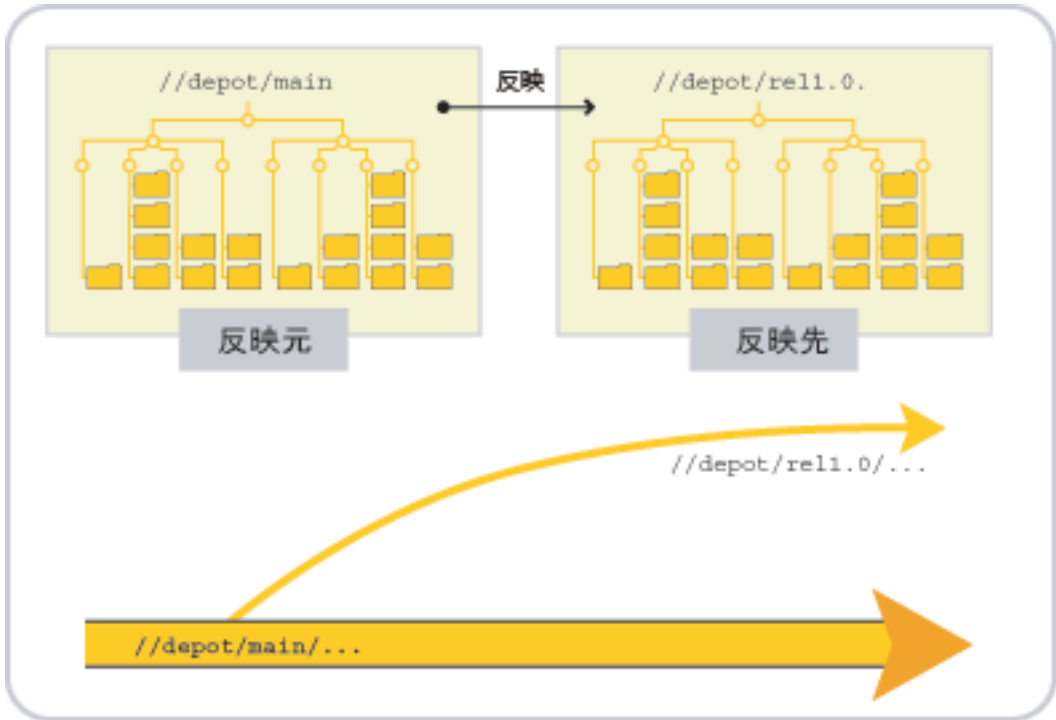
コードラインの作成

コードラインや開発ブランチを作成するには、どのファイルをブランチするかを決め（反映元ファイル）、それらのファイルを新しいコードラインに反映して反映先ファイルを作成します。PERFORCEサーバはチェンジリストで「反映先ファイルをブランチ / 同期目的で作業状態に」します。

ファイルをブランチ / 同期目的で作業状態にする方法は、追加 / 編集 / 削除目的で作業状態にする場合と同様です。チェンジリストでファイルを作業状態にし、クライアント・ワークスペース・ビューに反映先ファイルを含める必要があります。また、チェンジリストをサブミットするまではディポの変更は行われません。チェンジリストはアトミックであるため、コードラインの作成時にブランチしたすべてのファイルが確実に含まれます。

SCM システムを利用しない場合は、ファイルを一方のディレクトリから別のディレクトリへとコピーしてブランチを作成することになるでしょう。ファイルをコピーし、ディポ内の新しいディレクトリへファイルを追加することによって反映を行えば、1 つのコードラインから別のコードラインにファイルを反映させる際、PERFORCE が関連するファイル間のつながりを反映記録で追跡するため、2 つのファイル・セット間の変更の追跡や伝達を容易に行うことができます。

また、反映により PERFORCE でファイルの「遅延コピー」を処理することが可能です。ファイルをブランチしたとき、実際にサーバにファイルのコピーが 2 つ保存されるわけではありません。反映元ファイルと、反映先ファイルへのブランチが実行されたことを示すデータベース・レコードへのポインタが保存されるだけです。遅延コピーによりブランチ操作でのオーバヘッドが軽減され、サーバがファイルの複数コピーを追跡する必要がなくなります。



ファイルを反映元コードラインから反映先コードラインへ反映させるには、

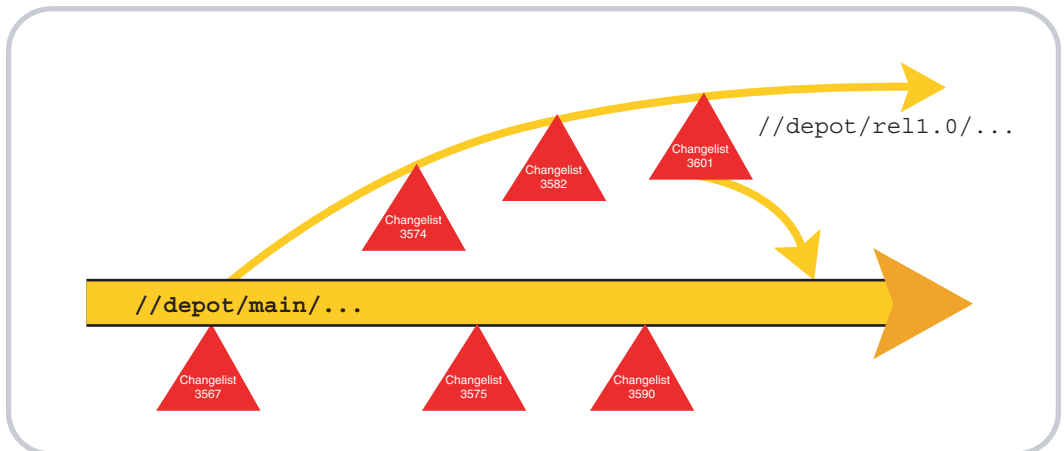
- ・ 反映先がクライアント・ワークスペース・ビュー内にある必要があります
- ・ 反映元がクライアント・ワークスペース・ビュー内にある必要はありません（ただし反映元ファイルの読み取り権限を持っている必要があります）

- ・ 反映により、新しいチェンジリスト内でファイルをブランチ目的の作業状態にします
- ・ チェンジリストのサブミットにより、新しいコードラインにファイルを作成します
- ・ 反映先ファイルのあるチェンジリストをサブミットすると、新しいコードラインの反映先ファイルはリビジョン #1 となります
- ・ 反映記録により、それらが反映元ファイルからの反映により作成されたという事実を含む、新しいコードラインのファイル履歴を調べることができます

コードライン間で変更を伝達する

反映により、関連するコードライン間の変更をコードラインの作成とほぼ同じ方法で伝達することができます。(コードラインの作成は、反映元ファイル全体を構成する変更のセットを、空の反映先ファイル・セットに伝達することに相当します。)

コードラインの作成時、反映先ファイルは当然 空の状態であり、変更が衝突する可能性はありません。既存のコードライン間で変更を伝達すると、反映元および反映先のコードラインに衝突する変更が加えられている可能性があるため、衝突が生じることがあります。



上記の例では、rel1.0 コードラインは、反映元ファイルを //depot/main から //depot/rel1.0 にブランチすることにより、チェンジリスト 3567 で作成されています。チェンジリスト 3574、3582、3601 はリリース・ブランチで行われた作業を表し、チェンジリスト 3575 および 3590 は main ラインで行われた作業を表します。

リリース・ブランチで行われた作業を main ラインに伝達するには、//depot/rel1.0 にある反映元ファイルを //depot/main へ反映し、リリース・ブランチで行われた作業と main ラインで行われた作業との間の衝突を解決します。

コードライン間の相違を解決する

チェンジリストを反映元コードラインから既存の反映先コードラインに反映させると、PERFORCE はファイル間の **衝突解決** または **3 ウェイ・マージ** をスケジュールします。

クライアント・ワークスペースにある反映先ファイルのリビジョンを、*yours* といいます。ディポにある反映元ファイルのリビジョンは *theirs* と呼ばれます。これらのファイル間の変更が衝突しない場合、自動的に変更をマージすることができます。変更が衝突する場合はどちらの変更をファイルに受け入れるかを選択しなければなりません。

チェンジリストはアトミックであるため、サブミットを成功させるにはチェンジリスト内のあらゆるファイルの衝突を解決しなければなりません。以下の 3 つのいずれかの方法により、衝突を解決することができます。

- ・ **自動**：多くの場合、「*yours*」（クライアント・ワークスペースにある反映先リビジョン）または「*theirs*」（ディポにある反映元リビジョン）のどちらの変更を受け入れるかを判断することになります。この「*yours*」あるいは「*theirs*」を承諾する衝突解決タイプは **マージを伴わない自動衝突解決** と呼ばれます。
- ・ **マージ結果を承諾**：「*theirs*」ファイルに対する変更と「*yours*」ファイルに対する変更とが衝突しない場合があります。そのような場合、PERFORCE は 2 つのファイルをマージし、マージ結果を受け入れるオプションを提供します。そのような解決方法を、「安全な」マージによる **自動衝突解決** といいます。
- ・ **手動マージ**：他に、「*theirs*」と「*yours*」とで同一の行が変更されている場合があります。そのような行は衝突していると言います。変更が衝突している場合、PERFORCE は可能な限り多くの違いを解決し、手動解決のための衝突マーカを含むマージファイルを生成します。マージされたファイルを手動で編集してサブミットするか、衝突マーカも含めマージ結果を承諾して以降のチェンジリストにおいて衝突を解決する必要があります。

複雑なブランチ構造をコピーする

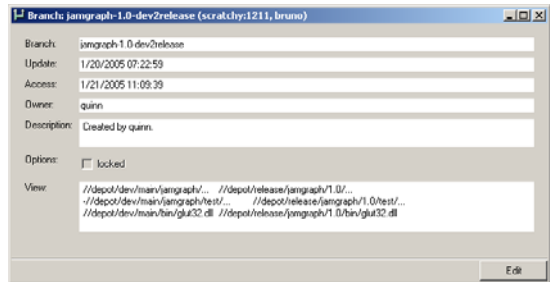
PERFORCE ではブランチのための 2 つのメカニズムとして、**ファイル指定を利用した反映**、および **ブランチ・マッピングを使用した反映** を提供しています。

単純なブランチ構造の場合、手動で反映元および反映先のファイルのパスを指定して、ファイル指定により反映元ファイルを反映先ブランチに反映させることができます。ファイル指定を使う場合、ブランチを行う度に反映元および反映先のコードラインを手動で指定する必要があります。

より複雑なブランチ構造の場合、**ブランチ・マッピング**を設定することにより、非常に複雑なブランチ構造であっても確実にコピーすることができます。ブランチ・マッピングには、反映元ブランチ内のファイルの、反映先ブランチへの反映方法を制御する **マッピング・ルール** (ブランチ・ビュー) のセットが含まれています。ブランチ・マッピングを設定したら、それを使用して反映を行うことにより、ブランチ・ビューで指定されているすべての反映を行うことができます。

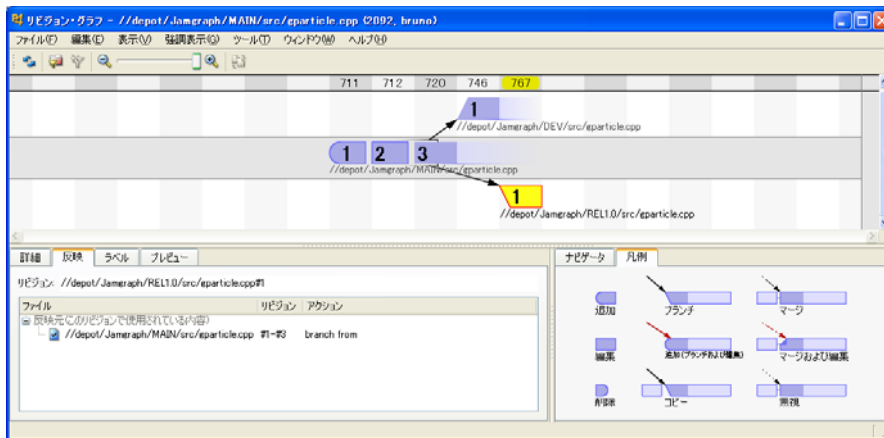
例えば、画面コピーのブランチ・マッピング
 グでは3つのマッピングが示されています。

1. //depot/dev/main/jamgraph にあるすべてのファイルの //depot/release/jamgraph/1.0 ディレクトリとのマッピング
2. /jamgraph/test にある作業が主要ラインにコピーされないようにする排他的マッピング
3. jamgraph プロジェクトとは無関係の共通の /bin ビルド・ディレクトリにある DLL (glut32.dll) デリバラブルを含むマッピング



コードライン間の変更履歴を追跡する

P4V のリビジョン・グラフ機能は、ブランチをまたがった（あるいはブランチ間の）ファイル履歴を視覚的に表現するのに便利です。



サンプル画面イメージは単純なりビジョン・グラフを示しています。リビジョン #1 からリビジョン #3 として表現されているファイル変更は、main コードライン (//depot/Jamgraph/MAIN/...) からリリース・ブランチ (//depot/Jamgraph/REL1.0/...) および開発ブランチ (//depot/Jamgraph/DEV/...) に反映されています。

ブランチに関する詳細説明

PERFORCE のブランチ機能は比較的単純ですが、ブランチの理論はきわめて複雑です。いつブランチを作成するのがよいか、どの時点でコード変更を 1 つのコードラインからもう 1 つのコードラインに伝達するのがよいか、誰がマージ実行の責任を負うのかなど、どの SCM システムを使用しているてもこのような疑問は発生するでしょうし、その答えも単純ではありません。

PERFORCE のブランチ機能の最適な実装方法および技術的詳細が記述されている、インター・ファイル・ブランチの報告書は下記から入手できます。

<http://www.perforce.com/perforce/branch.html>

ラベルに関する注意

ほとんどの SCM システムには、ラベルをファイルに貼り付けるタスクがあり、これはタグ付けとも呼ばれます。しかし、PERFORCE での動作は少し異なります。ラベルとは個々のファイルに適用される属性ではなく、ファイル・リビジョンのリストから構成される名前付きのコンテナです。1 つのコマンドによってファイルのグループにラベルを付けることができ、そのラベルを使用すれば、含まれるすべてのファイル・リビジョンを示すことが可能です。ラベルの典型的な用途の 1 つは、プログラムの特定バージョンを構築するために使用されるリビジョン集合を定義することです。PERFORCE では、日付やチェンジリスト番号による参照を含むいくつかの方法で、ファイル・リビジョンを示すことができます。

次の段階

作業と障害追跡

PERFORCE にはジョブと呼ばれる基本的な障害追跡システムが含まれています。PERFORCE ジョブは、バグ修正や変更要求など実行すべき作業の記述です。PERFORCE のジョブ追跡メカニズムにより、1 つ以上のジョブをジョブで指定された作業の実装を行うチェンジリストにリンクすることができます。ジョブをチェンジリストに関連付けることにより、作業が完了しているか、いつ完了したのか、だれがその作業を行い、どのファイル・リビジョンに影響しているのかをチーム・メンバが知ることができます。チェンジリストにリンクされたジョブは、チェンジリストをサブミットすると完了済みとしてマークされます。

ジョブ・システムにより追跡される情報の種類はカスタマイズ可能です。PERFORCE 管理者は、PERFORCE ジョブにより使用されるフィールドの追加、変更、削除を行うことができます。詳しくは『システム管理者ガイド』を参照してください。

PERFORCE では現在、PERFORCE とサードパーティの障害追跡/ワークフロー管理のシステムとを統合するために、2 つの独立したプラットフォームを提供しています。両プラットフォームによって、情報を PERFORCE のジョブ・システムと外部の障害追跡システムとの間で共有することができます。

P4DTG (PERFORCE 障害追跡ゲートウェイ) は、グラフィカルな構成エディタと複製エンジンを含む統合プラットフォームです。詳しくは、下記サイトを参照してください。

<http://www.perforce.com/perforce/products/p4dtg.html>

日本語版 PERFORCE は、P4DTG をサポートしていませんのでご注意ください。

ファイルにラベルをタグ付けする

PERFORCE のラベルはタグ付けされたファイル・リビジョンのセットです。ラベルにより特定のファイル・グループをクライアント・ワークスペース内に再現することができます。ラベルはチェンジリストとは異なります。チェンジリスト番号がチェンジリストをサブミットした時点のディポ内の全ファイルの状態を示すのに対し、ラベルはファイル・リビジョンが複数のチェンジリストでサブミットされた作業を表す場合でも、そのようなファイル・グループのタグ付けに使用されます。

チェンジリストとラベルのもう 1 つの違いは、チェンジリストは PERFORCE が規定する番号が付けられるのに対し、ラベルにはユーザが指定した名前を付けられることです。例えば、特定のリリースを構成するファイル・リビジョンに `rel12.1` というラベルをタグ付けたいとします。後に、`rel12.1` というタグの付いたリビジョンを更新して、以降のチェンジリストで行われた修正を反映することができます。そしてワークスペースをラベルと同期させることによって、すべてのタグ付けされたリビジョンをクライアント・ワークスペースに取得することができます。

ラベルに関して詳しくは、『P4 ユーザ・ガイド』を参照してください。

エディタとマージ・ツール

ほとんどの PERFORCE クライアント・プログラムには、好みのテキスト・エディタまたはマージ・ツールを指定できるオプションがあります。例えば、コマンドライン・クライアントでは環境変数の `P4EDITOR` および `P4MERGE` を使用することにより好みのエディタやマージ・ツールを起動できます。

詳細については各 PERFORCE クライアント・プログラムの関連文書を参照してください。

プロテクションと権限付与

PERFORCE ではディボへの無許可のアクセスや不注意によるアクセスを防ぐためのプロテクション機能が提供されています。プロテクション機能により、どの PERFORCE コマンドを、どのファイルに対して、誰によって、どのクライアント・ワークステーションから実行可能であるかが決定されます。管理者は PERFORCE コマンドライン・クライアントで `p4 protect` コマンドを使用するか、PERFORCE 管理ツールの P4Admin を使用して、プロテクションを設定することができます。

詳しくは『システム管理者ガイド』を参照してください。

ユーザとライセンス

PERFORCE サーバがサポートするユーザ数に応じて、ライセンスが発行されます。ライセンス情報は、サーバ・ルート・ディレクトリの `license` というファイルに格納されています。`license` ファイルは PERFORCE Software により提供されるプレーンテキスト・ファイルです。`license` ファイルが存在しない場合、PERFORCE サーバはその利用を 2 ユーザと 5 クライアント・ワークスペースに制限します。

詳しくは『システム管理者ガイド』を参照するか、技術サポートにお問い合わせください。

PERFORCE の詳細情報

すべての PERFORCE クライアント・プログラムから以下の方法でオンラインヘルプを参照できます。

- ・ グラフィカル PERFORCE クライアント・プログラムからヘルプメニューを使用する
- ・ コマンドライン・クライアントのヘルプは、コマンドラインから `p4 help` と入力する

PERFORCE 関連文書は以下のウェブサイトから入手できます。

<http://www.perforce.com/perforce/technical.html>

PERFORCE ナレッジベース、掲載記事の完全なリストは以下のサイトにあります。

<http://kb.perforce.com/Technotes>

PERFORCE では研修コースも提供しています。詳細は以下のサイトをご覧ください。

<http://www.perforce.com/perforce/services/training/index.html>

メーリングリストにより他の PERFORCE ユーザと意見や質問のやり取りをすることもできます。

<http://maillist.perforce.com/mailman/listinfo/perforce-user>

PERFORCE サポート担当者にメールまたは電話でお問い合わせいただくことも可能です。

<http://www.perforce.com/perforce/support.html>